DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

Facultad de Informática
Universidad Politécnica de Madrid

PhD THESIS

Contributions to Bayesian network learning with applications to neuroscience

Author

Pedro L. López-Cruz
MS Computer Science
MS Artificial Intelligence

PhD supervisors

Pedro Larrañaga
PhD Computer Science

Concha Bielza
PhD Computer Science

2013

Thesis Committee

President: Serafín Moral

External Member: Helge Langseth

Member: Javier DeFelipe

Member: Luis de Campos

Secretary: Hanen Borchani

A mis padres, Alcázar y Luis

Acknowledgements

This work would not have been possible without the help and support of many people. First of all, I would like to thank my supervisors, Pedro Larrañaga and Concha Bielza, for their trust, encouragement and guidance. Their supervision and hard work have deeply inspired me during this period.

I am grateful to Javier DeFelipe and Ruth Benavides-Piccione for their help and patience while introducing me to the neuroscience field and interdisciplinary research. I extend my gratitude to all the people working in the Cajal Blue Brain project and to the group of experts who participated in the experiments carried out in this research. Also, I would like to thank Thomas D. Nielsen and the members of the Machine Intelligence group at Aalborg University for their warm welcome and hospitality during the cold Danish winter.

I would like to thank my colleagues at the Computational Intelligence Group at UPM for their help and for providing a friendly and exciting work environment: Rubén Armañanzas, Diego Vidaurre, Alfonso Ibáñez, Luis Guerra, Hossein Karshenas, Laura Antón, Bojan Mihaljevic and Hanen Borchani.

This dissertation would not have been possible without the financial support offered by the Spanish Ministry of Education through a personal FPU grant AP2009-1772. I would also like to thank the Cajal Blue Brain (C080020-09), TIN2007-62626 and TIN2010-20900-C04-04 projects that supported my research during these years. I thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Visualization Center of Madrid (CeSViMa).

Lastly, I thank all my family and friends for their support and encouragement. My greatest gratitude goes to my mother and my father, who are my role models and guiding lights. This work is dedicated to them.

Abstract

Neuronal morphology is a key feature in the study of brain circuits, as it is highly related to information processing and functional identification. Neuronal morphology affects the process of integration of inputs from other neurons and determines the neurons which receive the output of the neurons. Different parts of the neurons can operate semi-independently according to the spatial location of the synaptic connections. As a result, there is considerable interest in the analysis of the microanatomy of nervous cells since it constitutes an excellent tool for better understanding cortical function. However, the morphologies, molecular features and electrophysiological properties of neuronal cells are extremely variable. Except for some special cases, this variability makes it hard to find a set of features that unambiguously define a neuronal type. In addition, there are distinct types of neurons in particular regions of the brain. This morphological variability makes the analysis and modeling of neuronal morphology a challenge.

Uncertainty is a key feature in many complex real-world problems. Probability theory provides a framework for modeling and reasoning with uncertainty. Probabilistic graphical models combine statistical theory and graph theory to provide a tool for managing domains with uncertainty. In particular, we focus on Bayesian networks, the most commonly used probabilistic graphical model. In this dissertation, we design new methods for learning Bayesian networks and apply them to the problem of modeling and analyzing morphological data from neurons.

The morphology of a neuron can be quantified using a number of measurements, e.g., the length of the dendrites and the axon, the number of bifurcations, the direction of the dendrites and the axon, etc. These measurements can be modeled as discrete or continuous data. The continuous data can be linear (e.g., the length or the width of a dendrite) or directional (e.g., the direction of the axon). These data may follow complex probability distributions and may not fit any known parametric distribution. Modeling this kind of problems using hybrid Bayesian networks with discrete, linear and directional variables poses a number of challenges regarding learning from data, inference, etc.

In this dissertation, we propose a method for modeling and simulating basal dendritic trees from pyramidal neurons using Bayesian networks to capture the interactions between the variables in the problem domain. A complete set of variables is measured from the dendrites, and a learning algorithm is applied to find the structure and estimate the parameters of the probability distributions included in the Bayesian networks. Then, a simulation algorithm is used to build the virtual dendrites by sampling values from the Bayesian networks, and a thorough evaluation is performed to show the model's ability to generate realistic dendrites. In this first approach, the variables are discretized so that discrete Bayesian networks can be learned and simulated.

Then, we address the problem of learning hybrid Bayesian networks with different kinds of variables. Mixtures of polynomials have been proposed as a way of representing probability densities in hybrid Bayesian networks. We present a method for learning mixtures of polynomials approximations of one-dimensional, multidimensional and conditional probabil-

ity densities from data. The method is based on basis spline interpolation, where a density is approximated as a linear combination of basis splines. The proposed algorithms are evaluated using artificial datasets. We also use the proposed methods as a non-parametric density estimation technique in Bayesian network classifiers.

Next, we address the problem of including directional data in Bayesian networks. These data have some special properties that rule out the use of classical statistics. Therefore, different distributions and statistics, such as the univariate von Mises and the multivariate von Mises–Fisher distributions, should be used to deal with this kind of information. In particular, we extend the naive Bayes classifier to the case where the conditional probability distributions of the predictive variables given the class follow either of these distributions. We consider the simple scenario, where only directional predictive variables are used, and the hybrid case, where discrete, Gaussian and directional distributions are mixed. The classifier decision functions and their decision surfaces are studied at length. Artificial examples are used to illustrate the behavior of the classifiers. The proposed classifiers are empirically evaluated over real datasets.

We also study the problem of interneuron classification. An extensive group of experts is asked to classify a set of neurons according to their most prominent anatomical features. A web application is developed to retrieve the experts' classifications. We compute agreement measures to analyze the consensus between the experts when classifying the neurons. Using Bayesian networks and clustering algorithms on the resulting data, we investigate the suitability of the anatomical terms and neuron types commonly used in the literature. Additionally, we apply supervised learning approaches to automatically classify interneurons using the values of their morphological measurements.

Then, a methodology for building a model which captures the opinions of all the experts is presented. First, one Bayesian network is learned for each expert, and we propose an algorithm for clustering Bayesian networks corresponding to experts with similar behaviors. Then, a Bayesian network which represents the opinions of each group of experts is induced. Finally, a consensus Bayesian multinet which models the opinions of the whole group of experts is built. A thorough analysis of the consensus model identifies different behaviors between the experts when classifying the interneurons in the experiment. A set of characterizing morphological traits for the neuronal types can be defined by performing inference in the Bayesian multinet. These findings are used to validate the model and to gain some insights into neuron morphology.

Finally, we study a classification problem where the true class label of the training instances is not known. Instead, a set of class labels is available for each instance. This is inspired by the neuron classification problem, where a group of experts is asked to individually provide a class label for each instance. We propose a novel approach for learning Bayesian networks using count vectors which represent the number of experts who selected each class label for each instance. These Bayesian networks are evaluated using artificial datasets from supervised learning problems.

Resumen

La morfología neuronal es una característica clave en el estudio de los circuitos cerebrales, ya que está altamente relacionada con el procesado de información y con los roles funcionales. La morfología neuronal afecta al proceso de integración de las señales de entrada y determina las neuronas que reciben las salidas de otras neuronas. Las diferentes partes de la neurona pueden operar de forma semi-independiente de acuerdo a la localización espacial de las conexiones sinápticas. Por tanto, existe un interés considerable en el análisis de la microanatomía de las células nerviosas, ya que constituye una excelente herramienta para comprender mejor el funcionamiento de la corteza cerebral. Sin embargo, las propiedades morfológicas, moleculares y electrofisiológicas de las células neuronales son extremadamente variables. Excepto en algunos casos especiales, esta variabilidad morfológica dificulta la definición de un conjunto de características que distingan claramente un tipo neuronal. Además, existen diferentes tipos de neuronas en regiones particulares del cerebro. La variabilidad neuronal hace que el análisis y el modelado de la morfología neuronal sean un importante reto científico.

La incertidumbre es una propiedad clave en muchos problemas reales. La teoría de la probabilidad proporciona un marco para modelar y razonar bajo incertidumbre. Los modelos gráficos probabilísticos combinan la teoría estadística y la teoría de grafos con el objetivo de proporcionar una herramienta con la que trabajar bajo incertidumbre. En particular, nos centraremos en las redes bayesianas, el modelo más utilizado dentro de los modelos gráficos probabilísticos. En esta tesis hemos diseñado nuevos métodos para aprender redes bayesianas, inspirados por y aplicados al problema del modelado y análisis de datos morfológicos de neuronas.

La morfología de una neurona puede ser cuantificada usando una serie de medidas, por ejemplo, la longitud de las dendritas y el axón, el número de bifurcaciones, la dirección de las dendritas y el axón, etc. Estas medidas pueden ser modeladas como datos continuos o discretos. A su vez, los datos continuos pueden ser lineales (por ejemplo, la longitud o la anchura de una dendrita) o direccionales (por ejemplo, la dirección del axón). Estos datos pueden llegar a seguir distribuciones de probabilidad muy complejas y pueden no ajustarse a ninguna distribución paramétrica conocida. El modelado de este tipo de problemas con redes bayesianas híbridas incluyendo variables discretas, lineales y direccionales presenta una serie de retos en relación al aprendizaje a partir de datos, la inferencia, etc.

En esta tesis se propone un método para modelar y simular árboles dendríticos basales de neuronas piramidales usando redes bayesianas para capturar las interacciones entre las variables del problema. Para ello, se mide un amplio conjunto de variables de las dendritas y se aplica un algoritmo de aprendizaje con el que se aprende la estructura y se estiman los parámetros de las distribuciones de probabilidad que constituyen las redes bayesianas. Después, se usa un algoritmo de simulación para construir dendritas virtuales mediante el muestreo de valores de las redes bayesianas. Finalmente, se lleva a cabo una profunda evaluación para verificar la capacidad del modelo a la hora de generar dendritas realistas. En esta primera aproximación, las variables fueron discretizadas para poder aprender y muestrear las redes bayesianas.

A continuación, se aborda el problema del aprendizaje de redes bayesianas con diferentes tipos de variables. Las mixturas de polinomios constituyen un método para representar densidades de probabilidad en redes bayesianas híbridas. Presentamos un método para aprender aproximaciones de densidades unidimensionales, multidimensionales y condicionales a partir de datos utilizando mixturas de polinomios. El método se basa en interpolación con *splines*, que aproxima una densidad como una combinación lineal de *splines*. Los algoritmos propuestos se evalúan utilizando bases de datos artificiales. Además, las mixturas de polinomios son utilizadas como un método no paramétrico de estimación de densidades para clasificadores basados en redes bayesianas.

Después, se estudia el problema de incluir información direccional en redes bayesianas. Este tipo de datos presenta una serie de características especiales que impiden el uso de las técnicas estadísticas clásicas. Por ello, para manejar este tipo de información se deben usar estadísticos y distribuciones de probabilidad específicos, como la distribución univariante von Mises y la distribución multivariante von Mises–Fisher. En concreto, en esta tesis extendemos el clasificador naive Bayes al caso en el que las distribuciones de probabilidad condicionada de las variables predictoras dada la clase siguen alguna de estas distribuciones. Se estudia el caso base, en el que sólo se utilizan variables direccionales, y el caso híbrido, en el que variables discretas, lineales y direccionales aparecen mezcladas. También se estudian los clasificadores desde un punto de vista teórico, derivando sus funciones de decisión y las superficies de decisión asociadas. El comportamiento de los clasificadores se ilustra utilizando bases de datos artificiales. Además, los clasificadores son evaluados empíricamente utilizando bases de datos reales.

También se estudia el problema de la clasificación de interneuronas. Desarrollamos una aplicación web que permite a un grupo de expertos clasificar un conjunto de neuronas de acuerdo a sus características morfológicas más destacadas. Se utilizan medidas de concordancia para analizar el consenso entre los expertos a la hora de clasificar las neuronas. Se investiga la idoneidad de los términos anatómicos y de los tipos neuronales utilizados frecuentemente en la literatura a través del análisis de redes bayesianas y la aplicación de algoritmos de clustering. Además, se aplican técnicas de aprendizaje supervisado con el objetivo de clasificar de forma automática las interneuronas a partir de sus valores morfológicos.

A continuación, se presenta una metodología para construir un modelo que captura las opiniones de todos los expertos. Primero, se genera una red bayesiana para cada experto y se propone un algoritmo para agrupar las redes bayesianas que se corresponden con expertos con comportamientos similares. Después, se induce una red bayesiana que modela la opinión de cada grupo de expertos. Por último, se construye una multired bayesiana que modela las opiniones del conjunto completo de expertos. El análisis del modelo consensuado permite identificar diferentes comportamientos entre los expertos a la hora de clasificar las neuronas. Además, permite extraer un conjunto de características morfológicas relevantes para cada uno de los tipos neuronales mediante inferencia con la multired bayesiana. Estos descubrimientos se utilizan para validar el modelo y constituyen información relevante acerca de la morfología neuronal.

Por último, se estudia un problema de clasificación en el que la etiqueta de clase de los datos de entrenamiento es incierta. En cambio, disponemos de un conjunto de etiquetas para cada instancia. Este problema está inspirado en el problema de la clasificación de neuronas, en el que un grupo de expertos proporciona una etiqueta de clase para cada instancia de manera individual. Se propone un método para aprender redes bayesianas utilizando vectores de cuentas, que representan el número de expertos que seleccionan cada etiqueta de clase para cada instancia. Estas redes bayesianas se evalúan utilizando bases de datos artificiales de problemas de aprendizaje supervisado.

Contents

List of Figures

List of Tables

Acronyms

ANN artificial neural network

BBA basic belief assignment

BIC Bayesian information criterion

BN Bayesian network

BNC Bayesian network classifier

BRAIN Brain Research through Advancing Innovative Neurotechnologies project

CLG conditional linear Gaussian

CoMEM counts multinomial expectation maximization algorithm

CPT conditional probability table

DAG directed acyclic graph

EM expectation maximization

FMM finite mixture model

GTT greedy thick-thinning Bayesian network structure learning algorithm

HBP Human Brain Project

HC hill-climbing algorithm

JPD joint probability distribution

KDD knowledge discovery in databases

KDE kernel density estimation

KL Kullback-Leibler divergence

LIP Lagrange interpolation polynomial

MAP maximum a posteriori

MG multivariate Gaussian

MI mutual information

ML maximum likelihood

MM max-min algorithm

MoP mixture of polynomials

MoTBF mixture of truncated basis functions

MPE most probable explanation

MTE mixture of truncated exponentials

NB naive Bayes

$PABA\kappa$ Prevalence-Adjusted Bias-Adjusted kappa index

PGM probabilistic graphical model

PLEM probabilistic label expectation maximization algorithm

RS 2-phase restricted search max-min algorithm

SEM structural expectation-maximization

SVM support vector machine

TA tabu search algorithm

TAN tree-augmented naive Bayes

TSE Taylor series expansion

vMNB von Mises naive Bayes

vMFNB von Mises–Fisher naive Bayes

Part I

INTRODUCTION

Chapter 1

Introduction

The technological advance in the last decades has exponentially increased the amount of data available for both companies and researchers. Therefore, analyzing this information by hand using classical statistical tools is not feasible anymore. Machine learning provides the tools for properly managing and working with these large amounts of data.

The study of the brain is one of the most important scientific challenges nowadays. The new tools and techniques discovered in the fields of chemistry, physics, etc. have enabled the acquisition of detailed data about the morphology and the chemical, electrical and physiological features of the neurons and other structures in the brain. Additionally, the development of computer science has provided neuroscientists with tools for visualizing, reconstructing, measuring and storing a large amount of neurological data. Machine learning tools can be used for analyzing and studying these data.

We focus on the problem of modeling, analyzing, simulating, classifying, etc. the morphological structure of nervous cells (neurons). The number of neurons in the human brain is estimated to be in the order of 86.1×10^9 [25]. However, there are no two neurons with the same morphology. Despite recent advances in molecular biology and new discoveries related to neuronal development, current knowledge about neuron structure is still incomplete. Neuronal morphology shows a huge variability across neuron types, brain areas and animal species [140]. This morphological variability makes it difficult to find the anatomical traits that define neuron types [413].

In this dissertation, we propose using probabilistic graphical models to address some of the problems regarding the study of neuronal morphology. Probabilistic graphical models combine probability theory and graph theory into a single framework that is able to manage complex real-world domains. The probabilistic part models the variables in the problem domain and their relationships using probability distributions. The graphical component represents these variables and their relationships, so experts in the domain can interpret it to obtain new insights into the problem. In particular, we will focus on Bayesian networks, the most frequently used type of probabilistic graphical model.

Chapter outline

This chapter is organized as follows. The main hypothesis and objectives in this dissertation are presented in Section 1.1. Then, the organization of this manuscript is explained in Section 1.2.

1.1 Hypothesis and objectives

Based on the evidences and motivation presented above, the main research hypothesis guiding this dissertation can be stated as follows:

"The variability of neuronal morphology makes it difficult to find a set of anatomical traits that unambiguously define and describe the different neuron types. The characterization of the neuronal morphology should be approached from a statistical point of view to overcome this variability. Within statistics, probabilistic graphical models are appropriate tools for tackling this problem."

Based on this hypothesis, the main objectives of this dissertation are:

Proposing methods for modeling neuronal morphology. To accurately model the complete neuronal morphology, the models will need to include different types of variables (discrete, continuous, directional). Additionally, these variables may not follow any known parametric densities. Therefore, one objective of this research is studying flexible hybrid Bayesian networks that are able to accurately model the neuronal morphology.

Analyzing, describing and managing data from the main neuronal types. In particular, we will focus on the subset of GABAergic interneurons from the neocortex. The main objective is identifying the main interneuron types. We will ask a group of neuroscientists to classify a set of interneurons according to their main morphological traits. Then, we will analyze the agreement between the experts and analyze the data from different perspectives: supervised learning, unsupervised learning, statistical modeling, etc. New Bayesian network models will be designed for solving these machine learning tasks.

1.2 Document organization

The manuscript includes 13 chapters grouped into six parts:

Part I. Introduction

This is the current part.

- Chapter 1 introduces this dissertation, stating the research hypothesis and objectives and summarizing the document organization.

Part II. Background

This part includes five chapters introducing the basic concepts, definitions and nomenclature used throughout this dissertation. The chapters explain the basic theory behind the models and tools used in the following chapters. The state-of-the-art is discussed in each of these chapters.

- Chapter 2 presents an overview of machine learning. The main machine learning problems addressed in this dissertation are discussed in depth, i.e., dependency modeling, supervised learning, unsupervised learning, etc. The different methods and approaches for solving each of these problems are briefly reviewed, and some notes are given on how to evaluate the performance of the approaches for the different problems. A list of the machine learning software used in this dissertation is provided.
- Chapter 3 introduces probabilistic graphical models, with a special focus on Bayesian networks, which are the main topic in this dissertation. The chapter includes the theoretical foundations of Bayesian networks, and discusses some of the issues that will be addressed during this dissertation, e.g., parameterization, learning from data, inference, etc. Specific Bayesian network models for solving supervised learning problems (Bayesian classifiers) and unsupervised learning problems (finite mixture models) are also reviewed. A list of the software related to Bayesian networks and used in this dissertation is included.
- Chapter 4 deals with the problem of analyzing the consensus between a group of experts on a classification task. This chapter reviews the most commonly used agreement indices for quantifying the consensus between a group of experts. The assumptions, advantages and disadvantages of each of these measures are highlighted.
- Chapter 5 includes an introduction to directional statistics, which provides the theoretical background and the techniques for properly analyzing data related to directions or angles of a given phenomenon. The chapter details how to compute summary statistics of a dataset of directional data, and introduces the two most commonly used probability distributions for modeling directional data, i.e., the von Mises and the von Mises–Fisher distributions.
- Chapter 6 introduces the basic concepts related to neuroscience. The chapter starts with a brief historical note on the beginnings and the developments of modern neuroscience. Some notions on brain organization and neuronal morphology are included for reference. The chapter also discusses the problems regarding neuron classification and the morphological variability of neurons. Some of the current efforts in neuroscience research are also highlighted.

Part III. Contributions to Bayesian network learning

This part of the dissertation includes our proposals regarding Bayesian network learning and the application to modeling and analyzing basal dendritic trees in pyramidal neurons.

- Chapter 7 applies Bayesian networks to the problem of modeling and simulating basal dendritic trees from pyramidal cells from the mouse neocortex. The Bayesian networks are learned from data and then used to sample virtual dendritic trees, which are compared to the real dendritic trees to evaluate the model. The analyses of the models reveal relationships that conform to current neuroanatomical knowledge and support model correctness. At the same time, studying the relationships in the models helps identifying new interactions between variables related to dendritic morphology.
- In Chapter 8 we propose methods for learning mixture of polynomials approximations of one-dimensional, multidimensional and conditional probability densities from data. The methods are empirically evaluated using artificial datasets and then used as a non-parametric density estimation technique for solving supervised learning problems with Bayesian networks.
- Chapter 9 introduces Bayesian classifiers where the predictive variables are defined in a directional or circular domain. We also study hybrid classifiers with linear, directional and discrete variables. The decision functions of the classifiers are derived, analyzed and compared from a geometrical point of view.

Part IV. Contributions to consensus analysis

This part of the dissertation includes our proposals on probabilistic graphical models for modeling and analyzing the consensus between experts when classifying interneuron types.

- Chapter 10 studies the problem of the classification of GABAergic interneurons from the neocortex. The agreement between experts when classifying interneuron types is quantified using the techniques introduced in Chapter 4. The data collected in the experiment is analyzed from different perspectives, e.g., as a statistical modeling problem, as a supervised learning problem, as an unsupervised learning problem, etc.
- In Chapter 11 we propose a method for modeling the consensus between a group of experts by building a Bayesian multinet. The proposed approach is applied to the interneuron classification problem.
- Chapter 12 studies the problem of learning Bayesian classifiers when the true class labels of the training instances are not known. Instead, the class information is modeled as a count vector modeling the number of votes given by a group of experts to each class label for each instance.

Part V. Conclusions

This part concludes this dissertation.

- Chapter 13 summarizes the contributions of this dissertation and the scientific results derived from it. The chapter also discusses the research lines opened in this work and summarizes future research topics.

Part VI. Appendices

This part includes three appendices with derivations and supplementary results.

- Appendix A includes the derivations of the decision functions and their corresponding decision surfaces for the von Mises naive Bayes classifiers proposed in Chapter 9.
- Appendix B includes the derivations of the decision functions and their corresponding decision surfaces for the von Mises–Fisher naive Bayes classifiers proposed in Chapter 9.
- Appendix C includes supplementary results of the Bayesian network classifiers proposed in Chapter 12.

Part II

BACKGROUND

Chapter 2

Machine learning

2.1 Introduction

The amount of data generated and stored in databases has grown exponentially in the last decades. Therefore, it is not possible to manually handle and analyze such amount of data using classical statistical techniques. The *knowledge discovery in databases* (KDD) is the discipline of Artificial Intelligence defined as [169]:

> "The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Here, data are a set of facts (for example, cases in a database), and pattern is an expression in some language describing a subset of the data or a model applicable to the subset. Hence, in our usage here, extracting a pattern also designates fitting a model to data; finding structure from data; or, in general, making any high-level description of a set of data."

The KDD process is an interactive process and can be divided into nine steps [168]: 1) Understanding the domain of the problem, 2) Generating a dataset, 3) Cleaning and preprocessing the data, 4) Reducing, projecting and selecting data, 5) Identifying the aim of the KDD process, 6) Selecting the appropriate methods and algorithms, 7) Data mining, 8) Interpreting the discovered patterns and 9) Exploiting the new knowledge. Data mining is the main step of the KDD process. Therefore, the term data mining is frequently used to refer to the whole process.

Alpaydin [13] defines *machine learning* as "programming computers to optimize a performance criterion using example data or past experience." *Data mining* is the application of machine learning techniques to large databases. Machine learning techniques can be used to solve different tasks [168]:

> Summarization refers to the compact description of the main features of a dataset.
>
> Dependency modeling searches for valuable patterns in the data and models the relationships between features of objects.

Supervised learning, also known as classification, deals with the problem of assigning a discrete class label to an object based on a set of predictive features describing its main properties.

Unsupervised learning, also known as clustering, addresses the problem of grouping a dataset into clusters of objects with similar properties.

Regression studies the prediction of a continuous value from a set of predictive features.

Time series analysis studies the changes and behavior of some phenomenon over time.

Machine learning techniques can be predictive and/or descriptive. Predictive techniques can be used to answer questions about unknown information in a problem, e.g., to estimate the temperature in a weather forecast or to predict how likely it is for a patient to develop a certain disease. On the other hand, descriptive models can be used to find interesting relationships between the features or the instances in the datasets, e.g., to find correlations between abnormal gene expressions and diseases or to group instances or cases with similar properties. The division into these two groups is not strict, and a lot of techniques can be used to perform both tasks to some extent. For instance, classification trees can be transformed to rules that describe relevant relationships between the features. Also, probabilistic graphical models, which describe the probabilistic relationships between the features of the problem, can be adapted for solving prediction problems.

Chapter outline

This chapter reviews some of the most popular algorithms and approaches for solving machine learning tasks. Section 2.2 deals with the problem of finding frequent patterns and dependencies in datasets. Section 2.3 introduces supervised learning and reviews some of the available approaches and algorithms. Some methods for unsupervised learning are explained in Section 2.4. Other machine learning problems are discussed in Section 2.5. Finally, the chapter ends in Section 2.6 with a brief description of the available software for solving machine learning tasks.

2.2 Dependency modeling

Dependency modeling searches for frequent patterns and dependencies in the data. Given a dataset $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with N objects , where each object $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})$ is a vector of values for the variables $\mathbf{X} = (X_1, \ldots, X_n)$ encoding some of their relevant properties, interesting patterns in the data can be represented as association rules of the form:

$$\texttt{IF } X_1 = x_1 \texttt{ THEN } X_2 = x_2.$$

Association rule mining algorithms find relationships between values of the objects that occur frequently in the dataset. The aim is finding a set of association rules with high support,

confidence and interest (also known as lift) [66]. The support of a rule is the proportion of objects where $X_1 = x_1$ and $X_2 = x_2$ appear together in the dataset. The confidence of a rule is the proportion of objects with $X_2 = x_2$ out of those where $X_1 = x_1$. The lift of a rule is the support of the rule divided by the product of the supports of $X_1 = x_1$ and $X_2 = x_2$, respectively. These three values measure, respectively, the usefulness, certainty and interest of the rules recovered from the dataset. For an association rule to be considered interesting, it has to yield high values for these three metrics. Several algorithms have been proposed for mining association rules from a dataset. Agrawal and Srikant [4] proposed one of the first algorithms for mining Boolean association rules. Several modifications and enhancements have been proposed in the literature to apply the same ideas to categorical data, optimizing the efficiency of the procedures and the quality of the recovered rules, e.g., see [184, 240, 534]. Association rules have also been studied for continuous [24, 512] and temporal data [360, 361]. Recently, metaheuristics such as evolutionary computation or swarm-based approaches have been used for learning association rules, e.g., see [125] for a review.

Bayesian networks [300, 402] are the other most frequently used techniques for finding and modeling the dependencies in a problem. A Bayesian network is a kind of probabilistic graphical model with two main components. The graphical structure of the network encodes the probabilistic conditional independence relationships between the variables in the domain. The probabilistic component models the strength of these probabilistic relationships using probability distributions. These models are the representation of choice in domains with uncertainty. The graphical structure compactly represents the problem domain and can be easily interpreted by experts. Moreover, the information encoded in the probabilistic component can be used to perform inference with mathematically sound methods. These models will be reviewed in depth in Chapter 3.

2.3 Supervised learning

Supervised learning, also known as classification, addresses the problem of predicting the class of an object based on a set of features describing its main properties. For instance, an everyday classification problem can be found in weather forecasting. Let's imagine that the goal is predicting whether or not it will rain tomorrow based on what the weather is like today. That is, we want to select a class label ("rain" or "no rain") based on some data describing the weather today, e.g., temperature, humidity, wind direction, wind speed, air pressure, month of the year, etc. These measurements are collected using meteorological stations and stored in a database for years, e.g., see Table 2.1. Weather forecasting services analyze these data and build classification models which are able to predict with high accuracy whether or not it is going to rain.

Supervised learning [46, 147, 153] is the field of machine learning which studies how to solve this kind of problems. Formally, the features describing the objects (e.g., temperature, humidity, etc.) are encoded in a vector of n variables $\mathbf{X} = (X_1, \ldots, X_n)$. The class labels of the problem (e.g., "rain" or "no rain") are encoded as a class variable C with values

Table 2.1: Example of a database with records for weather forecasting

Day	Temperature	Humidity	...	Wind speed	Rain
1/Jan/2000	4.5	10.2	...	5	yes
2/Jan/2000	6	7.8	...	2.3	no
3/Jan/2000	5.2	20.1	...	3.8	no
...		...			...

in a set $\Omega_C = \{1, \ldots, K\}$. The training dataset is a set of N objects whose class label is known $\mathcal{D}^{\text{train}}_{\mathbf{X},C} = \{(\mathbf{x}_j, c_j)\}_{j=1,\ldots,N}$, where $\mathbf{x}_j = (x_{j1}, \ldots, x_{jn})$ are the values of the predictive variables for the jth instance and c_j is the true class label for the jth instance. Table 2.2 shows an abstract representation of a training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$. Then, a classifier is a function $\gamma : \Omega_{\mathbf{X}} \longrightarrow \Omega_C$ that maps the vector of values of the predictive variables $\mathbf{x}$ to a class label $c \in \Omega_C$, i.e., $\gamma(\mathbf{x}) = c$. Machine learning proposes algorithms for finding such a function γ based on the information contained in the training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$.

Table 2.2: Training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ for a supervised learning problem

Instance	X_1	X_2	$\cdots$	X_n	C
1	x_{11}	x_{12}	$\cdots$	x_{1n}	c_1
2	x_{21}	x_{22}	$\cdots$	x_{2n}	c_2
...		...			...
N	x_{N1}	x_{N2}	$\cdots$	x_{Nn}	c_N

2.3.1 Supervised learning approaches

Supervised learning is the most widely studied problem in machine learning. It has been covered in a number of texts and a huge number of approaches and algorithms have been proposed, e.g., see [13, 153, 303, 519]. A basic description of the main approaches for supervised classification used in this thesis follows.

Bayesian network classifiers (BNCs) [199] are a class of Bayesian networks specially designed to solve supervised classification problems. BNCs model the joint probability distribution over the predictive variables and the class $(\mathbf{X}, C)$. To classify an instance $\mathbf{x}$, the Bayes rule is used to compute the posterior probability $p_{C|\mathbf{X}}(c|\mathbf{x}) = \frac{p_{\mathbf{X}|C}(\mathbf{x}|c)p_C(c)}{p_{\mathbf{X}}(\mathbf{x})}$ of each class label $c \in \Omega_C$ given the values of the predictive variables $\mathbf{x}$. The class with maximum posterior probability is selected as the class label for the instance $\mathbf{x}$. Bayesian network classifiers will be studied in depth in Section 3.4.

Decision or classification trees [33, 384, 385] are hierarchical models that sequentially divide the problem domain into subregions and assign a class label to each subregion. The inner nodes in a decision tree represent predictive features in the problem, and branches going out of the nodes represent values of the predictive features. The leaf nodes of a decision tree contain the class label assigned to each subregion of the problem

domain. To classify an instance, the instance travels the tree from the root node to a leaf according to the values of its features, and the class label at the leaf node is assigned. Decision trees are popular because they are efficient, interpretable, easy to train and reasonably accurate. Some problems of decision trees include overfitting [421], replication of subtrees [28] and fragmentation of the data [507].

Decision or classification rules are a kind of association rules (see Section 2.2) for solving supervised learning problems [204, 486]. The goal is to find the smallest set of rules that is consistent with the training data. Decision trees can be transformed into classification rules. However, rule-based classifiers can yield higher accuracies than decision trees in some problems [332, 335]. Classification rules usually obtain more comprehensive sets of rules, but they can be more difficult to interpret and can yield inconsistent sets of rules when more than two class labels are available in the problem [336, 447].

Instance-based learning approaches [7, 111, 344] do not generate a model or classifier. Instead, they classify a new instance by looking for the most similar instances in the training dataset and returning their labels. If the most similar instances have different class labels, combination rules have been proposed [6], e.g., majority vote, weighted majority vote, etc. Different dissimilarity measures can be used depending on the kind of data: continuous, categorical, temporal, etc.

Artificial neural networks (ANN) [45, 246, 427] are a learning paradigm based on biological neural networks. ANNs have one or more layers of processing units (neurons) connected with each other. Each processing unit aggregates the inputs it receives from other processing units and sends the result to other processing units. The connections between processing units are modeled with weights. ANN learning algorithms find an appropriate structure by establishing connections between the processing units and fit the weights that model the strength of those connections. ANNs usually provide higher accuracies than other methods. However, they operate as a black box and they are difficult to interpret.

Support vector machines (SVM) [1, 69] transform the training dataset by mapping the observations into high dimensional spaces so that the problem becomes linearly separable and can be solved using decision hyperplanes. SVMs have yielded very accurate results in a lot of different settings and are less prone to overfitting than other methods. However, training time is very high and, in addition, the models cannot be easily interpreted.

2.3.2 Evaluation

Correctly evaluating the performance of a classifier is a key step in any machine learning problem. The aim is to estimate how well the classifier predicts the class values for a new instance $\mathbf{x}$ with an unknown class.

The classification accuracy is the most frequently used measure for evaluating the performance of a classifier. Given a labeled dataset $\mathcal{D}_{\mathbf{X},C} = \{(\mathbf{x}_j, c_j)\}_{j=1,\ldots,N}$, the accuracy of a classifier is the proportion of correctly classified instances in $\mathcal{D}_{\mathbf{X},C}$. Formally,

$$acc(\gamma, \mathcal{D}_{\mathbf{X},C}) = \frac{Card(\{j|c_j = c_j^*, j = 1, \ldots, N\})}{N},$$

where $c_j^* = \gamma(\mathbf{x}_j)$ is the class predicted by the classifier for the jth instance.

Unfortunately, using the same training set $\mathcal{D}_{\mathbf{X},C}^{\text{train}}$ for learning γ and estimating its performance yields an optimistic value of the accuracy. That is, if we classify a test dataset $\mathcal{D}_{\mathbf{X},C}^{\text{test}}$ different from the one used for training, the accuracy $acc(\gamma, \mathcal{D}_{\mathbf{X},C}^{\text{test}})$ will be lower than $acc(\gamma, \mathcal{D}_{\mathbf{X},C}^{\text{train}})$. Therefore, other methods have to be used for honestly evaluating the performance of a classifier.

The most frequently used evaluation method is called k-fold cross validation [475]. In a k-fold cross-validation method, the training dataset $\mathcal{D}_{\mathbf{X},C}^{\text{train}}$ is divided into k disjoint sets called folds: $\mathcal{D}_{\mathbf{X},C}^{(m)}, m = 1, \ldots, k$, with $\mathcal{D}_{\mathbf{X},C}^{\text{train}} = \cup_{m=1}^{k} \mathcal{D}_{\mathbf{X},C}^{(m)}$. For each fold m, a classifier $\gamma^{(m)}$ is learned using the data in the $k-1$ folds $\cup_{l \neq m} \mathcal{D}_{\mathbf{X},C}^{(l)}$. Then, the accuracy is computed over the mth fold $acc\left(\gamma^{(m)}, \mathcal{D}_{\mathbf{X},C}^{(m)}\right)$. Finally, an unbiased estimate of the accuracy of the classifier γ learned over the complete dataset $\mathcal{D}_{\mathbf{X},C}^{\text{train}}$ is the mean of the accuracies obtained in the k folds:

$$acc(\gamma, \mathcal{D}_{\mathbf{X},C}^{\text{train}}) = \frac{\sum_{m=1}^{k} acc\left(\gamma^{(m)}, \mathcal{D}_{\mathbf{X},C}^{(m)}\right)}{k}.$$

Figure 2.1 graphically represents the k-fold cross-validation process. This process can be repeated several times to reduce the variance of the estimate. Then, the final estimate of the accuracy is the mean of the estimates computed in each repetition. Typical schemes of k-fold cross validations are 10-fold cross-validation, 5 repetitions of 2-fold cross-validation and 10 repetitions of 10-fold cross-validation.

Stratified k-fold cross-validation is recommended when the class labels are imbalanced, i.e., when the number of instances belonging to each class label is very different. Then, the dataset is divided trying to preserve the proportion of instances of each class in every fold.

Another popular evaluation technique is the leave-one-out method [379]. Leave-one-out is a special case of k-fold cross-validation where the number of folds is equal to the number of instances in the training dataset, i.e., $k = N$. This method is useful for datasets with few instances, i.e., where N is small.

2.4 Unsupervised learning

Unsupervised learning, also known as clustering, studies the problem of finding groups of similar objects in a dataset. This problem appears frequently in machine learning when working with very large datasets where the training instances are unlabeled because it is very costly to obtain the true class labels of all the instances. For instance, imagine that

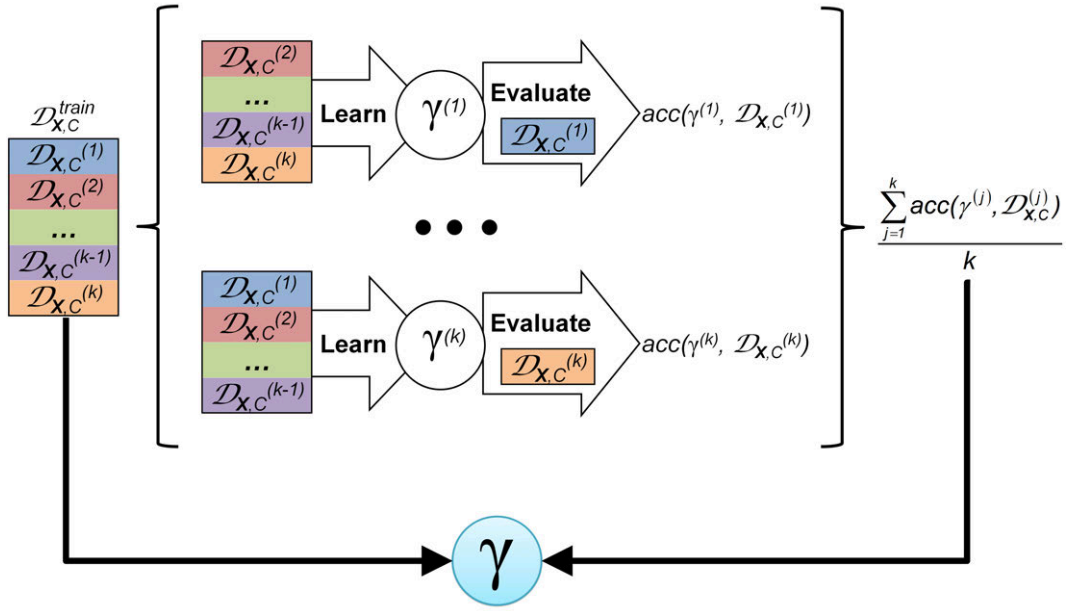


Figure 2.1: The figure shows the k-fold cross validation scheme for honestly estimating the accuracy of a classifier γ from a training dataset $\mathcal{D}^{train}_{\mathbf{X},C}$.

the advertising division of a big supermarket chain wants to study their customers' shopping habits so that they can design more effective advertising campaigns. The supermarket has a large database with the products bought by each customer, and the advertising division would like to group the customers to send them personalized advertisements according to their preferences. However, they do not know how many groups of customers there are, to which group do the customers belong a priori, or what are the main features of each group. This is the typical scenario where unsupervised learning techniques can be applied.

Unsupervised learning methods [165, 525] generate clusters of objects so that the objects within the same cluster have similar features, whereas objects in different clusters have different features. Formally, the definition of an unsupervised learning problem is similar to that of a supervised learning problem. The features describing the main properties of the instances in the dataset are encoded as a vector of variables $\mathbf{X} = (X_1, \ldots, X_n)$. The clusters of instances are modeled with a cluster variable C, and each value $c \in \Omega_C$ denotes a cluster. A clustering algorithm is a function $\gamma : \Omega_{\mathbf{X}} \longrightarrow \Omega_C$ that maps the vector of values of the predictive variables $\mathbf{x}$ to a cluster $c \in \Omega_C$, i.e., $\gamma(\mathbf{x}) = c$. The main difference between an unsupervised and a supervised learning scenario is that function γ is learned from a dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C} = \{(\mathbf{x}_i, c_i)\}_{i=1,\ldots,N}$ where the true cluster $c_i, i = 1, \ldots, N$ of the instances is unknown. Table 2.3 shows a typical dataset used for solving an unsupervised learning problem. Unsupervised learning algorithms analyze the dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ to find a function γ that yields accurate and meaningful clusters.

Table 2.3: Training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ for an unsupervised learning problem

i	X_1	X_2	$\cdots$	X_n	C
1	x_{11}	x_{12}	$\cdots$	x_{1n}	?
2	x_{21}	x_{22}	$\cdots$	x_{2n}	?
$\cdots$		$\cdots$			$\cdots$
N	x_{N1}	x_{N2}	$\cdots$	x_{Nn}	?

2.4.1 Unsupervised learning approaches

Unsupervised learning is the most frequently studied machine learning problem after supervised classification. Therefore, the scientific literature contains many algorithms for unsupervised learning, e.g., see [524] for a review. Some of these approaches adapt supervised learning techniques to the scenario where the training instances are unlabeled. Also, specific methods for unsupervised learning have been proposed. Classical unsupervised learning techniques can be divided into three groups:

Hierarchical paradigms rely on the definition of a distance or dissimilarity measure between the observations. A classical agglomerative (bottom-up) hierarchical clustering algorithm starts with one cluster per observation and iteratively merges the two most similar clusters according to some criterion, called linkage function, which depends on the distances of the observations in the clusters. Therefore, hierarchical clustering techniques do not generate a single partition but a hierarchy of clusters. Different dissimilarity measures and linkage functions yield different hierarchies of clusters. Therefore, these decision should be carefully made taking into account the nature of the data.

Partitional clustering techniques generate a single partition of the objects into clusters by applying an optimization process which maximizes/minimizes an objective function. This objective function usually measures the distances between the objects in the same cluster (minimization) and/or the distance between objects in different clusters (maximization). The number of clusters to be generated is a free parameter that has to be set by the expert. Usually, solutions with different number of clusters are obtained and the final solution is selected according to one or more quality measures [18, 227, 371]. Also, an appropriate distance measure has to be chosen depending on the nature of the data.

Probabilistic clustering deals with the problem of fitting a finite mixture of distributions [367], where each component is the probability distribution which models the observations belonging to the cluster. Probabilistic clustering offers a number of advantages. First, it generates a probabilistic model which describes the data. Using that model, one can compute the (posterior) probability of a given observation belonging to each cluster. Also, it is able to formally address the problem of model selection (finding an appropriate number of clusters). Finite mixture models will be further discussed in Section 3.5.

2.5 Other machine learning problems

Supervised and unsupervised learning are the two extremes of a continuum of learning problems where different amounts of information about the class are available. These problems are interesting because it is frequently the case that some (partial, noisy, uncertain and/or incomplete) information about the true class labels can be obtained with less effort. The development of the Internet has significantly reduced the cost and time needed to obtain such information, and a number of collaborative and crowdsourcing tools are available for that purpose [141]. *Semi-supervised learning* [79] deals with the problem of inducing classifiers when only a (usually small) subset of the training data is labeled. Table 2.4 shows an example of a dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ where the first i instances are labeled and the last $N-i$ instances are unlabeled. Semi-supervised learning problems have been solved using a large number of techniques and by adapting some supervised and unsupervised learning algorithms [2, 539].

Table 2.4: Training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ for a semi-supervised learning problem

i	X_1	X_2	$\cdots$	X_n	C
1	x_{11}	x_{12}	$\cdots$	x_{1n}	c_1
$\cdots$		$\cdots$			$\cdots$
i	x_{i1}	x_{i2}	$\cdots$	x_{in}	c_i
$i+1$	$x_{(i+1)1}$	$x_{(i+1)2}$	$\cdots$	$x_{(i+1)n}$	?
$\cdots$		$\cdots$			$\cdots$
N	x_{N1}	x_{N2}	$\cdots$	x_{Nn}	?

In *partially supervised learning*, also known as *multi-label learning* [99], a subset of all the possible class labels, which contains the true class, is given for each instance. Table 2.5 shows an example of a dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ used for partially supervised learning, where the class can take values in the set $\Omega_C = \{k_1, k_2, k_3\}$ and a subset $\mathbf{c}_i \subseteq \Omega_C$ is given for each instance $i = 1, \ldots, N$. Different machine learning techniques have been used to solve this problem, e.g., see [536] for a review.

Table 2.5: Training dataset $\mathcal{D}^{\text{train}}_{\mathbf{X},C}$ for a partially supervised learning problem

i	X_1	X_2	$\cdots$	X_n	C
1	x_{11}	x_{12}	$\cdots$	x_{1n}	$\{k_1, k_2\}$
2	x_{21}	x_{22}	$\cdots$	x_{2n}	$\{k_3\}$
$\cdots$		$\cdots$			$\cdots$
N	x_{N1}	x_{N2}	$\cdots$	x_{Nn}	$\{k_1, k_2, k_3\}$

Multiple instance learning [136] addresses the classification problem when the instances are grouped into "bags of points", and each one of those bags is labeled as positive or negative depending on whether or not it contains any positive instances, respectively.

2.6 Software

A large number of software packages and suites are available for solving machine learning problems. A brief description of the software used in this dissertation follows. Software for working with Bayesian networks will be reviewed in Section 3.6.

WEKA[1] [236] is a multi-platform data mining suite written in Java. It includes a large collection of algorithms for solving supervised and unsupervised learning problems, and methods for finding association rules. Preprocessing and visualization features are available to explore and analyze the data. The algorithms can be run using the graphical user interface, through the command-line interface or called from a Java code. Complex experiments where different algorithms are run and compared over several datasets can be defined using the `Experimenter` module.

KEEL[2] [8] is an open-source cross-platform suite written in Java. It focuses on evolutionary algorithms for learning classification and regression models. It includes techniques for preprocessing data and statistical tests to analyze the results.

R[3] [422] is a free environment for statistical computing and graphics. A large number of packages implementing machine learning algorithms are available at the CRAN repository. Additionally, the Bioconductor project [216] provides more than 600 R packages for analyzing biological data and includes many machine learning and statistical methods. Also, R scripts implementing different machine learning algorithms can be found on the Internet.

Matlab[4] is a general numerical computing environment developed by the company MathWorks. Its statistics toolbox includes machine learning algorithms for supervised and unsupervised learning. Other toolboxes are available for working with neural networks, optimization techniques, etc. Additionally, scientists and researches all over the world share code implementing machine learning algorithms on the Internet.

GNU Octave[5] is a free high-level interpreted language for numerical computations. The language is similar to Matlab so that most programs are easily portable.

[1] Available at: http://www.cs.waikato.ac.nz/ml/weka/
[2] Available at: http://www.keel.es/
[3] Available at: http://www.r-project.org/
[4] Available at: http://www.mathworks.com/
[5] Available at: http://www.gnu.org/software/octave/

Chapter 3

Probabilistic graphical models

3.1 Introduction

Uncertainty is a key feature in many complex real-world problems. Probability theory provides a framework for modeling and reasoning with uncertainty. The problem domain is encoded in terms of a set of random variables, and the aim is to model the joint probability distribution over the set of random variables. Then, probability theory can be used to reason about the values of the variables. However, complex domains containing many variables yield high-dimensional joint probability distributions, and reasoning with it becomes intractable.

Probabilistic graphical models (PGMs) [300, 402] combine probability theory and graph theory into a single framework that is able to manage complex real-world domains. A PGM has two components: a graphical component and a probabilistic component. The graphical component is a graph where the nodes represent the variables in the problem domain and the edges show the conditional (in)dependence relationships between the variables in the PGM. The probabilistic component models these dependence relationships using (conditional) probability distributions. Summing up these two components, a PGM encodes a factorization of the joint probability distribution over the set of variables. Therefore, PGMs compactly represent the problem domain and can perform any kind of reasoning (causal, diagnostic, abductive, bidirectional, etc.) efficiently because of the local computations allowed by the probability factorization.

There are two families of PGMs: Bayesian networks and Markov networks. Bayesian networks use a directed acyclic graph in the graphical component of the PGM. On the other hand, Markov networks use an undirected graph in the graphical component of the PGM. Bayesian networks are the main focus of this work, as it is the most frequently used model for reasoning with uncertainty, and they have been successfully applied across a large number of problems from very different domains [418].

Chapter outline

Section 3.2 defines notation and useful concepts. Section 3.3 introduces Bayesian networks and explains how to learn them from data and use them for inference tasks. Section 3.4 deals with Bayesian network classifiers, a class of Bayesian networks for solving supervised learning problems. Finally, Section 3.5 briefly discusses finite mixture models, a class of Bayesian networks for solving unsupervised learning problems.

3.2 Notation and definitions

The following conventions and definitions are used throughout the dissertation:

Random variables are denoted using uppercase letters, e.g., X or Y.

Values of a random variable are denoted using lowercase letters, e.g., x or y.

Multidimensional vectors are denoted using boldfaced letters, either uppercase for variables (e.g., $\mathbf{X}$) or lowercase for values (e.g., $\mathbf{x}$).

A random variable X is said to be discrete if it may take a countable number of distinct values.

A random variable X is said to be continuous if it can take values in the real domain $\mathbb{R}$ or a subinterval $[a, b] \in \mathbb{R}$.

The domain of a random variable X, i.e., the set of values it can take, is denoted Ω_X.

The cardinality of a set Ω, i.e., the number of elements contained in it, is denoted $Card(\Omega)$.

A discrete probability distribution function for a discrete random variable X is denoted $p_X(x; \boldsymbol{\theta}_X), x \in \Omega_X$, with $\sum_{x \in \Omega_X} p_X(x; \boldsymbol{\theta}_X) = 1$. Parameters $\boldsymbol{\theta}_X$ will be dropped when they are clear from the context.

A continuous probability density function for a continuous random variable X is denoted $f_X(x; \boldsymbol{\theta}_X), x \in \Omega_X$, where $\boldsymbol{\theta}_X$ are the parameters of the probability density function. A continuous probability density function is continuous in Ω_X, non-negative and integrates to one, i.e., $f_X(x; \boldsymbol{\theta}_X) \geq 0, x \in \Omega_X$, and $\int_{\Omega_X} f_X(x; \boldsymbol{\theta}_X) dx = 1$. Parameters $\boldsymbol{\theta}_X$ will be dropped when they are clear from the context.

A general probability distribution function for a random variable X being either discrete or continuous is denoted as $\rho_X(x; \boldsymbol{\theta}_X), x \in \Omega_X$, where $\boldsymbol{\theta}_X$ are the parameters of the probability distribution. Parameters $\boldsymbol{\theta}_X$ will be dropped when they are clear from the context.

A graph $\mathcal{G}$ is a pair $(\mathbf{X}, \mathbf{E})$, where $\mathbf{X}$ is the set of nodes and $\mathbf{E}$ is the set of edges connecting the nodes in $\mathbf{X}$.

A directed acyclic graph (DAG) is a graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ that contains only directed edges, called arcs $\mathbf{A} = \{(X_i, X_j) \mid X_i, X_j \in \mathbf{X}\}$, where the ordered pair (X_i, X_j) encodes an arc from X_i to X_j. Additionally, a DAG cannot have cycles, i.e., there is no directed path $(X_i, X_j) \dots (X_m, X_k)$ where $X_i = X_k$.

Given a DAG $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, the set of parents of a variable X_i contains the variables in $\mathbf{X} \setminus \{X_i\}$ with arcs towards X_i: $\mathbf{Pa}(X_i) = \{X_j \mid j \neq i, (X_j, X_i) \in \mathbf{A}\}$.

3.3 Bayesian networks

A Bayesian network (BN) is defined as a pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$, where:

$\mathcal{G} = (\mathbf{X}, \mathbf{A})$ is the graphical component of the model, i.e., a DAG where the nodes ($\mathbf{X}$) represent the variables $\mathbf{X} = (X_1, \dots, X_n)$ in the problem domain, and the arcs ($\mathbf{A}$) encode the probabilistic conditional (in)dependence relationships between the variables.

$\boldsymbol{\theta}$ is the probabilistic component of the model. $\boldsymbol{\theta}$ includes the parameters of the (conditional) probability functions of each variable $X_i, i = 1, \dots, n$ given the values of its parents $\mathbf{Pa}(X_i) = \mathbf{pa}(x_i)$ in $\mathcal{G}$. Therefore, $\boldsymbol{\theta} = \left(\boldsymbol{\theta}_{X_1|\mathbf{Pa}(X_1)}, \dots, \boldsymbol{\theta}_{X_n|\mathbf{Pa}(X_n)}\right)$.

A BN encodes a factorization of the joint probability distribution (JPD) over all the variables in $\mathbf{X}$ according to the structure in $\mathcal{G}$:

$$\rho_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n} \rho_{X_i|\mathbf{Pa}(X_i)}\left(x_i|\mathbf{pa}(x_i); \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}\right). \tag{3.1}$$

Bayesian networks are both interpretable and efficient. The graphical component of a Bayesian network is a compact representation of the problem domain. Additionally, the factorization of the JPD reduces the computational workload of using high-dimensional probability distributions.

Figure 3.1 shows an example of a discrete BN with five variables and four arcs. Variables X_1 and X_2 have no parents, whereas $\mathbf{Pa}(X_3) = \{X_1, X_2\}$, $\mathbf{Pa}(X_4) = \{X_2\}$ and $\mathbf{Pa}(X_5) = \{X_3\}$. The probabilistic component of the BN includes parameters $\boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}$ of the discrete conditional probability distributions for each variable X_i given all the combinations of the values of its parents $\mathbf{Pa}(X_i)$. Variables X_1 and X_2 are modeled using marginal discrete probability distributions because they have no parents. The BN in Figure 3.1 encodes the following factorization of the JPD:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{X_1}(x_1)p_{X_2}(x_2)p_{X_3|X_1X_2}(x_3|x_1, x_2)p_{X_4|X_2}(x_4|x_2)p_{X_5|X_3}(x_5|x_3).$$

3.3.1 Parameterization

The probabilistic component $\boldsymbol{\theta}$ determines the kind of probability distributions used in a Bayesian network $\mathcal{B}$. Different parameterizations of Bayesian networks have been proposed

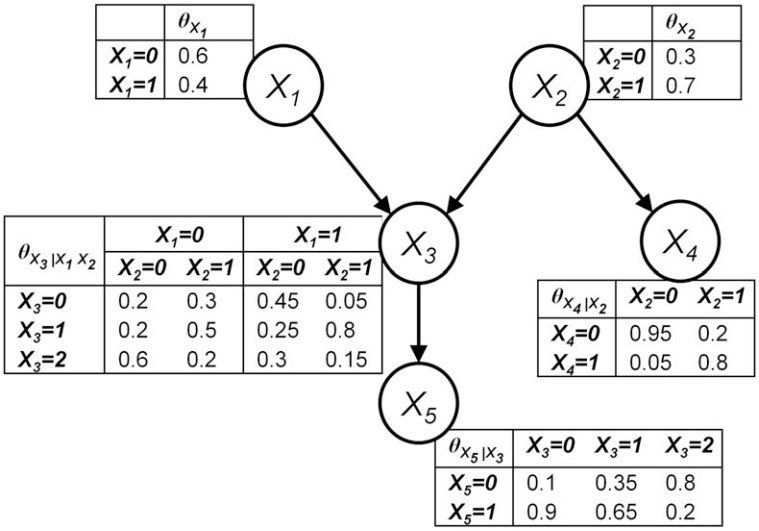


Figure 3.1: An example of a Bayesian network with five discrete variables $\mathbf{X} = (X_1, X_2, X_3, X_4, X_5)$ and four arcs $\mathbf{A} = \{(X_1, X_3), (X_2, X_3), (X_2, X_4), (X_3, X_5)\}$. The tables show the probabilistic component of the BN containing the parameters of the discrete conditional probability distributions $p_{X_i|\mathbf{Pa}(X_i)}\left(x_i|\mathbf{pa}(x_i); \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}\right)$.

depending on the nature of the domain of the random variables they include, i.e., discrete, continuous or hybrid domains including both discrete and continuous variables.

3.3.1.1 Discrete Bayesian networks

Discrete Bayesian networks having all its variables $\mathbf{X}$ defined in discrete domains are the most frequently used Bayesian network models. In this parameterization, the statistical relationship between a variable X_i and its parents $\mathbf{Pa}(X_i)$ is encoded using discrete (conditional) probability distributions. In a discrete BN, one discrete probability distribution of variable X_i is defined for each combination of the values of its parents $\mathbf{Pa}(X_i) = \mathbf{pa}(x_i)$. Usually, a tabular representation known as conditional probability table (CPT) is used to store the parameters of the discrete (conditional) probability distributions of X_i for all the combinations of the values of its parents (see Figure 3.1). Let $r_i = Card(\Omega_{X_i})$ be the number of possible values of variable X_i and $q_i = Card(\Omega_{\mathbf{Pa}(X_i)})$ be the number of possible combinations of the values of the parents. Then, a CPT contains the parameters $\theta_{ijk} = p_{X_i|\mathbf{Pa}(X_i)}\left(x_i^{(j)}|\mathbf{pa}(x_i)^{(k)}\right)$, where $x_i^{(j)}$ is the jth value of variable X_i and $\mathbf{pa}(x_i)^{(k)}$ is the kth combination of values of the parents of X_i. The number of parameters to be estimated in a CPT is

$$(r_i - 1)q_i = (r_i - 1) \prod_{m=1}^{Card(\mathbf{Pa}(X_i))} r_m.$$

Therefore, the total number of parameters in a discrete Bayesian network is

$$\sum_{i=1}^{n}(r_i - 1) \prod_{m=1}^{Card(\mathbf{Pa}(X_i))} r_m.$$

3.3.1.2 Continuous Bayesian networks

Continuous Bayesian networks have all their variables $\mathbf{X}$ defined in continuous domains. Although these continuous domains appear frequently in real-world problems, most of the Bayesian network learning and inference algorithms only work with discrete variables. Three main approaches can be distinguished for dealing with continuous variables in Bayesian networks [203]: discretization approaches, Gaussian Bayesian networks and other methods.

Discretization-based approaches

Discretization-based approaches transform the continuous variables into discrete variables, so that classical algorithms for Bayesian network learning and inference can be applied. The standard approach consists of discretizing the continuous variables into discrete variables prior to and independently from the learning or inference algorithm. Once the variables have been discretized, one proceeds as in a regular discrete Bayesian network. Discretization is one of the preprocessing techniques most broadly used in data mining, and numerous proposals of discretization procedures can be found in the literature, e.g., see [208] for a recent review and taxonomy. Some works have studied the effect of discretization procedures in Bayesian networks [148, 190, 203, 265, 266, 526]. More complex methods can also be found in the literature. For instance, a number of works have considered the discretization of the continuous variables inside the learning algorithm, e.g., [197, 374, 471].

Gaussian Bayesian networks

Gaussian Bayesian networks are BNs where all the variables are continuous and their conditional probability distributions are linear Gaussian distributions [454, 516]. These models are the most commonly used continuous Bayesian networks because the Gaussian assumption provides interesting properties: exact inference in closed form [321, 322], fast approximate inference algorithms [151], tractable learning algorithms [212], etc. Given a multivariate Gaussian (MG) density, it is possible to obtain an equivalent Gaussian BN and vice versa [454].

Given a vector of n continuous variables $\mathbf{X} = (X_1, \ldots, X_n)$, the conditional density for variable X_i with parents $\mathbf{Pa}(X_i)$ is the linear Gaussian distribution $f_{X_i|\mathbf{Pa}(X_i)}(x_i|\mathbf{pa}(x_i)) = \mathcal{N}\left(\beta_{0X_i|\mathbf{Pa}(X_i)} + \boldsymbol{\beta}^T_{X_i|\mathbf{Pa}(X_i)}\mathbf{pa}(x_i), \sigma^2_{X_i|\mathbf{Pa}(X_i)}\right)$, where $\beta_{0X_i|\mathbf{Pa}(X_i)}$ and $\boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)}$ are the linear regression coefficients of X_i over $\mathbf{Pa}(X_i)$, and $\sigma^2_{X_i|\mathbf{Pa}(X_i)}$ is the conditional variance of X_i

given $\mathbf{Pa}(X_i)$. These parameters are computed as

$$\begin{aligned}
\beta_{0X_i|\mathbf{Pa}(X_i)} &= \mu_{X_i} - \boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)}\boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)}\boldsymbol{\mu}_{\mathbf{Pa}(X_i)},\\
\boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)} &= \boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)}\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)X_i},\\
\sigma^2_{X_i|\mathbf{Pa}(X_i)} &= \Sigma_{X_i} - \boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)}\boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)}\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)X_i},
\end{aligned}$$

where μ_{X_i} and $\boldsymbol{\mu}_{\mathbf{Pa}(X_i)}$ are, respectively, the mean values of variables X_i and $\mathbf{Pa}(X_i)$, $\boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)}$ is the vector with the covariances of each variable X_i and its parents $\mathbf{Pa}(X_i)$, and $\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)}$ is the covariance matrix between parents $\mathbf{Pa}(X_i)$ of X_i. Therefore, only the means and covariances of each variable X_i and its parents $\mathbf{Pa}(X_i)$ and the covariances between these parents have to be estimated, which is performed by the corresponding sample means and variances. The joint probability density over variables $\mathbf{X}$ (Equation (3.1)) of a Gaussian BN is given by the product

$$f_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n} f_{X_i|\mathbf{Pa}(X_i)}\left(x_i|\mathbf{pa}(x_i); \beta_{0X_i|\mathbf{Pa}(X_i)}, \boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)}, \sigma^2_{X_i|\mathbf{Pa}(X_i)}\right). \tag{3.2}$$

The total number of parameters to be estimated in a Gaussian BN is given by

$$2n + \sum_{i=1}^{n}\left(Card(\mathbf{Pa}(X_i)) + \frac{1}{2}Card(\mathbf{Pa}(X_i))(Card(\mathbf{Pa}(X_i)) - 1)\right).$$

Other methods

More complex methods have been proposed for learning continuous Bayesian networks without the need of discretization and avoiding the assumption of Gaussianity. Non-parametric Bayesian networks based on kernel density estimation (KDE) were studied, e.g., in [26, 259, 279, 409]. Gurwicz and Lerner [230] used spline smoothing to alleviate the computational burden of kernel-based Bayesian classifiers. Non-parametric regression models were used for modeling the conditional probability distributions in continuous BNs in [272, 273]. Monti and Cooper [373] proposed using neural networks for conditional density estimation in continuous BNs.

3.3.1.3 Hybrid Bayesian networks

Hybrid domains including both discrete and continuous variables occur frequently in science. Hybrid Bayesian networks are useful for modeling these problems defined in hybrid domains. However, this class of BNs pose a number of challenges regarding the representation of conditional probability distributions, inference, learning from data, etc. Several approaches using these networks have been proposed in the literature.

Conditional linear Gaussian networks

Conditional linear Gaussian (CLG) networks [324, 390] are the most widely used Bayesian network models for hybrid domains. The continuous variables are modeled using conditional linear Gaussian distributions in a CLG network. Also, CLG networks impose some restrictions on the network structure, i.e., discrete variables cannot have continuous parents. CLG networks model the joint probability over the variables in the hybrid domain as a mixture of MG distributions with one component for each combination of values of the discrete variables. CLG networks have been studied at length because they provide tractable inference [321, 322] and learning [52]. Also, they provide a framework including a large number of related models [433], e.g., mixtures of Gaussian densities, principal component analysis, hidden Markov models, etc.

In a CLG network with n_c continuous variables and n_d discrete variables, the vector of $n = n_c + n_d$ variables $\mathbf{X} = (X_1, \ldots, X_n)$ is divided into a set of discrete variables $\mathbf{X}^{(d)}$ and a set of continuous variables $\mathbf{X}^{(c)}$. The discrete variables $X_i \in \mathbf{X}^{(d)}$ can only have discrete parents $\mathbf{Pa}(X_i)^{(d)} \subseteq \mathbf{X}^{(d)} \setminus \{X_i\}$, and the discrete conditional probability distributions are categorical distributions $p_{X_i|\mathbf{Pa}(X_i)^{(d)}}\left(x_i|\mathbf{pa}(x_i)^{(d)}; \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)^{(d)}}\right)$ that can be represented using a CPT (see Section 3.3.1.1). The continuous variables $X_i \in \mathbf{X}^{(c)}$ can have both continuous $\mathbf{Pa}(X_i)^{(c)} \subseteq \mathbf{X}^{(c)} \setminus \{X_i\}$ and discrete parents $\mathbf{Pa}(X_i)^{(d)} \subseteq \mathbf{X}^{(d)}$. Then, the conditional probability distribution for a continuous variable X_i is defined as a CLG distribution

$$f_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{Pa}(X_i)^{(d)}}\left(x_i|\mathbf{pa}(x_i)^{(c)}, \mathbf{pa}(x_i)^{(d)}; \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)^{(d)},\mathbf{Pa}(X_i)^{(c)}}\right) = \mathcal{N}\left(\beta_{0X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}} + \boldsymbol{\beta}^T_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}\mathbf{pa}(x_i)^{(c)}, \sigma^2_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}\right),$$

with $\boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)^{(d)}\mathbf{Pa}(X_i)^{(c)}} = \left\{\beta_{0X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}, \boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}, \sigma^2_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}\right\}$, $\mathbf{pa}(x_i)^{(d)} \in \Omega_{\mathbf{Pa}(X_i)^{(d)}}$. Parameters $\beta_{0X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}$ and $\boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}$ are the linear regression coefficients of X_i over $\mathbf{Pa}(X_i)^{(c)}$ for each combination of the values of the discrete parents $\mathbf{pa}(x_i)^{(d)}$, and $\sigma^2_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}}$ is the conditional variance of X_i given $\mathbf{Pa}(X_i)^{(c)}$ for $\mathbf{Pa}(X_i)^{(d)} = \mathbf{pa}(x_i)^{(d)}$. These parameters are computed as

$$\begin{aligned}
\beta_{0X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}} &= \mu_{X_i|\mathbf{pa}(x_i)^{(d)}} - \boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}\boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}\boldsymbol{\mu}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}},\\
\boldsymbol{\beta}_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}} &= \boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)^{(c)}X_i|\mathbf{pa}(x_i)^{(d)}},\\
\sigma^2_{X_i|\mathbf{Pa}(X_i)^{(c)}\mathbf{pa}(x_i)^{(d)}} &= \Sigma_{X_i|\mathbf{pa}(x_i)^{(d)}}\\
&\quad - \boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}\boldsymbol{\Sigma}^{-1}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)^{(c)}X_i|\mathbf{pa}(x_i)^{(d)}},
\end{aligned}$$

where $\mu_{X_i|\mathbf{pa}(x_i)^{(d)}}$ and $\boldsymbol{\mu}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}$ are, respectively, the mean values of variables X_i and $\mathbf{Pa}(X_i)^{(c)}$ given $\mathbf{Pa}(X_i)^{(d)} = \mathbf{pa}(x_i)^{(d)}$, $\boldsymbol{\Sigma}_{X_i\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}$ is the vector with the covariances of each variable X_i and its continuous parents $\mathbf{Pa}(X_i)^{(c)}$ given $\mathbf{Pa}(X_i)^{(d)} = \mathbf{pa}(x_i)^{(d)}$, and $\boldsymbol{\Sigma}_{\mathbf{Pa}(X_i)^{(c)}|\mathbf{pa}(x_i)^{(d)}}$ is the covariance matrix between the continuous parents $\mathbf{Pa}(X_i)^{(c)}$ of X_i given $\mathbf{Pa}(X_i)^{(d)} = \mathbf{pa}(x_i)^{(d)}$. The joint probability density over variables $\mathbf{X}$ (Equation (3.1))

of a CLG network is given by the product

$$\rho_{\mathbf{X}}(\mathbf{x}) = \prod_{X_i \in \mathbf{X}^{(d)}} p_{X_i|\mathbf{Pa}(X_i)^{(d)}} \left(x_i|\mathbf{pa}(x_i)^{(d)}; \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)^{(d)}}\right) \times \prod_{X_i \in \mathbf{X}^{(c)}} f_{X_i|\mathbf{Pa}(X_i)^{(c)},\mathbf{Pa}(X_i)^{(d)}} \left(x_i|\mathbf{pa}(x_i)^{(c)}, \mathbf{pa}(x_i)^{(d)}; \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)^{(d)},\mathbf{Pa}(X_i)^{(c)}}\right).$$

The total number of parameters to be estimated in a CLG network is given by

$$2n_c + \sum_{X_i \in \mathbf{X}^{(d)}} \left[(Card(\Omega_{X_i}) - 1) \prod_{X_j \in \mathbf{Pa}(X_i)^{(d)}} Card(\Omega_{X_j}) \right] + \sum_{X_i \in \mathbf{X}^{(c)}} \left[Card\left(\Omega_{\mathbf{Pa}(X_i)^{(d)}}\right) \left(Card\left(\mathbf{Pa}(X_i)^{(c)}\right) + \frac{1}{2} Card\left(\mathbf{Pa}(X_i)^{(c)}\right) \left(Card\left(\mathbf{Pa}(X_i)^{(c)}\right) - 1 \right) \right) \right].$$

The parametric assumption of Gaussian distributions in CLG networks might not hold in real domains. Also, the constraints on the network structure might limit their applicability in some problems. Therefore, other methods have been studied to avoid the structural constraints of CLG networks, e.g., see [274, 312, 381, 382].

Mixtures of truncated basis functions

Recently, a family of related models including mixtures of truncated basis functions (MoTBFs) [315], mixtures of truncated exponentials (MTEs) [376] and mixtures of polynomials (MoPs) [458] have been proposed for probability estimation in hybrid Bayesian networks. Given a continuous random variable X with a probability density function $f_X(x)$, the goal is to find an approximation of $f_X(x)$ over a closed domain $\Omega_X \subset \mathbb{R}$. MoTBFs approximate $f_X(x)$ as a (piecewise) linear combination of truncated basis functions. MTEs and MoPs are particular scenarios of MoTBFs when using exponential or polynomial functions, respectively, as basis functions. MoTBFs, MTEs and MoPs are closed under multiplication, addition and integration. Therefore, exact probabilistic inference can be performed using algorithms based on the Shenoy-Shafer architecture [457]. These methods provide a more flexible approach than discretization for dealing with hybrid Bayesian networks in domains with both continuous and discrete variables. MoTBFs, MTEs and MoPs can model both continuous and discrete variables in a unified fashion. Therefore, unlike CLG networks, no constraints are set on the structure of the networks. Also, they provide a non-parametric density estimation technique and can model domains where the Gaussian assumption of CLG networks does not hold.

First, different methods have been proposed for approximating with MTEs. Cobb et al. [93] provided MTE approximations of seven standard parametric probability density functions. Rumí et al. [435] proposed an iterative least squares algorithm for learning MTE approximations of one-dimensional and conditional probability densities from data. This approach used exponential regression and empirical histograms for density estimation. Romero et al. [432] enhanced the algorithm by applying a Gaussian kernel smoothing of the probability densities obtained with the empirical histograms. Langseth et al. [314] provided a ML

estimation approach for MTE approximations. Second, polynomial approximation and interpolation techniques have been used to obtain MoPs. Thus, Shenoy and West [458] found MoP approximations of known parametric densities by computing their Taylor series expansions (TSE). Later, Lagrange interpolating polynomials (LIPs) were used to obtain MoPs [456]. Third, regarding MoTBFs, Langseth et al. [315] proposed a method for finding approximations of one-dimensional and conditional densities by minimizing the Kullback-Leibler divergence (KL) from the MoTBF to the true distribution. Recently, the KL approach was combined with kernel density estimation techniques to approximate MoTBFs from data [316]. A more detailed study of these models will be given in Chapter 8.

3.3.2 Learning

Bayesian network learning is a two-step procedure [68, 108, 247]: structural search and parameter fitting. These two steps correspond to the specification of the graphical component $\mathcal{G}$ and the parameters in the probabilistic component $\boldsymbol{\theta}$ of $\mathcal{B}$, respectively. Both the structure and the parameters of the probability distributions can be obtained in two ways. First, one can elicit both the structure [192] and the parameters [209] of $\mathcal{B}$ using expert knowledge. Second, if a dataset with observations of the random variables is available, one can learn a BN from the data. The combination of both approaches has also been explored [248, 362]. Eliciting BNs from expert knowledge is out of the scope of this dissertation. Therefore, the remaining of the section focuses on methods for learning BNs from data.

3.3.2.1 Structure learning

Learning Bayesian network structures has been proven to be NP-hard [85, 86]. There are two main methods for learning the structure $\mathcal{G}$ of a Bayesian network: constraint-based methods and score+search methods.

Constraint-based methods rely on performing statistical tests to find conditional independence relationships between groups of variables in the network. Then, an undirected independence graph is built, and edge orientation discovers a Bayesian network structure which encodes those conditional independence relationships. A large number of constraint-based algorithms for learning the structure of a BN can be found in the literature, e.g., see Chapter 5 in [466].

Score+search methods approach the structure learning problem as an optimization problem. They use a heuristic search algorithm to explore the space of network structures, and a score function to evaluate the candidate network structures and guide the search procedure.

The rest of the section focuses on score+search approaches. Using a score+search approach to structure learning requires the specification of two elements: the score function which evaluates the candidate structures and the search procedure which explores the space of network structures.

Score functions

Given a dataset of observations $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_j = (x_{j1}, \ldots, x_{jn}),\ j = 1, \ldots, N$, the score function evaluates how good a candidate network structure $\mathcal{G}$ models the dataset. Several score functions have been proposed in the literature:

The *log-likelihood* of dataset $\mathcal{D}_{\mathbf{X}}$ given model $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ is the most natural choice for a scoring function. The log-likelihood measures the probability that dataset $\mathcal{D}_{\mathbf{X}}$ has been sampled from model $\mathcal{B}$:

$$\ell\left(\mathcal{D}_{\mathbf{X}}|\mathcal{B}\right) = \sum_{j=1}^{N} \ln \rho_{\mathbf{X}}(\mathbf{x}_j). \tag{3.3}$$

The log-likelihood monotonically increases with the complexity of the network structure. Therefore, using the log-likelihood $\ell\left(\mathcal{D}_{\mathbf{X}}|\mathcal{B}\right)$ as a score for structure learning tends to select complex models and overfit the data.

The *Bayesian information criterion* (BIC) score [443] avoids the overfitting problem by penalizing the log-likelihood according to the size of the dataset and the number of parameters in the model:

$$BIC\left(\mathcal{D}_{\mathbf{X}}, \mathcal{B}\right) = \sum_{j=1}^{N} \ln \rho_{\mathbf{X}}(\mathbf{x}_j) - \frac{1}{2} Dim(\mathcal{B}) \ln N, \tag{3.4}$$

where $Dim(\mathcal{B})$ refers to the dimension of the model, i.e., the number of independent parameters that have to be estimated from $\mathcal{D}_{\mathbf{X}}$. This criterion is a special case of the minimum description length criterion. The BIC score is asymptotically optimal, i.e., it is guaranteed to retrieve the true model when enough data are available.

For a discrete BN, the *K2 scoring function* [97] measures the joint probability of the network structure $\mathcal{G}$ and the dataset $\mathcal{D}_{\mathbf{X}}$ as

$$p(\mathcal{D}_{\mathbf{X}}, \mathcal{B}) = p(\mathcal{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{3.5}$$

where $p(\mathcal{G})$ is the prior probability of the network structure $\mathcal{G}$, $r_i = Card(\Omega_{X_i})$ is the number of distinct values of X_i, $q_i = Card(\Omega_{\mathbf{Pa}(X_i)})$ is the number of possible configurations of $\mathbf{Pa}(X_i)$, N_{ij} is the number of instances in dataset $\mathcal{D}_{\mathbf{X}}$ where the set of parents $\mathbf{Pa}(X_i)$ takes it jth configuration, and N_{ijk} is the number of instances where variable X_i takes the kth value and $\mathbf{Pa}(X_i)$ takes it jth configuration ($N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$).

Search procedures

The search procedure explores the space of network structures and tries to find a high-scoring network structure. The search can be performed in three different spaces: DAGs, partial DAGs or variable orderings. We focus on search procedures in the space of DAGs. The number of possible DAGs increases exponentially with the number of variables and an exhaustive evaluation becomes infeasible. Two search procedures are reviewed here:

The *K2 search algorithm* [97] implements a greedy search that adds one arc in each iteration of the procedure (see Algorithm 3.1). K2 needs the variables in $\mathbf{X}$ to be ordered (step 1). Leray and Francois [327] proposed using the maximum weight spanning tree algorithm to compute such ordering. First, each possible edge is given a weight that corresponds to the score variation when the variables are related. Then, Prim's algorithm uses those weights to build an undirected tree, and a root node is selected so that the edges can be oriented. Finally, a topological sorting method is applied to partially order the variables. Other approaches for finding an ordering of the variables can be found in [54, 115, 229, 267, 319].

After ordering the variables, the K2 algorithm incrementally builds the Bayesian network. At each iteration, one variable is selected according to the ordering of variables (step 4 of Algorithm 3.1). All the previous variables in the ordering are considered as candidate parents. The candidate variable that gives the highest score is selected as a parent, until the score decreases or the maximum number of parents allowed per variable is reached. The process is repeated until all variables have been considered.

Algorithm 3.1 (K2 search algorithm)
Inputs:

- $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$*: A dataset*
- u*: The maximum number of parents allowed for each variable*

Output: A DAG $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ *learned from* $\mathcal{D}_{\mathbf{X}}$

Steps:

1. $Order(\mathbf{X})$.
2. *Create an empty Bayesian network* $\mathcal{B} = (\mathcal{G} = (\mathbf{X}, \mathbf{A}), \boldsymbol{\theta}), \mathbf{A} = \emptyset$.
3. $Score_{\max} = Score(\mathcal{D}_{\mathbf{X}}, \mathcal{B})$.
4. *Repeat for each variable* X_i *following* $Order(\mathbf{X})$*:*
   (a) *If the number of parents of* X_i *is equal to* u*, go to next variable in* $Order(\mathbf{X})$ *(step 4).*
   (b) *Find* $X_j, j = 1, \ldots, i-1$ *that maximizes* $Score(\mathcal{D}, \mathcal{B}' = (\mathcal{G} = (\mathbf{X}, \mathbf{A}'), \boldsymbol{\theta}))$ *where* $\mathbf{A}' = \mathbf{A} \cup \{(X_j, X_i)\}$.

(c) If $Score(\mathcal{D}, \mathcal{B}') < Score_{\max}$*, go to next variable in* $Order(\mathbf{X})$ *(step 4). Else, set* $\mathcal{B} = \mathcal{B}'$*,* $Score_{\max} = Score(\mathcal{D}, \mathcal{B}')$ *and go to step 4.(a).*

The *greedy thick-thinning* (GTT) algorithm [112] implements a two-step procedure for discovering a Bayesian network structure (see Algorithm 3.2). Given an initial (empty) graph $\mathcal{G}$, it iteratively adds the arc which maximizes the increase in the score function (thicking step). When no further increment is possible by adding arcs, the algorithm iteratively removes arcs until no arc deletion yields a positive increase in the score function (thinning step). Then, the algorithm stops and the resulting Bayesian network structure is returned. The GTT algorithm has a number of advantages, e.g., unlike other methods [97, 198] it does not require an ordering of the variables. Also, it is simple, computationally efficient and avoids overfitting by removing arcs in the thinning step.

Algorithm 3.2 (Greedy thick thinning algorithm)
Inputs:

- $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$*: A dataset*
- $\mathcal{G} = (\mathbf{X}, \mathbf{A})$*: An initial graph*

Output: A DAG $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ *learned from* $\mathcal{D}_{\mathbf{X}}$

Steps:

1. *Thicking step: While the score function increases:*
   (a) *Find the arc* (X_i, X_j) *which maximizes the score function when included in* $\mathcal{G}' = (\mathbf{X}, \mathbf{A}')$ *with* $\mathbf{A}' = \mathbf{A} \cup \{(X_i, X_j)\}$.
   (b) *Set* $\mathcal{G} \leftarrow \mathcal{G}'$.
2. *Thinning step: While the K2 score function increases:*
   (a) *Find the arc* (X_i, X_j) *which maximizes the score function when deleted in* $\mathcal{G}' = (\mathbf{X}, \mathbf{A}')$ *with* $\mathbf{A}' = \mathbf{A} \setminus \{(X_i, X_j)\}$.
   (b) *Set* $\mathcal{G} \leftarrow \mathcal{G}'$.
3. *Return* $\mathcal{G}$.

3.3.2.2 Parameter fitting

Once the network structure $\mathcal{G}$ has been found, parameters $\boldsymbol{\theta}$ have to be estimated from dataset $\mathcal{D}_{\mathbf{X}}$. There are two main approaches for estimating parameters: maximum likelihood and Bayesian estimation.

The maximum likelihood (ML) approach looks for the values of the parameters $\widehat{\boldsymbol{\theta}}$ from the set of possible parameter values $\Omega_{\boldsymbol{\Theta}}$ which maximize the log-likelihood of the dataset given the model (Equation (3.3)):

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \Omega_{\boldsymbol{\Theta}}} \ell\left(\mathcal{D}_{\mathbf{X}} | \mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})\right).$$

The Bayesian estimation approach is able to include prior knowledge into the parameter estimation problem. Parameters $\boldsymbol{\theta}$ are modeled with a random variable $\boldsymbol{\Theta}$, and a probability distribution $f_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$ is used to encode our prior knowledge about the value of $\boldsymbol{\theta}$. Then, the joint probability of the parameters and the dataset $\rho_{\mathcal{D}_{\mathbf{X}},\boldsymbol{\Theta}}(\mathcal{D}_{\mathbf{X}},\boldsymbol{\theta})$ is specified. The aim is to find the value of the parameters $\widehat{\boldsymbol{\theta}}$ which maximizes the posterior distribution of $\boldsymbol{\Theta}$ given the dataset $\mathcal{D}_{\mathbf{X}}$:

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\Theta}}} f_{\boldsymbol{\Theta}|\mathcal{D}_{\mathbf{X}}}(\boldsymbol{\theta}|\mathcal{D}_{\mathbf{X}}).$$

3.3.3 Inference

Bayesian networks provide a powerful tool for modeling and reasoning in domains with uncertainty. Any kind of reasoning task can be performed using Bayesian networks: causal, diagnostic, abductive, bidirectional, etc. Therefore, any kind of probabilistic query can be answered using BNs.

The most common inference task is evidence propagation. In evidence propagation, we identify a subset of variables $\mathbf{X}_e \subset \mathbf{X}$ (called evidence variables) whose values $\mathbf{x}_e$ have been observed, and the goal is to reason about another subset of variables $\mathbf{X}_q \subseteq \mathbf{X} \setminus \mathbf{X}_e$ (called query variables) given the observed values of the evidence variables. Therefore, the goal is obtaining the conditional probability

$$\rho_{\mathbf{X}_q|\mathbf{X}_e}(\mathbf{x}_q|\mathbf{x}_e) = \frac{\rho_{\mathbf{X}_q,\mathbf{X}_e}(\mathbf{x}_q,\mathbf{x}_e)}{\rho_{\mathbf{X}_e}(\mathbf{x}_e)}.$$

Computing the conditional probability $\rho_{\mathbf{X}_q|\mathbf{X}_e}(\mathbf{x}_q|\mathbf{x}_e)$ involves the computation of the joint probability $\rho_{\mathbf{X}_q,\mathbf{X}_e}(\mathbf{x}_q,\mathbf{x}_e)$ and then marginalizing out all the variables in $\mathbf{X} \setminus \mathbf{X}_q$ [457].

Another interesting inference task is the most probable explanation (MPE) problem. The MPE problem deals with the problem of finding the values $\mathbf{x}_q^*$ of a subset of variables $\mathbf{X}_q = \mathbf{X} \setminus \mathbf{X}_e$ which yield the maximum posterior probability

$$\mathbf{x}_q^* = \arg\max_{\mathbf{x}_q\in\Omega_{\mathbf{X}_q}} \rho_{\mathbf{X}_q|\mathbf{X}_e}(\mathbf{x}_q|\mathbf{x}_e),$$

given the observed values $\mathbf{x}_e$ for the evidence variables $\mathbf{X}_e \subset \mathbf{X}\setminus\mathbf{X}_e$. An easier but related task is called the maximum a posteriori (MAP) problem. MAP inference deals with the problem of finding the values $\mathbf{x}_q^*$ of only a subset of the unobserved variables $\mathbf{X}_q \subset \mathbf{X}$ which yield the maximum posterior probability given the observed values $\mathbf{x}_e$ for the evidence variables $\mathbf{X}_e \in \mathbf{X} \setminus \mathbf{X}_q$.

Unfortunately, these approaches to inference are exponential in the worst case scenario, and it has been proved that both exact and approximate inference in Bayesian networks is $\mathcal{NP}$-hard [98, 107]. However, this worst case scenario is not common, and the inference task is tractable and efficient for many real-world applications. Exact and approximate methods have been proposed for performing inference in Bayesian networks [108, 228, 252]. Exact methods

are available when the Bayesian network structure is a polytree [222, 295, 401, 402]. When the BN structure is not a polytree, different approaches have been proposed for tackling the exact inference problem: global conditioning methods [479, 480], local conditioning methods [137], variable elimination methods [451–453], clique methods [277, 278, 323, 457, 465], polynomial compilation [110, 393], etc. Approximate inference methods have been proposed when the previous exact inference methods are not suitable, e.g., see [56, 75, 250, 253].

3.4 Bayesian network classifiers

Bayesian network classifiers (BNCs) [199] are a kind of Bayesian networks specially designed to solve supervised classification problems. In a BNC, the class labels in the supervised classification problem are modeled with a discrete random variable C (called class variable) with values $\Omega_C = \{1, \ldots, K\}$ representing the class labels. The prior probability of the class labels is modeled using a discrete categorical probability distribution over the class values $p_C(c) > 0, \sum_c p_C(c) = 1, c \in \Omega_C$. The features describing the object properties which are used for classification are called predictive variables and modeled using a vector of random variables $\mathbf{X} = (X_1, \ldots, X_n)$, where each variable X_i takes values $x_i \in \Omega_{X_i}, i = 1, \ldots, n$. Like in a regular BN, the BNC structure encodes the conditional (in)dependencies between the variables in $\mathbf{X}$.

When classifying a new instance $\mathbf{x} \in \Omega_{\mathbf{X}}$, a BNC yields a posterior probability $p_{C|\mathbf{X}}(c|\mathbf{x})$ for each class label $c \in \Omega_C$. Then, the maximum a posteriori decision rule is used so that $\mathbf{x}$ is assigned to the class c^* with maximum posterior probability

$$c^* = \arg\max_{c \in \Omega_C} p_{C|\mathbf{X}}(c|\mathbf{x}) = \arg\max_{c \in \Omega_C} p_C(c)\rho_{\mathbf{X}|C}(\mathbf{x}|c),$$

where $\rho_{\mathbf{X}|C}(\mathbf{x}|c)$ factorizes according to the BNC structure as in a regular BN (see Equation (3.1)).

3.4.1 Parameterization

Most of the proposals for BNCs, like in regular BNs, are focused on problems where all the predictive variables are discrete. Empirical studies show that discretizing continuous predictive variables can yield higher accuracies than assuming Gaussian densities like in CLG networks [148]. Some theoretical analysis of this phenomenon has been given in [265, 266, 526]. Also, it has been shown that the discretization algorithm used does not affect the ranking of the Bayesian classifiers [190]. Bayesian networks with continuous predictive variables modeled using CLG distributions and conforming to the CLG network structure (see Section 3.3.1.2) have been studied, e.g., in [408]. Extensions of the AODE classifier (see next Section 3.4.2) to problems with continuous and discrete predictive variables can be found in [189, 359]. Non-parametric KDE has been used for modeling the continuous variables in BNCs in [279, 409]. Bayesian classifiers using MTEs for modeling the continuous predictive variables have been studied in [188, 191].

3.4.2 Learning

3.4.2.1 Structure learning

Different BNCs can be identified according to the network structure encoding the conditional independence relationships between the predictive variables and the class variable.

The Bayesian classifier

A general multivariate Bayesian classifier [153] jointly models the predictive variables $\mathbf{X}$ using a multidimensional conditional probability distribution $\rho_{\mathbf{X}|C}(\mathbf{x}|c)$ for each class value $c \in \Omega_C$. Figure 3.2 shows the graphical structure of the Bayesian classifier. This Bayesian classifier does not exploit the conditional independence relationships between the predictive variables $\mathbf{X}$ given the class variable C. When the number of predictive variables n increases, the number of parameters to estimate also increases and using this classifier may become infeasible.

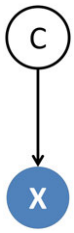


Figure 3.2: Bayesian network structure of a Bayesian classifier.

Duda et al. [153] showed that the decision surface of the Bayesian classifier is a hyperplane when the conditional joint probability distributions of the predictive variables $\rho_{\mathbf{X}|C}(\mathbf{x}|c)$ is modeled with a MG with class-independent covariance matrices, i.e., the covariance matrices are the same for each class value. On the other hand, the decision surfaces are hyperquadrics when the covariance matrices are different for each class value $c \in \Omega_C$.

The naive Bayes classifier

The naive Bayes (NB) classifier [372] is one of the most popular methods for supervised classification. It assumes the predictive variables to be conditionally independent given the class. Figure 3.3 shows the network structure for a NB classifier. Although the conditional independence assumption in NB classifiers seems very restrictive, it has shown competitive results against other BNCs and supervised classification approaches in a number of real-world problems [142, 199, 241, 310, 330]. The MAP decision rule for classifying an observation $\mathbf{x} = (x_1, \ldots, x_n)$ according to the NB classifier structure is given by

$$c^* = \arg\max_{c \in \Omega_C} p_{C|\mathbf{X}}(c|\mathbf{x}) = \arg\max_{c \in \Omega_C} p_C(c) \prod_{i=1}^{n} \rho_{X_i|C}(x_i|c).$$

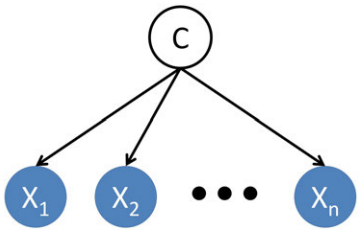


Figure 3.3: Bayesian network structure of a naive Bayes classifier.

The decision function of a NB classifier with two class values $\Omega_C = \{1, 2\}$ is given by the function $r(x)$ [153]:

$$\begin{aligned} p_{C|\mathbf{X}}(1|\mathbf{x}) &= p_{C|\mathbf{X}}(2|\mathbf{x}), \\ r(\mathbf{x}) = & p_{C|\mathbf{X}}(1|\mathbf{x}) - p_{C|\mathbf{X}}(2|\mathbf{x}) \\ = & p_C(1) \prod_{i=1}^{n} \rho_{X_i|C}(x_i|1) \\ & - p_C(2) \prod_{i=1}^{n} \rho_{X_i|C}(x_i|2). \end{aligned} \tag{3.6}$$

If the class has more than two values, a decision surface is considered for each pair of values, and the subregions defined by all the surfaces are labeled accordingly. The decision surfaces of a NB classifier with binary predictive variables are hyperplanes [372]. Later on, the same result was shown for general discrete variables [407]. Duda and Hart [152] found polynomial decision surfaces when the NB classifier has ordinal predictive variables. Duda et al. [153] showed that the decision surface is also a hyperplane when the conditional joint probability distributions of the predictive variables is modeled with a MG with class-independent covariance matrices, i.e., the covariance matrices are the same for each class value. On the other hand, the decision surfaces are hyperquadrics when the covariance matrices are different for each class value.

The tree-augmented naive Bayes classifier

The tree-augmented naive Bayes (TAN) classifier [199] tries to alleviate the conditional independence assumption in the NB classifier while avoiding the high complexity of the general Bayesian classifier. The TAN classifier allows relationships between pairs of predictive variables in the network. At the same time, it controls the complexity of the learning algorithm by imposing a tree structure over the subgraph structure of the predictive variables. In a TAN classifier, the class variable C is a parent of all the predictive variables $X_i, i = 1, \ldots, n$ as in the NB classifier. However, unlike the NB classifier, each predictive variable (except for the root node of the tree) has one additional predictive variable as a parent. Figure 3.4 shows a possible network structure for a TAN classifier, where the root node of the tree is X_1.

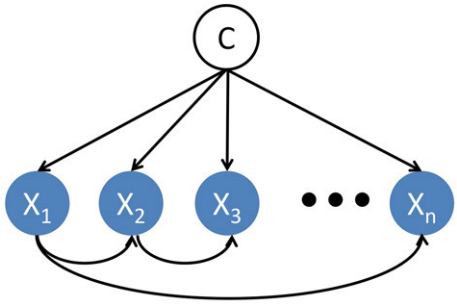


Figure 3.4: Bayesian network structure of a tree-augmented naive Bayes classifier.

Friedman et al. [199] proposed a procedure based on the Chow-Liu algorithm [88] for learning TAN classifiers. The conditional mutual information (MI) between two variables X_1 and X_2 given the class C is computed as a measure of the strength of their dependence:

$$MI(X_1, X_2|C) = \int_{\Omega_{X_1}} \int_{\Omega_{X_2}} \sum_{c \in \Omega_C} \rho_{X_1,X_2|C}(x_1, x_2|c) p_C(c) \log \frac{\rho_{X_1,X_2|C}(x_1, x_2|c)}{\rho_{X_1|C}(x_1|c)\rho_{X_2|C}(x_2|c)} \, dx_1 \, dx_2, \tag{3.7}$$

where $\rho_{X_1,X_2|C}(x_1, x_2|c)$ is the joint conditional probability distribution of (X_1, X_2) given the class label $C = c$ and $\rho_{X_1|C}(x_1|c)$ and $\rho_{X_2|C}(x_2|c)$ are the conditional probability distributions of X_1 given $C = c$ and X_2 given $C = c$, respectively. If X_1 or X_2 (or both) variables are discrete, then the integrals in Equation (3.7) are changed to sums over all the values in their domains, Ω_{X_1} and Ω_{X_2}, respectively. First, the algorithm builds a maximum weight spanning tree using the conditional MI values $MI(X_1, X_2|C)$ as edge weights. Then, a root variable $X_u, u \in \{1, \ldots, n\}$ is chosen, and the arcs are directed away from the root variable. Finally, the directed arcs of this tree are combined with a NB structure to obtain the final TAN classifier structure. Therefore, the root variable X_u has only one parent, i.e., the class variable, whereas the other predictive variables $X_v, v \neq u$, have two parents, i.e., the class variable and another predictive variable.

Once the TAN classifier has been built, a new instance $\mathbf{x}$ is classified by applying the MAP rule:

$$c^* = \arg\max_{c \in \Omega_C} p_C(c) \rho_{X_u|C}(x_u|c) \prod_{\substack{v=1,\ldots,n \\ v \neq u}} \rho_{X_v|X_w,C}(x_v|x_w, c),$$

where X_u is the predictive variable selected as the root of the tree during the TAN learning algorithm, and $X_w, w \neq v$, is the predictive variable which is a parent of variable $X_v, v \neq u$.

The aggregated one-dependence estimator classifier

The aggregated one-dependence estimator (AODE) classifier [513] learns n Bayesian classifiers with a specific TAN structure. The ith TAN classifier has variable X_i as a parent of all the other predictive variables $X_j, j \neq i$. Figure 3.5 shows a TAN classifier where variable X_1 is a parent of all the other variables $X_j, j = 2, \ldots, n$. When classifying a new instance, AODE computes the posterior probability of each class label as the mean of the posterior probabilities

yielded by each individual TAN classifier. Webb et al. [513] showed that the AODE classifier yielded low variances and competitive accuracies against state-of-the-art Bayesian classifiers.

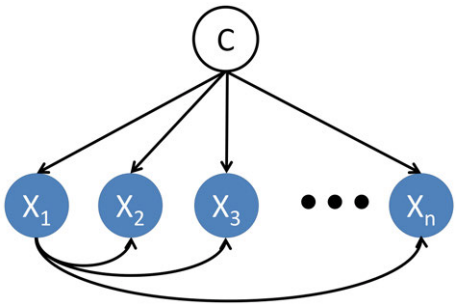


Figure 3.5: Bayesian network structure of the first tree-augmented naive Bayes classifier in an aggregated one-dependence estimator classifier where the predictive variable X_1 is a parent of all the other predictive variables $\{X_2, \ldots, X_n\}$.

Other Bayesian network classifier structures

Other proposals for Bayesian network classifiers can be found in the literature. The semi-naive Bayes [302, 394] considers statistical relationships between subsets of predictive features by joining them into a multidimensional variable. An extension to the TAN classifier is the k-dependence Bayesian classifier [299, 438, 538], that allows a predictive variable to have at most k other predictive variables as parents. Hierarchical naive Bayes classifiers including latent variables are studied in [311]. Naive Bayes and feature subset selection have been jointly studied in [309]. Full BNCs [477] do not exploit the conditional independence relationships between the predictive variables, but use decision trees to efficiently encode the big CPTs. Cheng and Greiner [84] studied unconstrained Bayesian classifiers that can adopt any BN structure. They also propose Bayesian multinet-based classifiers in [84].

3.4.2.2 Parameter fitting

Two main approaches can be identified for fitting the parameters in Bayesian classifiers: the generative approach and the discriminative approach. The generative approach is the classical approach to parameter fitting in BNCs. In the generative approach, the parameters of the Bayesian classifiers are found as in a regular BN, i.e, using ML estimates or Bayesian estimates (see Section 3.3.2.2). However, BNCs are designed to solve different tasks than BNs. BN learning is interested in accurately modeling the data and, thus, looks for parameters maximizing the log-likelihood of the dataset given the BN. In supervised classification, however, the aim is to maximize classification accuracy, not likelihood. Discriminative approaches to parameter fitting in BNCs look for parameters which maximize the classification accuracy or, alternatively, the conditional log-likelihood of the class variable given the predictive variables [77, 225, 226, 410, 478].

3.5 Finite mixture models

Finite mixture models (FMM) [367] are a formalism for modeling probability distributions as a weighted sum of K parameterized probability distributions:

$$\rho_{\mathbf{X}}(\mathbf{x}) = \sum_{i=1}^{K} \pi_i \rho_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{X}|i}), \tag{3.8}$$

where $\pi_i > 0, \sum_{i=1}^{K} \pi_i = 1$ are the weights of the components of the finite mixture model and $\rho_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{X}|i})$ are the multidimensional probability distributions for each component $i = 1, \ldots, K$ with parameters $\boldsymbol{\theta}_{\mathbf{X}|i}$. The FMM in Equation (3.8) can be seen as a special case of a Bayesian classifier (see Section 3.4.2.1) [395]. In that representation, the class variable C encodes the number of components (K) of the FMM, i.e., $\Omega_C = \{1, \ldots, K\}$. The probabilities of the class values $p_C(c)$ represent the weights π_i of each component in the FMM. The multidimensional probability distribution for each component $\rho_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{X}|c})$ is the conditional probability distribution of $\mathbf{X}$ given $C = c$. Using the BN notation and the BN structure of the Bayesian classifier in Figure 3.2, the joint probability of the predictive variables $\mathbf{X}$ is given by

$$\rho_{\mathbf{X}}(\mathbf{x}) = \sum_{c \in \Omega_C} p_C(c; \boldsymbol{\theta}_C) \rho_{\mathbf{X}|C}(\mathbf{x}|c; \boldsymbol{\theta}_{\mathbf{X}|c}). \tag{3.9}$$

FMMs are one of the most popular models for solving unsupervised learning tasks (clustering). Each component of the FMM models the features of a cluster of objects, and the probability distribution for the class variable encodes the prior probabilities of the clusters. FMMs are a method for probabilistic clustering, where the posterior distribution $p_{C|\mathbf{X}}(c|\mathbf{x})$ yields the probability that an object $\mathbf{x}$ belongs to cluster $c \in \Omega_C$. FMMs have a number of advantages over other clustering algorithms. First, FMMs provide a descriptive model of the clusters through the conditional probability distributions $\rho_{\mathbf{X}|C}(\mathbf{x}|c)$. Second, the objects are not assigned to a single cluster. FMMs assign a probability that an object belongs to each one of the clusters. Third, the problem of identifying a correct number of clusters can be formally addressed by fitting FMMs with different number of components and selecting the final model according to a penalized log-likelihood score, e.g., the BIC score (Equation (3.4)). Recently, infinite mixtures of distributions have been proposed as a way to automatically select the number of components in an FMM, e.g., see [61, 425, 482, 485].

3.5.1 Parameterization

Different FMMs have been proposed to model different kinds of data depending on their nature. FMMs usually differ in the kind of conditional probability distributions used for modeling each component $\rho_{\mathbf{X}|C}(\mathbf{x}|c)$. For discrete data, FMMs have been studied using different distributions: Bernoulli [285], Poisson [90], multinomial [388], Dirichlet compound multinomial [59], etc. FMMs with continuous predictive variables most commonly use MG densities [367, 395, 503], although other distributions have also been studied, e.g., Student's t

[404], Weibull [493], generalized Dirichlet [63], inverted Dirichlet [36]. In the special scenario where directional data are considered (see Chapter 5), mixture models have been proposed using von Mises [71, 356], von Mises-Fisher [30] or Kent [405] distributions.

3.5.2 Learning

3.5.2.1 Structure learning

The conditional independence relationships between the predictive variables $\mathbf{X}$ given the class variable C can be exploited in FMMs in a similar way to BNCs (see Section 3.4.2.1). Therefore, clustering problems have been solved using BNs with different structures: naive Bayes [80], tree-augmented naive Bayes [397], etc. More complex BN models have been used recently, e.g., Bayesian multinets [399, 415], estimation of distribution algorithms [400]. The structural expectation-maximization (SEM) algorithm [196] can be used to learn the BN structure when it is not set a priori. Some modifications and refinements to the SEM algorithm have been proposed, e.g., in [399].

3.5.2.2 Parameter fitting

In a clustering problem, the cluster labels of the objects are not available in the training dataset, and the aim is to group similar objects into clusters. Classical ML estimation techniques for fitting the parameters of a BNC or a regular BN cannot be applied in this setting, because the values of the hidden class variable C are missing in the training dataset. The expectation maximization (EM) algorithm [128] is usually applied to find the ML parameters of an FMM. We consider a Bayesian network $\mathcal{B}$ with a fixed network structure $\mathcal{G}$ and unknown parameter values $\boldsymbol{\theta}$. Since the network structure is fixed, the complete log-likelihood of a dataset $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ given the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\mathbf{X}|1}, \ldots, \boldsymbol{\theta}_{\mathbf{X}|K}\}$ is

$$\ell\left(\mathcal{D}_{\mathbf{X}}|\boldsymbol{\theta}\right) = \ln(\rho(\mathcal{D}_{\mathbf{X}}|\boldsymbol{\theta})) = \sum_{j=1}^{N} \ln\left(\sum_{c\in\Omega_C} p_C(c;\boldsymbol{\theta}_C)\rho_{\mathbf{X}|C}(\mathbf{x}_j|c;\boldsymbol{\theta}_{\mathbf{X}|c})\right).$$

The EM algorithm iterates between two steps (the expectation step and the maximization step) maximizing the log-likelihood of the dataset given the parameters, $\ell\left(\mathcal{D}_{\mathbf{X}}|\boldsymbol{\theta}\right)$. The expectation step at iteration q computes the posterior probability that the jth instance $\mathbf{x}_j$ belongs to class c given parameters $\boldsymbol{\theta}^{(q)}$ as

$$t_{jc}^{(q)} = p_{C|\mathbf{X}}\left(c|\mathbf{x}_j;\boldsymbol{\theta}^{(q)}\right) = \frac{p_C\left(c;\boldsymbol{\theta}_C^{(q)}\right)\rho_{\mathbf{X}|C}\left(\mathbf{x}_j|c;\boldsymbol{\theta}_{\mathbf{X}|c}^{(q)}\right)}{\sum_{k\in\Omega_C} p_C\left(k;\boldsymbol{\theta}_C^{(q)}\right)\rho_{\mathbf{X}|C}\left(\mathbf{x}_j|k;\boldsymbol{\theta}_{\mathbf{X}|k}^{(q)}\right)}.$$

Then, the expected log-likelihood of the complete data $\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right)$ is computed as the sum

of the expected log-likelihood of each instance $\mathcal{Q}_j\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right), j = 1,\ldots,N$:

$$\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right) = \sum_{j=1}^{N}\mathcal{Q}_j\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right) = \sum_{j=1}^{N}\sum_{c\in\Omega_C} t_{jc}^{(q)}\left[\ln p_C\left(c;\boldsymbol{\theta}_C^{(q)}\right) + \ln \rho_{\mathbf{X}|C}\left(\mathbf{x}_j|c;\boldsymbol{\theta}_{\mathbf{X}|c}^{(q)}\right)\right].$$

Finally, the maximization step computes the parameters $\boldsymbol{\theta}^{(q+1)}$ which maximize the expected log-likelihood of the complete data $\mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right)$:

$$\boldsymbol{\theta}^{(q+1)} = \arg\max_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\Theta}}} \mathcal{Q}\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(q)}\right).$$

The EM algorithm iteratively computes the expectation and maximization steps until it reaches a local maximum in the log-likelihood. Therefore, there is no guarantee that the algorithm will find the global maximum. Usually, several runs of the algorithm are performed with different initial values of parameters $\boldsymbol{\theta}^{(1)} = \left\{\boldsymbol{\theta}_C^{(1)}, \boldsymbol{\theta}_{\mathbf{X}|1}^{(1)},\ldots,\boldsymbol{\theta}_{\mathbf{X}|K}^{(1)}\right\}$ in the first expectation step of the algorithm, and the model with the highest log-likelihood in all the runs is returned.

3.6 Software

Computer tools and software solutions are available for working with PGMs. A brief description of the programs used in this work follows.

SMILE and GeNIe[1] are developed by the Decision Systems Laboratory of the University of Pittsburgh. SMILE (Structural Modeling, Inference, and Learning Engine) is a platform-independent library of C++ classes implementing graphical probabilistic and decision-theoretic models, such as Bayesian networks, influence diagrams, and structural equation models. Wrappers for Java and .NET are also available. GeNIe is the graphical interface to SMILE. Probabilistic graphical models can be created using the graphical editor and inference can be performed interactively using GeNIe.

The Bayes net toolbox for Matlab[2] was written by Kevin Murphy [383]. It supports discrete and continuous (Gaussian) nodes as well as deterministic and decision nodes. It implements several methods for learning the structure and the parameters of PGMs, and both exact and approximate algorithms for performing inference. The BNT structure learning package [327] implements a large number of structure learning algorithms for Murphy's toolbox. The probabilistic modeling toolkit[3] extends the functionality of the Bayes net toolbox to other probabilistic models, e.g., linear and logistic regression.

Implementing PGMs in R has been extensively covered in two recent books [262, 386]. Packages `gRbase` [133], `gRain` [260] and `gRim` [261] provide methods for building and

[1] Available at http://genie.sis.pitt.edu/
[2] Available at http://bnt.googlecode.com/
[3] Available at https://code.google.com/p/pmtk3/

making inference with discrete PGMs and other probabilistic models, e.g., log-linear models. Algorithms for learning Bayesian networks can be found in packages `deal` [53] and `bnlearn` [446]. Additionally, several packages are available for implementing and analyzing graphs, e.g., see [101, 215].

Chapter 4

Consensus analysis

4.1 Introduction

Supervised learning tasks are by far the most commonly studied problems in machine learning. Supervised learning deals with the problem of learning classifiers when the true class label for all the training instances is available. However, obtaining the true class label of all the training instances can be both difficult, time-consuming, expensive or even impossible. For example, in a medical setting, the diagnostic test to know if a patient suffers from a given disease may be unaffordably costly, have undesirable side effects or be too invasive for the patient. Also, different tests may yield different conclusions. In supervised learning settings, it is customary to manually label the training instances by asking an expert to provide class labels for all the instances in the training dataset. However, the class labels provided by one expert can be subjective, e.g., in medical practice, several physicians could have different opinions about the diagnosis, prognosis or the most appropriate treatment for a given patient. Therefore, a set of class labels including the opinion of each expert could be generated for that instance. This is a common scenario in sentiment analysis, where different people tend to label documents differently according to their subjective opinions [19]. A similar scenario arises frequently in document, image and video categorization, where a set of class labels can be automatically obtained from captions, titles and other sources of information [32]. The development of the Internet has significantly reduced the cost and time needed to obtain such information, and a number of collaborative and crowdsourcing tools are available for that purpose [141].

Consensus analysis [19, 31, 187, 234, 387] provides the statistical tools to analyze the level of agreement between a group of experts when labeling a dataset. These techniques have been used in a wide range of different fields: medicine [264, 461], content analysis [19], clinical practice [65], image classification [193], etc. They have also been used as a measure of accuracy in supervised learning scenarios, e.g., the machine learning suite WEKA [236] reports Cohen's kappa values when testing classification algorithms.

Chapter outline

This chapter reviews some of the most commonly used techniques for analyzing agreement between experts in a classification experiment. Section 4.2 details expert agreement measures and discusses some of their advantages, disadvantages and modifications. Section 4.3 describes statistical tests for checking whether or not the agreement values obtained in a classification experiment are higher than those expected by chance.

4.2 Agreement indices

In an expert classification experiment, a group of R experts classifies a set of N items according to a feature X of interest. The values $\Omega_X = \{1, \ldots, Q\}$ of feature X are called categories. Each expert i has to assign a category $q \in \{1, \ldots, Q\}$ to each item j. agreement indices are used to measure and analyze the degree of agreement between experts when categorizing the items. We denote R_{jq} as the number of experts who assigned the jth item to category q. We denote R_j as the number of experts who classified the jth item:

$$R_j = \sum_{q=1}^{Q} R_{jq}.$$

The number of items out of N that expert i assigned to category q is denoted N_{iq}.

4.2.1 Overall observed agreement

The most straightforward way to assess consensus is by computing the observed agreement:

$$P_o = \frac{\sum_{q=1}^{Q} \sum_{j=1}^{N} R_{jq}(R_{jq} - 1)}{\sum_{j=1}^{N} R_j(R_j - 1)}. \tag{4.1}$$

The observed agreement takes values in the interval $[0, 1]$. An observed agreement value $P_o = 1$ means perfect agreement, whereas $P_o = 0$ means complete disagreement. This overall observed agreement has been widely criticized in the literature, e.g., see [19, 76, 445]. The observed agreement favors experiments with a low number of categories, Q. In addition, it does not take into account the different distributions of items among categories. Two solutions have been proposed to this problem: First, adjusting the observed agreement for chance agreement; and second, computing category-specific observed agreement.

4.2.2 Chance-corrected agreement

A solution to the problem of analyzing the agreement between experts in a classification study is correcting the observed value to erase the influence of chance agreements. According to Hayes and Krippendorff [245], Popping [417] identified more than 40 different proposals of chance-corrected agreement indices. In general, most of the chance-corrected agreement

indices have the following expression:

$$A = \frac{A_o - A_e}{1 - A_e}, \tag{4.2}$$

where A_o is the observed agreement and A_e is the expected agreement by chance. The numerator encodes the observed agreement beyond chance, and the denominator encodes the maximum agreement that can be achieved beyond chance. An index value of $A = 1$ means perfect agreement, whereas a value of $A = 0$ shows chance agreement. Negative values of A indicate agreement below chance. The two most frequently used chance-corrected agreement indices are reviewed next: Cohen's kappa (and some of its variants) and Fleiss' pi indices.

4.2.2.1 Cohen's kappa

Cohen's kappa index [94] is defined for a two-expert ($R = 2$) and two-category ($Q = 2$) experiment. Only those items classified by both experts are considered. The results of the experiment can be reported as a cross-classification table (Table 4.1).

Table 4.1: Cross-classification table for an experiment with two experts and two categories (+ and −)

		Expert 2		Frequency
		+	−	
Expert 1	+	a	b	N_{1+}
	−	c	d	N_{1-}
	Frequency	N_{2+}	N_{2-}	N

Cohen's kappa index A^κ has the structure of Equation (4.2), with

$$A_o^\kappa = \frac{a + d}{N},$$

and

$$A_e^\kappa = \frac{N_{1+}N_{2+} + N_{1-}N_{2-}}{N^2}. \tag{4.3}$$

Cohen's kappa index is developed under three assumptions: 1) the classified items are independent, 2) the categories are independent, exhaustive, and mutually exclusive, and 3) the experts operate independently and have different distributions for the categories. Cohen's kappa index is negatively affected by the different prevalence of the categories (prevalence problem) and by the degree of disagreement between the two experts (bias problem), e.g., see [171]. Interpreting the magnitude of Cohen's kappa index is challenging because of these effects. Several standards have been proposed for interpreting the strength of agreement (see Figure 4.1). Note that important differences can be identified between the scales: Some scales set $\kappa \approx 0.4$ as the minimum value for considering moderate agreement [14, 31, 158, 308], whereas other scales consider higher values for an acceptable agreement [243, 387]. These approaches are necessarily subjective and arbitrary, since the interpretation of Cohen's index

depends on the field of science, the nature of the experiment, and the prevalence and bias effects in the data [19, 387]. Some variants of Cohen's kappa index follow.

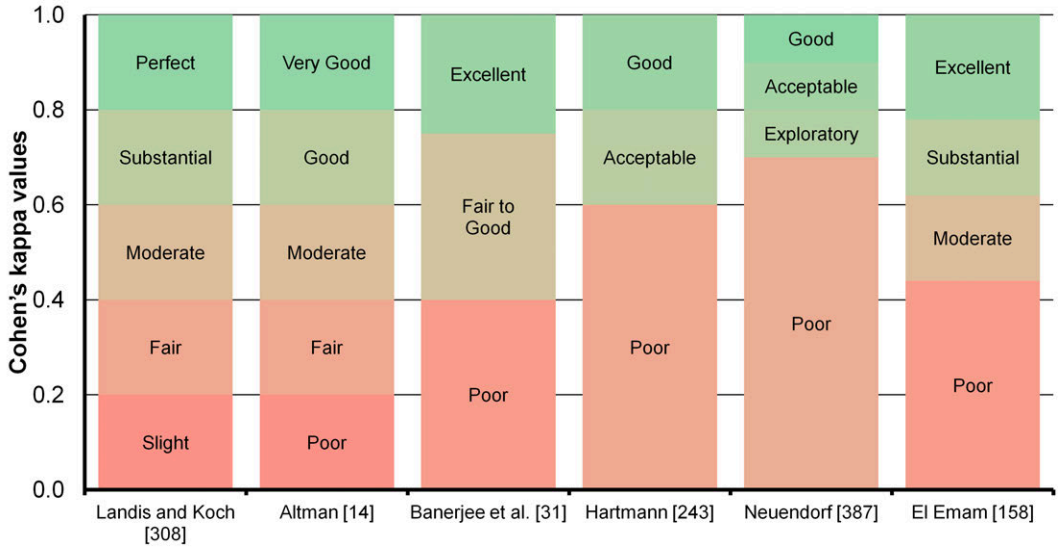


Figure 4.1: Different scales for interpreting the values of Cohen's κ index.

Prevalence-Adjusted Bias-Adjusted kappa index

Byrt et al. [70] propose a Prevalence-Adjusted Bias-Adjusted kappa index ($PABA\kappa$) to minimize the effects of prevalence and bias. This value can be reported alongside Cohen's kappa to show the effects of prevalence and bias on the index value and to determine the sources of disagreement. To compute $PABA\kappa$, the cross-classification table is modified as in Table 4.2. The agreement cells a and d (main diagonal in Table 4.1) are changed to their mean $(a+d)/2$, removing the prevalence effect. The disagreement cells c and b (secondary diagonal in Table 4.1) are also changed to their mean value $(b+c)/2$, adjusting for the bias effect. $PABA\kappa$ is Cohen's kappa index computed with the values in the modified Table 4.2.

Table 4.2: Modified cross-classification table for minimizing prevalence and bias effects and computing $PABA\kappa$

		Expert 2		Frequency
		+	−	
Expert 1	+	$(a+d)/2$	$(b+c)/2$	N'_{1+}
	−	$(b+c)/2$	$(a+d)/2$	N'_{1-}
	Frequency	N'_{2+}	N'_{2-}	N

Maximum kappa index

Another approach for interpreting Cohen's kappa value is comparing it to the maximum value $\kappa_{\max}$ that can be achieved when the marginal frequencies of each expert are fixed [463].

To compute $\kappa_{\max}$, a modified cross-classification table is used, where the agreement cells (main diagonal) are set to the minimum of the marginal frequencies for their corresponding categories, as in Table 4.3. The disagreement cells (secondary diagonal) are adjusted to maintain the marginal frequencies. The value of $\kappa_{\max}$ shows the maximum possible agreement taking into account the different prevalence and bias of the experts. The ratio $\kappa/\kappa_{\max}$ is usually computed as a measure of the proportion of agreement that was achieved in the experiment taking into account the differences between experts.

Table 4.3: Modified cross-classification table for computing $\kappa_{\max}$

		Expert 2		
		+	−	Frequency
Expert 1	+	$\min(N_{1+}, N_{2+})$	$N_{1+} - \min(N_{1+}, N_{2+})$	N_{1+}
	−	$N_{2+} - \min(N_{1+}, N_{2+})$	$\min(N_{1-}, N_{2-})$	N_{1-}
Frequency		N_{2+}	N_{2-}	N

4.2.2.2 Fleiss' pi

When more than two experts join the experiment ($R > 2$), Fleiss' [185] generalization of Scott's [445] pi index is the most commonly used chance-corrected agreement coefficient. When missing values are allowed (not all the experts have to classify all the items), Fleiss' pi (A^{π}) can be adapted to give equal weight to each judgment or equal weight to each item [19]. Here, we focus on giving an equal weight to each item. Then, Fleiss' pi follows the structure in Equation (4.2) with

$$A_o^{\pi} = \frac{1}{N} \sum_{j=1}^{N} \sum_{q=1}^{Q} \frac{R_{jq}(R_{jq} - 1)}{R_j(R_j - 1)},$$

and

$$A_e^{\pi} = \sum_{q=1}^{Q} \left(\frac{1}{N} \sum_{j=1}^{N} \frac{R_{jq}}{R_j} \right)^2.$$

Fleiss' pi assumes that the marginal distribution of the categories is the same for each expert given the assumption that they are operating by chance. This is the main difference with Cohen's kappa index, where it is assumed that the marginal distributions of the categories for each expert are different.

4.2.3 Category-specific agreement indices

The inter-expert agreement (observed and chance-corrected Fleiss' pi index) can also be studied for each category individually.

4.2.3.1 Observed agreement

We can compute specific observed agreement values for each category $q = 1, \ldots, Q$, using a similar approach as with specificity and sensitivity:

$$P_{oq} = \frac{\sum_{j=1}^{N} R_{jq}(R_{jq} - 1)}{\sum_{j=1}^{N} R_j(R_j - 1)}.$$

Several authors advocate the use of these category-specific indices, e.g., see [91, 317, 467]. Reporting these category-specific indices overcomes the problems of the overall observed agreement and avoids the need to correct for chance agreement.

4.2.3.2 Chance-corrected Fleiss' pi index

A different chance-corrected agreement index can be computed for each category using Fleiss' pi index. The chance-corrected Fleiss' pi index (π_q) for a category $q = 1, \ldots, Q$ is given by Equation (4.2) with

$$A_o^{\pi_q} = \frac{\sum_{j=1}^{N} R_{jq}(R_{jq} - 1)}{\sum_{j=1}^{N} R_j(R_j - 1)},$$

and

$$A_e^{\pi_q} = \frac{1}{N} \sum_{j=1}^{N} \frac{R_{jq}}{R_j}.$$

4.3 Statistical tests for agreement indices

Statistical tests have been proposed for testing whether or not the values reported by different agreement measures are significant.

Cohen's kappa

For large $N \to \infty$, the standard error of κ is estimated by [186]:

$$SE(\kappa) = \frac{1}{(1 - A_e^{\kappa})\sqrt{N}} \sqrt{A_e^{\kappa} + (A_e^{\kappa})^2 - \sum_{q=1}^{Q} N_{1q} N_{2q}(N_{1q} + N_{2q})},$$

where A_e^{κ} is the expected agreement in Equation (4.3), N is the number of items in the classification experiment, and N_{1q} and N_{2q} are the number of items classified by the first and second experts into category $q = 1, \ldots, Q$, respectively. Then, under the null hypothesis of no agreement $H_0 : \kappa = 0$, the statistic

$$z = \frac{\kappa}{SE(\kappa)}$$

approximately follows a standard Gaussian density $\mathcal{N}(0, 1)$ by the central limit theorem. This statistic can be used to test whether or not the agreement is higher than that expected by

chance. According to [187], a one-sided test is more appropriate here than a two-sided test, i.e., $H_1 : \kappa > 0$.

In most real-world classification experiments an agreement beyond chance is expected, and some authors discuss that specifying a value of $\kappa = 0$ is not very meaningful, e.g., see [463]. Therefore, it is customary to test whether or not the agreement values are higher than a given value taken as the minimum value for considering significant agreement (see Figure 4.1). The standard error of κ for testing $H_0 : \kappa = \kappa_0$ is estimated by [186]:

$$
\begin{aligned}
SE(\kappa) &= \frac{\sqrt{A + B - C}}{(1 - A_e^{\kappa})\sqrt{N}}, \\
A &= \sum_{q=1}^{Q} N_{1q,2q} \left[1 - (N_{1q} + N_{2q})(1 - \kappa)\right]^2, \\
B &= (1 - \kappa)^2 \sum_{q=1}^{Q-1} \sum_{\substack{k=1 \\ k \neq q}}^{Q} N_{1q,2k} (N_{1q} + N_{2k})^2, \\
C &= \left[\kappa - A_e^{\kappa}(1 - \kappa)\right]^2,
\end{aligned}
$$

where $N_{1q,2q}$ is the number of items classified by both experts in category q, and $n_{1q,2k}$ is the number of items classified by the first expert in category q and the second expert in category k. Then, the null hypothesis $H_0 : \kappa = \kappa_0$ is tested by computing

$$
z = \frac{|\kappa - \kappa_0|}{SE(\kappa)},
$$

which approximately follows a standard Gaussian density $\mathcal{N}(0, 1)$.

Fleiss' pi

Statistical tests have also been proposed for Fleiss' pi agreement measure. Fleiss [185] showed that the variance of π can be estimated by

$$
Var(\pi) = \frac{2}{NR(R-1)} \frac{\sum_{q=1}^{Q} p_q^2 - (2R-3)\left(\sum_{q=1}^{Q} p_q^2\right)^2 + 2(R-2)\sum_{q=1}^{Q} p_q^3}{\left(1 - \sum_{q=1}^{Q} p_q^2\right)^2},
$$

where $p_q = \frac{1}{NR}\sum_{i=1}^{N} R_{jq}$. Then, the statistic

$$
z = \frac{\pi}{\sqrt{Var(\pi)}}
$$

is approximately distributed according to a standard Gaussian density $\mathcal{N}(0, 1)$. This statistic can be used to test the null hypothesis $H_0 : \pi = 0$. Recently, Gwet [233] proposed two non-parametric estimators of the variance of π obtained using a linearization technique and a jackknife technique, respectively.

Fleiss [185] estimated the variance of the category-specific Fleiss' π index by

$$Var(\pi_q) = \frac{(1 + 2(R-1)p_q)^2 + 2(R-1)p_q(1-p_q)}{NR(R-1)^2 p_q(1-p_q)}.$$

Then, the null hypothesis $H_0 : \pi_q = 0$ can be tested using the statistic

$$z = \frac{\pi_q}{\sqrt{Var(\pi_q)}},$$

which approximately follows a standard Gaussian density $\mathcal{N}(0, 1)$.

Permutation tests

Permutation and resampling tests based on bootstrapping approaches can be used to assess how significant the values of the agreement indices are [156, 329]. For performing such a test, a random classification experiment is generated by randomly sampling a category q for each one of the N items in the experiment. The categories are sampled according to their relative frequency in the classifications provided by the experts. Then, the agreement index value (e.g., Cohen's κ) is computed for that randomly sampled classification dataset. The sampling procedure is repeated a large number of times M. Thus, M agreement index values $\kappa_i, i = 1, \ldots, M$ are computed and stored. The cumulative distribution of the values of the agreement index obtained with the randomly generated experiments is computed. The value of the agreement index computed using the experts' classifications κ is compared with the cumulative distribution of the agreement index values $\kappa_i, i = 1, \ldots, M$ obtained with the sampled classification experiments. A p-value is estimated by computing the proportion of agreement index values with a value $\kappa_i \geq \kappa$. Finally, the p-value is compared with a significance level α. If p-value $< \alpha$ then the null hypothesis that the experts classified the items randomly is rejected.

Chapter 5

Directional statistics

5.1 Introduction

Directional data can be found in almost every field of science [35, 180, 181]. Information measured as angles is commonly used to capture the direction of some phenomenon of interest, e.g., biologists study the movement of animals, meteorologists measure the direction of air currents, geologists observe the orientation of magnetic fields in rocks, etc. Modern visualization techniques manifest valuable three-dimensional information in a number of domains, e.g., neuroscientists are interested in the direction of neuronal axons and dendrites, microbiologists analyze the angles formed by protein structures and astrologists study the position and movement of celestial bodies. Directional statistics [276, 355] provides the theoretical background and the techniques to successfully work with this information. The roots of directional statistics can be found in the beginnings of the classical linear statistics, as explained by R. A. Fisher on a historical note reported in [355]:

> "The theory of errors was developed by Gauss primarily in relation to the needs of astronomers and surveyors, making rather accurate angular measurements. Because of this accuracy it was appropriate to develop the theory in relation to an infinite linear continuum, or, as multivariate errors came into view, to a Euclidean space of the required dimensionality. The actual topological framework of such measurements, the surface of a sphere, is ignored in the theory as developed, with a certain gain in simplicity.
>
> It is, therefore, of some little mathematical interest to consider how the theory would have had to be developed if the observations under discussion had in fact involved errors so large that the actual topology had had to be taken into account. The question is not, however, entirely academic, for there are in nature vectors with such large natural dispersions."

Directional information can be captured in two ways: angular or directional data. Angular or circular data refers to information measured in radians (or compass degrees) [325].

Directional data is a more general term which refers to information measured as directional vectors in an n-dimensional Euclidean space. We should note that there is a correspondence between the two representations by transforming the Cartesian coordinates of a point in the circumference or the sphere to its polar coordinates. We will use the term linear, as opposed to directional, circular or angular, to refer to common continuous information, e.g., wind speed measured in kilometers per hour, mass measured in kilograms, etc.

Chapter outline

Section 5.2 introduces the basic techniques for analyzing circular data and reviews the von Mises density function for modeling angles. Section 5.3 focuses on the statistical analysis of directional vectors. It includes a brief discussion of summary statistics for directional vectors and reviews the von Mises-Fisher density function.

5.2 Statistics for circular data

Circular data can arise in two basic ways [355]: angles measured with the compass or time measured with the clock. These two ways have in common that the domain in which they are defined is circular. A random variable Φ is called "angular" or "circular" if it is defined in the unit circumference, i.e., the domain of the random variable is $\Omega_\Phi = [-\pi, \pi)$. Alternatively, we can work on the domain $[0, 2\pi)$. The main property of a circular domain is that it is periodic, e.g., the values 0 and 2π refer to the same point in the circle. Therefore, it is standard to work with angular data after applying modulo 2π. The periodical behavior that comes from having a directional domain makes linear statistics unsuitable for this kind of data. Special techniques are necessary to work with circular information due to its distinctive properties [276, 355]. For instance, the classical linear mean of angles $5\pi/6$ and $-5\pi/6$ is 0 (see dashed line in Figure 5.1), which points in exactly the opposite direction. It is clear that the mean angle should be π (see solid line in Figure 5.1). Also, different visualization tools are necessary to convey directional information. In this section, some basic notions of statistics for analyzing circular data are introduced.

5.2.1 Summary statistics and graphical representations

Given a set of N angular values $\{\phi_1, ..., \phi_N\}$ defined in the unit circle $\phi_i \in [-\pi, \pi), i = 1, \ldots, N$ and having vectors of Cartesian coordinates $\mathbf{x}_i = (\cos\phi_i, \sin\phi_i)$, the mean angle $\overline{\phi} \in [-\pi, \pi)$ is defined as the angle given by the center of mass $\overline{\mathbf{x}} = \left(\overline{C}, \overline{S}\right)$:

$$\overline{\phi} = \arctan \frac{\overline{C}}{\overline{S}}, \tag{5.1}$$

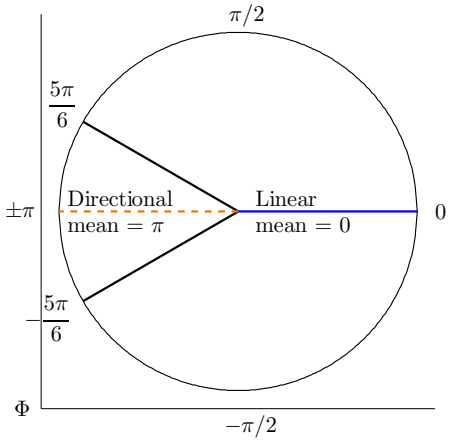


Figure 5.1: Classical linear mean (dashed) and circular mean (solid) of angles $5\pi/6$ and $-5\pi/6$.

where

$$\begin{aligned} \overline{C} &= \frac{1}{N}\sum_{i=1}^{N}\cos\phi_i, \\ \overline{S} &= \frac{1}{N}\sum_{i=1}^{N}\sin\phi_i. \end{aligned} \tag{5.2}$$

The mean resultant length $\overline{R}$ is the length of the center of mass $\overline{\mathbf{x}} = \left(\overline{C}, \overline{S}\right)$, i.e.,

$$\overline{R} = \sqrt{\overline{C}^2 + \overline{S}^2}. \tag{5.3}$$

The mean resultant length $\overline{R}$ is defined in the interval $[0, 1]$ and can be used as a measure of the concentration of a dataset. Values of $\overline{R}$ close to one show tightly clustered values, whereas values of $\overline{R}$ close to zero show uniformly distributed values. For a perfectly uniformly distributed dataset, the center of mass is not defined $\overline{\mathbf{x}} = (0, 0)$, so $\overline{R} = 0$ and $\overline{\phi}$ is not defined.

Directional statistics provides specific visualization tools to convey angular data. The most basic representation of a set of circular data is the circular plot. A circular plot shows the angular observations in a dataset as points in a unit circumference. Figure 5.2 shows the circular plot of the following set of angles: $\{0, \pi/6, \pi/4, 10\pi/24, \pi/2\}$.

Rose diagrams are used instead of classical linear histograms to show the distribution around the circle of a set of angular observations. In a rose diagram, the circle is divided into equal-width sectors. The radius of each sector is proportional to the square root of the number of observations in the bin. This ensures that the area of the sector is proportional to the frequency of the observations. Like in linear histograms, selecting an appropriate number of bins in a rose diagram is important to accurately convey the underlying distribution of the data. Figure 5.3 shows a classical histogram (a) and a rose diagram (b) of a set of angles defined in $[0, 2\pi)$ with the mean angle at $\overline{\phi} = 0$. The classical histogram ignores the periodical

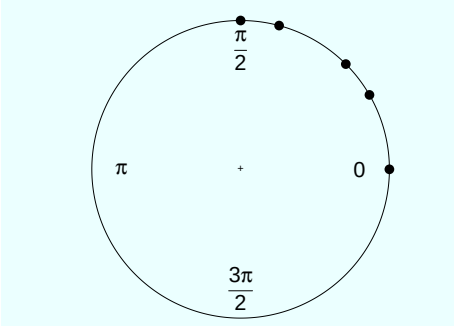
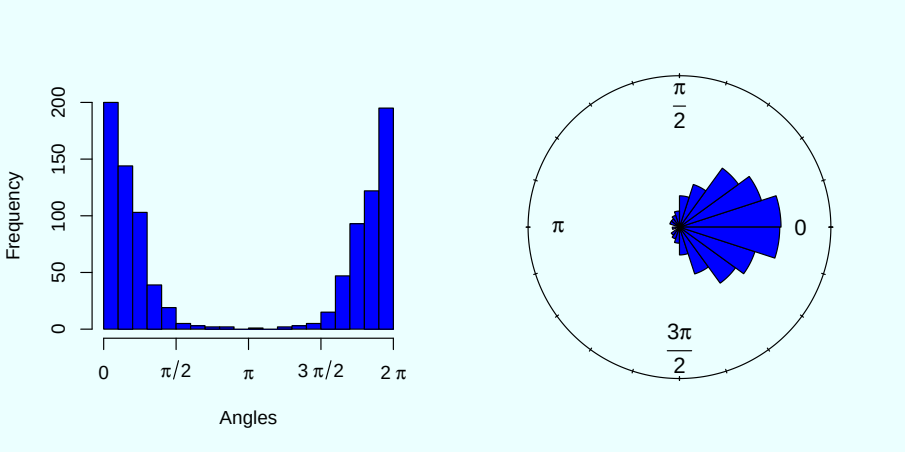


Figure 5.2: Circular plot showing a dataset of angles $\{0, \pi/6, \pi/4, 10\pi/24, \pi/2\}$.



(a) (b)

Figure 5.3: Linear histogram (a) and rose diagram (b) of a symmetric dataset of angles with the mean at $\overline{\phi} = 0$.

nature of the domain and shows a U shaped distribution with two modes (Figure 5.3(a)). On the other hand, the rose diagram clearly portrays that the dataset has only one mode (Figure 5.3(b)).

5.2.2 Probability densities for circular data

A number of probability densities have been proposed for modeling circular data (see Figure 5.4). The most straightforward way to model directional data is to adjust linear densities by wrapping them around the circle. Several probability densities have been proposed using this approach, e.g., the wrapped normal density [124, 411] or the wrapped Cauchy density [328]. However, the interest in this kind of information has led statisticians to propose special probability densities for angular data. The von Mises density [508] is the most frequently used probability density for modeling angular data. It is unimodal, symmetric and it is the maximum entropy density on the circle given the mean $\overline{\phi}$ and the mean resultant length $\overline{R}$. Recently, more flexible densities for modeling angular data have been proposed in the liter-

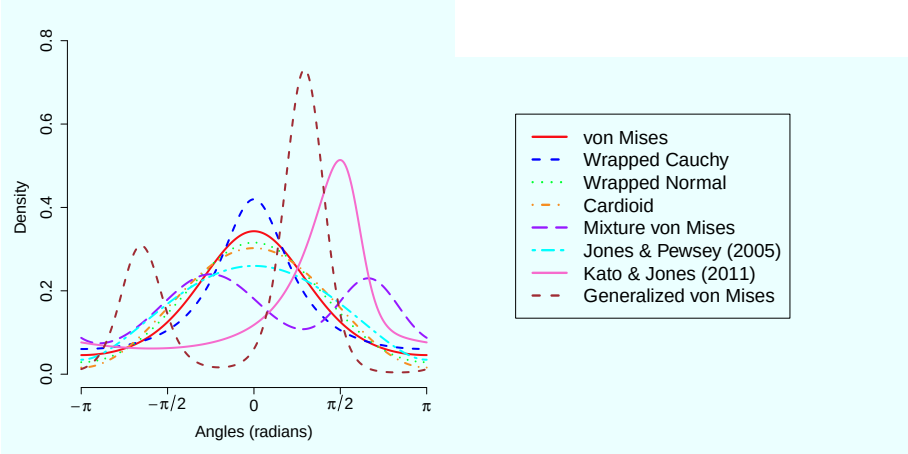


Figure 5.4: Probability density functions defined in a circular domain $[-\pi, \pi)$.

ature. Jones and Pewsey [282] proposed a family of symmetric densities on the circle that includes the uniform, von Mises, cardioid and wrapped Cauchy densities as special cases. Probability densities adopting both symmetric and asymmetric shapes in the circle can be found in a number of works, e.g., [211, 289, 290, 414, 528]. Also, probability densities for modeling both unimodal and bimodal angular data are available [211, 289, 290, 528]. Section 5.2.2.1 reviews the von Mises density and discusses the maximum likelihood estimates of its parameters.

5.2.2.1 The von Mises density

The one-dimensional von Mises density [508] is the best known angular density, as it is the circular analogue of the Gaussian density. A circular variable Φ which follows the von Mises density on the unit circle is denoted by $\Phi \sim M(\mu_\Phi, \kappa_\Phi)$ and its probability density function for a given angle ϕ is

$$f_\Phi(\phi; \boldsymbol{\theta}_\Phi) = \frac{\exp(\kappa_\Phi \cos(\phi - \mu_\Phi))}{2\pi I_0(\kappa_\Phi)}, \tag{5.4}$$

where $\boldsymbol{\theta}_\Phi = \{\mu_\Phi, \kappa_\Phi\}$ are the parameters of the distribution, μ_Φ is the mean direction angle, $\kappa_\Phi \geq 0$ is the concentration of the values around μ_Φ, and $I_\nu(x)$ is the modified Bessel function of the first kind of order $\nu \in \mathbb{R}$:

$$I_\nu(x) = \frac{1}{2\pi} \int_0^{2\pi} \cos(\nu\phi) \exp(x \cos\phi) d\phi. \tag{5.5}$$

The density of the points in the circle becomes uniform when $\kappa_\Phi = 0$, whereas high values of κ_Φ yield points tightly clustered around μ_Φ. The von Mises density is unimodal and symmetric around the mean direction. The mean direction is also the mode, and the antimode is at $\mu_\Phi \pm \pi$. The von Mises density is close to the Gaussian density when the concentration κ_Φ is high and to the Cauchy density when κ_Φ is low. The von Mises density arises naturally in several ways, e.g., as the maximum entropy density on the circle given $\boldsymbol{\mu}_\Phi$

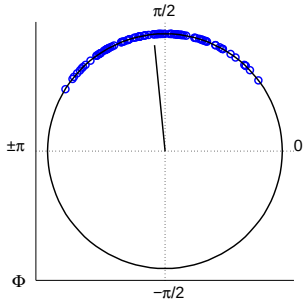


Figure 5.5: Sample of $N = 100$ points from a von Mises density $M(\pi/2, 5)$. The black line shows the sample mean direction $\widehat{\mu}_\Phi$ and its length is the mean resultant length $\overline{R}$.

and $\overline{R}$ [355]. Figure 5.5 shows a random sample of $N = 100$ points from a von Mises density. We used the functions provided in the Circular Statistics Toolbox for Matlab [41] to sample the set of angles from the von Mises densities (see Section 5.4).

Given a set of N values $\{\phi_1, ..., \phi_N\}$ randomly sampled from $\Phi \sim M(\mu_\Phi, \kappa_\Phi)$, the maximum likelihood estimator of the parameters of the density are the sample mean direction (Equation 5.1)

$$\widehat{\mu}_\Phi = \overline{\phi}. \tag{5.6}$$

and the concentration parameter $\widehat{\kappa}_\Phi = A^{-1}(\overline{R})$, where

$$A(\widehat{\kappa}_\Phi) = \frac{I_1(\widehat{\kappa}_\Phi)}{I_0(\widehat{\kappa}_\Phi)} = \overline{R} = \sqrt{\overline{C}^2 + \overline{S}^2}.$$

Unfortunately, the value of $\widehat{\kappa}_\Phi$ cannot be found analytically, and approximations have to be computed numerically [469]. For instance, Fisher [181] proposed the following approximation:

$$\widehat{\kappa}_\Phi = \begin{cases} 2\overline{R} + \overline{R}^3 + \frac{5\overline{R}^5}{6} & 0 \leq \overline{R} < 0.53, \\ -0.4 + 1.39\overline{R} + \frac{0.43}{1-\overline{R}} & 0.51 \leq \overline{R} < 0.85, \\ \frac{1}{\overline{R}^3 - 4\overline{R}^2 + 3\overline{R}} & 0.85 \leq \overline{R} \leq 1. \end{cases}$$

5.3 Statistics for directional data

Directional data can be modeled using unit vectors in a sphere [180, 355]. The unit hypersphere centered at the origin is defined by the set of n-dimensional vectors $\mathbb{S}^{n-1} = \left\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}^T\mathbf{x} = 1\right\}$. When $n = 2$, the sphere reduces to the unit circle. Directional data can be represented using circular data by transforming the Cartesian coordinates of the unit vectors into their spherical or polar coordinates, e.g., for a unit vector $\mathbf{x} = (x_1, x_2, x_3)$ in the sphere $\mathbb{S}^2$ we have (Figure 5.6):

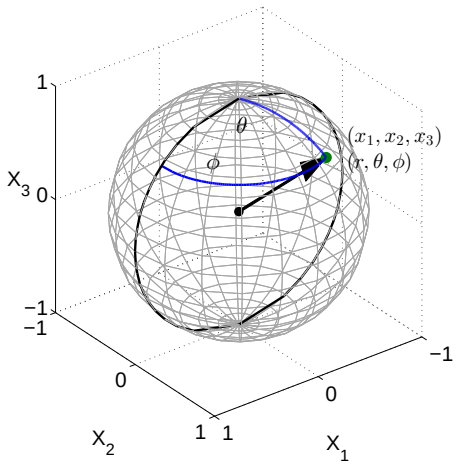


Figure 5.6: Equivalence between Cartesian and polar coordinates in the sphere.

$$
\begin{aligned}
x_1 &= r \sin\theta \cos\phi, \\
x_2 &= r \sin\theta \sin\phi, \\
x_3 &= r \cos\theta,
\end{aligned}
$$

with $r = 1$. Basic concepts regarding the modeling of spherical data are introduced in the remaining of the section.

5.3.1 Summary statistics

Given a set of n-dimensional unit vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_i = (x_{i1}, \ldots, x_{in}), i = 1, \ldots, N$ defining points in the hypersphere $\mathbb{S}^{n-1}$, the mean direction vector $\overline{\mathbf{x}} \in \mathbb{S}^{n-1}$ is defined as

$$
\overline{\mathbf{x}} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{\left\| \sum_{i=1}^{N} \mathbf{x}_i \right\|}, \tag{5.7}
$$

where $\|\|$ denotes the ℓ^2-norm. Equation (5.7) sums all the unit vectors and then normalizes the result so that $\|\overline{\mathbf{x}}\| = 1$. Similarly to Equation (5.3), the mean resultant length $\overline{R}$ is the length of the mean vector computed over the set of unit vectors:

$$
\overline{R} = \frac{R}{N} = \frac{\left\| \sum_{i=1}^{N} \mathbf{x}_i \right\|}{N}.
$$

The mean resultant length $\overline{R} \in [0, 1]$ is a bounded measure of the concentration of the set of unit vectors. When the set of unit vectors is uniformly distributed around the sphere, the mean resultant length $\overline{R}$ becomes zero and the mean direction $\overline{\mathbf{x}}$ is not defined.

5.3.2 Probability densities for directional data

Like the circular scenario, a number of probability densities have been proposed for modeling directional unit vectors in the sphere $\mathbb{S}^2$, e.g., see Table 9.2 in [355]. The von Mises-Fisher density is the most frequently used probability density in this scenario. It is unimodal, symmetric and isomorphic. Densities with more flexible shapes have been proposed for modeling directional vectors in the three-dimensional sphere $\mathbb{S}^2$, e.g., the Kent density and the Fisher-Bingham densities [294]. However, these densities need more parameters and their extension to high dimensions is challenging [30].

5.3.2.1 The von Mises-Fisher density

A directional variable $\mathbf{X} = (X_1, X_2, ..., X_n)$ which follows a multivariate von Mises-Fisher density on the unit hypersphere $\mathbb{S}^{n-1}$ is denoted by $\mathbf{X} \sim M_n(\boldsymbol{\mu}_\mathbf{X}, \kappa_\mathbf{X})$, and its probability density function for a given unit n-dimensional vector $\mathbf{x}$ is

$$f_\mathbf{X}(\mathbf{x}; \boldsymbol{\theta}_\mathbf{X}) = \frac{\kappa_\mathbf{X}^{\frac{n}{2}-1}}{\sqrt{(2\pi)^n} I_{\frac{n}{2}-1}(\kappa_\mathbf{X})} \exp(\kappa_\mathbf{X} \boldsymbol{\mu}_\mathbf{X}^T \mathbf{x}), \tag{5.8}$$

where $\boldsymbol{\theta}_\mathbf{X} = \{\boldsymbol{\mu}_\mathbf{X}, \kappa_\mathbf{X}\}$ are the parameters of the distribution, $\boldsymbol{\mu}_\mathbf{X}$ is the population mean direction vector satisfying $\boldsymbol{\mu}_\mathbf{X}^T \boldsymbol{\mu}_\mathbf{X} = 1$ (i.e., $\|\boldsymbol{\mu}_\mathbf{X}\| = 1$), $\kappa_\mathbf{X} \geq 0$ is the concentration parameter around $\boldsymbol{\mu}_\mathbf{X}$, and $I_\nu(x)$ is the modified Bessel function of the first kind and order $\nu \in \mathbb{R}$ (Equation (5.5)).

The von Mises-Fisher density reduces to the von Mises density when $n = 2$ and to the Fisher density [183] when $n = 3$. Like the von Mises density, the von Mises-Fisher density is also unimodal and symmetric around $\boldsymbol{\mu}_\mathbf{X}$, having the mode at $\boldsymbol{\mu}_\mathbf{X}$ and the antimode at $-\boldsymbol{\mu}_\mathbf{X}$. The von Mises-Fisher density is the spherical analogue to an isomorphic multivariate Gaussian density, i.e., the equal-density lines of a von Mises-Fisher density are concentric circumferences in the surface of the sphere. Figure 5.7 shows a set of $N = 100$ points from the density $M_3((0, 0, 1)^T, 5)$ defined in $\mathbb{S}^2$ and sampled using Jung's implementation[1] of the algorithm proposed in [521].

The maximum likelihood estimators of the parameters of the density $M_n(\boldsymbol{\mu}_\mathbf{X}, \kappa_\mathbf{X})$ given a sample of unit vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ are the sample mean direction (Equation (5.7))

$$\widehat{\boldsymbol{\mu}}_\mathbf{X} = \overline{\mathbf{x}},$$

and the concentration parameter $\widehat{\kappa}_\mathbf{X} = A_n^{-1}(\overline{R})$, where

$$A_n(\widehat{\kappa}_\mathbf{X}) = \frac{I_{\frac{n}{2}}(\widehat{\kappa}_\mathbf{X})}{I_{\frac{n}{2}-1}(\widehat{\kappa}_\mathbf{X})} = \overline{R} = \frac{R}{N} = \frac{\left\| \sum_{i=1}^N \mathbf{x}_i \right\|}{N}.$$

Like in the von Mises density, $\widehat{\kappa}_\mathbf{X}$ cannot be found analytically, and approximations have

[1] The source code is available at: http://www.unc.edu/~sungkyu

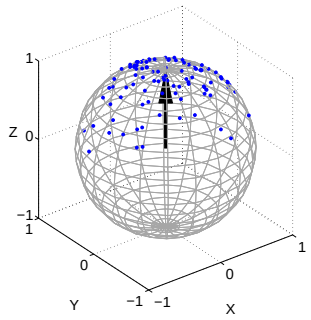


Figure 5.7: Sample of $N = 100$ points from a von Mises-Fisher density $M_3((0, 0, 1)^T, 5)$. The black arrow shows the sample mean direction $\widehat{\boldsymbol{\mu}}_{\mathbf{X}}$.

been proposed in [30, 469, 484].

5.4 Software

The following list briefly reviews the software used in this dissertation for working with directional statistics.

The CircStat toolbox for Matlab [41] implements several summary statistics and plotting methods for angular data. The von Mises density function is implemented, and parameter fitting functions are provided. Also, it includes several correlation measures and a large collection of statistical tests for angular data.

The circular package for R [3] is available at CRAN repository. It implements several probability density functions for angular data, including the von Mises density and other more flexible densities (see Section 5.2.2). It provides methods for computing summary statistics, plotting data and testing different hypotheses.

Finally, Oriana[2] is a statistical software for analyzing circular data written for Microsoft Windows. It provides a graphical user interface that allows you to import, analyze and work with angular data in a similar way as other frequently used statistical suites.

[2]Available at http://www.kovcomp.co.uk/oriana

Chapter 6

Neuroscience

6.1 Introduction

Reverse engineering the human brain has been identified as one of the main challenges in science by the National Academy of Engineering in USA[1]. Neuroscience is the field of science which studies the nervous system. The technological development in the last decades has boosted the research in neuroscience and our understanding of the brain has greatly improved. However, this knowledge is still very limited due to the huge complexity of the brain: neurons receive information from thousands of other neurons, they show very different activation behaviors, they form complex circuits with feedback and synchronization patterns, etc.

Computational neuroscience [113, 174, 442, 448, 492] has been defined as "the theoretical study of the brain used to uncover the principles and mechanisms that guide the development, organization, information-processing and mental abilities of the nervous system". Modern computing technologies have enabled important advances in our understanding of how the brain works. Software suites have been developed for semi-automating some time-consuming processes, e.g., the reconstruction of neuronal morphologies from microscope images, obtaining counts of neuron somas from tissue images, etc. Data mining methods have been used for analyzing the large amounts of data produced by neuroscientists with those new techniques. The morphology, electrical properties and activation behaviors of the neurons have been encoded and simulated using computer models. Powerful visualization tools have been designed to graphically represent the models and results obtained.

The work developed in this dissertation has been mainly applied to the study of neuronal morphology. Neuronal morphology shows a huge variability across neuron types, brain areas and animal species. Therefore, statistical analysis and machine learning tools seem like a promising approach for modeling and analyzing neuron morphology. This chapter provides an introduction to the biological notions used throughout this dissertation.

[1]Website available at: http://www.engineeringchallenges.com/

Chapter outline

Section 6.2 starts with a brief note about the history of modern neuroscience. Some basic concepts on the organization of the brain and the structure of neuronal cells are included in Section 6.3. The problem of the morphological variability of neurons and how it affects the classification of neurons in the cortex are reviewed in Section 6.4. Finally, Section 6.5 enumerates some of the most prominent research efforts that have been developed by the neuroscience community in the last decade.

6.2 Historical note

The first theories of neuronal organization were postulated in the 1830s, but it was not until the second half of the XIX century when the advances in citohistology allowed neuroscientists to start studying the neuron's structure [345]. Christian G. Ehrenberg (1795–1876) and Johannes E. Purkinje (1787–1869) provided the first descriptions of cells in the nervous systems. Later, the German scientist Joseph von Gerlach (1820–1896) proposed the first widely accepted neuronal theory, the reticular theory. This theory postulated that the brain was formed by a continuum of nervous fibers and that there was no physical separation between one neuron and the next one.

The technological advances in the construction of microscopes, combined with new techniques for cell staining, allowed the identification of the different parts of a neuron. In 1883, Camillo Golgi (1843–1926) discovered the method known as Golgi's method or black reaction. This method was based on hardening the cellular tissue by introducing it in potassium dichromate and osmic acid. Then, the tissue was stained using a silver nitrate solution. As a result, a small number of neurons appeared stained in black over a yellow background. One of the main advantages of this method was that only few neurons were stained randomly (only 1% of the neurons in a specific area of the tissue). Otherwise, the huge density of neuronal branches in the brain would have made it impossible to clearly distinguish the neuronal morphology. Also, the level of detail offered by the new technique was higher than in previous methods [118]. Golgi received, along with the Spanish neuroscientist Santiago Ramón y Cajal, the Nobel Prize in Medicine in 1906 for his neuroanatomy studies.

Wilhelm His (1831–1904), Fridtjof Nansen (1861–1930) or Auguste H. Forel (1848–1931) were some of the first scientists who criticized the reticular theory at the end of the XIX century. However, Santiago Ramón y Cajal was the first scientist who gave evidence suggesting that neurons were independent units and they did not form a continuous net. He improved Golgi's method and applied it to neuronal tissue from different areas in the brain. Cajal introduced the basic concepts of a new neuron doctrine. Also, he established the roots of the dynamic polarization law, the specific connectivity principle and he foresaw the concept of synapsis [345].

From Cajal's discoveries, most of the scientific community adopted the new theory and helped spreading it with new research. The electronic microscope, invented in the middle of the XX century, was able to obtain more detailed images of the nervous tissue. These

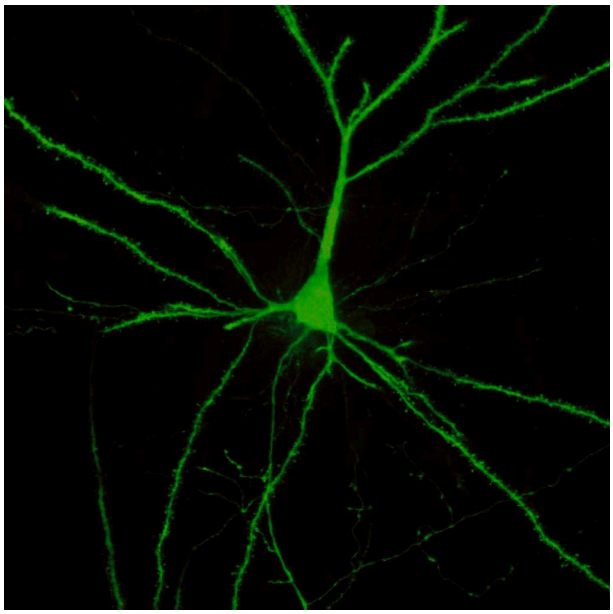


Figure 6.1: Confocal microscope image of a pyramidal neuron from the rat's neocortex. Source: Instituto Cajal (CSIC).

new tools and research works confirmed the Cajal's neuronal theory and allowed the in-depth study of the synaptic connections.

Nowadays, the confocal microscope is the most commonly used tool for studying the neuronal structure. This microscope takes a set of pictures of a given neuron using different angles and different depths (see Figure 6.1). Then, the neuronal morphology is traced using a (semi-)automatic software tool [146, 235, 483]. Once the neuron structure has been obtained, the different parts of the neuron can be visualized, labeled, edited, etc.

6.3 Brain organization and neuronal morphology

The central nervous system can be divided into seven main parts [288]: the spinal cord, the medulla oblongata, the pons, the cerebellum, the midbrain, the diencephalon and the cerebral hemispheres. These parts can be subdivided according to different criteria, e.g., their anatomic features, their location, etc. In general, each one of these distinguished parts can be assigned a specific functional role. All cognitive functions, such as the ability to talk, the movement of each part of the body or the memory, are located in different areas of the cerebral cortex, which is placed in the outer part of the hemispheres.

The cerebral cortex is a structure divided in layers and including a great variety of neurons. This structure can be found in several types of non-mammals vertebrates, although it has greatly evolved and differentiated in mammals. The differences in the size and the organization of the neocortex, the newest and most complex part of the cerebral brain, is believed to allow the wide range of behaviors and abilities in mammals [37].

In general, the cerebral cortex can be divided into six layers enumerated from I (the most

superficial) to VI (the deepest). Also, the cortex is divided into cortical areas according to their main features. However, different researchers have identified very different cortical areas. All the cortical areas studied in the literature seem to include the same basic elements: neurons, glial cells, fibers and blood vessels. The characterization of the different types of neurons is one of the main goals in the study of the cerebral cortex.

The cortical column has been shown to be the basic organization and functional circuit in the neocortex. The term column is vague [122, 423, 431], since it can refer to small-scale minicolumns (diameter $\sim 50\mu m$), to larger-scale macrocolumns (diameter $\sim 300 - 500\mu m$), and to multiple different structures within both categories (including barrel columns and ocular dominance columns, extent of arborization of single thalamic afferent fibers, cytochrome oxidase blobs, individual dendritic arbors of pyramidal cells, and tangential widths of axonal patches originated from pyramidal cells). However, a diameter of $\sim 300\mu m$ remains rather similar across several species and cortical areas for many of these cortical columns [350, 380].

6.3.1 Neuron structure

The neuron is the basic structural unit in the brain. The number of neurons in the human brain is estimated to be in the order of 86.1×10^9 [25]. Neuron morphology exhibits a huge variability across areas and species [140, 413]. A neuron can be structurally divided into three parts: the soma, the dendrites and the axon.

The soma or cell body includes the cell nucleus, along with other cellular structures such as ribosomes, mitochondria, lysosomes, etc. The neurites grow away from the soma and can be classified into: dendrites and axon. The morphology of the soma greatly varies depending on the number and the orientation of the neurites, although it is usually modeled as a sphere or an ovaloid. The soma can appear centered with respect to the rest of the neuron morphology or displaced. Its size can greatly vary, with diameters between 4 and $135\mu m$.

Most of the chemical activity in the neuron takes place in the soma. Therefore, the soma has traditionally been given the role of receiving and processing the information that goes into the neuron through the dendrites and sending the result to other neurons through the axon. However, neuronal behavior is extremely complex and depends on a lot of other factors, e.g., dendrite and axonal morphology, the cell environment, etc.

The dendrites are prolongations that profusely bifurcate forming branched structures called dendritic trees. These structures increase the neuronal surface for receiving synapses from other neurons. The dendritic branching patterns greatly differs between neuron types, species and areas [39, 163, 275, 301]. Some dendrites extend away from the soma without bifurcating until they reach a specific area where they frequently branch. Other dendrites bifurcate close to the soma, whereas other dendrites grow away from the soma and end without bifurcating. Recent studies have postulated the idea that dendritic branching patterns follow optimal dendritic wiring [102–105, 499, 515].

However, the principles that determine the geometric shapes of dendrites are currently unknown.

In general, dendritic branching nodes give raise to two branches, although branching nodes with three or more branches have been observed. These multifurcations can represent up to 10% of the total branching nodes in some neuronal types [44]. However, it is not clear whether these multifurcations are indeed branching nodes with more than two branches, or if they should be considered as several bifurcations located at a minimal distance [504].

Some kinds of neurons show dendritic spines in the surface of their dendrites [424, 531]. The function of these spines is not known [532], although it has been proposed that their main goal is to increase the dendritic surface so that a higher number of connections can be achieved. It has also been shown that the position, shape and number of dendritic spines changes in time. This morphological plasticity is said to be related to memory, learning or neurodegenerative diseases [12, 40, 126, 529].

The functional role of dendrites is receiving and integrating the information coming from other neurons. The shape and size of the dendritic tree determine which neurons are connected to a given neuron and the place where the synapses occur. Traditionally, the study and simulation of neurons considered dendrites as indivisible nodes that did not perform any kind of information processing. Thus, the dendritic morphology was not taken into account. However, it has been proved that the branch geometry and the spatial location of the synapses allow different parts of the dendritic tree to work in a semi-independent way [459]. Therefore, dendritic morphology is a relevant element to understand neuronal behavior [244, 296, 297, 349, 506].

The axon is a neurite that can extend very long distances away from the soma. Its length can hugely vary and its diameter is usually small (between 0.2 and 20µm). Axons are covered by myelin sheaths which are regularly interrupted by the Rainver nodes [288]. Usually, the axon grows away from the soma until it reaches a specific area where it branches to connect with the dendrites of other neurons in the same or a different region. Also, axons can grow collateral branches to connect with closer neurons. The main function of the axon is sending the information to other neurons. This information is codified in electric pulses called action potentials or spikes. The action potentials are generated in the initial part of the axon and propagate through it until they reach the axon terminals where the synapses occur. Therefore, the axon morphology also defines which neurons are communicated and their functional roles. Notably, some neurons are axonless, such as retina amacrine cells and olfactory granule cells [522].

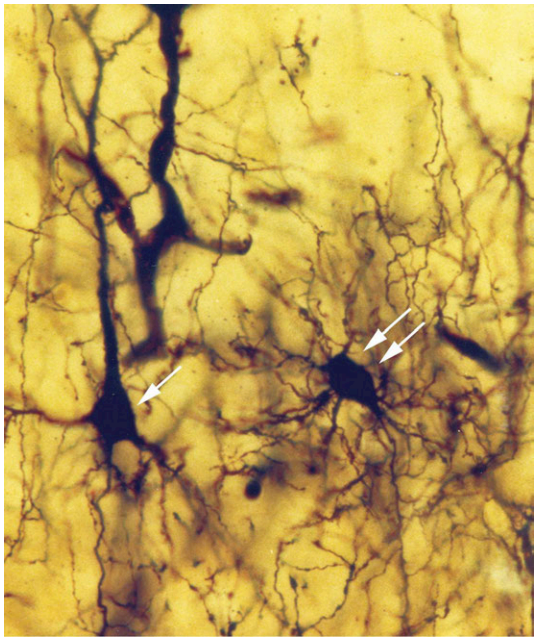


Figure 6.2: Photomicrograph from Cajal's preparation of the occipital pole of a cat stained with the Golgi method, showing a pyramidal cell (one arrow) and an interneuron (neurogliaform cell) (two arrows). From DeFelipe and Jones [118].

6.4 Neuron classification

The morphologies, molecular features and electrophysiological properties of neuronal cells are extremely variable [140, 413, 436, 472]. Neuronal morphology is a key feature in the study of brain circuits, as it is highly related to information processing and functional identification. Except for some special cases, this variability makes it hard to find a set of features that unambiguously define a neuronal type [413]. In addition, there are distinct types of neurons in particular regions of the brain. Indeed, neurons in the cerebral cortex can be classified into two main categories based on their morphology: pyramidal neurons and interneurons (Figure 6.2):

> Pyramidal neurons are excitatory (glutamatergic) cells which display spines in their dendrites and have an axon which projects out of the white matter. Their name refers to the pyramidal shape of their soma and was introduced by Rudolf Berlin in 1958. Pyramidal neurons are key elements in the functional organization of the cerebral cortex, where they are the most frequent neuronal type (70-85%) and the main source of cortical excitatory synapses. Pyramidal neurons show a characteristic morphology (see Figure 6.3). A big apical dendrite grows from the apex of the soma towards the cortical surface, usually extending until it reaches layer I, where it frequently branches forming a dendritic tuft. A set of radially oriented basal dendrites grow from the base of the soma. The basal dendritic arbors of pyramidal cells represent about 90% of the dendritic length of cortical pyramidal neurons from layers II/III and V [318]. A single axon

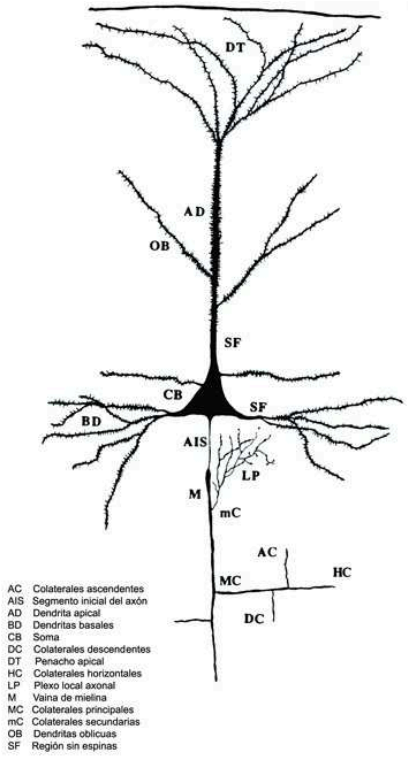


Figure 6.3: Schema of the morphology of a pyramidal neuron. The schema shows the main properties defining this kind of neuron: a pyramidally-shaped soma, an ascending apical dendrite, radially-oriented basal dendrites and a descending axon. Source: Benavides-Piccione [37].

grows from the base of the soma. Another important feature of pyramidal cells is that their dendritic surfaces are covered by spines, which represent the major postsynaptic elements of excitatory synapses [121, 172, 468, 517, 533].

Interneurons are cells with short axons that do not leave the white matter and their dendrites show few or no spines. These interneurons appear to be mostly GABAergic (inhibitory) and constitute ∼15–30% of the total neuron population, but they display chemical, physiological and synaptic heterogeneity [413]. Most cortical interneurons lack the typical somatodendritic morphological characteristics used to identify projection neurons, namely a pyramidal-shaped cell body and an apical dendritic tree that is distinct from and lies opposite to the basal dendritic arbor. However, the absence of these features should not be used to define interneurons, as they are neither necessary nor sufficient for distinguishing interneurons from projection neurons. Indeed, there are interneurons that have a somatodendritic morphology resembling that of pyramidal cells, e.g., the so-called "pyramidal basket cells" [15], and projection neurons that have a non-pyramidal appearance in their somata and dendrites [412].

Traditionally, interneurons have been subdivided into two main groups [412]: spiny non-pyramidal cells and aspiny or sparsely spiny non-pyramidal cells. Spiny non-pyramidal cells are located in the middle cortical layers, especially in layer IV of primary sensory cortices. They comprise a morphologically heterogeneous group of interneurons with ovoid, fusiform, and triangular somata. Most spiny non-pyramidal cells are excitatory (glutamatergic [173]), and their axons are distributed within layer IV or in the

adjacent layers above or below the somatic location [470]. Aspiny or sparsely spiny non-pyramidal cells usually have axons that remain near the parent cell, although some run prominent collaterals in the horizontal (parallel to the cortical surface) or vertical dimension (ascending and/or descending, reaching other cortical layers). These interneurons appear to be mostly GABAergic and constitute 10-30% of the total neuron population, the percentage varying substantially between cortical layers, areas and species [118, 369]. They are the main component of inhibitory cortical circuits. The identification of classes and subclasses of interneurons is clearly critical for gaining a better understanding of how these cell shapes relate to cortical functions in both health and disease. GABAergic interneurons also show a remarkable morphological variability between species, layers and areas [117].

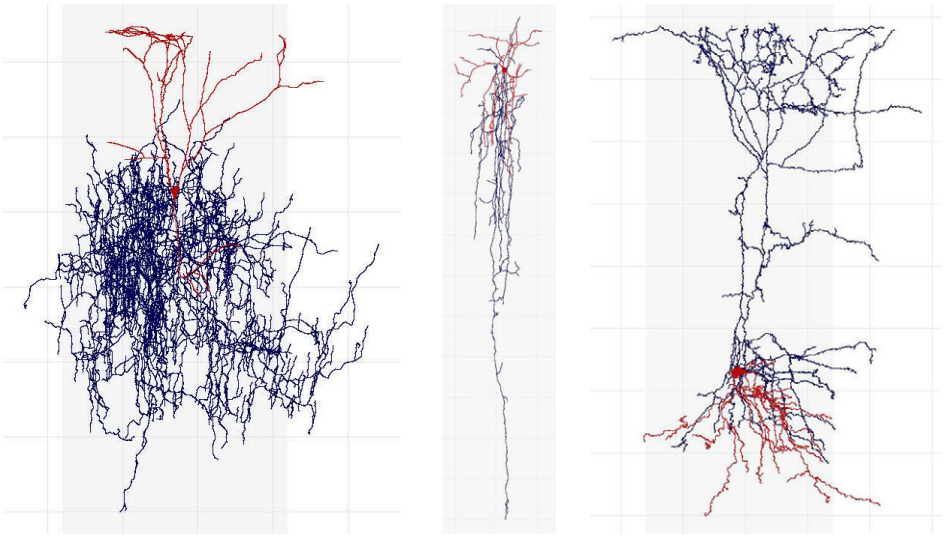
6.4.1 Neuronal variability

The Internet has made it possible for researchers to share digital three-dimensional reconstructions of neuronal morphology in publicly accessible databases [20, 23]. With such amount of available data, a common nomenclature for naming cortical neurons is a crucial prerequisite for advancing in our knowledge of neuronal structure [51]. However, the wide neuron morphological variability makes it difficult to find a set of morphological properties which clearly define neuronal types [413]. Therefore, the characterization of the different neuronal types should be statistical in nature [143].

With few exceptions, no general consensus has emerged for naming cortical neurons. For example, at present most neuroscientists agree on the usage of terms such as pyramidal neuron, non-pyramidal neuron, interneuron, and chandelier (or axo-axonic) cell. These cell types are readily distinguished by their clear morphological attributes. However, other common names, such as double bouquet cell, Martinotti cell, neurogliaform cell, and basket cell, seem to lack a consensual definition. In these cases, the same name is often assigned to neurons of varying morphologies by different authors, and a variety of terms are inconsistently adopted in different laboratories to represent the same cell classification. As a consequence, virtually every author has his/her own classification scheme and neuron terms, making the comparison and exchange of information among laboratories rather difficult, if not impossible. In fact, it is not possible to find two neurons with the same exact morphology.

Two types of variability can be identified: interclass variability and intraclass variability. Figure 6.4 shows the three-dimensional reconstructions of three neurons corresponding to very different morphological classes as an example of interclass variability. Chandelier cells (Figure 6.4(a)) are neurons with the soma in layers II-VI; multipolar or bitufted dendritic arbors; and distinguished by pre-terminal axon branches that form short vertical rows of boutons resembling candlesticks. Double-bouquet or horse-tail cells (Figure 6.4(b)) are neurons with the soma in layers II-III; multipolar or bitufted dendrites; and distinguished by their "horse-tail" axons forming tightly intertwined bundles of long descending vertical collaterals. Finally, Martinotti cells (Figure 6.4(c)) are neurons with the soma in layers III-VI; multipolar, bitufted or bipolar dendrites; and distinguished by their ascending axons that give rise to

two axonal arborizations, one near the cell body and the other at a variable distance above the cell body. This second plexus may be very dense (axonal tuft) or more diffuse, and it can be found either in the same cortical layer as the cell body of origin or in the layers above (ascending axons can travel from layer VI to layer I). We can also observe how the dendritic morphology also varies in the different neurons.



(a) (b) (c)

Figure 6.4: Three cells belonging to different neuron types. The axon is shown in blue whereas the dendrites are shown in red. (a) Chandelier cell from the rat neocortex [249]. (b) Horse-tail cell from the rat neocortex [511]. (c) Martinotti cell from the mouse neocortex [221]. Source: NeuroMorpho.Org [23].

Figure 6.5 shows the three-dimensional reconstructions of the basal dendrites of four pyramidal neurons from the primary motor cortex of the mouse. Some similarities can be identified between the reconstructions, e.g., there are dendritic trees oriented in all directions, dendrites bifurcate close to the soma and they have long ending segments which do not intertwine. However, there are differences in the number of dendritic trees, their sizes or the number of branching nodes in each tree.

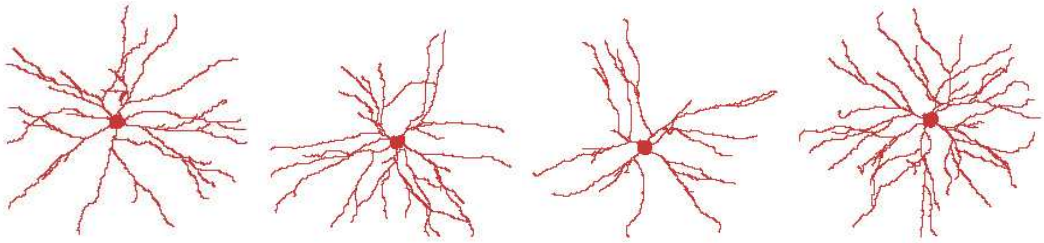


Figure 6.5: Basal dendrites of four pyramidal neurons from the primary motor cortex of the mouse. Source: Instituto Cajal (CSIC).

6.5 Current research efforts in neuroscience

The beginning of the XXI century is seeing a renewed interest in the study of the brain. The Blue Brain project[2] [358] is an international project launched on 1st July 2005 by the Brain Mind Institute at the École Polytechnique Fédérale de Lausanne and IBM (International Business Machines). The aim of this initiative is to use the enormous computing power of IBM's supercomputers to simulate the brains of mammals with a high level of biological accuracy. To achieve this goal, highly detailed computer models of neurons are generated including three-dimensional reconstructions of their morphology, the distribution and composition of their ion channels, the number and the location of their synapses, their physiological and pharmacological properties, etc. Then, the number and locations of the different types of neurons are studied. The first model of a rat's neocortical column was announced in the summer of 2006. This virtual cortical column included 10000 units of a simplified neuronal model. This computer model was used to simulate the electrical activity in the cortical column.

The Cajal Blue Brain project[3] is the Spanish participation in the Blue Brain project, involving the Universidad Politécnica de Madrid and the Centro Superior de Investigaciones Científicas among other partners. The Cajal Blue Brain project officially started in 2008. It focuses on two topics: the study of the functional and anatomical microorganization of the cortical column, and the development of new biomedical technology with potential applications in other fields. The main long term goals in the Cajal Blue Brain project can be summarized in four points:

Studying the synaptome [120], i.e., the detailed map of the synaptic connections in the cortical column.

Understanding how the cortical column works, both in health and disease, specially focusing on Alzheimer disease.

Developing new methods for processing and analyzing biological data.

Developing new graphical methods for visualizing and simulating brain function.

In 2013, two big projects focusing on the study of the brain have been announced. In january 2013, the European Research Commission announced the Human Brain Project[4] as one of the two projects selected for the EU FET Flagship Program. The project goals can be grouped into four topics: First, generating complete and accurate brain atlases and building brain models from them. Second, identifying the mathematical principles underlying the relationships between different levels of brain organization. Third, building a system that integrates all the knowledge about the human brain. Fourth, developing new technologies and tools to produce results with immediate value for basic neuroscience, medicine and computing technology.

[2] Website available at: http://bluebrain.epfl.ch/
[3] Website available at: http://cajalbbp.cesvima.upm.es/
[4] Website available at: http://www.humanbrainproject.eu/

Almost simultaneously, the Brain Research through Advancing Innovative Neurotechnologies (BRAIN) project, formerly known as the Brain Activity Map project [10, 11], was announced by the United States of America's President Obama in february 2013 as a decade-long large-scale effort to study the functional mapping and neural activity in brains. It has three main goals [11]: First, building tools to simultaneously record the activity of all neurons in a circuit. Second, developing methods for influencing the activity of individual neurons in the circuit. Third, understanding the circuits' function.

Computer science and statistics lay at the core of these new research projects. These projects serve as an example of the kind of synergies stemming from multidisciplinary research when real problems drive research in artificial intelligence. Indeed, we expect statistics and machine learning methods to play a major role in solving some of the most challenging problems in neuroscience and medicine, such as the study of low-level interactions between neurons in the brain and their relationships to perception, learning or brain diseases.

Part III

CONTRIBUTIONS TO BAYESIAN NETWORK MODELING

Chapter 7

Bayesian network modeling of pyramidal basal dendritic trees

7.1 Introduction

In Chapter 6 we saw that dendritic morphology is essential for understanding neuronal connectivity and is a crucial feature in information processing and brain function. Pyramidal neurons are key elements in the functional organization of the cerebral cortex and the structure of their dendritic trees affects the process of information integration, whereas their size influences the mixing of inputs [515]. The branching patterns of the dendritic trees are related to the processing of synaptic inputs [244, 296, 297] and define the electric behavior of the neurons [82, 349, 506]. Different parts of the dendrites can operate semi-independently according to the spatial location of synaptic connections [459]. As a result, there is considerable interest in the analysis of the microanatomy of pyramidal cells since it constitutes an excellent tool for better understanding cortical information processing.

Despite recent advances in molecular biology and new discoveries related to neuronal development, current knowledge about neuron structure is still incomplete, and it is hard to find a set of anatomical traits that unambiguously define a neuron type (see Section 6.4 for a discussion on neuronal morphological variability and the problem of neuron classification). Computational stochastic models have been used for the last two decades to measure geometric parameters of real neuronal arborizations and simulate virtual neuron morphologies. These simulations can be used to identify the basic structures and important features in neuronal classes, to study neuronal development and neurite outgrowth, or to examine relationships between morphology and neuronal function. As 3D reconstruction of real cells is a time- and resource-consuming process, the data compression and amplification that can be achieved with these techniques are also important advantages [20, 22]. However, a major obstacle to the creation of virtual neurons is method validation because data on the complete dendritic tree of real neurons is rather scarce. Indeed, labeled processes are frequently incomplete because, during the tissue slicing procedures some parts of the neuron morphology

are missing in a varying degree, depending on the thickness of the sections and the relative localization of the labeled neuron within the slice. Usually, this problem can only be overcome using serial sections to reconstruct the cell in 3D. However, neuronal processes are not always easy to trace and they may at times get lost in the background noise [119]. Together, these obstacles make it very laborious and time-consuming to obtain meaningful measurements from neurons. In the study of this chapter, we have used data from fully reconstructed basal dendrites of pyramidal cells. The basal dendritic arbors of pyramidal cells represent about 90% of the dendritic length of cortical pyramidal neurons from layers II/III and V [318]. The whole basal dendritic arbor can be fully reconstructed in single horizontal sections [162]. Thus, they are particularly valuable for validating the simulated virtual neurons.

In this chapter, we present a novel methodology for the 3D simulation of dendritic trees using BNs. The goal is to simulate virtual dendrites that are visually and statistically indistinguishable from real ones. This approach has a number of advantages over previous models. First, BNs use the conditional (in)dependencies between the variables defining the dendritic morphology to model their JPD. In fact, the possible use of BNs to consider the relationships between morphological variables has already been noted [491]. Since these statistical relationships are found automatically by analyzing the data, the whole process is data-driven and widely applicable. Thus, instead of changing the model to consider different kinds of relationships and analyzing the results to gather relevant information, we let the data speak. The resulting model is studied to gain insights into the processes underlying dendritic morphology. We believe that this approach is less constrained by a priori assumptions based on current biological knowledge and is not affected by disagreements between domain experts.

Second, the model learns and uses a BN for each part of the dendrite. This way, the relationships can change to take into account the dendritic tree location. This is an important characteristic, since there is evidence supporting the idea that heterogeneous parts of the dendrites could be regulated by different developmental factors [144]. The model is flexible enough to capture and exploit low-level relationships.

Third, the model uses an extensive set of variables that include commonly used dendritic tree measurements along with new features to model context factors, such as the morphology of the subdendrite or the distance to the nearest segment. Finally, the use of several univariate tests and a novel multivariate test makes the evaluation more robust and reliable.

The research included in this chapter has been published in López-Cruz et al. [338].

Chapter outline

The chapter is organized as follows. Section 7.2 reviews computational models for analyzing and simulating neuronal morphology. Section 7.3 details the proposed methodology for modeling and simulating basal dendritic trees from pyramidal neurons. Section 7.4 shows the results of the extensive evaluation of the methodology. Discussion and final comments are included in Section 7.5.

7.2 Related work

Existing models for simulating dendritic morphology can be grouped into two categories: growth and reconstruction models [498]. Growth models try to capture the behavior of growth cones during neuron development and, thus, are able to simulate dendritic structure at its different stages of maturity. These models usually consider that the neurite tips elongate and taper as they grow away from the soma until a bifurcation occurs or the neurite ends. They estimate the probabilities for each of these events taking into account some of the different factors involved in neuron development, e.g., molecular gradients [251], electric field presence [429], neuritic tension [331], segment length or centrifugal order [500], neurotrophic particles [346], etc. One of the recent works that implements this kind of model [298] has simulated complete networks of neurons. The probability functions include complex elements, e.g., the influence of competition between dendrites when deciding if a bifurcation should occur, the distance between dendrites and axons when establishing synaptic connections, etc.

On the other hand, reconstruction models measure relevant variables from real neurons and use their statistical distributions to describe the dendritic tree structure. Then, a simulation algorithm samples the distributions to output virtual dendrites that should be indistinguishable from real ones. Donohue and Ascoli [143, 144] propose an algorithm that samples 2D virtual dendrites from the univariate marginal probability distributions of some basic parameters (segment length, width and bifurcation probability). Later, they consider conditional relationships between the variables and three other fundamental parameters (centrifugal order, segment radius and path length) and compare the models at length [145]. They use common predefined parametric distributions, like Gaussian, Gamma or uniform distributions, to fit the data.

Parametric distributions might not accurately fit the real distributions, and other models use non-parametric approaches to avoid that problem. KDE is used in [334] to simulate 2D dendritic structures taking into account conditional relationships between features. Torben-Nielsen et al. [491] use conditional KDE to generate dendrites and angle information is included to obtain 3D simulations.

Variations of L-Systems [434] have also been used in neuronal morphology simulations because of their ability to create branching structures [21, 239]. Recently, evolutionary computation has been used to output L-Systems that generate virtual neurons that are similar to a single real one [488–490]. They also include a postprocessing step to filter out non-biologically plausible neurons.

The above models only measure univariate marginal probability distributions or define a priori conditional relationships ad hoc, e.g., see [17, 334]. Considering variables to be independent keeps models simpler, which makes them easy to analyze. However, the independence assumption does not hold since complex interactions with extracellular elements and intrinsic factors have been widely reported for real neurons [366, 444]. Other works define relationships between model parameters according to some predefined criterion and study the simulated neurons to check whether or not the hypotheses are correct, e.g., see

[145, 439]. This methodology is more likely to be biased towards expert knowledge and disregards important information that could be inferred from the data.

7.3 Models and simulation of basal dendrites with Bayesian networks

Our proposal can be classed as a reconstruction model. The whole process is summarized in Figure 7.1. First, we measure key features in 3D reconstructions of real pyramidal neurons and estimate their JPD to define the BNs used to build up the model. We then simulate from the BNs to output a set of virtual dendrites, which we compare with the original data to verify the model's ability to capture the dendritic tree structures. In the evaluation step, statistical tests are performed to check whether or not the variables included in the model have the same distribution in the original and simulated data. Other emergent features measured from the whole dendritic trees and branches, not used in model learning, are also compared, e.g., total dendritic length, asymmetry index, etc. Wilcoxon rank-sum, Kolmogorov-Smirnov and KL-based tests are used to compare each variable independently. We also propose a multivariate test that uses a KL estimation technique to compare the JPD over a set of variables. Finally, examples of real and simulated dendrites are visually compared.

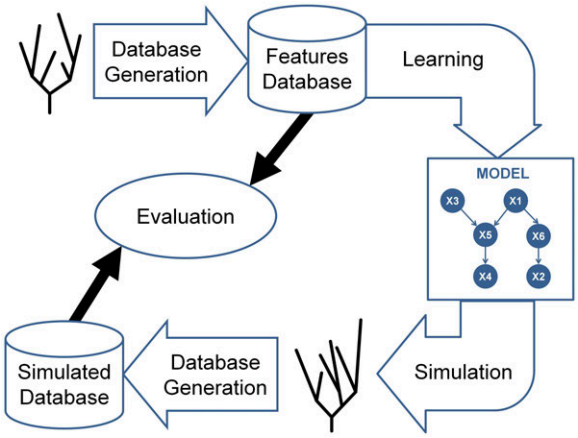


Figure 7.1: Our reconstruction model approach.

The following sections provide details about each of the steps involved in the methodology. First, Section 7.3.1 explains the data acquisition and preparation process, i.e., how the variables used in the model were measured from the real dendrites and the data were preprocessed. Then, the model is built by learning one BN for each part of the dendritic tree in Section 7.3.2. Then, the simulation algorithm that uses the BNs to generate virtual 3D dendrites is shown in Section 7.3.3. Finally, the evaluation methodology comparing virtual and real dendrites is presented in Section 7.3.4. Figure 7.2 shows a schematic overview of the whole process.

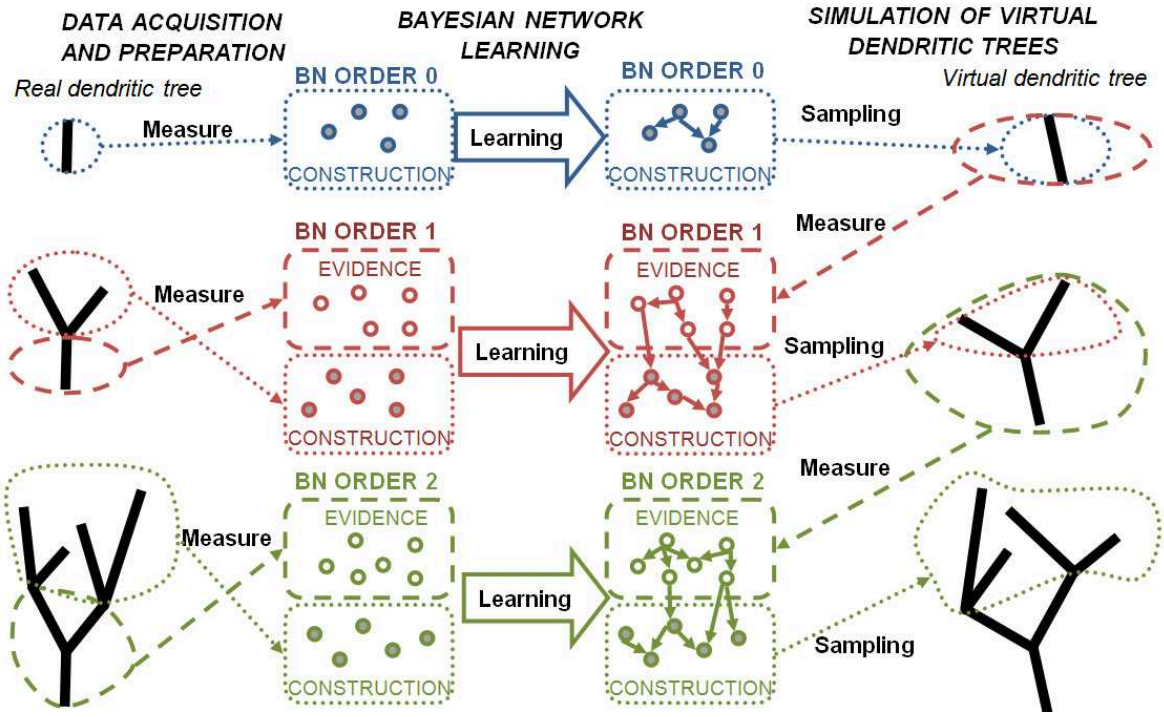


Figure 7.2: Application of BNs to the modeling and simulation of basal dendritic trees. The figure shows the measuring of variables from the real trees, the learning of the BNs and the sampling of values to simulate the virtual dendritic trees.

7.3.1 Data acquisition and preparation

We have used a set of 3D reconstructions of 90 pyramidal neurons from the mouse neocortex (two BC57 Black mice, 2 months old). These neurons were labeled with Lucifer Yellow using an intracellular injection method that covers the full extent of the basal dendritic arbor. The neurons were located in layer III of different cortical regions: the secondary motor cortex (M2), the secondary somatosensory cortex (S2) and the lateral secondary visual cortex and association temporal cortex (V2L/TeA). Therefore, three databases[1] of reconstructions were built according to their cortical area. The whole basal dendritic trees of the neurons were traced using the Neurolucida package [219] and stored in digital files in the ASC Neurolucida format. The tissue preparation and injection process are detailed in [39]. The reconstructions are publicly available at `www.neuromorpho.org` [23] as part of DeFelipe's laboratory archive. Each basal dendritic arbor is made up of approximately 6 (mean $\pm$ SD, 5.7 ± 0.9, range 4-8) main trunks, which are in turn made up of several dendrites, as shown in Figure 7.3. For the sake of simplicity and unless otherwise stated, we called these single trunks of basal dendritic arbors *dendritic trees*.

The 3D reconstructions were made up of the Cartesian coordinates of the points where

[1]The term "database" refers to the sets of 3D reconstructions of basal dendrites from each of the three cortical areas. The term "dataset" is used to refer to the values of the variables measured for each pair of sibling segments in those reconstructions.

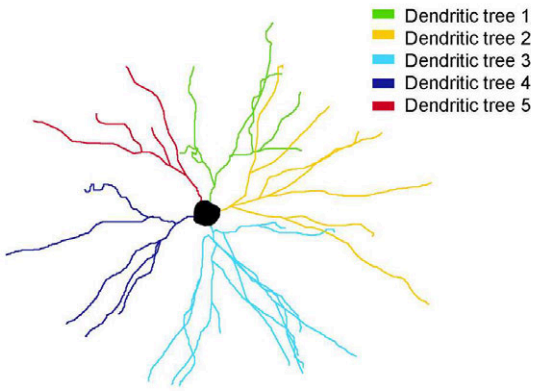


Figure 7.3: Basal dendritic arbor of a pyramidal neuron from the secondary motor cortex. Each dendritic tree is drawn in a different color.

the dendrites branched. Each dendritic tree was isolated and its coordinates were moved and rotated such that the root point was placed at the coordinate system origin and the root segment was on the vertical axis. We considered a segment as the straight line between two branch points. The dendritic trees with multifurcations (branch points that are the source of three or more segments) were discarded.

A set of 41 variables was measured for each pair of sibling segments (Table 7.1). Some of the variables were selected because they have been widely used to describe dendritic morphology [67, 257, 494, 505], whereas other new variables have been included to capture context influence and neuritic competition. Two types of variables were identified: evidence (E) and construction (C) variables. Construction variables define the morphology of a segment. In the simulation step, construction variables are sampled by the model to incrementally build the virtual dendritic trees. On the other hand, evidence variables measure the part of the dendritic tree morphology previous to a pair of sibling segments. Evidence variables are measured during the simulation process and used as information to accurately sample the construction variable values.

Evidence variables (E), which provide information about the context of the segments and how the dendritic tree is constructed (variables 1-33). These features include (Figure 7.4):

- Morphological data from the subtree (variables 1-13). Given two sibling segments with order $= x$, the subtree is defined as the part of the dendritic tree considering all the segments with order $< x$. For example, Figures 7.4(a) and (b) show a pair of sibling segments with a centrifugal order value of 5 (in red). The subtree (blue area) includes all the segments with an order value from 0 to 4. This information could be interesting for considering dendritic size when deciding if a segment should branch or to control the spread and direction of the dendritic tree.
- Variables that describe the subdendrite (variables 14-21). Given two sibling seg-

ments, the subdendrite is the path from the soma to the segments' branching point. The dotted area in Figure 7.4(a) represents the subdendrite for the two sibling segments (in red), including the segments along the path with lower centrifugal order, i.e., the segments in the path with order 0 to 4. These data could be used as a way to capture neuron tropism and determine segment direction or to bound segment length.

- Information about the nearest segment (neighbor in Figure 7.4(c)) in the dendritic tree that is not a part of the subdendrite (22-26). These variables have been inspired by previous studies about the branching patterns of trees in the field of ecology [370, 481], where several competition indexes and measures are given to model tree growth when the influence of neighboring trees is acknowledged. Following these approaches, we calculated five variables to take into account possible competition for resources between different branches, e.g., distance to the neighboring segment, angle between the subbranches of the two segments, etc.

- Variables 27-29 describe parent segment morphology and enable interactions between the three segments that are involved in a bifurcation (Figure 7.4(a)). These variables are included to consider relationships between consecutive segments.

- Root segment variables (30-32) correspond to the first segment of the dendrite that grows away from the soma (Figure 7.4(a)). These variables could also help to capture dendritic tropism. Subdendrite information, along with parent and root segment measures, have already been used to model neurite growth direction [439].

- The centrifugal order of the segment (branch order), i.e. the number of bifurcations along the path to the soma (variable 33).

Construction variables (C) completely specify segment morphology (variables 34-41). This group determines whether or not the segments branch, as well as the spherical coordinates of each end point taking the starting point as the origin (Figure 7.4(d)). A distinction between the two segments in a bifurcation was made based on the azimuth angle of their end points. We defined the left segment in a bifurcation as the one having a higher absolute azimuth value. Therefore, the right segment is the one with a smaller absolute value of the azimuth angle.

We formed four datasets by centrifugal order of segments: root segments (order 0), first-order segments (order 1), second-order segments (order 2) and segments with a higher order (order > 2). Finally, each of the 41 variables in each dataset was discretized by mimicking their histograms. Two or three discrete values were defined for each variable trying to preserve the shape of the empirical distributions while ensuring that enough data was available in each interval to accurately estimate BN parameters.

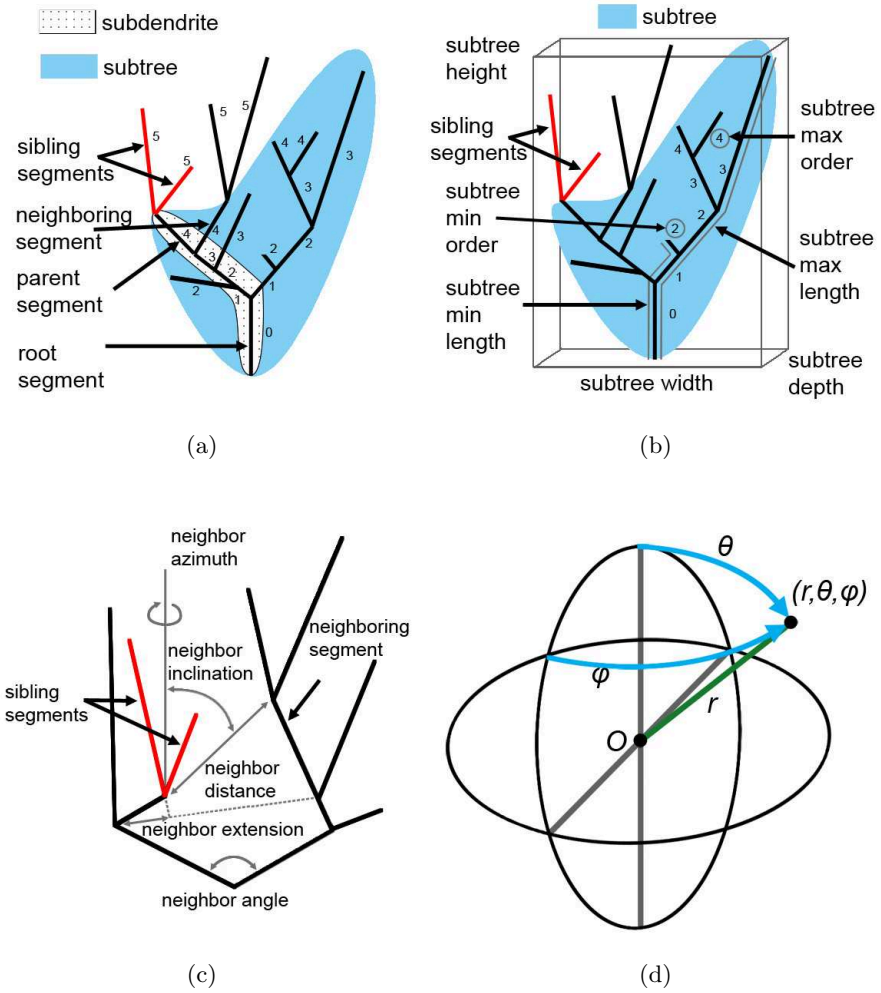


Figure 7.4: Scope of the variables used in the model. (a) Subtree (blue area), subdendrite (dotted area), parent and root segments for the two sibling segments (in red). The numbers refer to the centrifugal order of the segments. (b) Variables measured from the sibling segments subtree. (c) Variables related to the segment closest to the segment starting point (neighbor). (d) A spherical coordinate system where a segment is defined by the spherical coordinates (r, θ, φ) of its end point taking the starting point as the origin: r is the Euclidean distance between the two points, θ is the inclination angle and φ is the azimuth angle.

7.3.2 Bayesian network learning and model construction

The proposed model is made up of a BN (see Chapter 3) with discrete variables for each one of the four datasets. The BNs were learned from data using a score+search approach (see Section 3.3.2). The BIC score (Equation (3.4) on page 30) was used as a score function to evaluate the network structures. The K2 heuristic search algorithm [97] was applied to efficiently examine the space of network structures. Algorithm 3.1 on page 31 outlines the K2 procedure at a high level.

Table 7.1: Variables measured from the real dendritic trees and used for learning the model. Two types of variables are considered: evidence variables (E), which provide information about the subtree and subdendrite; and construction variables (C), which describe the segment length, orientation and bifurcation.

No.	Type	Variable	No.	Type	Variable
1	E	subtree degree (no. endings)	22	E	neighbor distance
2	E	subtree no. bifurcations (no. nodes)	23	E	neighbor inclination
3	E	subtree total length	24	E	neighbor azimuth
4	E	subtree width	25	E	neighbor extension
5	E	subtree height	26	E	neighbor angle
6	E	subtree depth	27	E	parent segment length
7	E	subtree box volume	28	E	parent segment inclination
8	E	subtree max distance between nodes	29	E	parent segment azimuth
9	E	subtree max distance to soma	30	E	root segment length
10	E	subtree max length	31	E	root segment inclination
11	E	subtree min length	32	E	root segment azimuth
12	E	subtree max order	33	E	segment centrifugal order
13	E	subtree min order	34	C	left segment length
14	E	subdendrite length	35	C	left segment inclination
15	E	subdendrite width	36	C	left segment azimuth
16	E	subdendrite height	37	C	left segment bifurcates
17	E	subdendrite depth	38	C	right/root segment length
18	E	subdendrite box volume	39	C	right/root segment inclination
19	E	subdendrite distance to soma	40	C	right/root segment azimuth
20	E	subdendrite inclination	41	C	right/root segment bifurcates
21	E	subdendrite azimuth			

The maximum number of parents (u) allowed for each variable was set to three in our experiments. K2 needs the variables in $\mathbf{X}$ to be ordered (step 1). The maximum weight spanning tree algorithm was used to compute such ordering. The centrifugal order (variable 33) was always used as the root node of the tree. Then, a topological sorting method was applied to partially order the variables. Evidence variables were forcibly placed at the top of the list, followed by the construction attributes.

Once the structures were known, the probabilistic components of the BNs were found by computing the maximum likelihood estimates of the CPTs in the nodes of the BNs. The Bayes Net Toolbox for Matlab was used to run these algorithms [383], and a BN was learned for each of the four datasets. Thus, the model comprises four BNs that capture the relationships between the variables at the different levels of the dendritic trees (see Figure 7.2).

7.3.3 Simulation algorithm for generating virtual dendritic trees

The simulation process uses the BNs included in the model to generate the virtual dendritic trees. Algorithm 7.1 shows the main steps of the procedure. The iterative algorithm measures the evidence variables and uses that information to increasingly build the virtual dendrite. The algorithm simulates the dendritic tree in a breadth-first way according to the centrifugal order (see Figure 7.2), i.e. first the root segment is created, then first order segments are generated, followed by second order segments for the previous branching segments, etc.

Algorithm 7.1 (Simulation algorithm for generating virtual dendritic trees)
Repeat while there are incomplete dendrites:

1. *Select the appropriate BN depending on the centrifugal order of the segment to be sampled.*
2. *Measure evidence variables from the dendritic tree built so far.*
3. *Discretize the variables and set their values in the BN.*
4. *Sample the construction variable values from the BN.*
5. *Transform the spherical coordinates of the segments back to Cartesian coordinates to build the segments.*
6. *If a segment bifurcates, consider that the dendrite is still incomplete. Else, the dendrite has ended.*

During the data acquisition and preparation step (Section 7.3.1), the variables were discretized and the BNs were learned with those discrete values. Therefore, the values sampled from the BNs were also discrete (step 4 in Algorithm 7.1). These discrete values had to be converted back to continuous values in order to build a virtual dendritic tree.

BN learning prevents the discretization process from making a high number of bins and ensures that enough data is available for accurately estimating the probability distributions. However, a low number of discrete values produces wide intervals with complex and heterogeneous data distributions. Therefore, it is not appropriate to use a central tendency measure as the mean or the median to convert a discrete into a continuous value. Parametric fitting of the data to some theoretical distribution was also avoided because of the high complexity of the models that work with such distributions and a low goodness of fit due to unusual data shapes.

Thus, we applied a method that samples a continuous value exploiting the original data without making any assumption about its shape. Figure 7.5 shows an example where, as the result of the sampling, the construction variable `segment_length` is given the value `short` and the parent variable `subdendrite_length` has the value `long`. The method was based on the conditional histograms of the real continuous values. For each simulated construction variable $X_i, i = 34, \ldots, 41$ that takes the discrete value x_i and whose parents values are

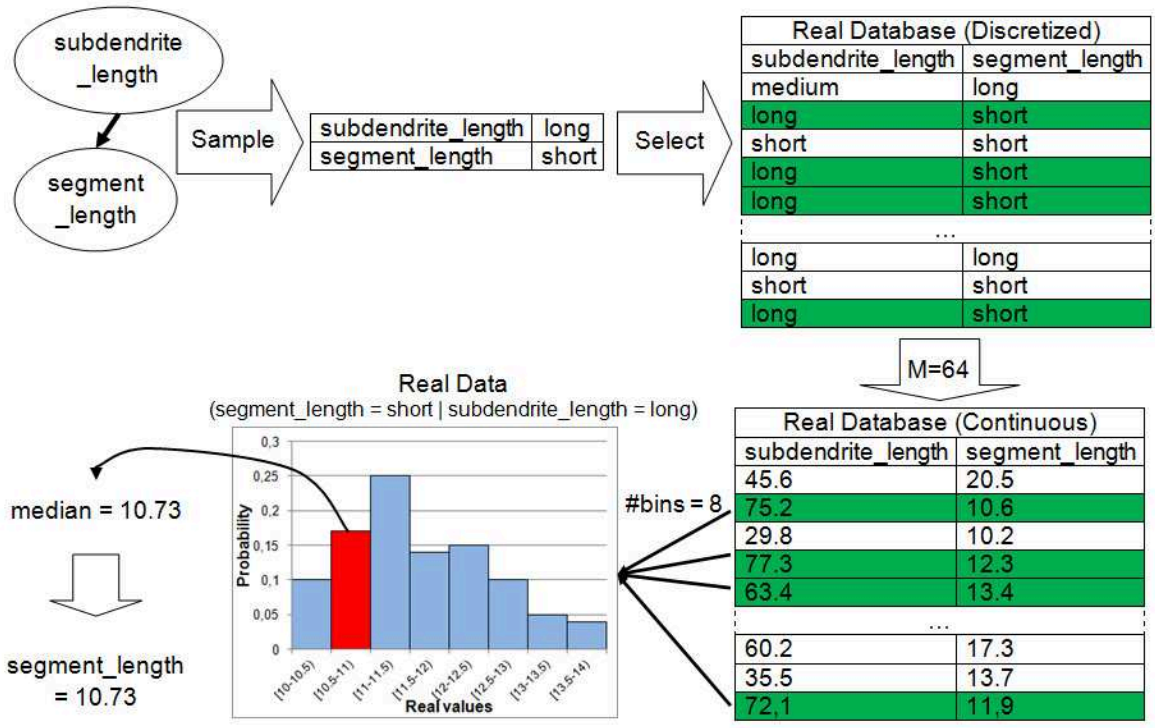


Figure 7.5: Example of the transformation of discrete values to continuous values. The conditional histogram of the real continuous values that were discretized to the values `subdendrite_length=long` and `segment_length=short` is calculated. Then, one bin is selected randomly and the median of its values is the corresponding continuous value.

$\mathbf{Pa}(X_i) = \mathbf{pa}(x_i)$, the values $(X_i = x_i, \mathbf{Pa}(X_i) = \mathbf{pa}(x_i))$ were found in the discretized real dataset. Then, the corresponding continuous real values were retrieved (M is the number of selected samples). We built a histogram with those continuous values using $\sqrt{M}$ equal-width bins. One bin was then selected randomly, where the bin probability was proportional to the number of data in the interval. The final continuous value corresponding to the discrete value x_i was the median of the data in the selected bin.

The continuous values were finally converted to the Cartesian coordinates used to simulate the virtual dendrites (step 5 in Algorithm 7.1).

7.3.4 Evaluation methodology

The last step in the methodology is to check if the virtual dendrites are statistically and visually indistinguishable from the real ones. This would mean that the model is able to accurately capture the processes underlying dendritic morphology and can be analyzed to extract relevant knowledge. Van Pelt and Uylings [497] compared real and simulated dendritic trees using both optimized parameters included in the model and predicted variables that emerged as an outcome of their growth model. Later on, Ascoli et al. [22] proposed three ways of validating a reconstruction model: comparing real and simulated probability distributions

of the variables used in the model, comparing real and simulated emergent parameters not included in the model, and performing a visual inspection by experts in neuroanatomy.

We can evaluate each variable independently using univariate statistical tests. However, since BNs model the JPD over all the variables in the problem, a method that compares this multivariate distribution in real and simulated data would be desirable. Therefore, a test based on KL that can be applied to both univariate and multivariate data was designed. KL [305] measures the "distance" from a true probability distribution $p_X(x)$ to a reference distribution $q_X(x)$. It is a frequently used technique to quantify the difference between distributions. We refer to these tests as the univariate and multivariate KL tests, respectively.

In this research, we counted the number of dendritic trees in the original database and simulated the same number of virtual dendritic trees from the model. Each run was repeated 100 times to consider statistical variability. After that, a sign test was performed for each test to check if the number of rejections was significant in the 100 repetitions. A significance level of $\alpha = 0.05$ was used for all statistical tests.

7.3.4.1 Univariate analysis

For each feature, we applied three univariate statistical tests to assess whether or not the simulated and original variables were significantly different. The three tests are non-parametric so they can be applied without making any assumption about the shape of data:

The two-sample Wilcoxon rank-sum test checks if two samples have an equal median, and can also be used to test for equal distribution of samples [167, 518].

The Kolmogorov-Smirnov test checks the hypothesis that two samples come from the same underlying distribution.

The univariate KL test uses the KL value to compare two univariate marginal probability distributions $p_X(x)$ and $q_X(x)$ on $\mathbb{R}$. The KL divergence is

$$KL(p_X, q_X) = \int_{-\infty}^{\infty} p_X(x) \log \frac{p_X(x)}{q_X(x)} dx. \tag{7.1}$$

Computing the expression in Equation (7.1) when a closed form cannot be found is not a trivial task, so we used the bioDist package [139] for R to estimate the KL for each continuous feature. The design of the statistical test that uses the divergence values is detailed below in this section.

7.3.4.2 Multivariate analysis

The above tests are univariate, i.e., they only consider each variable independently. However, a test using all the variables at the same time would be useful as we could compare the JPDs on real and simulated data. We used the multivariate KL estimator for continuous data

proposed in [510], which is based on k nearest neighbors density estimation:

$$\widehat{KL}(p_{\mathbf{X}}, q_{\mathbf{X}}) = \frac{n}{N_p} \sum_{j=1}^{N_p} \log \frac{\upsilon_{\mathcal{D}^q_{\mathbf{X}}}(j)}{\rho_{\mathcal{D}^p_{\mathbf{X}}}(j)} + \log \frac{N_q}{N_p - 1}. \tag{7.2}$$

This expression estimates the divergence between the densities $p_{\mathbf{X}}(\mathbf{x})$ and $q_{\mathbf{X}}(\mathbf{x})$ from two datasets $\mathcal{D}^p_{\mathbf{X}}$ and $\mathcal{D}^q_{\mathbf{X}}$ with n-dimensional samples of sizes N_p and N_q, respectively. The term $\rho_{\mathcal{D}^p_{\mathbf{X}}}(j)$ represents the Euclidean distance from the sample $\mathbf{x}_j \in \mathcal{D}^p_{\mathbf{X}}$ to its nearest neighbor in $\mathcal{D}^p_{\mathbf{X}} \setminus \{\mathbf{x}_j\}$. On the other hand, $\upsilon_{\mathcal{D}^q_{\mathbf{X}}}(j)$ is the Euclidean distance from $\mathbf{x}_j \in \mathcal{D}^p_{\mathbf{X}}$ to its nearest neighbor in $\mathcal{D}^q_{\mathbf{X}}$. Each feature was first scaled to the interval $[0, 1]$ to avoid the different scales in the features affecting the Euclidean distance measure.

We performed a statistical test based on a bootstrap method [157] that uses the KL to check whether or not the simulated and real distributions are different. If the real dataset $\mathcal{D}^r_{\mathbf{X}}$ contains N samples, two datasets $\mathcal{D}^{r1}_{\mathbf{X}}$ and $\mathcal{D}^{r2}_{\mathbf{X}}$ of size N are sampled with replacement from $\mathcal{D}^r_{\mathbf{X}}$, and both univariate (Equation (7.1)) and multivariate (Equation (7.2)) KL values are calculated. This process was repeated 100 times and the 95-percentile was stored as a threshold for each set of KL values. When applying the KL-based test, the divergence value between the simulated and real dataset had to be higher than the threshold for the null hypothesis stating distributions are equal to be rejected. A sign-test was applied to check if the number of rejections was significant.

Finally, the model can be visually validated by comparing the pictures of real and virtual dendritic trees. Since it is not possible to portray 3D data through 2D projections, a website has been set up with examples of real and simulated dendritic trees (see Section 7.4.5).

7.4 Results

This section examines simulation results with different databases of basal dendritic trees. We established three databases of neurons according to their cortical area, and each dendritic tree was considered independently. Table 7.2 shows the number of dendritic trees included in each database. This section illustrates the statistical comparison for the three areas. The analysis of the BN structures focused on the M2 area, since the basal trees in this area have more complex branching patterns and dimensions [39]. The BN structures for the S2 and V2L/TeA areas are available at the supplementary results webpage (see Section 7.4.5).

These experiments were run on an Intel Core2 Quad CPU at 2.49Ghz with 6GB RAM. The algorithms were implemented in Matlab under Windows Vista. Table 7.3 shows runtime intervals for each algorithm, i.e., the minimum and maximum runtime in the 12 runs (4 BNs for each one of the three cortical areas).

7.4.1 Analysis of Bayesian networks

The BN structure learned from the data encodes the conditional (in)dependence relationships between the variables in the problem. The model can be validated by verifying if those

Table 7.2: Number of trees included in each database according to the cortical region from where the neurons were sampled.

Region	Database	#Dendritic trees
Motor cortex	M2	104
Somatosensory cortex	S2	103
Lateral visual and association temporal cortex	V2L/TeA	156

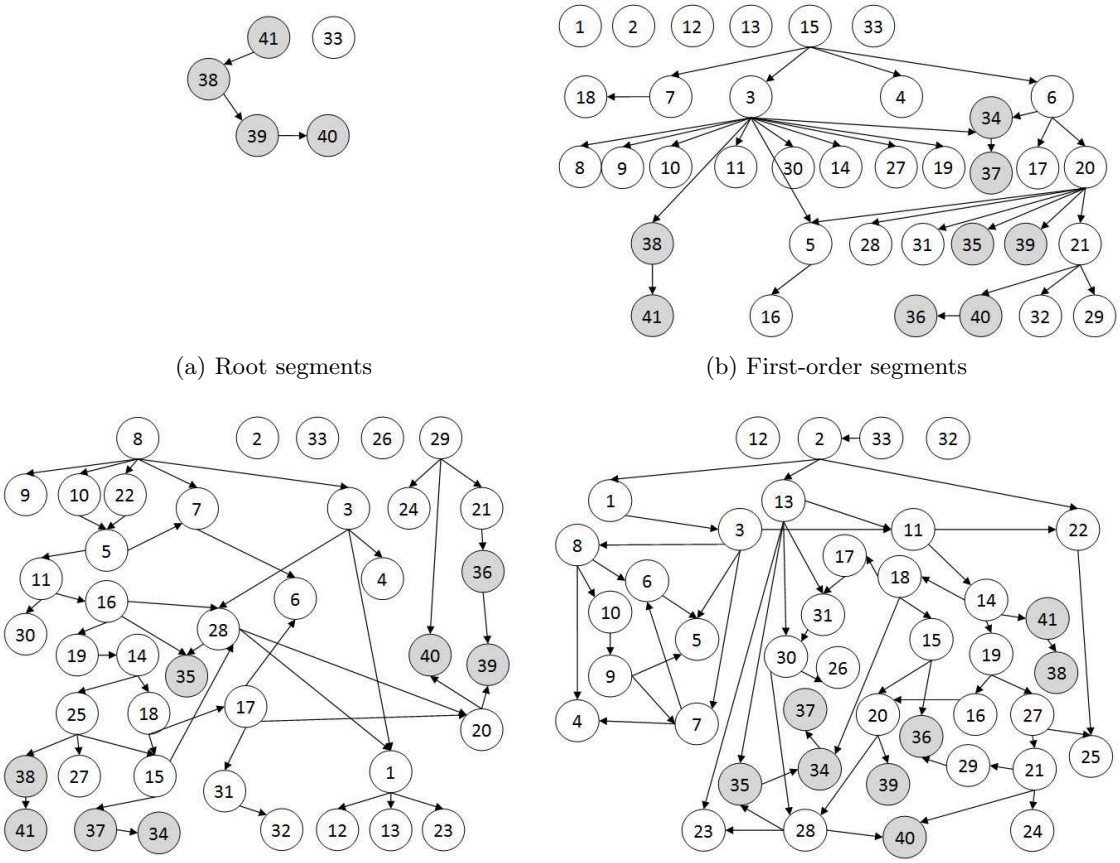
Table 7.3: Runtimes of the different algorithms implemented in Section 7.3.

Algorithm	Runtime (seconds)
MWST algorithm	3.02 - 4.52
K2 search algorithm	4.29 - 9.76
Simulation algorithm (100 virtual dendritic trees)	1.53 - 3.86

relationships conform to current biological knowledge. On the other hand, a thorough analysis of the relationships could help to discover new factors involved in neuron development and dendritic morphology. Figure 7.6 shows the structure of the four BNs built from the M2 database, corresponding to the root segments, first-order segments, second-order segments and the other segments with higher centrifugal order. As described earlier (see Section 7.3.2), the ordering established during BN learning ensured that the construction variables (shaded) were always influenced by either evidence variables or other construction variables. On the other hand, evidence variables (with a white background) could only be influenced by other evidence variables.

In the first BN (Figure 7.6(a)) only four construction variables and the centrifugal order evidence variable are shown. This BN was used to generate the root segment of the dendrite. There were no evidence variables to measure because no dendritic segment had yet been simulated. At this level, only one segment (the root segment) has to be sampled, and it is not possible to distinguish between left and right segments. Therefore, we decided to use variables 38-41 to encode the root segment morphology. Variables 38-41 refer to the right segment in the other BN structures (Figures 7.6(b), (c) and (d)). In the BN for first-order segments (Figure 7.6(b)) only variables 22 to 26 were unavailable. Those variables could not be measured at this level because the neighboring segment could not be defined, since the root segment was the only segment that had so far been simulated. On the other hand, the other two structures (Figures 7.6(c) and (d)) contained all the variables in Table 7.1 because the simulated part of the dendritic tree was complex enough to measure all the evidence features.

At first sight, it is clear that network complexity grew as we considered segments of a higher centrifugal order. For example, the number of relationships (line plot in Figure 7.7) steadily increased from three relationships in the first to 55 relationships in the last BN. When we counted the number of variables with 0 to 3 parents (bars in Figure 7.7), we found that variables with a higher number of parents were also more frequent in the higher-order networks. Since 0 and first-order BNs did not include all the variables, fewer relationships



(a) Root segments

(b) First-order segments

(c) Second-order segments

(d) Segments with higher order

Figure 7.6: Structure of the four BNs learned from the M2 database. The numbers in the nodes refer to the variables in Table 7.1. Shaded nodes represent construction variables and evidence variables are shown on a white background.

were expected. However, the complexity growth was higher than the number of variables added, and it also increased in the last two networks, including all the variables. The subtree from where the evidence attributes were measured grows and gets more complex as we consider high-order segments. Thus, a more widely connected network was necessary to model the relationships in the dendritic tree structure. Besides, the last BN was learned for segments with different centrifugal order and a higher variability in the data was observed. The BN needs more information to model this variability. This is a possible explanation for why variables had more parents.

One way to distinguish between global and centrifugal order-specific relationships is to count the number of times two variables are connected in the different networks, without taking into account the direction of the relationship. Frequent edges could represent order-independent global relationships and scarce edges could encode context-specific interactions that only appear at a certain level. Additionally, the conditional probability distributions in each variable (not shown) could be analyzed to check how the variable values change depend-

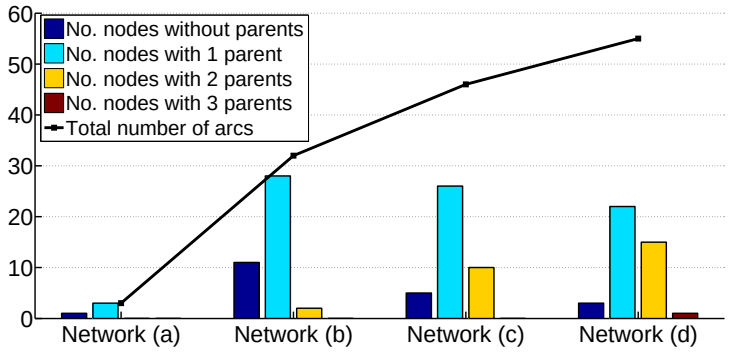


Figure 7.7: BNs complexity analysis. The line plot shows the number of arcs in each BN from Figure 7.6. The bars represent the number of variables with different numbers of parents in each network.

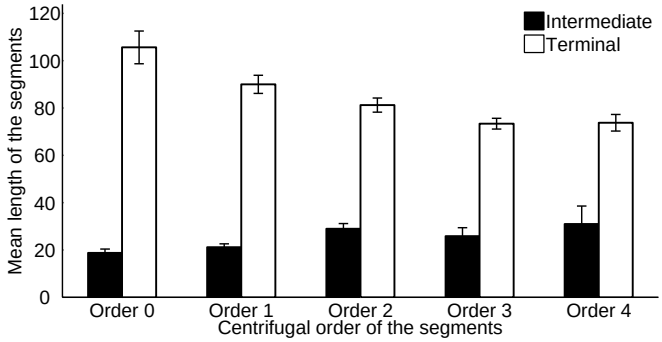


Figure 7.8: Mean length of intermediate and terminal segments of real dendrites. The figure shows the mean length of the intermediate (black) and terminal (white) segments of each centrifugal order in real basal dendrites of pyramidal neurons from the M2 area. Differences between intermediate and terminal segments can be identified.

ing on the parent values. The only edge that appeared in all BNs (Figure 7.6) was the one that related segment length and bifurcation occurrence for both the left (variables 34 and 37) and the right segments (variables 38 and 41) in a bifurcation. This relationship encoded the knowledge that terminal segments in basal dendrites were longer than intermediate segments. In fact, dendrites usually branch when they are close to the soma, producing short segments; whereas the segments that do not branch spread away from the soma. Figure 7.8 confirms this relationship, as there is a clear difference between terminal and intermediate segments at each branch order.

Segment angles also exhibited consistent relationships in the different BNs. Segment azimuth (variables 36 and 40) was related in different ways to the subdendrite azimuth (variable 21) and the parent segment azimuth (variable 29). Subdendrite inclination (variable 20) and parent segment inclination (variable 28) also influenced segment inclination (variables 35 and 39). Parent and subdendrite angles (variables 28-29 and 20-21) were frequently interrelated too. Samsonovich and Ascoli [439] used both parent and subdendrite vectors

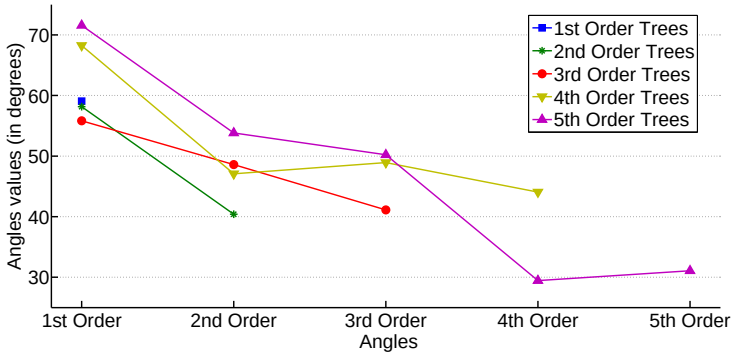


Figure 7.9: Angles formed between sibling segments. Mean values of the angles formed between the two sibling segments in a bifurcation for real M2 basal dendrites. The dendritic trees are grouped by their maximum centrifugal order. Each data point shows the mean value of the angles (in degrees) between segments at each bifurcation node.

to successfully model dendritic tropism. They hypothesize that dendritic guidance might be controlled by the host cell rather than by exclusively external factors, so our model found relationships that had already been shown to be relevant. We show that segment orientation is mainly controlled by the orientation of the previous segments, reflecting how the dendrites extend away from the soma without making any sharp change of direction. We believe that the fact that our model simultaneously considers and makes a distinction between the two sibling segments at a bifurcation is possibly a key strength. Segment angles in a bifurcation are assumed to be related since they do not grow in the same direction. Thus, it could be important to consider these interactions when modeling dendritic orientation, e.g., the azimuth and inclination of different segments were related (arc from 36 to 39 in Figure 7.6(c)).

Other subdendrite measurements can also directly influence some construction variables. In Figure 7.6(c), the width component of the subdendrite (variable 15) influenced segment bifurcation (variable 37), preventing it from splitting when this parameter showed high values. This helped to constrain tree size and could be related to resource division and competition between branches. In the above BN again, inclination angles (variable 35) were more likely to be small when subdendrite height (variable 16) took higher values. This makes the dendrite grow straight in one direction instead of the branches spreading when the segments are far from the soma and helps to model dendritic tropism. On the other hand, the subdendrite width was used to regulate segment azimuth (variable 36) and control the dendritic tree spread in the next network structure (Figure 7.6(d)). Therefore, the dendritic trees tend to first spread rapidly when they are close to the soma and then, once they have reached a minimum size, grow straight away from the soma. Figure 7.9 shows two interesting facts that support these explanations. First, when considering bifurcations of the same order, complex dendritic trees with a high maximum centrifugal order (fourth and fifth order trees) have wider angles than less complex trees (first, second and third order trees). Second, when considering dendritic trees of the same complexity, the mean angle between sibling segments steadily decreases as we consider higher order bifurcations. Therefore, sibling segments grow

to fill up the area defined by the angle of the first-order segment.

Subdendrite length also bounded dendritic size (arcs 14 → 41 in Figure 7.6(d) and 3 → 38 in Figure 7.6(b)). Note that variable 3 equaled variable 14 in Figure 7.6(b) since only the root segment had been simulated. When the subdendrite is short, the simulation is still near the soma and more bifurcations are supposed to occur. On the other hand, a long subdendrite tends to stop the simulation by ending the dendrite with a long terminal segment.

In general, information related to the subdendrite appears to be more important for the simulation process than the whole subtree measurements because construction variables were almost always related to subdendrite variables. Summing up, segment orientation was defined by the orientation of previous segments (subdendrite) to ensure that the dendrites showed no angles causing a rough change of direction. On the other hand, the length of the path to the soma was used to control segment size and whether or not it bifurcated, bounding the total length of the dendrites. Although this could lead to the understanding that dendrites grow independently from each other, influences from other branches could be important for modeling competition and resource availability factors.

7.4.2 Evaluation of features used in the model

We compared the original and simulated values of the features included in the BNs to check if the model accurately reproduces those variables. Figure 7.10 shows the results in the three areas for the datasets compared using the multivariate KL-based test designed in Section 7.3.4. At least 61 rejections were considered to be significant, as this corresponded to the rejection region limit of a sign test with 100 observations. In the three cortical areas, a significant number of rejections was found when comparing the real and simulated values in the last dataset. This could be caused by its higher data variability. This dataset was the only one that contained segments with different centrifugal order (from three to five) so the subtrees and subdendrites where evidence features were measured are also more diverse.

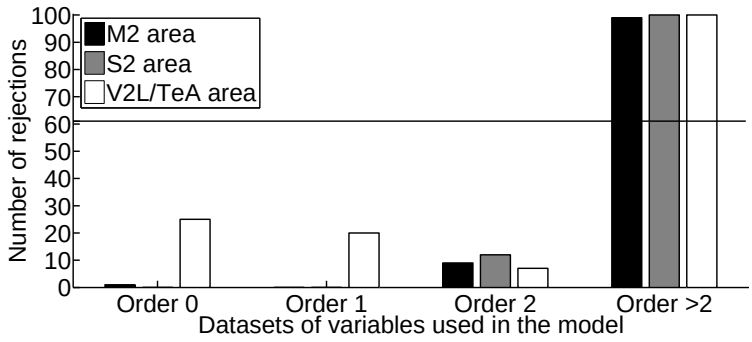


Figure 7.10: Multivariate comparison of variables used in the models. The number of rejections in the KL-based multivariate test is shown for each of the four datasets according to the centrifugal order of the segments. The horizontal line represents the threshold (61 rejections) for rejections to be considered significant.

The univariate comparison of features could help to analyze which variables were causing such a result. The number of rejections in the three univariate tests are summarized in Figure 7.11(a) for the M2 database. Only 14 variables out of the 123 variables (11.38%) used in the four BNs had a significant number of rejections in at least one of the tests: the rejected variables correspond to variables 5, 6, 7, 8, 9, 10, 11, 14, 16, 17, 18, 19, 26 and 30 in Table 7.1. If we only count the variables where at least two tests gave a significant number of rejections, nine variables are selected: 9, 10, 11, 14, 16, 18, 19, 26 and 30. All those variables belong to the fourth dataset, so they might have decreased the performance of the whole dataset when multivariate comparison was applied (Figure 7.10).

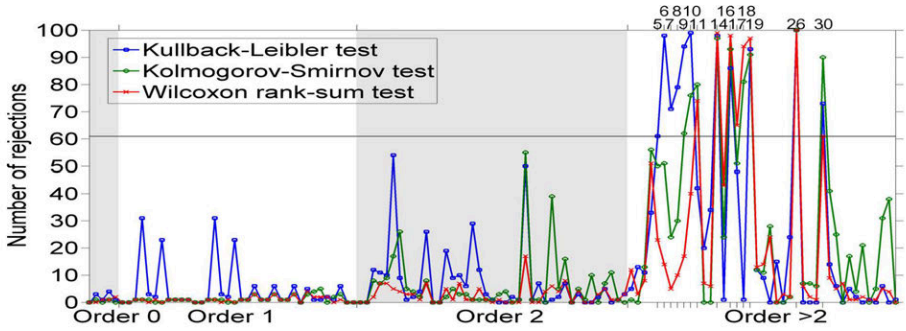
The variables with rejections were evidence features, mostly from the subtree and the subdendrite. As previously mentioned, this dataset includes information from different orders, and the higher variability could be an obstacle to their simulation. We tried to decrease this variability by further dividing this dataset according to the centrifugal order of the segments. We created a dataset for each order, and learned as many BNs as the maximum order value. However, this was not a feasible solution due to the low number of higher order segments, as there has to be enough data to accurately estimate the probabilities included in the BNs.

Despite the rejections in evidence variables, the number of rejections for construction variables was never significant. In fact, only two of the rejected variables, i.e., subdendrite length (14) and subdendrite box volume (18), were directly related to any construction variable in Figure 7.6(d). Moreover, the simulated dendritic trees with a maximum order value greater than two were indistinguishable from real ones (see the following Section 7.4.3). This means that the model was robust against false or heterogeneous evidence and was still able to simulate correct values for the variables used to build the virtual dendrites.

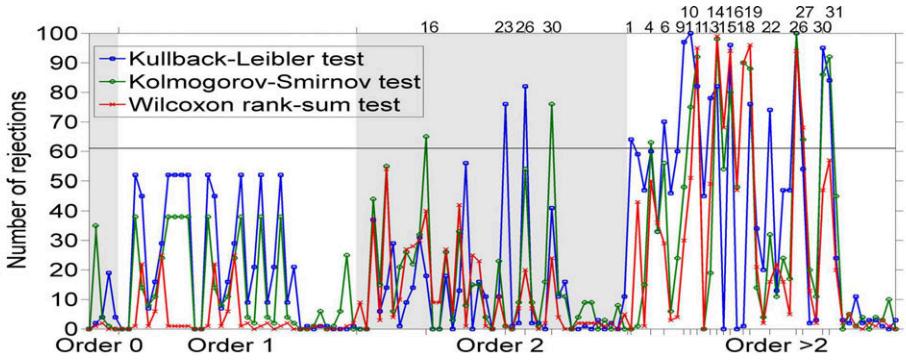
The results for the S2 area were quite similar (Figure 7.11(b)), where 21 of 123 variables (17.07%) had a significant amount of rejections. Most of these rejections appeared in the last dataset, like M2 results. On the other hand, the tests in the V2L/TeA area (Figure 7.11(c)) gave 20 variables with significant rejections (16.26%). Note, however, that in the first-order dataset, the subdendrite, subbranch, parent segment and root segment refer to the same information, since only one segment has been simulated. Thus, rejections for variables 21, 29 and 32 in that dataset were interrelated and also related to the azimuth rejections in the zero order dataset (variable 40). Similarly, pairs of variables 4-15, 5-16 and 7-18 represent the same values. Moreover, when we analyzed the meaning of these variables, we could see that they were related to the azimuth value that was rejected in the root segments dataset, i.e. width, height and box volume closely depend on the azimuth value. Azimuth angle is defined in a circular domain so it is a difficult feature to model, specially when considering discrete values, and could be contributing to the rejections. However, it was the only case in which a construction variable had a significant number of rejections.

7.4.3 Comparison of emergent parameters not used in the model

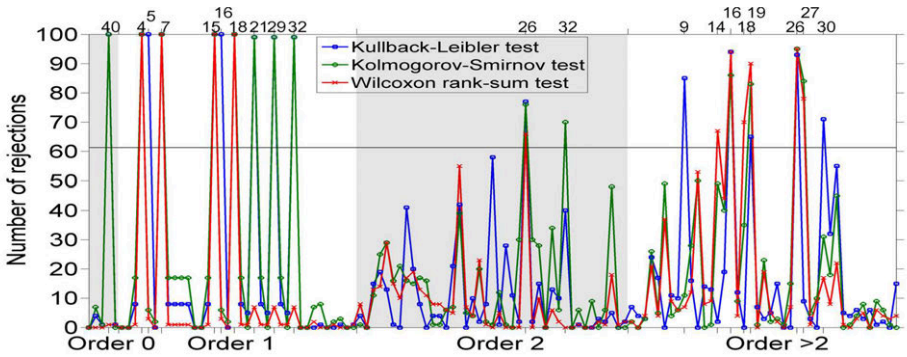
A set of 12 new variables was measured from the complete real and simulated dendritic trees (see Table 7.4). Ascoli et al. [22] proposed the term “emergent parameter” to refer to these



(a) M2 area



(b) S2 area



(c) V2L/TeA area

Figure 7.11: Univariate comparison of variables used in the model. The number of rejections for the KL-based test (blue squares), Kolmogorov-Smirnov test (green circles) and Wilcoxon rank-sum test (red crosses) are shown. The horizontal axis represents the variables in each BN ordered according to their number in Table 7.1. The numbers at the top of the graph indicate the rejected variables. At least 61 rejections are considered to be significant as indicated by the horizontal line.

global variables not included in the model that can be used to describe and compare the dendritic morphology. Model variables and "predicted variables" were previously compared in [497].

In this work, the emergent parameters are variables that can be measured at the level of the whole tree, as opposed to subdendrite or subtree variables, which are measured from

Table 7.4: Emergent parameters not included in the model. The values of these features are used to describe and compare the morphology of the whole dendritic trees.

No.	Variable	No.	Variable
1	degree (no. endings)	7	dendritic tree depth
2	no. of bifurcations	8	box volume
3	total length	9	max distance between nodes
4	mean asymmetry index	10	max distance to soma
5	dendritic tree width	11	max centrifugal order
6	dendritic tree height	12	min centrifugal order

a part of the tree. First, these emergent parameters are measured from the real dendritic trees. Then, the simulation step generates a set of virtual dendritic trees. Finally, the emergent parameters are calculated for the virtual dendritic trees, and both the multivariate and univariate statistical analyses are conducted again.

The number of rejections when all the real and virtual dendritic trees were tested was very high in the three areas. We hypothesized that it might be wrong to consider all the dendrites together since we could find a high diversity of dendritic morphologies. For example, comparing a dendritic tree that extends away from the soma without branching (has only a root segment) with a complex dendritic tree that branches extensively is expected to return a high number of mismatches. To check that statement we repeated the analyses comparing only real and virtual trees with the same maximum centrifugal order. The bars in Figure 7.12 show a low number of rejections when we compared these subgroups of dendritic trees, suggesting that simulated dendrites were in fact similar to real ones.

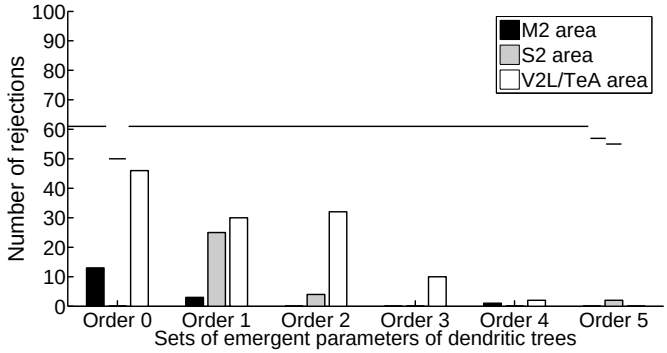
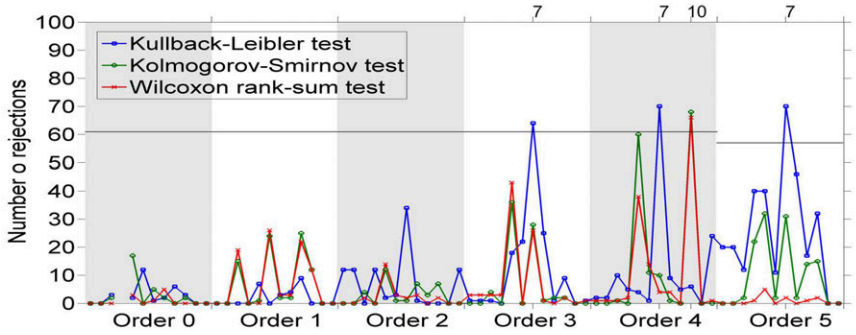
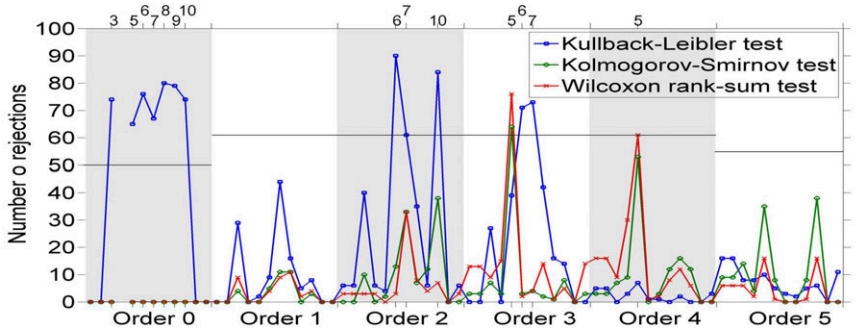


Figure 7.12: Multivariate comparison of emergent parameters not included in the model. The number of rejections in the KL-based multivariate test is shown when only trees with the same maximum centrifugal order are compared. The horizontal line represents the threshold as of which the rejections are considered significant.

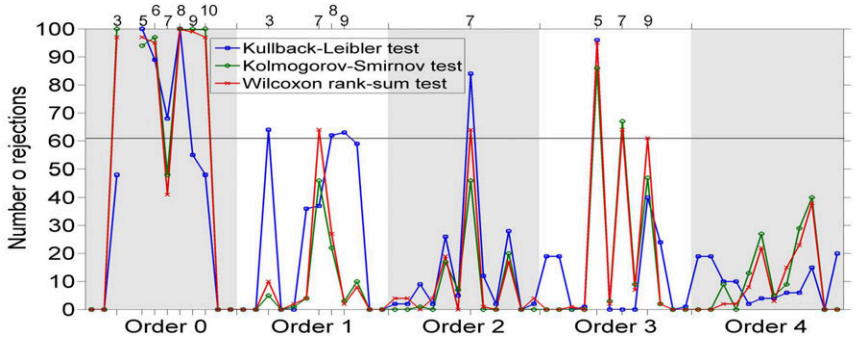
The horizontal line in Figure 7.12 shows the rejection threshold (61 in a sign test with 100 observations). However, there were some runs in the 100 repetitions where no dendritic trees with a given centrifugal order value were simulated. In the M2 area, only 94 repetitions generated dendritic trees with order 5 and the rejection threshold was decreased to 57. In



(a) M2 area



(b) S2 area



(c) V2L/TeA area

Figure 7.13: Univariate comparison of emergent parameters not used in the model. Results for 12 emergent parameters (Table 7.4) using univariate tests: KL-based (blue squares), Kolmogorov-Smirnov (green circles) and Wilcoxon rank-sum (red crosses). This figure shows the number of rejections when the dendritic trees are grouped according to their maximum centrifugal order. The horizontal line indicates the number as of which rejections are considered significant.

the S2 area, dendritic trees with a maximum centrifugal order value of 0 were simulated in 81 repetitions and trees with order 5 were generated in 90 repetitions. Therefore, the rejection threshold was set to 50 and 55, respectively. The breaks in the horizontal line in Figure 7.12 (and Figure 7.13) show the above changes in the rejection threshold.

The analysis of each feature independently (Figure 7.13) showed that the number of rejections also decreased when grouping the dendritic trees by their centrifugal order. The

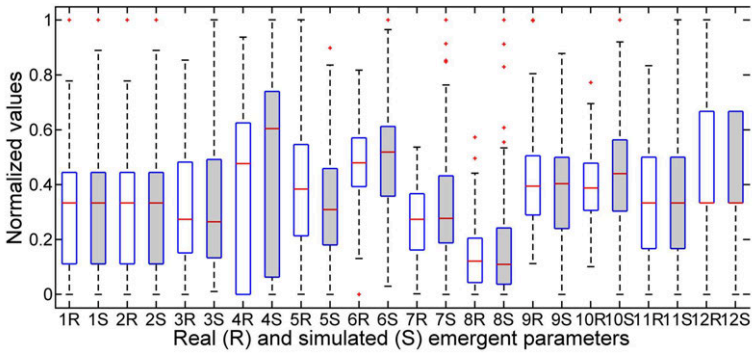


Figure 7.14: Boxplots for the real and simulated values of the emergent parameters. The boxes show the values when all the dendritic trees from the M2 area are compared and no subgroups are considered. The numbers refer to the variables in Table 7.4, whereas R and S stand for real (with white background) and simulated (shaded) data, respectively.

most rejected variable in the three areas and in the different groups was the dendritic tree depth (variable 7 in Table 7.4). Dendritic depth is highly dependent on the resolution of the z-dimension when tracing the neurons, which is lower than the x and y dimensions. Thus, the z-dimension measurements are a common source of uncertainty and errors [473] that could also be limiting the model's ability to accurately capture dendritic depth.

In the M2 database, dendritic depth (7) and maximum Euclidean distance from the soma to the terminal tips (10) were the only variables with a significant number of rejections (Figure 7.13(a)). Although there were more rejections for these two features than for the other features in the three tests, only the KL-based test yielded a significant number of rejections. Figure 7.14 shows that, while the median value in real and simulated data was quite similar, the whiskers in the simulated data were wider, and the number of outliers was also higher. These mismatches could be due to the discretization process, since it might attach a higher probability to some infrequent data.

In the S2 and V2L/TeA areas (Figures 7.13(b) and (c)), a high number of rejections were identified in zero-order dendrites. There were very few such dendrites in the S2 area, where only 3 out of 103 dendritic trees were selected (2.91%), so the tests might be influenced by this shortage of data. On the other hand, zero-order dendrites were more frequent in the V2L/TeA area (27 out of 156, 17.31%). As previously noted in the evaluation of the model variables, the difficulty of modeling root segment azimuth in this area might be causing these rejections.

7.4.4 Visual comparison

A visual comparison between real and simulated dendrites (Figure 7.15) confirms that the model is able to generate virtual dendrites that show some of the defining features of pyramidal basal dendritic trees, e.g., long terminal segments or more frequent bifurcations at closer distances from the soma. 3D morphologies can be accessed (see Section 7.4.5) to further

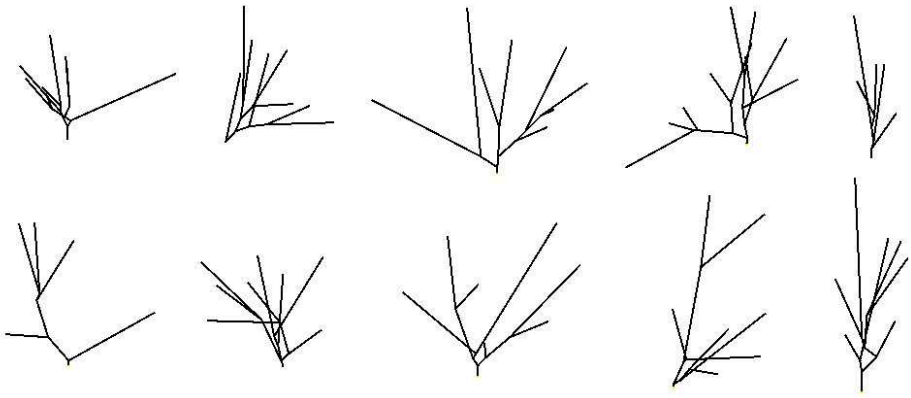


Figure 7.15: Examples of real and virtual dendritic trees. The figure shows 2D projections of real (top) and simulated (bottom) main trunks of basal dendrites from M2 pyramidal neurons. Virtual dendrites are similar to the real ones and show some distinctive traits of basal dendritic trees from pyramidal cells.

check this correspondence and allow for a qualitative validation of the simulated dendritic trees.

7.4.5 Supplementary results

A website[2] has been set up containing all the information about the experiments reported in this chapter. The BNs learned for the different cortical areas (M2, S2 and V2L/TeA) and the results of the statistical evaluation of the models are available. An analysis of the stability of the network structures obtained by the learning algorithm and the study of the sample size are included. The results of the tests performed to assess the quality of the real and simulated data and to evaluate the multivariate KL test can also be accessed. Examples of the 3D dendritic morphologies of real and virtual dendrites from the three areas can be explored online using an applet based on CVAPP software [74]. The code has mainly been developed using Matlab.

7.5 Conclusion

This chapter has presented a new reconstruction approach for the simulation of 3D dendritic morphology. The methodology uses BNs to capture the interactions between the variables in the problem domain. A complete set of evidence and construction variables is measured from the dendrites, and a learning algorithm is applied to find the structure and estimate the probability distributions included in the BNs. Then, a simulation algorithm is used to build the virtual dendrites by sampling values from the BNs, and a thorough evaluation is performed to show the model's ability to generate realistic dendrites.

[2] Available at: http://www.dia.fi.upm.es/~concha/dendriticsimulation/

This proposal has an important advantage over previous research because the relationships between the variables in the model are found by directly mining the data. Therefore, it is not necessary to previously specify all the interactions between the morphological features. Instead, we have a computer algorithm find the model that best fits the data. Applying machine learning algorithms to discover dependencies between parameters and simulate virtual morphologies is useful when complete information on the structure of the problem is not available. Since the model is learned from the data, the methodology can be applied to any neuronal class without the need for any modification. We should note that the model presented here is implicitly conditioned by the centrifugal order of the segments. The datasets are split during the data acquisition and preparation step. This means that different BNs (different relationships and probability distributions) are learned for different orders. This is a commonly used feature in the literature, e.g., see [17, 134, 145]. However, once the datasets have been created, the BN learning process is completely data-driven.

Other reconstruction models either do not explicitly include relationships between variables, which is an overly simplistic approach, or include predefined relationships, e.g., see [17, 145, 298, 334]. However, it is very difficult to find common relationships that apply to every neuronal class, and, if this is possible, their validity would have to be checked every time a new class is modeled. On the other hand, if a model has explicit relationships to accurately represent a neuronal class, changes will have to be made to fit the model to other classes that do not comply with those hypotheses. The methodology proposed here is generally applicable, whereas the models are data specific. Common interactions among neuronal classes can be identified by looking for similarities between the different models, whereas rare relationships could capture unique features in a given class. This approach addresses knowledge extraction from the models in a more straightforward way.

Furthermore, although reconstructions of neurons from diverse classes are available online in several public databases [23], the high tracing process heterogeneity between labs and researchers can negatively affect the model's behavior [20]. In our study, we have used data from basal dendritic arbors of pyramidal cells from the same laboratory. Additionally, they are fully reconstructed basal dendritic arbors, so they are particularly valuable for validating simulated virtual neurons. On the other hand, segment diameter is commonly used in the literature to determine segment length or branching probability, as it is highly correlated with structural factors like the microtubule density [257]. Since measuring segment diameter is prone to errors and noise [473], diameter values were not considered in the reconstruction of the real neurons; and, consequently, have not been included in our simulation model. All these factors will have to be taken into account when we study dendrites from other neuronal classes traced in different labs.

Our model comprises four BNs that encode the relationships between the variables at different levels of the dendritic tree. A number of stages have been described during neuron development, and different parts might have different morphologies and be regulated by different factors [144]. Making a distinction between segments at different centrifugal orders is useful for identifying context-specific interactions. In fact, the BNs found for the different

levels are quite different (see Section 7.4.1). Bayesian multinets [213], a generalization of BNs where different relationships can be considered depending on the values of some variables are also worth investigating for this problem. These models can identify the changes in the problem structure for different parts of the dendrites without the need for more than one BN.

The analysis of the BNs showed some interesting results. Segment length is always related to the branching probability, indicating a statistical difference between intermediate and terminal segments [318, 495, 501]. In fact, some of the models in previous works separate root, intermediate and terminal segments when modeling or evaluating segment length, e.g., see [17, 334, 491]. In our case, that distinction was inferred automatically from the data by the learning algorithm and not predefined on the model. Segment orientation is mainly controlled by parent segment and subbranch azimuth and inclination angles to successfully capture dendritic tropism. Other relationships with subbranch measurements were found, supporting the idea that host cell influence might be more important in neuron development and neurite outgrowth than expected. It is important to keep in mind that the pattern of dendritic arborization is the result of a complex interaction between intrinsic genetic programs and external modulators, e.g., neurotransmitters or patterns of activity [29, 38, 92, 292, 515]. Thus, adding variables to model some environmental factors and checking the relationships that are established with them will help to clarify this issue [491]. If relationships between construction variables and subdendrite or subtree evidence variables still occur in the model when environmental features are included, it will suggest that intrinsic factors are relevant and cannot be ignored.

The results were fully evaluated to verify that the virtual dendrites were accurate. Univariate and multivariate analyses were conducted to compare model variables and emergent features. A statistical test using KL was designed based on a bootstrapping method to compare both univariate and JPDs over the variables. Applying several statistical tests proved to be useful because some of the mismatches were only detected by one method. The univariate KL test was applied along with Kolmogorov-Smirnov test and Wilcoxon rank-sum test to check if the values of the real and virtual dendrites came from the same distribution. On the other hand, the multivariate KL test allows for a comparison of the JPD over all the features at the same time. As far as we know, this is the first time that such a multivariate test has been used to evaluate the JPD over a set of variables that describe dendritic morphology. Usually, univariate statistical tests that compare each variable independently are used [334, 491], or plots are visually inspected to evaluate bivariate or conditional densities [144, 298]. More complex evaluation methods are still needed in order to accurately compare dendritic morphology. Measures for quantifying dendritic branching patterns and tree structures are rather simple and not very informative, although some efforts are being made to propose such advanced measures, as in [255].

The analyses of the robustness of the network structures [200] and the minimum sample size that guarantees that there is a high probability of the learned and the true distributions being close to each other [197] confirmed the stability of the models (see Section 7.4.5).

Future work will focus on using BNs that can directly manage continuous features without the need for discretization. Hybrid BNs, which include both discrete and continuous variables, usually impose some assumptions on the shape of the data, e.g., CLG networks (see Section 3.3.1.3). However, our data may not fit any of these distributions, so non-parametric alternatives will also be considered, e.g., mixtures of polynomials. Learning and simulating from this kind of models are difficult and interesting tasks. Therefore, in Chapter 8 we propose a method for learning mixtures of polynomials from data that can be used in hybrid BNs.

Chapter 8

Learning mixtures of polynomials from data

8.1 Introduction

In real life problems, continuous data may not fit any standard parametric distribution. Therefore, the assumption of a parametric shape might yield misleading conclusions or results. Non-parametric density estimation is used to avoid the parametric assumptions in probabilistic modeling and reasoning [218, 462]. Non-parametric estimation techniques can be classified in four categories [201]: histograms, orthogonal series, kernels and splines. Histograms are based on transforming the continuous data into discrete data. Discretization is one of the most widely used approaches for data transformation, and a large number of discretization techniques have been proposed in the literature [208, 527]. However, important information can be lost during the discretization process. A different approach approximates probability densities by an orthogonal series expansion using, e.g., Hermite, Fourier or trigonometric orthonormal systems of functions. Their main drawback is that the resulting estimate is frequently not a proper density (non-negative and integrating to one). Recently, KDE has received a lot of attention because it provides a flexible and powerful tool for non-parametric density estimation. However, KDE has to save and analyze the complete training dataset to evaluate the density of each data point. Also, bandwidth selection for KDE can be challenging and a lot of different approaches have been proposed for finding bandwidth values that accurately model the data without overfitting [87, 283]. Finally, splines are piecewise polynomial curves frequently used for approximating arbitrary functions.

Also, we saw in Section 3.3.1.3 that hybrid BNs pose a number of challenges regarding the representation of conditional probability distributions, inference, learning from data, etc. CLG networks model the variables using CLG distributions and impose some restrictions on the network structure. However, the parametric assumption of Gaussian distributions in CLG networks might not hold in real domains. Also, the constraints on the network structure might limit their applicability in some problems.

Here, we focus on a family of related models including mixtures of truncated exponentials (MTEs) [376], mixtures of polynomials (MoPs) [458] and mixtures of truncated basis functions (MoTBFs) [315]. Given a continuous random variable X with a probability density function $f_X(x)$, the goal is to find an approximation of $f_X(x)$ over a closed domain $\Omega_X \subset \mathbb{R}$. Additionally, the domain of approximation Ω_X can be divided into subintervals, and $f_X(x)$ is approximated (piecewise) in each subinterval. MoTBFs approximate $f_X(x)$ as a (piecewise) linear combination of truncated basis functions. MTEs and MoPs are particular scenarios of MoTBFs when using exponential or polynomial functions, respectively, as basis functions. MoTBFs, MTEs and MoPs are closed under multiplication, addition and integration. Therefore, exact probabilistic inference can be performed using the Shenoy-Shafer architecture [457].

In this chapter, we present a method for learning MoP approximations of one-dimensional, multidimensional and conditional probability densities directly from data. The proposed method is based on a probability density estimation technique [540, 541] that uses basis spline (B-spline) interpolation [441]. Then, MoPs are used as a non-parametric density estimation technique in BNCs. We study the use of MoPs for density estimation in two of the best-known BNCs analyzed in Section 3.4, i.e., the NB classifier [372] and the TAN classifier [199]. The proposed BNCs using MoPs as a non-parametric density estimation technique are compared to other BNCs using CLG networks [408], KDE [409] or discretization.

The research included in this chapter has been published in López-Cruz et al. [339, 343].

Chapter outline

The remainder of the chapter is organized as follows. Section 8.2 reviews the methods reported in the literature for learning MoPs, MTEs and MoTBFs. Section 8.3 introduces notation and definitions for MoPs. Section 8.4 details the proposed method for learning MoP approximations of probability densities from data. Section 8.5 reports the experimental evaluation of the proposed methods. Finally, Section 8.6 ends with some conclusions.

8.2 Related work

Different methods have been proposed for approximating with MTEs. Cobb et al. [93] provided MTE approximations of seven standard parametric probability density functions. Rumí et al. [435] proposed an iterative least squares algorithm for learning MTE approximations of one-dimensional and conditional probability densities from data. This approach used exponential regression and empirical histograms for density estimation. Romero et al. [432] enhanced the algorithm by applying a Gaussian kernel smoothing of the probability densities obtained with the empirical histograms. Langseth et al. [314] provided a ML estimation approach for MTE approximations.

Regarding MoTBFs, Langseth et al. [315] proposed a method for finding approximations of one-dimensional and conditional densities by minimizing the KL from the MoTBF to the

true distribution. Recently, the KL-based approach was combined with KDE techniques to approximate MoTBFs from data [316].

Polynomial approximation and interpolation techniques have been used to obtain MoPs. Shenoy and West [458] found MoP approximations of parametric probability density functions by computing the TSE around the middle point of each subinterval in the MoP. The mathematical expression of the probability density $f_X(x)$ needs to be known for computing the TSE. However, real data might not fit any known parametric density, so the TSE cannot be used in practice. Also, TSE cannot ensure that MoP approximations are valid densities, i.e., they are continuous, non-negative and integrate to one. Later, Shenoy [456] proposed estimating MoPs using the LIPs over the Chebyshev points defined in each subinterval. However, the true probability densities of the Chebyshev points in each subinterval need to be known or estimated beforehand (e.g., using empirical histograms or KDE techniques). LIPs can ensure non-negativity by increasing the order of the polynomials, and continuity by putting interpolation points at the limits of the intervals. However, it cannot ensure that the resulting MoP integrates to one.

Learning approximations of conditional densities has only been given limited attention [313, 316]. The general approach shared by existing methods for learning conditional densities is that the conditioning variables are discretized, and a one-dimensional approximation of the density of the conditional variable is found for each combination of the (discretized) values of the conditioning variables. Thus, the estimation of a conditional density reduces to the estimation of a collection of marginal densities, where the correlation between the variable and the conditioning variables is captured by the discretization procedure.

Regarding the use of the proposed methods as a non-parametric density estimation technique, BNCs have traditionally dealt with problems where the predictive features are discrete. Pérez et al. [408] studied hybrid BNCs with both continuous and discrete predictive features using CLG-based BNCs. Later, they proposed a non-parametric alternative by using KDE in BNCs [409]. BNCs using MTEs have been studied in [188, 192].

8.3 Mixtures of polynomials

Let X be a one-dimensional continuous random variable with probability density $f_X(x)$. Shenoy and West [458] defined a one-dimensional MoP approximation of $f_X(x)$ over a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$ as an L_X-piece d_X-degree piecewise function of the form

$$\varphi_X(x) = \begin{cases} pol_{l_X}(x) & \text{for } x \in A_{l_X},\ l_X = 1, \ldots, L_X \\ 0 & \text{otherwise,} \end{cases} \tag{8.1}$$

where $pol_{l_X}(x) = b_{0,l_X} + b_{1,l_X}x + b_{2,l_X}x^2 + \cdots + b_{d_X,l_X}x^{d_X}$ is a polynomial function with degree d_X (and order $r_X = d_X + 1$), $\{b_{0,l_X}, \ldots, b_{d_X,l_X}\}$ are constants and $A_1, \ldots, A_{L_X}$ are disjoint intervals in Ω_X, which do not depend on x with $\Omega_X = \cup_{l_X=1}^{L_X} A_{l_X}$, $A_i \cap A_j = \emptyset, i \neq j$.

Given a vector of n random variables $\mathbf{X} = (X_1, \ldots, X_n)$ and an approximation domain

$\Omega_{\mathbf{X}} = \Omega_{X_1} \times \cdots \times \Omega_{X_n}$, Shenoy and West [458] defined an n-dimensional MoP as the product of one-dimensional MoPs as defined in (8.1):

$$\varphi_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n} \varphi_{X_i}(x_i). \tag{8.2}$$

We should note that the product in Equation (8.2) assumes that the variables of the muldimensional MoP are independent. MoPs are closed under multiplication, integration, differentiation and addition. Therefore, the Shenoy-Shafer architecture [457] can be used to perform exact inference in the associated hybrid BN.

8.4 Learning mixtures of polynomials using B-spline interpolation

In this section we detail a method for learning MoP approximations of one-dimensional, multidimensional and conditional probability densities from data. The proposal does not assume any prior knowledge about the true density underlying the data, as opposed to previously proposed methods such as TSE or LIPs. Also, it ensures that the resulting MoP approximation is continuous, non-negative and integrates to one. Finally, it provides ML estimators of some of the parameters in the approximation. The BIC is used as a score for finding appropriate values for the other parameters in a principled way. Section 8.4.1 introduces B-spline interpolation. Section 8.4.2 studies MoP approximations of one-dimensional probability densities. Section 8.4.3 extends the proposed approach to multidimensional densities. Section 8.4.4 proposes two methods for learning MoP approximations of conditional densities. Section 8.4.5 addresses the model selection problem for finding appropriate values for the order and the number of intervals of a MoP.

8.4.1 B-spline interpolation

B-splines or basis splines [441] are polynomial curves that form a basis for the space of piecewise polynomial functions over a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$ [166]. Therefore, any piecewise polynomial function can be written as a linear combination of B-splines. Zong [540] proposed a method for finding B-spline approximations of one-dimensional and two-dimensional probability density functions from data.

Given a non-decreasing knot sequence of L_X+1 real numbers $\boldsymbol{\delta}_X = \{a_{X,0}, a_{X,1}, \ldots, a_{X,L_X}\}$ in the approximation domain $\Omega_X = [\epsilon_X, \xi_X]$ with $a_{X,i-1} < a_{X,i}$, $\epsilon_X = a_{X,0}$ and $\xi_X = a_{X,L_X}$, one can define $M_X = L_X + r_X - 1$ different B-splines with order r_X spanning the whole

domain Ω_X. The j_Xth B-spline $B^{r_X}_{X,j_X}(x)$, $j_X = 1, \ldots, M_X$ is written as

$$B^{r_X}_{X,j_X}(x) = (a_{X,j_X} - a_{X,j_X-r_X})H(x - a_{X,j_X-r_X})\times \sum_{t=0}^{r_X} \frac{(a_{X,j_X-r_X+t} - x)^{r_X-1} H(a_{X,j_X-r_X+t} - x)}{w'_{j_X-r_X}(a_{X,j_X-r_X+t})}, \quad x \in \Omega_X, \quad (8.3)$$

where $w'_{j_X-r_X}(x)$ is the first derivative of $w_{j_X-r_X}(x) = \prod_{u=0}^{r_X}(x - a_{X,j_X-r_X+u})$ and $H(x)$ is the Heaviside function

$$H(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases}$$

A recursive definition of B-splines and an efficient and well conditioned evaluation algorithm are available, e.g., in [114]. B-splines have a number of interesting properties for learning MoP approximations of probability densities, e.g., $B^{r_X}_{X,j_X}(x)$ is right-side continuous, differentiable, positive in and zero outside $(a_{X,j_X}, a_{X,j_X-r_X})$ [419]. B-splines form a basis in the space of piecewise polynomials and MoPs are piecewise polynomials. Therefore, every MoP can be written as a linear combination of B-splines. Also, given a continuous function $f_X(x)$ defined in a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$, the Stone-Weierstrass approximation theorem [476] states that there is a polynomial function $pol_X(x)$ that uniformly converges to $f_X(x)$ with an error less than ζ, i.e., there is a polynomial function $pol_X(x)$ so that $\sup_{x\in\Omega_X} |f_X(x) - pol_X(x)| < \zeta$.

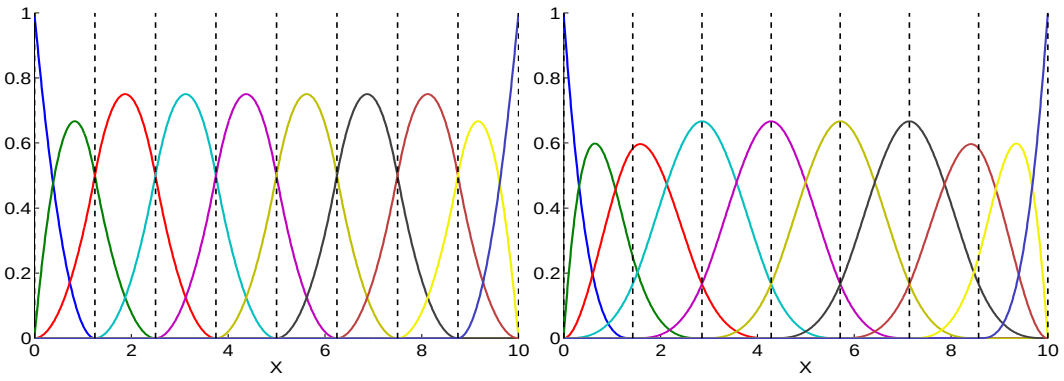
When the points in the knot sequence $\boldsymbol{\delta}_X$ are equally spaced, the B-splines are called uniform. A B-spline $B^{r_X}_{X,j_X}(x)$ can be written as a MoP function (Equation (8.1)) with L_X pieces, where each piece $pol_{l_X}(x)$ is defined as the expansion of Equation (8.3) in the interval $A_{l_X} = [a_{X,l_X-1}, a_{X,l_X}), l_X = 1, \ldots, L_X$. Figure 8.1 shows ten uniform B-splines defined in $\Omega_X = [0, 10]$ for orders (a) $r_X = 3$ and (b) $r_X = 4$. With the exception of the B-splines in the limits of Ω_X, we find that each B-spline is non-zero in r_X intervals and zero in the rest. Also, for each interval A_{l_X}, we find r_X non-zero B-splines.

8.4.2 Learning one-dimensional mixtures of polynomials

Zong [540] proposed using B-spline interpolation to find an approximation of the density $f_X(x)$ as a linear combination of $M_X = L_X + r_X - 1$ B-splines

$$\varphi_X(x\ ;\ \boldsymbol{\alpha}) = \sum_{j_X=1}^{M_X} \alpha_{j_X} B^{r_X}_{X,j_X}(x), \quad x \in \Omega_X, \quad (8.4)$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{M_X})$ are the mixing coefficients and $B^{r_X}_{X,j_X}(x), j_X = 1, \ldots, M_X$ are B-splines with order r_X (degree $d_X = r_X - 1$) as defined in Equation (8.3). MoPs are closed under multiplication and addition. Thus, the linear combination of M_X B-splines with order



(a) B-splines with $r_X = 3$ (b) B-splines with $r_X = 4$

Figure 8.1: Ten uniform B-splines defined in the domain $\Omega_X = [0, 10]$. Each B-spline is shown in a different color. The vertical dashed lines show the knot sequence $\boldsymbol{\delta} = (a_0, \ldots, a_{L_X})$, where $L_X = M_X - r_X + 1$ and $M_X = 10$ B-splines.

r_X (Equation (8.4)) yields a MoP function with L_X pieces, where each piece $pol_{l_X}(x)$ is a polynomial with order r_X defined in the interval A_{l_X}: $pol_{l_X}(x) = \sum_{j_X=1}^{M_X} \alpha_{j_X} B_{X,j_X}^{r_X}(x), \forall x \in A_{l_X} = [a_{X,l_X-1}, a_{X,l_X})$.

Therefore, four elements need to be specified to define a MoP using B-spline interpolation: the order (r_X), the number of intervals/pieces (L_X), the knot sequence ($\boldsymbol{\delta}_X$) and the mixing coefficients ($\boldsymbol{\alpha}$). We used uniform B-splines so the intervals A_{l_X} have an equal width of $a_{X,l_X} - a_{X,l_X-1} = \frac{\xi_X - \epsilon_X}{L_X}$, and $\boldsymbol{\delta}_X$ is easily found. The values of the order (r_X) and the number of intervals (L_X) of the MoP were found by testing different values and selecting the ones with the highest BIC score (see Section 8.4.5). Zong [540] derived an iterative procedure for computing the ML estimators of the mixing coefficients, $\widehat{\boldsymbol{\alpha}}$, in Equation (8.4). To ensure that the resulting linear combination of B-splines is a valid density (continuous, non-negative and integrating to one), the optimization procedure introduces two constraints: $\sum_{j_X=1}^{M_X} \alpha_{j_X} c_{j_X} = 1$ and $\alpha_{j_X} \geq 0, j_X = 1, \ldots, M_X$, where

$$c_{j_X} = \int_{a_{X,j_X-r_X}}^{a_{X,j_X}} B_{X,j_X}^{r_X}(x)dx = \frac{a_{X,j_X} - a_{X,j_X-r_X}}{r_X}.$$

Given a dataset $\mathcal{D}_X = \{x_1, \ldots, x_N\}$ with N observations of variable X, the ML estimates of the mixing coefficients are computed using the formula:

$$\widehat{\alpha}_{j_X}^{(q)} = \frac{1}{N c_{j_X}} \sum_{x \in \mathcal{D}_X} \frac{\widehat{\alpha}_{j_X}^{(q-1)} B_{X,j_X}^{r_X}(x)}{\varphi_X\left(x; \widehat{\boldsymbol{\alpha}}^{(q-1)}\right)}, \quad j_X = 1, \ldots, M_X, \tag{8.5}$$

where q is the iteration number in the optimization process. Zong showed that Equation (8.5) yields the only maximum of the log-likelihood $\ell(\mathcal{D}_X | \varphi_X(x; \boldsymbol{\alpha}))$ (Equation (3.3)) of $\mathcal{D}_X$ given the approximation $\varphi_X(x; \boldsymbol{\alpha})$ (Equation (8.4)). The initial values $\widehat{\alpha}_{j_X}^{(0)}$ are set to $1/\sum_{j_X=1}^{M_X} c_{j_X}$. The relative change in the log-likelihood of $\mathcal{D}_X$ given $\varphi_X\left(x; \widehat{\boldsymbol{\alpha}}^{(q)}\right)$ is used as a stopping crite-

rion, i.e., Equation (8.5) iterates until $\left|\frac{\ell^{(q)}-\ell^{(q-1)}}{\ell^{(q)}}\right| < e$, where $\ell^{(q)} = \ell\left(\mathcal{D}_X|\varphi_X(x;\widehat{\boldsymbol{\alpha}}^{(q)})\right)$ is the log-likelihood at iteration q. We used $e = 10^{-6}$ in our experiments. The computational complexity of this optimization process is $O(M_X N q_{\max})$, where $q_{\max}$ is the number of iterations of Equation (8.5) performed until the optimization converges. Algorithm 8.1 summarizes the whole process for obtaining a MoP approximation of a one-dimensional probability density function using a dataset.

Algorithm 8.1 (Learning a MoP approximation of a one-dimensional probability density from data)
Inputs:

$\mathcal{D}_X$: *A dataset with* N *observations* $\mathcal{D}_X = \{x_1, \ldots, x_N\}$

L_X: *The number of pieces of the MoP*

r_X: *The order of the polynomials*

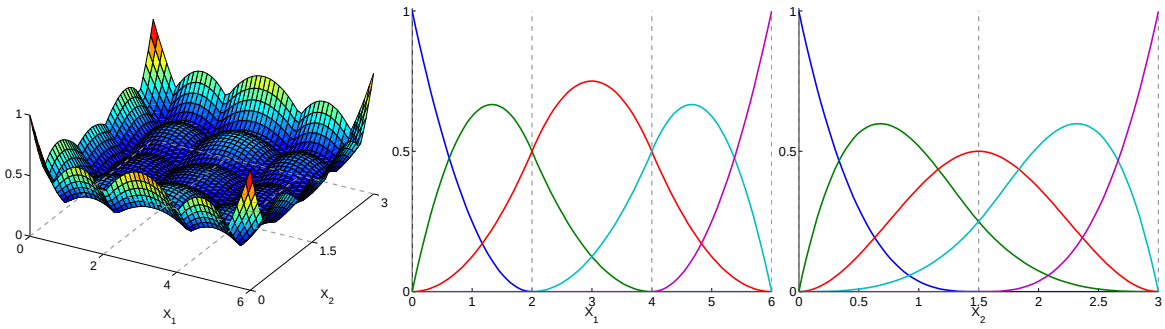
Output: An L_X*-piece* $(r_X - 1)$*-degree MoP approximation* $\varphi_X(x;\widehat{\boldsymbol{\alpha}})$ *of the probability density underlying the dataset* $\mathcal{D}_X$
Steps:

1. *Compute the domain of the approximation* $\Omega_X = [\epsilon_X, \xi_X]$ *, where the limits of the domain are defined as* $\epsilon_X = \min\{x_1, \ldots, x_N\}$ *and* $\xi_X = \max\{x_1, \ldots, x_N\}$.
2. *Compute the knot sequence* $\boldsymbol{\delta}_X = \{a_{X,0}, a_{X,1}, \ldots, a_{X,L_X}\}$ *and define the intervals* $A_{l_X} = [a_{X,l_X-1}, a_{X,l_X})$, $l_X = 1, \ldots, L_X$.
3. *Apply Equation (8.3) to build the* $M_X = L_X + r_X - 1$ *B-splines* $B^{r_X}_{X,j_X}(x)$.
4. *Apply Equation (8.5) iteratively to compute the ML estimators of the mixing coefficients* $\widehat{\boldsymbol{\alpha}}$.
5. *Compute the polynomials* $pol_{l_X}(x)$ *from Equation (8.1) as the linear combination of the B-splines defined for each interval* A_{l_X}*, and build the MoP.*

8.4.3 Learning multidimensional mixtures of polynomials

The approach in Section 8.4.2 can be intuitively extended to learn MoP approximations of multidimensional joint probability densities. Zong and Lam [541] and Zong [540] studied B-spline approximations of two-dimensional joint probability density functions. Here, we extend their work to general n-dimensional joint probability density functions. Given a vector of n random variables $\mathbf{X} = (X_1, \ldots, X_n)$, we can approximate the joint probability density function $f_{\mathbf{X}}(\mathbf{x})$ with a multidimensional linear combination of B-splines:

$$\varphi_{\mathbf{X}}(\mathbf{x};\boldsymbol{\alpha}) = \sum_{\substack{j_{X_1}=1,\ldots,M_{X_1} \\ \vdots \\ j_{X_n}=1,\ldots,M_{X_n}}} \alpha_{j_{X_1},\ldots,j_{X_n}} \prod_{i=1}^{n} B^{r_{X_i}}_{X_i,j_{X_i}}(x_i), \quad \mathbf{x} \in \Omega_{\mathbf{X}}, \tag{8.6}$$



(a) Two-dimensional approximation by a linear combination of the product of one-dimensional B-splines

(b) One-dimensional B-splines for X_1 with $r_{X_1} = 3$

(c) One-dimensional B-splines for X_2 with $r_{X_2} = 4$

Figure 8.2: (a) Two-dimensional B-splines defined as the product of one-dimensional B-splines. The domain of the approximation is $\Omega_{\mathbf{X}} = [0, 6] \times [0, 3]$. Five B-splines are used for each dimension ($M_{X_1} = M_{X_2} = 5$). Different orders are used for dimensions X_1 ($r_{X_1} = 3$) and X_2 ($r_{X_2} = 4$). Therefore, the number of intervals for each dimension is different, i.e., $L_{X_1} = 3$ and $L_{X_2} = 2$. The dashed lines show the two-dimensional subintervals (rectangles A_l) in which the two-dimensional B-splines are defined. (b) One-dimensional B-splines for X_1. (c) One-dimensional B-splines for X_2.

where r_{X_i} is the order of the B-splines for variable X_i, $M_{X_i} = L_{X_i} + r_{X_i} - 1$ is the number of B-splines for variable X_i, L_{X_i} is the number of intervals for variable X_i, and $\alpha_{j_{X_1},\ldots,j_{X_n}}$ is the mixing coefficient for the combination of B-splines given by the indices $j_{X_1}, \ldots, j_{X_n}$. Figure 8.2(a) shows an example of two-dimensional B-splines defined as a linear combination of the product of one-dimensional B-splines as in Equation (8.6). The corresponding one-dimensional B-splines are shown in Figures 8.2(b) and (c). As in the one-dimensional scenario, each B-spline $B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)$ can be written as a MoP, and each product of one-dimensional B-splines in Equation (8.6) yields a multidimensional MoP as defined in Equation (8.2). However, as opposed to Equation (8.2), the multidimensional MoP $\varphi_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\alpha})$ in Equation (8.6) does not assume that the variables $X_1, \ldots, X_n$ are independent. The dimensions of the final MoP are related through the mixing coefficients $\alpha_{j_{X_1},\ldots,j_{X_n}}$, one for each combination of B-splines.

As in the one-dimensional scenario, we have to specify the number of intervals for each dimension $(L_{X_1}, \ldots, L_{X_n})$, the order of the polynomials for each dimension $(r_{X_1}, \ldots, r_{X_n})$, the knot sequence $(\boldsymbol{\delta}_{\mathbf{X}})$ and the mixing coefficients $(\boldsymbol{\alpha})$ in order to completely define a multidimensional MoP. Here, we found the knot sequence as the Cartesian product of the knot sequences of each dimension $\boldsymbol{\delta}_{\mathbf{X}} = \boldsymbol{\delta}_{X_1} \times \cdots \times \boldsymbol{\delta}_{X_n}$, where $\boldsymbol{\delta}_{X_i}$ are defined to yield equal-width intervals as in the one-dimensional case (see Section 8.4.2). Similarly, the mixing coefficient vector has one value for each combination of one-dimensional B-splines, i.e., $\boldsymbol{\alpha} = (\alpha_{1,\ldots,1}, \ldots, \alpha_{M_{X_1},\ldots,M_{X_n}})$. The resulting MoP has $\prod_{i=1}^{n} L_{X_i}$ pieces, where each piece $pol_{l_{X_1},\ldots,l_{X_n}}(\mathbf{x})$ is defined in an n-dimensional hyperrectangle $A_{l_{X_1},\ldots,l_{X_n}} = \left[a_{X_1,l_{X_1}-1}, a_{X_1,l_{X_1}}\right] \times \cdots \times \left[a_{X_n,l_{X_n}-1}, a_{X_n,l_{X_n}}\right]$.

Given a dataset with N observations of n-dimensional vectors $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_z = (x_{z,1}, \ldots, x_{z,n}), z = 1, \ldots, N$, the ML estimates of the mixing coefficients $\widehat{\boldsymbol{\alpha}}$ in Equation (8.6) are approached by the iteration formula

$$\widehat{\alpha}^{(q)}_{j_{X_1},\ldots,j_{X_n}} = \frac{1}{N c_{j_{X_1},\ldots,j_{X_n}}} \sum_{z=1}^{N} \frac{\widehat{\alpha}^{(q)}_{j_{X_1},\ldots,j_{X_n}} \prod_{i=1}^{n} B^{r_{X_i}}_{X_i,j_{X_i}}(x_{z,i})}{\varphi_{\mathbf{X}}\left(\mathbf{x}_z \,;\, \widehat{\boldsymbol{\alpha}}^{(q-1)}\right)}, \tag{8.7}$$

subject to the constraints $\sum_{\substack{j_{X_1}=1,\ldots,M_{X_1} \\ \ldots \\ j_{X_n}=1,\ldots,M_{X_n}}} \alpha_{j_{X_1},\ldots,j_{X_n}} c_{j_{X_1},\ldots,j_{X_n}} = 1$ and $\alpha_{j_{X_1},\ldots,j_{X_n}} \geq 0$, where $j_{X_i} = 1, \ldots, M_{X_i}, i = 1, \ldots, n$ and

$$c_{j_{X_1},\ldots,j_{X_n}} = \prod_{i=1}^{n} \int_{a_{X_i,j_{X_i}-r_{X_i}}}^{a_{X_i,j_{X_i}}} B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)dx_i = \prod_{i=1}^{n} \frac{a_{X_i,j_{X_i}} - a_{X_i,j_{X_i}-r_{X_i}}}{r_{X_i}}.$$

Algorithm 8.2 details the steps for learning a MoP approximation of a multidimensional joint density from a dataset of observations.

Algorithm 8.2 (Learning a MoP approximation of a multidimensional joint probability density from data)
Inputs:

$\mathcal{D}_{\mathbf{X}}$*: A dataset with* N *observations* $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

$L_{X_1}, \ldots, L_{X_n}$*: The number of intervals of the MoP for each dimension*

$r_{X_1}, \ldots, r_{X_n}$*: The order of the polynomials for each dimension*

Output: A multidimensional MoP approximation $\varphi_{\mathbf{X}}(\mathbf{x}; \widehat{\boldsymbol{\alpha}})$ *of the joint probability density underlying the dataset* $\mathcal{D}_{\mathbf{X}}$
Steps:

1. *Compute the domain of the approximation for each dimension* $\Omega_{X_i} = [\epsilon_{X_i}, \xi_{X_i}], i = 1, \ldots, n$ *where* $\epsilon_{X_i} = \min\{x_{1,i}, \ldots, x_{N,i}\}$ *and* $\xi_{X_i} = \max\{x_{1,i}, \ldots, x_{N,i}\}$.
2. *Compute the multidimensional domain of the approximation* $\Omega_{\mathbf{X}} = \Omega_{X_1} \times \cdots \times \Omega_{X_n}$.
3. *Compute the knot sequence for each dimension* $\boldsymbol{\delta}_{X_i} = \{a_{X_i,0}, a_{X_i,1}, \ldots, a_{X_i,L_{X_i}}\}$ *and define the hyperrectangles* $A_{l_{X_1},\ldots,l_{X_n}} = \left[a_{X_1,l_{X_1}-1}, a_{X_1,l_{X_1}}\right] \times \cdots \times \left[a_{X_n,l_{X_n}-1}, a_{X_n,l_{X_n}}\right]$ *for each piece.*
4. *Apply Equation (8.3) to build the* $M_{X_i} = L_{X_i} + r_{X_i} - 1$ *B-splines* $B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)$ *for each dimension* $i = 1, \ldots, n$.
5. *Apply Equation (8.7) to compute the ML estimators of the mixing coefficients* $\widehat{\boldsymbol{\alpha}}$.
6. *Compute the polynomials* $pol_{l_{X_1},\ldots,l_{X_n}}(\mathbf{x})$ *from Equation (8.2) as the linear combination of the B-splines in Equation (8.6) and build the MoP.*

8.4.4 Learning conditional mixtures of polynomials

This section addresses the problem of learning MoPs of conditional densities from data. Following the terminology used for BNs, we consider the conditional random variable X as the child variable and the vector of conditioning random variables $\mathbf{Y} = (Y_1, \ldots, Y_n)$ as the parent variables. Given a sample $\mathcal{D}_{X,\mathbf{Y}} = \{(x_j, \mathbf{y}_j)\}$, $j = 1, \ldots, N$ from the joint density of $(X, \mathbf{Y})$, the aim is to learn a MoP approximation $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ of the conditional density $f_{X|\mathbf{Y}}(x|\mathbf{y})$ of $X|\mathbf{Y}$ from $\mathcal{D}_{X,\mathbf{Y}}$. We propose two methods for learning a MoP approximation of the conditional density of $X|\mathbf{Y}$ from data. Both approaches are based on learning MoP approximations of the joint density and the marginal density of the conditioning variables, but they differ as to how the MoP approximation of the quotient of the two densities is found. First, the method proposed in Section 8.4.4.1 is based on obtaining a sample from the conditional density of $X|\mathbf{Y}$ and learning a conditional MoP from it. Second, Section 8.4.4.2 reports a method based on multidimensional polynomial interpolation of the estimated values of the conditional density of $X|\mathbf{Y}$.

Our approach differs from previous methods in several ways. As opposed to [315, 456, 458], we learn conditional MoPs directly from data without any parametric assumptions. Also, we do not rely on a discretization of the conditioning variables to capture the correlation among the variables [313, 316]. On the other hand, our conditional MoPs are not proper conditional densities, hence posterior distributions established during inference have to be normalized so that they integrate to 1.

8.4.4.1 Learning conditional MoPs using sampling

The proposed method is based on first obtaining a sample from the conditional density of $X|\mathbf{Y}$ and then learning a conditional MoP density from the sampled values. Algorithm 8.3 shows the main steps of the procedure. First, we find a MoP representation of the joint density $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})$ (step 1) using the B-spline interpolation approach (Algorithm 8.2). Second, we obtain a MoP of the marginal density of the parents $\varphi_{\mathbf{Y}}(\mathbf{y})$ by marginalization (step 2). Next, we use a sampling algorithm to obtain a sample $\mathcal{D}_{X|\mathbf{Y}}$ from the conditional density of $X|\mathbf{Y}$ (step 3), where the conditional density values are obtained by evaluating the quotient $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$. More specifically, we have used a standard Metropolis-Hastings sampler for the reported experimental results. For the sampling process we generate uniformly distributed values over $\Omega_{\mathbf{Y}}$ for the parent variables $\mathbf{Y}$, whereas the proposed distribution for the child variable is a linear Gaussian distribution $\mathcal{N}(\boldsymbol{\beta}^T\mathbf{y}, \sigma^2)$, where $\boldsymbol{\beta}$ is an n-dimensional vector with all components equal to $1/n$. We used $\sigma^2 = 0.5$ in our experiments. Next, we find an (unnormalized) MoP approximation of the conditional density $X|\mathbf{Y}$ from $\mathcal{D}_{X|\mathbf{Y}}$ (step 4). Finally, we apply the partial normalization procedure proposed in [458] to obtain a MoP approximation $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ of the conditional density (steps 5 and 6). The complexity of the algorithm is dominated by the complexity of the multidimensional MoP learning method (Algorithm 8.2).

This method has some interesting properties. Algorithm 8.2 guarantees that the approximations are continuous, non-negative and integrate to one. Therefore, the conditional MoPs obtained with Algorithm 8.3 are also continuous and non-negative. Continuity is not required for inference in BNs, but it usually is a desirable property, e.g., for visualization purposes. The algorithm provides ML estimates of the mixing coefficients of the linear combination of B-splines when learning MoPs of the joint density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ and the marginal density $\varphi_{\mathbf{Y}}(\mathbf{y})$, hence the quotient $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ corresponds to a ML model of the conditional distribution. It should be noted, though, that this property is not shared by the final learned model as the partial normalization (steps 5 and 6) does not ensure that the learned MoP is a proper conditional density. Therefore, the MoP approximations of the posterior densities should be normalized to integrate to 1. Note that the same orders $r_X, r_{Y_1}, \ldots, r_{Y_n}$ and number of intervals $L_X, L_{Y_1}, \ldots, L_{Y_n}$ are used for learning the MoP of the joint density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ in step 1 and the unnormalized conditional MoP in step 4 $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$. This is an heuristic to reduce the number of parameters to find and decrease the number of models to fit. However, we should note that different orders and numbers of intervals could be considered in steps 1 and 4 without changing the outline of Algorithm 8.3.

Algorithm 8.3 (Learning conditional MoPs using sampling)
Inputs:

$\mathcal{D}_{X,\mathbf{Y}}$*: A training dataset* $\mathcal{D}_{X,\mathbf{Y}} = \{(x_j, \mathbf{y}_j)\}, j = 1, \ldots, N$

$r_X, r_{Y_1}, \ldots, r_{Y_n}$*: The order of the MoP for each dimension*

$L_X, L_{Y_1}, \ldots, L_{Y_n}$*: The number of intervals of the MoP for each dimension*

Output: The MoP approximation $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ *of the conditional density of* $X|\mathbf{Y}$
Steps:

1. *Learn a MoP* $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ *of the joint density of* $(X,\mathbf{Y})$ *from the dataset* $\mathcal{D}_{X,\mathbf{Y}}$ *using polynomials with orders* $r_X, r_{Y_1}, \ldots, r_{Y_n}$ *and* $L_X, L_{Y_1}, \ldots, L_{Y_n}$ *pieces (Algorithm 8.2).*

2. *Marginalize out* X *from* $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ *to yield a MoP* $\varphi_{\mathbf{Y}}(\mathbf{y})$ *of the marginal density of the parent variables* $\mathbf{Y}$*:*
$$\varphi_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega_X} \varphi_{X,\mathbf{Y}}(x,\mathbf{y})dx.$$

3. *Use a Metropolis-Hastings algorithm to yield a sample* $\mathcal{D}_{X|\mathbf{Y}}$ *with* M *observations from the conditional density* $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$.

4. *Learn an unnormalized conditional MoP* $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$ *from* $\mathcal{D}_{X|\mathbf{Y}}$ *using polynomials with orders* $r_X, r_{Y_1}, \ldots, r_{Y_n}$ *and* $L_X, L_{Y_1}, \ldots, L_{Y_n}$ *pieces (Algorithm 8.2).*

5. *Compute the partial normalization constant:*
$$c = \int_{\Omega_X} \int_{\Omega_{\mathbf{Y}}} \varphi_{\mathbf{Y}}(\mathbf{y}) \varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y}) d\mathbf{y} dx \ .$$

6. *Find the partially normalized MoP of the conditional density:*

$$\varphi_{X|\mathbf{Y}}(x|\mathbf{y}) = \frac{1}{c}\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y}) \ .$$

8.4.4.2 Learning conditional MoPs using interpolation

The preliminary empirical results obtained with Algorithm 8.3 show that the sampling approach can produce good approximations (see Section 8.5.1.6). However, it is difficult to control or obtain any guarantees about the quality of the approximation due to the partial normalization.

This shortcoming has motivated an alternative method for learning a MoP approximation of a conditional probability density for $X|\mathbf{Y}$. The main steps of the procedure are summarized in Algorithm 8.4. First, we find MoP approximations of both the joint density of $(X, \mathbf{Y})$ and the marginal density of $\mathbf{Y}$ in the same way as in Algorithm 8.3 (steps 1 and 2). Next, we build the conditional MoP $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ by finding, for each piece $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ defined in the hyperrectangle $A_{l_X,l_{Y_1},\ldots,l_{Y_n}}$, a multidimensional interpolation polynomial of the function given by the quotient of the joint and the marginal densities $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$.

Algorithm 8.4 (Learning conditional MoPs using interpolation)
Inputs:

$\mathcal{D}_{X,\mathbf{Y}}$*: A training dataset* $\mathcal{D}_{X,\mathbf{Y}} = \{(x_j, \mathbf{y}_j)\}, j = 1, \ldots, N$

$r_X, r_{Y_1}, \ldots, r_{Y_n}$*: The order of the MoP for each dimension*

$L_X, L_{Y_1}, \ldots, L_{Y_n}$*: The number of intervals of the MoP for each dimension*

Output: The MoP approximation $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ *of the conditional density of* $X|\mathbf{Y}$
Steps:

1. *Learn a MoP* $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ *of the joint density of* $(X, \mathbf{Y})$ *from the dataset* $\mathcal{D}_{X,\mathbf{Y}}$ *using polynomials with orders* $r_X, r_{Y_1}, \ldots, r_{Y_n}$ *and* $L_X, L_{Y_1}, \ldots, L_{Y_n}$ *pieces (Algorithm 8.2).*

2. *Marginalize out* X *from* $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ *to yield a MoP* $\varphi_{\mathbf{Y}}(\mathbf{y})$ *of the marginal density of the parent variables* $\mathbf{Y}$*:*

$$\varphi_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega_X} \varphi_{X,\mathbf{Y}}(x,\mathbf{y})dx.$$

3. *For piece* $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ *in the conditional MoP* $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$*, defined in* $A_{l_X,l_{Y_1},\ldots,l_{Y_n}}$ *with* $l_X = 1, \ldots, L_X$ *and* $l_{Y_i} = 1, \ldots, L_{Y_i}$*,* $i = 1, \ldots, n$*:*

   Find a multi-dimensional polynomial approximation $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ *of function* $g(x,\mathbf{y}) = \varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ *using an interpolation method.*

We consider two multidimensional interpolation methods, which can be used to obtain the polynomials of the pieces $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ in step 3 of Algorithm 8.4:

The multidimensional TSE around a point yields a polynomial approximation of any differentiable function g. The quotient of any two functions is differentiable as long as the two functions are also differentiable. In our scenario, polynomials are differentiable functions and, thus, we can compute the TSE of the quotient of two polynomials. Consequently, we can use multidimensional TSEs to find a polynomial approximation $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ of $g(x,\mathbf{y}) = \varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ for each piece. We computed these TSEs of $g(x,\mathbf{y})$ around the midpoint of the hyperrectangle $A_{l_X,l_{Y_1},\ldots,l_{Y_n}}$.

LIPs can approximate any function g. Before finding the LIP, we need to evaluate function g on a set of interpolation points. In the one dimensional scenario, the set of Chebyshev points are frequently used as interpolation points [242]. However, finding multidimensional LIPs is not a trivial task because it is difficult to find good interpolation points in a multidimensional space. Some works have been recently proposed in the two-dimensional scenario [73, 242]. To find a conditional MoP using LIPs, we first find and evaluate the conditional density function $g(x,\mathbf{y}) = \varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ on the set of interpolation points in $A_{l_X,l_{Y_1},\ldots,l_{Y_n}}$. Next, we compute the polynomial $pol_{l_X,l_{Y_1},\ldots,l_{Y_n}}(x,\mathbf{y})$ for the piece as the LIP over the interpolation points defined in $A_{l_X,l_{Y_1},\ldots,l_{Y_n}}$. Note that other approaches can also be used to evaluate the conditional density $g(x,\mathbf{y})$ on the set of interpolation points, e.g., kernel-based conditional estimation methods [34, 116, 202].

Compared with Algorithm 8.3, there are some apparent (dis)advantages. First, the conditional MoPs produced by Algorithm 8.4 are not necessarily continuous. Second, interpolation methods cannot in general ensure non-negativity, although LIPs can be used to ensure it by increasing the order of the polynomials. On the other hand, the learning method in Algorithm 8.4 does not need a partial normalization step. Thus, if the polynomial approximations are close to the conditional density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$, then the conditional MoP using these polynomial interpolations is expected to be close to normalized. As a result, we can more directly control the quality of the approximation by varying the order of the polynomials and the number of hyperrectangles.

8.4.5 Model selection

The number of pieces L_X and the order of the polynomials r_X have to be specified a priori in Algorithm 8.1. Similarly, the parameters L_{X_i} and r_{X_i} have to be specified for each variable $X_i, i = 1,\ldots,n$ in Algorithm 8.2. Also, the parameters $L_X, L_{Y_1},\ldots,L_{Y_n}$ and $r_X, r_{Y_1},\ldots,r_{Y_n}$ have to be provided as inputs in Algorithms 8.3 and 8.4. Since the ML estimators of the mixing coefficients, $\widehat{\boldsymbol{\alpha}}$, are computed in Equations (8.5) and (8.7), we can use a penalized likelihood score to perform model selection in a principled way. Here, we used the BIC score (Equation (3.4)) to find appropriate values for the order of the polynomials and the number of pieces of the MoP. Given the knot sequence $\boldsymbol{\delta}_{\mathbf{X}}$, the order $r_{X_1},\ldots,r_{X_n}$ of the B-splines and the number of pieces $L_{X_1},\ldots,L_{X_n}$, the only free parameters that need to be estimated for approximating one-dimensional (Section 8.4.2) and multidimensional (Section 8.4.3) MoPs

are the mixing coefficients $\boldsymbol{\alpha}$ of the linear combination of B-splines. Therefore, we used the number of mixing coefficients as a measure of the dimension of the MoP when approximating one-dimensional and multidimensional densities with Algorithms 8.1 and 8.2, i.e., $\dim(\varphi_{\mathbf{X}}(\mathbf{x})) = (\prod_{i=1}^{n} M_{X_i})$. For learning conditional MoPs (Section 8.4.4) we used the number of polynomial coefficients different from zero as a measure of the dimension of the MoP $\dim(\varphi_{X|\mathbf{Y}}(x|\mathbf{y}))$. In all our experiments, we selected the MoP approximation with the highest BIC score.

8.5 Experiments

In this section, we report the results of the experiments to evaluate the proposed methods. Section 8.5.1 includes the experiments related to MoP learning of probability densities from data. Section 8.5.2 reports the results of the BNCs using MoPs as a non-parametric density estimation technique. Section 8.5.3 shows a comparison of the evaluation time of MoPs vs. KDE.

8.5.1 Experiments with mixtures of polynomials approximations

In this section we report the experiments on the MoP approximations of one-dimensional, multidimensional and conditional probability densities from data.

8.5.1.1 Artificial datasets

We sampled datasets with different number of observations N from known densities with different shapes. The study included both known parametric probability densities and mixtures of densities. Table 8.1 shows the name of the datasets, the respective probability distributions and the domains of approximation.

8.5.1.2 Comparison measures

We used different measures to evaluate the quality of the MoPs learned from the datasets. First, two measures are reported to analyze the goodness of fit of the MoPs to the datasets from which they were learned:

> The log-likelihood $\ell(\mathcal{D}_{\mathbf{X}}|\varphi_{\mathbf{X}}(\mathbf{x}))$ of the dataset $\mathcal{D}_{\mathbf{X}}$ given the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ (see Equation (3.3)). We computed the log-likelihood using both the training dataset and a test dataset with the same size.
>
> The BIC score $BIC(\mathcal{D}_{\mathbf{X}}, \varphi_{\mathbf{X}}(\mathbf{x}))$ of the dataset $\mathcal{D}_{\mathbf{X}}$ given the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ (Equation (3.4)). We computed the BIC score using both the training dataset and a test dataset with the same size.

Additionally, we report three measures for evaluating how close the MoP approximation is to the true density which generated the training datasets (Table 8.1):

Table 8.1: Probability density functions used to sample artificial datasets and for learning mixtures of polynomials

Name	n	Distribution	Domain
		One-dimensional probability densities	
`Gauss`	1	$\mathcal{N}(0,1)$	$[-3,3]$
`Exp`	1	Exp	$[0,3]$
`Chisq`	1	χ^2_3	$[0,8]$
`MixGauss`	1	$0.5\mathcal{N}(0,1)+0.5\mathcal{N}(4,1)$	$[-3,7]$
`Mix1d`	1	$0.8\chi^2(3)+0.2\mathcal{N}(7,1)$	$[0,10]$
		Multidimensional probability densities	
`Gauss2d`	2	$\mathcal{N}\left((0,0),\begin{pmatrix}1&0\\0&1\end{pmatrix}\right)$	$[-3,3]\times[-3,3]$
`Mix2d`	2	$0.7\mathcal{N}\left((5,6),\begin{pmatrix}0.5&0.3\\0.3&0.5\end{pmatrix}\right)$ $+0.3\mathcal{N}\left((7,6.5),\begin{pmatrix}0.4&-0.2\\-0.2&0.4\end{pmatrix}\right)$	$[3,8]\times[4,8]$
`Mix3d`	3	$0.4\mathcal{N}\left((3,5.5,4),\begin{pmatrix}1&0.5&0.3\\0.5&0.5&0\\0.3&0&1\end{pmatrix}\right)$ $+0.6\mathcal{N}\left((4,4,2),\begin{pmatrix}1&0.75&-0.1\\0.75&1.5&-0.2\\-0.1&-0.2&2\end{pmatrix}\right)$	$[2,7]\times[1,7]\times[0,8]$
		Conditional probability densities	
`LinGauss`	2	$\mathcal{N}\left((0,0),\begin{pmatrix}2&1\\1&1\end{pmatrix}\right)$ corresponding to $Y\sim\mathcal{N}(0,1)$ and $X\|Y\sim\mathcal{N}(y,1)$	$[-3,3]\times[-2,2]$

The Kullback-Leibler (KL) divergence [305] of the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ from the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$KL(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x}))=\int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})\log\frac{f_{\mathbf{X}}(\mathbf{x})}{\varphi_{\mathbf{X}}(\mathbf{x})}d\mathbf{x}.$$

The mean squared error (MSE) between the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$MSE(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x}))=\int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})(f_{\mathbf{X}}(\mathbf{x})-\varphi_{\mathbf{X}}(\mathbf{x}))^2 d\mathbf{x}.$$

The maximum absolute error (MAE) between the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$MAE(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x}))=\max_{\mathbf{x}\in\Omega_{\mathbf{X}}}|f_{\mathbf{X}}(\mathbf{x})-\varphi_{\mathbf{X}}(\mathbf{x})|.$$

The MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ needs to be a proper density (non-negative and integrating to one in $\Omega_{\mathbf{X}}$) to compute the log-likelihood and the BIC scores. Similarly, both the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true

density $f_{\mathbf{X}}(\mathbf{x})$ need to be proper densities in the domain $\Omega_{\mathbf{X}}$ to compute the KL divergence. MoPs using B-splines learned with Algorithms 8.1 and 8.2 are ensured to be non-negative and integrate to one in $\Omega_{\mathbf{X}}$ (see Sections 8.4.2 and 8.4.3). Also, non-negative MoPs learned using LIPs were considered and normalized to integrate to one in $\Omega_{\mathbf{X}}$ (see Section 8.5.1.3). Finally, the true probability densities $f_{\mathbf{X}}(\mathbf{x})$ also need to be normalized because a (small) part of the density mass lays outside the domain of approximation $\Omega_{\mathbf{X}}$. Thus, a normalization constant $T = \int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})d\mathbf{x}$ was computed and the true density values $f_{\mathbf{X}}(\mathbf{x})$ were normalized by multiplying them by $1/T$ when computing KL, MSE and MAE.

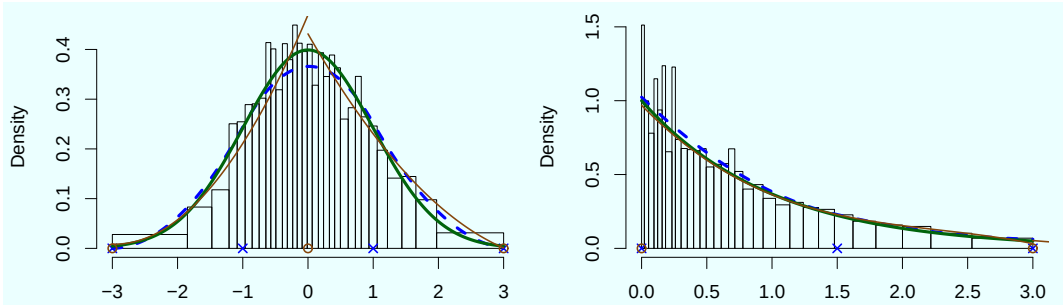
The integrals in the KL divergence and the MSE were computed using an adaptive quadrature integration procedure available in `R` [281, 422]. To avoid local maxima, the MAE was computed from the density differences on 1000 equally-spaced points defined in the domain $\Omega_{\mathbf{X}}$. Then, the 50 points yielding the maximum values of $|f_{\mathbf{X}}(\mathbf{x}) - \varphi_{\mathbf{X}}(\mathbf{x})|$ in the equally-spaced grid were used as starting values of a non-linear optimization algorithm bounded to the interval $\Omega_{\mathbf{X}}$ (`nlminb` function in `R`) to find the global maximum.

8.5.1.3 MoP approximation using Lagrange interpolating polynomials

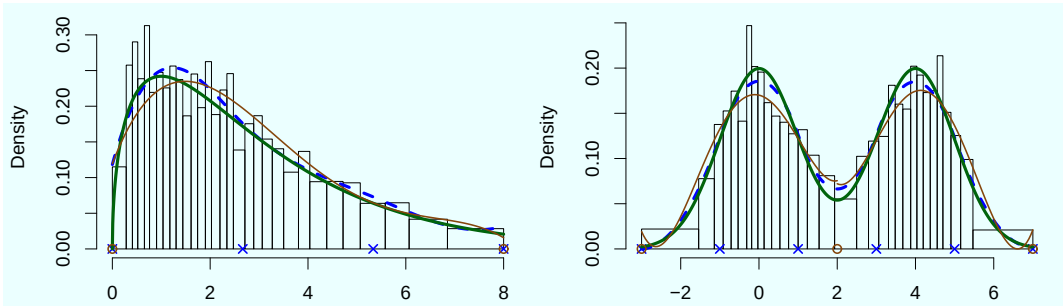
The results were compared with the MoPs obtained by computing the LIP over the Chebyshev points defined in each interval independently [456]. The density at the Chebyshev points was estimated using KDE. Gaussian kernels with the normal scale bandwidth were computed using the `ks` package [154]. We considered different values for the order r_X and the number of pieces L_X of the MoP. Equal-width intervals A_{l_X} were assumed for each piece $l_X = 1, \ldots, L_X$. The BIC score (see Section 8.4.5) was used to select appropriate values for the order r_X and the number of pieces L_X of the MoPs. Here, the number of polynomial coefficients different from zero was used as a measure of the dimension of the model $\dim(\varphi_X(x))$ in Equation (3.4).

We checked whether or not the MoP approximations learned with LIPs yielded negative values. To check this situation, the values of $\varphi_X(x)$ were computed at 1000 equally spaced points in Ω_X. Then, the points corresponding to local minima values of $\varphi_X(x)$ were found. These points were used as the starting points for a non-linear optimization procedure (`nlminb` function in `R`) to find the global minimum $\upsilon = \min_{x \in \Omega_X} \varphi_X(x)$. If the MoP yielded negative values, the minimum value of the function was added to the MoP: $\varphi_X(x) \leftarrow \varphi_X(x) - \upsilon$. Then, the normalization constant was computed as $T = \int_{\Omega_X} \varphi_X(x)dx$ and the MoP was normalized by computing $\varphi_X(x) \leftarrow \frac{1}{T}\varphi_X(x)$. These two steps (finding the minimum value and normalizing) were repeated until the final MoP $\varphi_X(x)$ was non-negative. Thus, we ensured that $\varphi_X(x)$ learned with LIPs was non-negative and integrated to one in Ω_X.

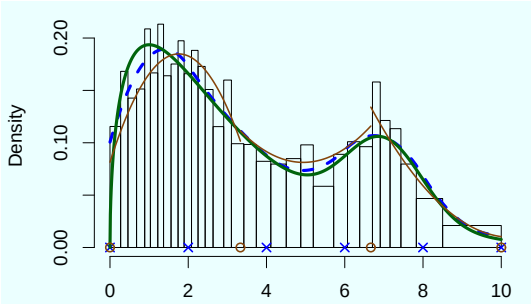
As opposed to the one-dimensional scenario, multidimensional LIPs are more challenging [210], although some approaches have been proposed for two-dimensional scenarios, e.g., see [50, 72, 242]. Additionally, estimating the densities at the interpolation nodes is more difficult in multidimensional domains than in the one-dimensional scenario.



(a) `Gauss` (b) `Exp`



(c) `Chisq` (d) `MixGauss`



(e) `Mix1d`

Figure 8.3: One-dimensional MoP approximations learned from a training dataset of $N = 1000$ observations. The MoPs with the highest BIC score for the first training dataset learned with Algorithm 8.1 (dashed lines) and using LIPs (thin solid lines) are shown. A thick solid line represents the true density used to generate the training datasets. An equal-frequency histogram of the training dataset is displayed. Crosses along the horizontal axis mark the limits of the intervals for the MoPs learned with Algorithm 8.1, whereas circles mark the intervals for the MoPs learned with LIPs.

8.5.1.4 Results for one-dimensional densities

We analyzed the behavior of Algorithm 8.1 used to build MoP approximations of probability densities from data using artificial examples. For each one of the datasets in Table 8.1, we sampled ten training sets and ten test sets with $N = 50, 100, 500, 1000$ observations each. From each training dataset, we found a MoP approximation of the probability density underlying the data using Algorithm 8.1. We considered different values for the order of the

polynomials $r_X = 2, \ldots, 5$ and the number of intervals/pieces $L_X = 1, \ldots, 10$.

Figure 8.3 shows the MoP approximations for each of the one-dimensional datasets in Table 8.1. For the first repetition, the MoPs with the highest BIC score in the training dataset with $N = 1000$ are shown. The MoPs learned with the proposed Algorithm 8.1 (dashed lines) are continuous, non-negative and integrate to one. Equation (8.8) is an example of the MoP with the highest BIC score learned with Algorithm 8.1 for the `Mix1d` dataset ($L_X = 5$ and $r_X = 3$) in Figure 8.3(e):

$$\varphi_X(x) = \begin{cases} 0.1006 + 0.1266x - 0.0451x^2 & 0 \leq x \leq 2, \\ 0.3038 - 0.0766x + 0.0057x^2 & 2 \leq x \leq 4, \\ 0.4843 - 0.1668x + 0.0169x^2 & 4 \leq x \leq 6, \\ -1.0028 + 0.3289x - 0.0244x^2 & 6 \leq x \leq 8, \\ 1.6187 - 0.3265x + 0.0166x^2 & 8 \leq x \leq 10. \end{cases} \tag{8.8}$$

It is easy to check that the MoP in Equation (8.8) is continuous for $x = 2, 4, 6, 8$ and that the integral of $\varphi_X(x)$ in $\Omega_X = [0, 10]$ is 1.

Table 8.2 shows the numerical comparison of the MoPs learned using B-splines and LIPs for the one-dimensional datasets in Table 8.1. For each of the ten experiments, we selected the MoP with the highest BIC score in training (see Section 8.4.5). Table 8.2 reports the mean of the ten values for the number of pieces (L_X), order of the polynomials (r_X), number of free parameters to be estimated from data (#par), KL, MSE and MAE. The best results are highlighted in bold. We performed a non-parametric Wilcoxon paired signed-rank test to check whether or not the differences between the two methods, B-splines and LIPs, in the ten repetitions were significant. Statistically significant differences between the two methods at a significance level $\alpha = 0.05$ are marked with an asterisk. We can see that MoPs learned using B-spline interpolation with Algorithm 8.1 clearly outperformed MoPs learned using LIPs in datasets with high sample sizes ($N = 500, 1000$), whereas the two methods performed similarly in datasets with fewer observations ($N = 50, 100$). According to the KL and MSE, MoPs learned with LIPs did not significantly outperform MoPs learned with B-splines in any experiment. On the other hand, MoPs learned with B-splines frequently outperformed MoPs learned with LIPs, specially in datasets with higher sample sizes ($N = 500, 1000$). Regarding the MAE values, MoPs learned with LIPs significantly outperformed MoPs learned with B-splines in only one scenario (`Mix1d` dataset with $N = 50$). On the other hand, MoPs learned with B-splines significantly outperformed MoPs learned with LIPs according to MAE in three scenarios (`Gauss` dataset with $N = 500, 1000$ and `MixGauss` dataset with $N = 1000$). In general, MoPs learned with B-splines had a lower order r_X than MoPs learned with LIPs, whereas MoPs learned with LIPs had less pieces L_X than MoPs learned with B-splines. MoPs learned with B-splines frequently outperformed MoPs learned with LIPs regarding the number of free parameters that need to be estimated from data (#par). MoPs learned with LIPs did not significantly outperform MoPs learned with B-splines in any scenario according to #par.

Table 8.2: Comparison of MoPs learned from data using B-splines (Algorithm 8.1) or LIPs (Section 8.5.1.3). For each of the ten repetitions, the MoP with the highest BIC score in training was selected, and the mean values of the performance measures were reported: order of the polynomials (r_X), number of pieces (L_X), number of free parameters to estimate (#par), Kullback-Leibler divergence (KL), mean squared error (MSE) and maximum absolute error (MAE). The best (lowest) value for each dataset, training size and performance measure is shown in bold. Statistically significant results of one method with respect to the other one at $\alpha = 0.05$ are marked with an asterisk.

	B-splines					LIP				
	Gauss	Exp	Chisq	MixGauss	Mix1d	Gauss	Exp	Chisq	MixGauss	Mix1d
					$N = 50$					
r_X	**2.1***	**2.3***	**2**	**2.6**	**2**	3.8	3.5	2.3	3	**2**
L_X	2.7	**1.1**	**1**	2.9	1.2	**1.4***	1.2	**1**	**1.3**	**1**
#par	**3.8**	**2.4***	**2**	4.5	2.2	4.6	3.9	2.3	**3.8**	**2**
KL	0.2303	**0.0342**	**0.0259***	0.1606	0.0434	**0.0728**	0.0555	0.0384	**0.1205**	**0.0417**
MSE	**0.0045**	0.0199	**0.0010**	**0.0024**	0.0008	0.0045	**0.0148**	0.0013	0.0028	**0.0008**
MAE	0.1064	0.3018	0.2461	0.0922	0.1889	**0.1049**	**0.2564**	**0.2330**	**0.0880**	**0.1715***
					$N = 100$					
r_X	**2.3***	**2.1***	**2**	**2.4**	**2**	3.9	3.4	2.1	3	**2**
L_X	3.6	1.5	1.4	4.1	1.2	**1.5***	**1.1**	**1.1**	**2.1***	**1**
#par	**4.9**	**2.6***	2.4	**5.5**	2.2	5.3	3.6	**2.3**	6	**2**
KL	0.0359	**0.0265***	0.0311	**0.0623**	**0.0410**	**0.0356**	0.0551	**0.0301**	0.0715	0.0416
MSE	0.0019	**0.0159**	0.0013	**0.0015**	0.0008	**0.0019**	0.0186	**0.0012**	0.0018	**0.0008**
MAE	**0.0880**	**0.2670**	**0.2368**	**0.0707**	0.1909	0.0948	0.3041	0.2397	0.0821	**0.1825**
					$N = 500$					
r_X	**2.8**	**2.7**	**2.6**	**2.8***	2.9	3.5	3.4	3.1	3.8	**2.4**
L_X	3.2	**1.3**	1.4	5.6	2.9	**1.8***	1.9	**1**	**2.7***	**1.4***
#par	**5***	**3***	**3**	**7.4***	4.8	6	5.3	3.1	9.8	**3.6**
KL	**0.0069***	**0.0041***	**0.0190**	**0.0078***	**0.0205**	0.0148	0.0131	0.0215	0.0201	0.0305
MSE	**0.0003***	**0.0030**	**0.0009**	**0.0002**	**0.0004**	0.0008	0.0039	0.0009	0.0005	0.0006
MAE	**0.0336***	**0.1434**	0.2079	**0.0277**	**0.1389**	0.0552	0.1486	**0.1926**	0.0401	0.1657
					$N = 1000$					
r_X	**2.9**	**2.8***	**3.6**	**2.9***	**3.3**	3.3	4.2	4	3.8	**3.3**
L_X	3.2	1.6	2.7	5.2	4.2	**2.2***	**1.5**	**1***	**3.1***	**2.5***
#par	**5.1***	**3.4***	5.3	**7.1***	**6.5***	7.2	5.6	**4**	11.1	7.7
KL	**0.0043***	**0.0031***	**0.0071***	**0.0047***	**0.0068***	0.0087	0.0074	0.0151	0.0115	0.0120
MSE	**0.0003***	0.0022	**0.0003***	**0.0001***	**0.0002***	0.0006	**0.0018**	0.0006	0.0002	0.0004
MAE	**0.0311***	0.1199	**0.1551**	**0.0188***	**0.0982**	0.0511	**0.0927**	0.1724	0.0288	0.0995

Table 8.3: Goodness of fit of MoPs learned from data using B-splines (Algorithm 8.1) or LIPs (Section 8.5.1.3). For each of the ten repetitions, the MoP with the highest BIC score in training was selected. The mean of the BIC and the log-likelihood values for both the training and the test datasets were reported. The best (highest) value for each dataset, training size and performance measure is shown in bold. Statistically significant results of one method with respect to the other one at $\alpha = 0.05$ are marked with an asterisk.

	B-splines					LIP				
	Gauss	Exp	Chisq	MixGauss	Mix1d	Gauss	Exp	Chisq	MixGauss	Mix1d
					$N = 50$					
ℓ training	**-74.24**	-92.92	-129.81	**-105.46***	**-121.86***	-75.23	**-56.92***	**-129.56**	-108.87	-122.43
BIC training	**-81.68***	-97.61	**-133.72**	**-114.26***	**-126.17***	-84.23	**-64.54***	-134.06	-116.30	-126.34
ℓ test	-79.58	-93.26	**-129.87**	-114.77	-123.34	**-77.70**	**-58.95***	-129.95	**-110.73**	**-122.69**
BIC test	-87.01	-97.96	**-133.79**	-123.57	-127.64	**-86.70**	**-66.58***	-134.44	**-118.17**	**-126.60**
					$N = 100$					
ℓ training	**-140.39***	-202.99	**-298.30***	**-204.59***	**-238.32***	-141.46	**-107.77***	-298.90	-207.82	-238.96
BIC training	**-151.68***	-208.98	**-303.82***	**-217.26***	**-243.38***	-153.67	**-116.06***	-304.19	-221.64	-243.56
ℓ test	**-146.34**	-204.18	-301.20	**-209.87**	-239.94	-146.66	**-111.99***	**-300.92**	-211.39	**-239.51**
BIC test	**-157.62**	-210.16	-306.73	**-222.54***	-245.01	-158.87	**-120.28***	**-306.22**	-225.20	**-244.12**
					$N = 500$					
ℓ training	**-742.09***	-936.48	**-1430.24***	**-1034.89***	**-1251.60***	-744.45	**-505.79***	-1431.03	-1039.73	-1259.12
BIC training	**-757.62***	-945.80	**-1439.56***	**-1057.88***	**-1266.51***	-763.09	**-522.26***	-1440.67	-1070.18	-1270.31
ℓ test	**-746.29**	-936.57	**-1435.59**	**-1039.93***	**-1259.30***	-748.06	**-511.83***	-1436.60	-1049.00	-1265.48
BIC test	**-761.82***	-945.89	**-1444.91**	**-1062.93***	**-1274.22**	-766.70	**-528.30***	-1446.23	-1079.46	-1276.67
					$N = 1000$					
ℓ training	**-1465.49**	-1894.64	**-2820.43***	**-2086.03***	**-2432.05**	-1467.97	**-1026.93***	-2828.60	-2089.98	-2435.64
BIC training	**-1483.10***	-1906.39	**-2838.74***	**-2110.55***	**-2454.50***	-1492.84	**-1046.27***	-2842.42	-2128.32	-2462.23
ℓ test	**-1469.70***	-1896.43	**-2824.97***	**-2091.79**	**-2435.90***	-1473.61	**-1029.49***	-2832.55	-2099.55	-2440.96
BIC test	**-1487.32***	-1908.17	**-2843.28***	**-2116.31***	**-2458.35***	-1498.47	**-1048.84***	-2846.36	-2137.89	-2467.56

Additionally, Table 8.3 shows the mean log-likelihood and mean BIC values in the training and the test datasets for both methods. The best results are highlighted in bold, and statistically significant results according to a Wilcoxon signed-rank test are marked with an asterisk. MoPs obtained using LIPs yielded significantly better log-likelihood and BIC scores in training and test datasets for the `Exp` dataset. On the other hand, MoPs computed using B-spline interpolation yielded significantly better results in the other four datasets. There were few statistically significant differences between the two methods for datasets with small sample sizes ($N = 50, 100$). MoPs learned with B-splines outperformed those learned with LIPs more frequently in datasets with large sample sizes ($N = 500, 1000$).

Finally, we assessed the accuracy of the coefficients of the polynomial functions in the MoP approximations learned with Algorithm 8.1. We considered the following one-dimensional polynomial functions:

$$g_X(x) = 2.25 - 7.5x + 7.5x^2, \quad 0 \leq x \leq 1,$$

and

$$h_X(x) = 1.0746 + 0.8955x - 3.5821x^2 + 2.6866x^3, \quad 0 \leq x \leq 1.$$

These two polynomial functions are non-negative and integrate to one in the domain $\Omega_X = [0, 1]$. Therefore, they are valid densities in Ω_X. We used an acceptance-rejection sampling algorithm to generate datasets from $g_X(x)$ and $h_X(x)$ with different sample sizes ($N = 100, 1000, 10000, 100000, 1000000$). We learned MoP approximations from the datasets by applying Algorithm 8.1. We considered different values for the order $r_X = 2, \ldots, 5$ and the number of intervals $L_X = 1, \ldots, 10$, and selected the MoP with the highest BIC score. Ten independent repetitions were performed for each polynomial function ($g_X(x)$ and $h_X(x)$) and each sample size N. Table 8.4 shows the mean and the standard deviation of the absolute value of the difference between the coefficients b_i of the true polynomials ($g_X(x)$ and $h_X(x)$) and the coefficients $\widehat{b}_i, i = 0, \ldots, r_X - 1$ of the polynomial functions in the MoP approximations. We observe that both the means and the standard deviations decrease as we consider higher sample sizes N. This reduction is clearer for the coefficients of higher order monomials $b_i, i > 1$. These coefficients are the most important because they control the shape of the polynomial functions. For instance, the MoP approximations computed for the first repetition with $N = 1{,}000{,}000$ are

$$\varphi_{g,X}(x) = \begin{cases} 2.2402 - 7.4484x + 7.4547x^2 & 0 \leq x \leq 0.5, \\ 2.2344 - 7.4251x + 7.4315x^2 & 0.5 \leq x \leq 1, \end{cases}$$

for the $g_X(x)$ polynomial density, and

$$\varphi_{h,X}(x) = 1.0687 + 0.9352x - 3.5967x^2 + 2.6505x^3, \quad 0 \leq x \leq 1,$$

for the $h_X(x)$ polynomial density. We observe that the selected MoP approximation $\varphi_{g,X}(x)$

Table 8.4: Accuracy of the estimates of the polynomial coefficients of the MoPs learned with Algorithm 8.1. The table shows the mean and the standard deviation of the absolute value of the difference between the true coefficients b_i of the polynomials and the coefficients $\widehat{b}_i$ of the MoP approximations learned from data. Each coefficient b_i corresponds to the monomial term $x^{i-1}, i = 1, \ldots, r_X$ in the polynomial.

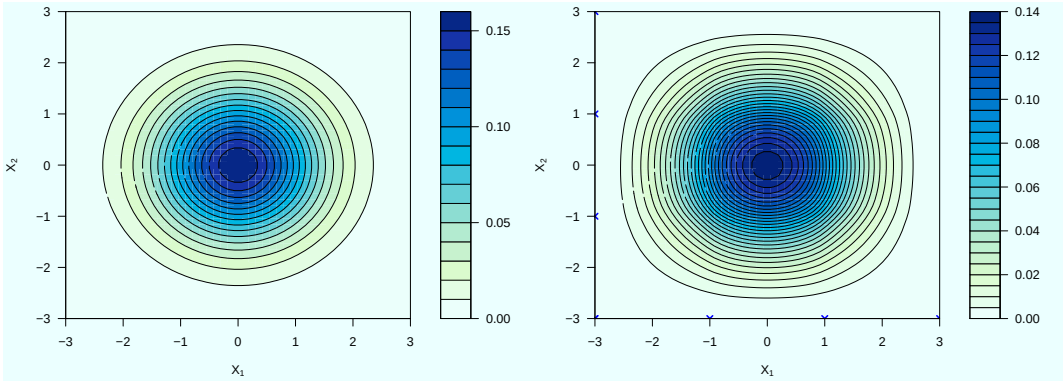
	$N = 100$	$N = 1{,}000$	$N = 10{,}000$	$N = 100{,}000$	$N = 1{,}000{,}000$
			$g_X(x) = 2.25 - 7.5x + 7.5x^2, 0 \leq x \leq 1$		
$\lvert b_0 - \widehat{b}_0 \rvert$	1.8962 ± 1.7411	0.5547 ± 1.1966	0.0636 ± 0.0618	0.0181 ± 0.0105	0.011 ± 0.0059
$\lvert b_1 - \widehat{b}_1 \rvert$	6.7028 ± 3.8739	1.8805 ± 3.0029	0.2435 ± 0.1909	0.0767 ± 0.0526	0.0586 ± 0.0238
$\lvert b_2 - \widehat{b}_2 \rvert$	6.4774 ± 2.3707	1.8563 ± 2.4548	0.2321 ± 0.1444	0.0714 ± 0.0547	0.0552 ± 0.0242
			$h_X(x) = 1.0746 + 0.8955x - 3.5821x^2 + 2.6866x^3, 0 \leq x \leq 1$		
$\lvert b_0 - \widehat{b}_0 \rvert$	0.0653 ± 0.0482	0.3546 ± 0.7428	0.3612 ± 0.4977	0.2747 ± 0.4189	0.0141 ± 0.0039
$\lvert b_1 - \widehat{b}_1 \rvert$	1.1466 ± 0.1293	1.3046 ± 0.6223	1.6654 ± 2.0572	1.3534 ± 1.7715	0.0815 ± 0.0263
$\lvert b_2 - \widehat{b}_2 \rvert$	3.5821 ± 0.0000	3.5821 ± 0.0000	2.9182 ± 2.4737	2.3931 ± 2.3841	0.0689 ± 0.0544
$\lvert b_3 - \widehat{b}_3 \rvert$	2.6866 ± 0.0000	2.6866 ± 0.0000	2.007 ± 1.0685	1.5805 ± 1.3263	0.0353 ± 0.0143

for the polynomial density $g_X(x)$ has two pieces ($L_X = 2$) instead of only one as it would be expected. However, both pieces have very similar coefficients to the true polynomial $g_X(x)$. We could use this kind of comparisons to simplify the MoP approximations obtained with Algorithm 8.1. On the other hand, the MoP approximation $\varphi_{h,X}(x)$ has only one piece, and its polynomial coefficients are very close to the coefficients of the true polynomial $h_X(x)$. We empirically conclude that the proposed Algorithm 8.1 asymptotically converges to the true probability density functions for these two polynomial densities, $g_X(x)$ and $h_X(x)$. Therefore, Algorithm 8.1 is able to retrieve these true polynomial models from the data.

8.5.1.5 Results for multidimensional densities

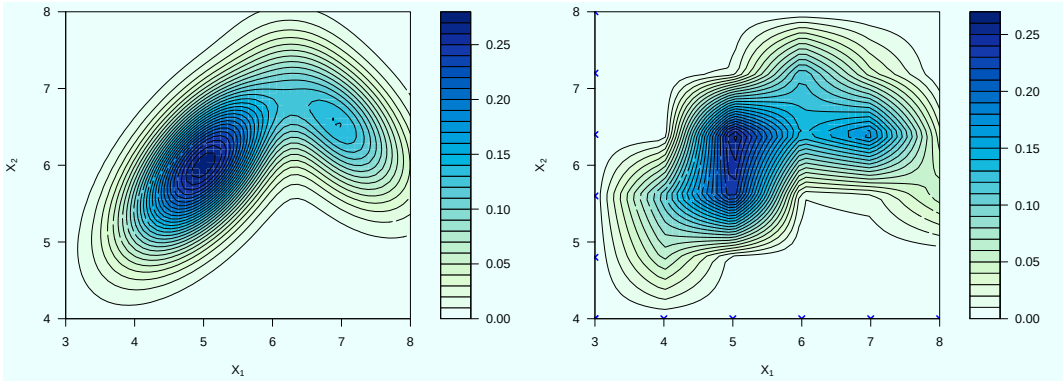
We analyzed the behavior of Algorithm 8.2 for learning MoP approximations of two-dimensional densities from data (see Table 8.1). For each probability density, we sampled ten datasets for each sample size $N = 50, 100, 500, 1000$. We only considered MoPs that had the same order and number of intervals for each dimension, i.e., $r_{X_1} = r_{X_2} = r_X$ and $L_{X_1} = L_{X_2} = L_X$. The values considered for these parameters were $r_X = 2, \ldots, 5$ and $L_X = 1, \ldots, 10$. For each combination of values of these parameters, we applied Algorithm 8.2 to learn a MoP approximation of the two-dimensional density from each of the ten training sets. As in the one-dimensional scenario, the multidimensional TSE cannot be used unless the mathematical expression of the true multidimensional joint density is known, so it is not applicable to learning directly from data. Additionally, estimating the densities at the interpolation nodes for LIPs is more difficult in multidimensional domains than in the one-dimensional scenario.

Figure 8.4 shows contour plots of the MoP approximations with the highest BIC score of one of the training sets and the respective true densities. Table 8.5 reports the values of the performance measures for the two-dimensional MoP approximations learned using B-splines. For the `Gauss2d` dataset, the proposed method yielded good results even with very small



(a) `Gauss2d`. True density

(b) `Gauss2d`. MoP approximation



(c) `Mix2d`. True density

(d) `Mix2d`. MoP approximation

Figure 8.4: Contour plots of the two-dimensional MoP approximations learned from a training dataset of $N = 1000$ observations. The true densities used to generate the training datasets are shown in (a) and (c). The MoPs with the highest BIC score learned with Algorithm 8.2 are shown in (b) and (d).

sample sizes ($N = 50, 100$), as the low KL divergence, MSE and MAE values show. On the other hand, the `Mix2d` dataset is more complex, and the algorithm needed more samples ($N = 500, 1000$) to yield good approximations to the true probability density. As expected, better approximations are obtained as we increase the training sample size. Regarding the complexity of the approximations, we observe that the MoPs frequently have low orders (r_X), whereas the number of parameters (#par) and the number of intervals (L_X) in each dimension increases with the sample size (N). We also observe that more parameters (#par) and intervals (L_X) are necessary to find MoPs approximations for the `Mix2d` dataset than for the `Gauss2d` dataset because the former is more complex than the latter.

We also show an example of a three-dimensional MoP learned with Algorithm 8.2. We used a dataset with $N = 1000$ observations sampled from the `Mix3d` density in Table 8.1. Figure 8.5 shows the contour plots of the true density and the MoP approximation for the domain $\Omega_{X_1} \times \Omega_{X_2}$ and three different values of X_3. We can see that the MoP approximation clearly replicates the two modes and is similar to the true multidimensional density.

Table 8.5: Evaluation of the two-dimensional MoPs learned from data using B-splines (Algorithm 8.2). The MoP with the highest BIC score was selected for each of the ten repetitions, and the mean values of the performance measures were reported: order of the polynomials (r_X), number of pieces (L_X), number of free parameters (#par), Kullback-Leibler divergence (KL), mean squared error (MSE) and maximum absolute error (MAE).

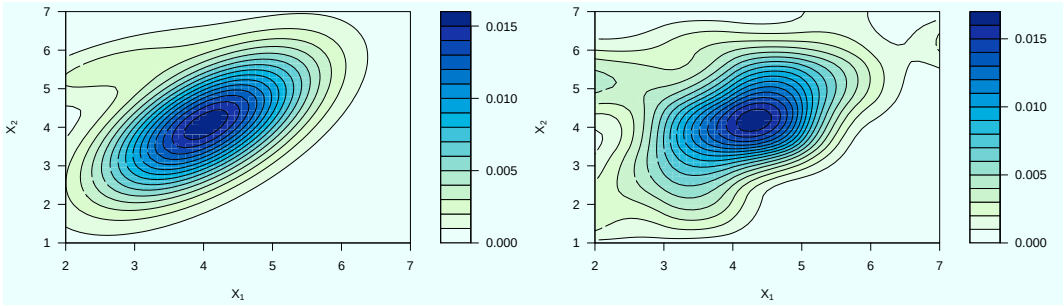
	$N = 50$		$N = 100$		$N = 500$		$N = 1000$	
	`Gauss2d`	`Mix2d`	`Gauss2d`	`Mix2d`	`Gauss2d`	`Mix2d`	`Gauss2d`	`Mix2d`
r_X	2	2	2	2	2.6	2	2.9	2
L_X	2	2	2	2.3	2.8	4.4	3.1	5.1
#par	9	9	9	11.1	20.2	29.8	25	37.3
KL	0.1182	0.3226	0.1127	0.2782	0.0433	0.1103	0.0116	0.0517
MSE	0.0010	0.0049	0.0010	0.0050	0.0003	0.0020	0.0001	0.0007
MAE	0.0583	0.1531	0.0574	0.1625	0.0364	0.1079	0.0276	0.0673

8.5.1.6 Results for conditional densities

In this section the two methods proposed for learning MoP approximations of conditional densities are shown. First, we show an example with two variables X and Y. We sampled a training dataset $\mathcal{D}_{X,Y}$ with $N = 5000$ observations from the two-dimensional Gaussian density corresponding to the `LinGauss` dataset in Table 8.1. Next, we applied Algorithm 8.3 to learn the MoP approximation of the conditional density of $X|Y$. The domain of the approximation was set to $\Omega_{X,Y} = [-3, 3] \times [-2, 2]$, which includes 0.9331 of the total Gaussian density mass. Note that $\sigma_Y^2 = 1$ is smaller than $\sigma_X^2 = 2$, thus the domain $\Omega_Y = [-2, 2]$ is smaller than Ω_X. We used the BIC score to greedily find the number of pieces (L_X, L_Y) and the order (r_X, r_Y) of the MoP. The number of pieces is considered to be the same for the two dimensions, i.e., $L_X = L_Y = L$. Similarly, the order is the same in the two dimensions ($r_X = r_Y = r$). We start considering one interval for each dimension ($L = 1$) and order $r = 2$ (linear polynomials). Then, we either increase the number of intervals to $L = 2$ or increase the order of the polynomials to $r = 3$. Finally, we choose the MoP with the highest BIC score out of the two MoPs (increasing L or r) and iterate until the BIC score does not increase.

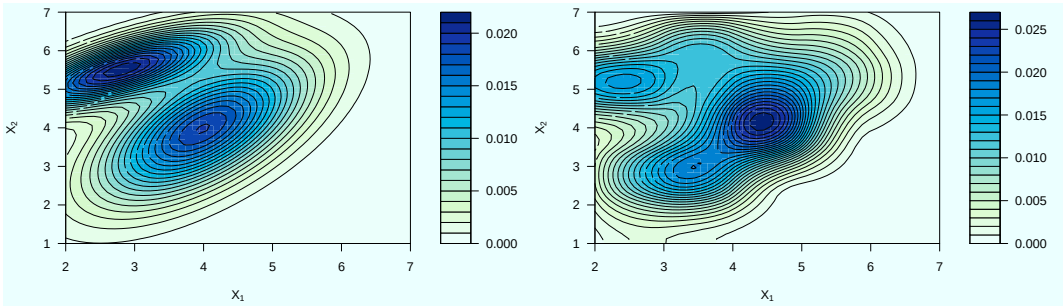
The conditional MoP $\varphi_{X|Y}(x|y)$ learned with Algorithm 8.3 is shown in Figure 8.6(a). The conditional MoP $\varphi_{X|Y}(x|y)$ had a total of 16 pieces ($L_X = L_Y = 4$) and order $r_X = r_Y = 2$, i.e., 64 polynomial coefficients. The true conditional density $f_{X|Y}(x|y)$ of $X|Y$ is the linear Gaussian density $\mathcal{N}(y, 1)$ shown in Figure 8.6(b). We can see that the conditional MoP $\varphi_{X|Y}(x|y)$ in Figure 8.6(a) is continuous and close to the true conditional density $f_{X|Y}(x|y)$. We observe high peaks at the "corners" of the domain $\Omega_{X,Y}$. These are due to numerical instabilities when evaluating the quotient $\varphi_{X,Y}(x, y)/\varphi_Y(y)$, caused by both the joint and the marginal MoPs yielding small values (close to zero) at the limits of the approximation domain.

Next, we performed inference based on the conditional MoP learned with Algorithm 8.3. Figures 8.6(c), (d) and (e) show the MoPs approximations $\varphi_{Y|X}(y|x)$ (solid) of the true posterior densities $f_{Y|X}(y|x)$ (dashed) for Y given three different values for X. The three



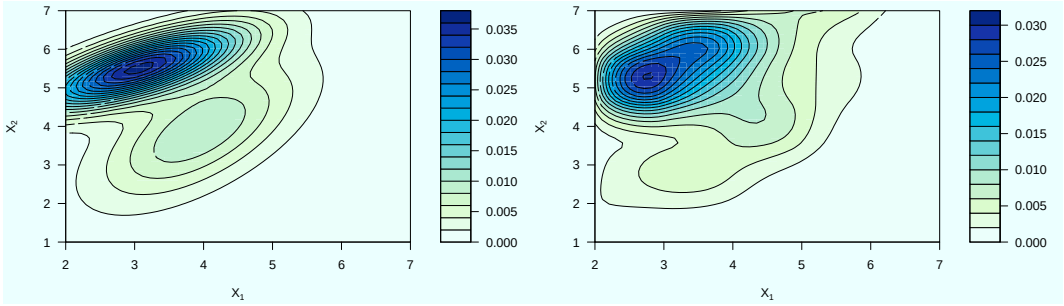
(a) $X_3 = 0.67$. True density

(b) $X_3 = 0.67$. MoP approximation



(c) $X_3 = 2.50$. True density

(d) $X_3 = 2.50$. MoP approximation



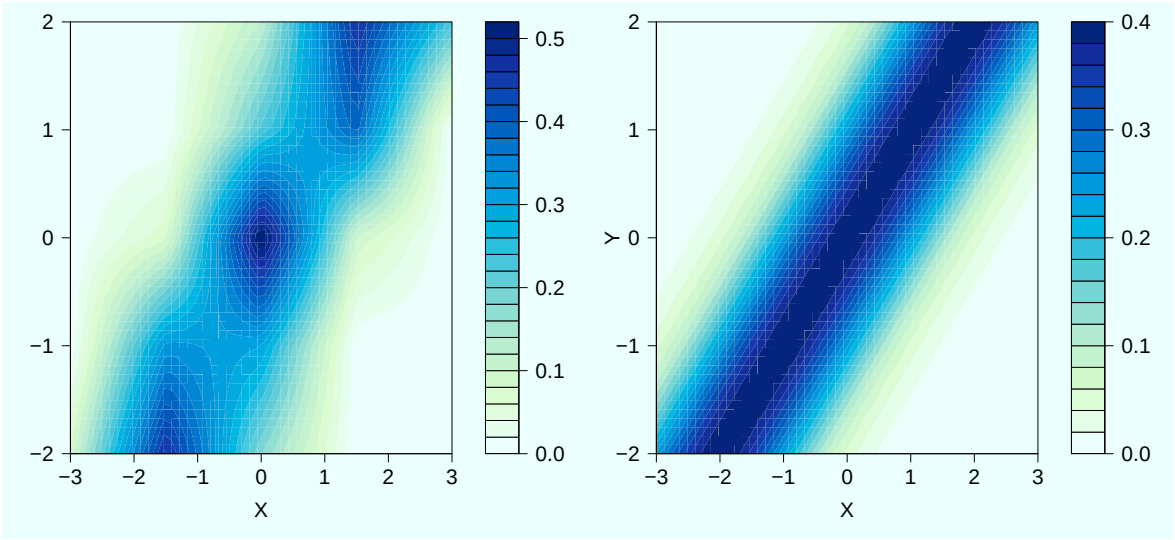
(e) $X_3 = 4.33$. True density

(f) $X_3 = 4.33$. MoP approximation

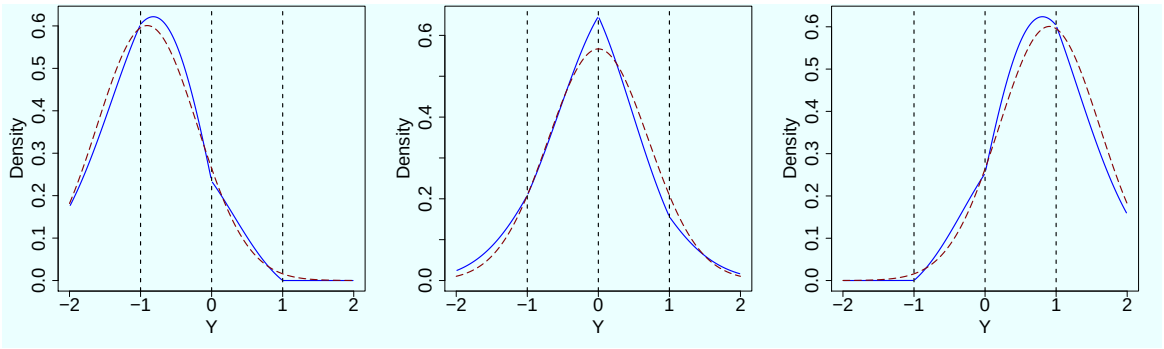
Figure 8.5: Contour plots of the three-dimensional MoP approximation learned from a training dataset of $N = 1000$ observations sampled from the `Mix3d` dataset. The true densities are shown in (a), (c) and (e). The respective densities of the MoP approximation with $r_{X_1} = r_{X_2} = r_{X_3} = 3$ and $L_{X_1} = L_{X_2} = L_{X_3} = 5$ are shown in (b), (d) and (f) for different values of X_3.

values $x = \{-1.81, 0, 1.81\}$ correspond to the percentiles 10, 50 and 90 of $X \sim \mathcal{N}(0, 2)$. Both the MoPs and the true posterior densities shown in Figures 8.6(c), (d) and (e) were normalized in the domain Ω_X so that they integrate to one. We can see that the MoPs of the posterior densities are also continuous and close to the true posterior densities.

We also applied Algorithm 8.4 to the `LinGauss` dataset (Table 8.1). We used the two-dimensional LIPs over the Padua points [73] as the polynomials $pol_{l_X,l_Y}(x, y)$ of the conditional MoP $\varphi_{X|Y}(x|y)$ (see Figure 8.7(a)). Algorithm 8.4 was used inside a greedy search to find the values of L and r as explained above. The conditional MoP with the highest BIC



(a) Conditional MoP $\varphi_{X|Y}(x|y)$ (b) True conditional density $f_{X|Y}(x|y)$

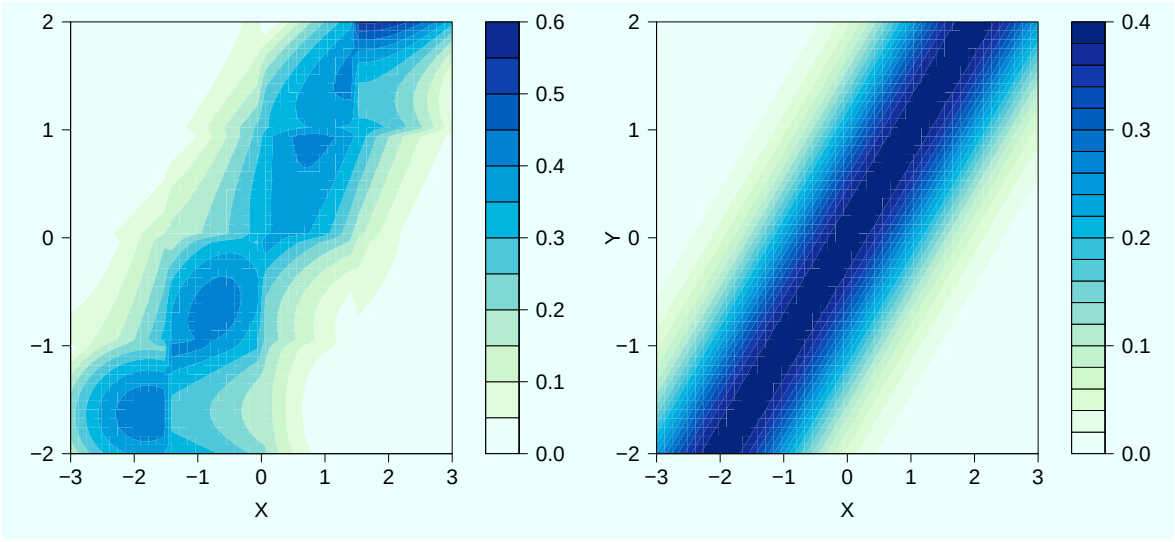


(c) Posterior MoP $\varphi_{Y|X}(y|-1.81)$ (d) Posterior MoP $\varphi_{Y|X}(y|0)$ (e) Posterior MoP $\varphi_{Y|X}(y|1.81)$

Figure 8.6: (a) Conditional MoP of $X|Y$ learned with Algorithm 8.3. (b) True conditional density of $X|Y \sim \mathcal{N}(y, 1)$. (c,d,e) MoP approximations (solid) and true posterior densities (dashed) of $Y|X$ for three values of X.

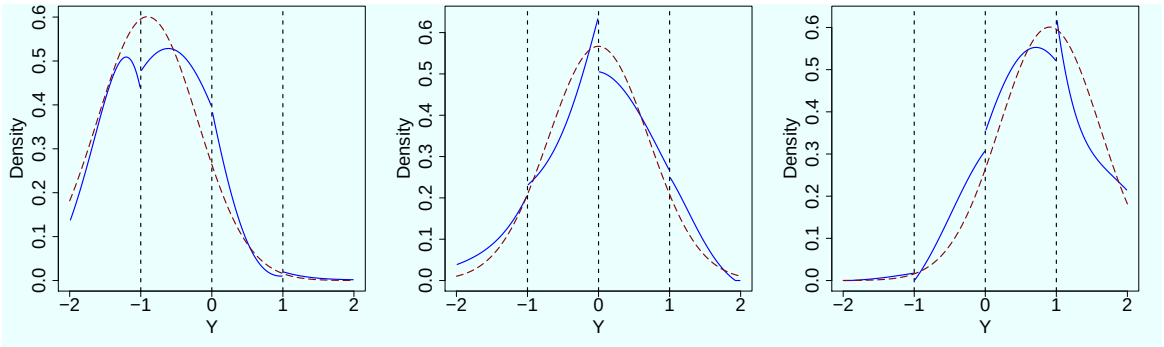
score had 16 pieces ($L_X = L_Y = L = 4$) and order $r_X = r_Y = r = 3$, i.e., 144 polynomial coefficients. We observe that the conditional MoP in Figure 8.7(a) is not continuous. Also, the MoPs of the posterior density in Figures 8.7(c), (d) and (e) are not continuous either.

Next, we compare the approaches proposed in this paper with that proposed in [316] for learning conditional MoTBFs from data. Figure 8.8(a) shows the MoTBFs of the conditional, and Figures 8.8(c), (d) and (e) show the posterior densities approximated using the `LinGauss` dataset (Table 8.1). The conditional MoTBF had $L_X = 6$ pieces and each piece defined a MoP with at most six parameters. MoTBF approximations of conditional densities are obtained by discretizing the parent variables and fitting a one-dimensional MoTBF for each combination of the discrete values of the parents. Compared with the two learning methods proposed in Algorithms 8.3 and 8.4, the method in [316] captures the correlation between the parent



(a) Conditional MoP $\varphi_{X|Y}(x|y)$

(b) True conditional density $f_{X|Y}(x|y)$



(c) Posterior MoP $\varphi_{Y|X}(y|-1.81)$

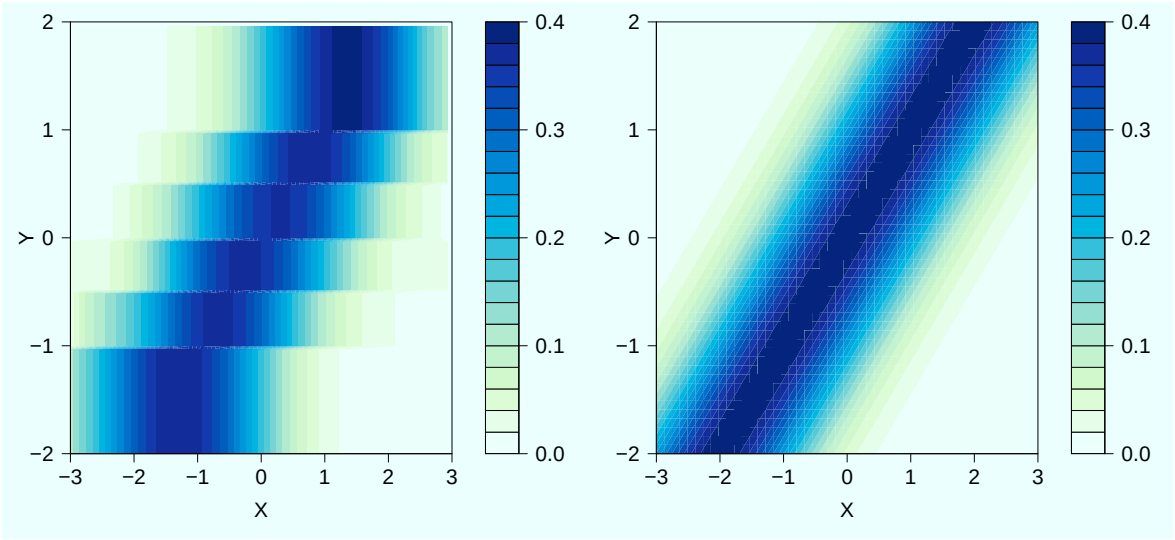
(d) Posterior MoP $\varphi_{Y|X}(y|0)$

(e) Posterior MoP $\varphi_{Y|X}(y|1.81)$

Figure 8.7: (a) Conditional MoP of $X|Y$ learned with Algorithm 8.4 using LIPs over the Padua points. (b) True conditional density of $X|Y \sim \mathcal{N}(y, 1)$. (c, d, e) MoP approximations (solid) and true posterior densities (dashed) of $Y|X$ for three values of X.

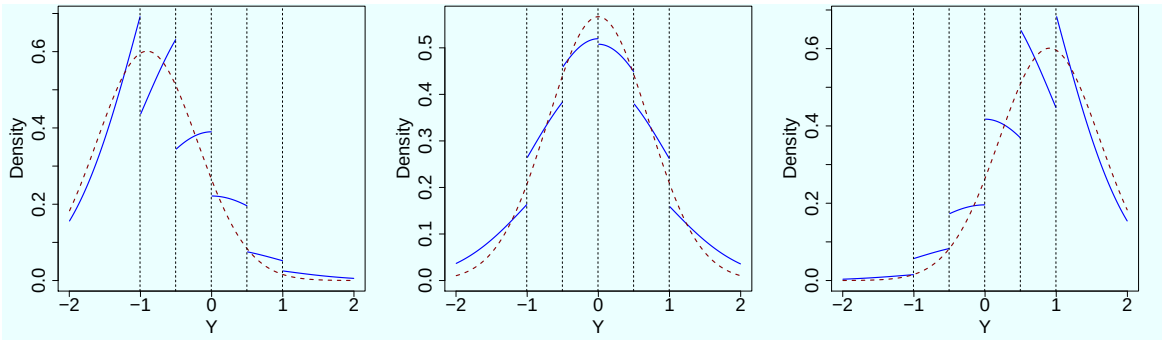
variables and the child variable through the discretization instead of directly in the functional polynomial expressions. Also, we observe that the posterior MoTBFs in Figures 8.8(c), (d) and (e) have $L_Y = 6$ pieces, whereas the posterior MoPs computed with Algorithms 8.3 and 8.4 in Figures 8.6 and 8.7, respectively, have only $L_Y = 4$.

If there is a weak correlation between the child and parent variables, then the conditional MoTBF approach in [316] is expected to yield approximations with few pieces. On the other hand, as the variables become more strongly correlated additional subintervals will be introduced by the learning algorithm. The MoTBF learning algorithm does not rely on a discretization of the child variable, but it rather approximates the density using a higher-order polynomial/exponential function. In contrast, Algorithms 8.3 and 8.4 yield conditional MoPs with more pieces because the domain of approximation $\Omega_{X,\mathbf{Y}}$ is split into hyperrectangles



(a) Conditional MoTBF $\varphi_{X|Y}(x|y)$

(b) True conditional density $f_{X|Y}(x|y)$



(c) Posterior MoTBF $\varphi_{Y|X}(y|-1.81)$

(d) Posterior MoTBF $\varphi_{Y|X}(y|0)$

(e) Posterior MoTBF $\varphi_{Y|X}(y|1.81)$

Figure 8.8: (a) Conditional MoTBF of $X|Y$ learned with the approach in [316]. (b) True conditional density of $X|Y \sim \mathcal{N}(y, 1)$. (c, d, e) MoTBF approximations (solid) and true posterior densities (dashed) of $Y|X$ for three values of X.

in all the dimensions. However, with the more fine-grained division of the domain into hyperrectangles, the polynomial functions of the conditional MoPs will usually have a low order.

We empirically compared the results of Algorithm 8.3, Algorithm 8.4 (using both TSEs and LIPs) and the method proposed in [316] for learning MoTBFs from data. We sampled ten datasets for each sample size ($N = 25, 500, 2500, 5000$) from the two-dimensional Gaussian density corresponding to the `LinGauss` dataset in Table 8.1. Table 8.6 shows the KL from the MoPs $\varphi_{Y|X}(y|x)$ to the true posterior densities $Y|X$ for three values of X. We applied a paired Wilcoxon signed-rank test and report statistically significant differences at a significance level $\alpha = 0.05$. The null hypothesis is that the two methods perform similarly. The alternative hypothesis is that the algorithm in the column outperforms the algorithm shown with a

Table 8.6: Kullback-Leibler divergences from the MoP approximations to the true posterior densities for the BN where $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$. The best results for each sample size are highlighted in bold. Statistically significant results of a Wilcoxon paired signed-ranks test ($\alpha = 0.05$) are shown with symbols ($*$, †, ‡, $\star$) indicating that the algorithm in the column outperforms the algorithm corresponding to the symbol.

$Y\|X$	Alg. 8.3 ($*$)		Alg. 8.4 TSE (†)		Alg. 8.4 LIP (‡)		MoTBF ($\star$)	
				$N = 25$				
$X = -1.81$	0.5032	†$\star$	0.7297		**0.3487**	$*$†$\star$	0.7084	†
$X = 0.00$	0.0746	‡$\star$	**0.0745**	$*$‡$\star$	0.1510		0.0939	‡
$X = 1.81$	**0.4952**	†‡$\star$	0.7297	‡	1.4582		0.7084	†‡
				$N = 500$				
$X = -1.81$	0.4194		0.2321	$*$‡	0.3161	$*$	**0.2191**	$*$‡
$X = 0.00$	**0.0239**	†‡$\star$	0.0646	$\star$	0.0453	†$\star$	0.0950	
$X = 1.81$	0.4141		0.2311	$*$‡	0.3701	$*$	**0.2170**	$*$‡
				$N = 2500$				
$X = -1.81$	0.1045		0.0850		0.1128		**0.0728**	$*$‡
$X = 0.00$	0.0387		0.0441		**0.0097**	$*$†$\star$	0.0272	$*$†
$X = 1.81$	0.0984		0.0978		0.1041		**0.0695**	$*$‡
				$N = 5000$				
$X = -1.81$	0.0575		0.0413		0.0341	$*$	**0.0308**	$*$
$X = 0.00$	**0.0196**		0.0262		0.0221		0.0210	
$X = 1.81$	0.0556		0.0425		0.0383		**0.0322**	$*$

symbol: $*$ for Algorithm 8.3, † for Algorithm 8.4 with TSE, ‡ for Algorithm 8.4 with LIPs, and $\star$ for conditional MoTBFs. For instance, a $\star$ in the column corresponding to Algorithm 8.3 in Table 1 shows that Algorithm 8.3 significantly outperformed MoTBFs for a given value of N and X. Algorithms 8.3 and 8.4 yielded competitive results against conditional MoTBFs.

8.5.1.7 Rmop: An R package for multidimensional mixture of polynomials learning from data

The proposed Algorithms 8.1 and 8.2 have been implemented in a freely available R package called `Rmop`. The package offers methods for learning multidimensional MoPs from data using B-spline interpolation. Also, it includes methods for managing MoPs, e.g., operations (sum, product, integration, marginalization, etc.), computation of statistics (mean, variance and covariance), comparison (KL divergence, MSE and MAE), plotting, etc. The package is available at http://cig.fi.upm.es/index.php/members/151-rmop/.

8.5.2 Experiments with Bayesian classifiers

In this section, we illustrate how to use the proposed methods for learning one-dimensional and multidimensional MoPs from data as a non-parametric density estimation technique in

Table 8.7: Datasets used in the BNC experiments.

Name	No. instances N	No. pred. variables n	No. class labels K
`appendicitis`	106	7	2
`fourclass`	862	2	2
`glass`	214	9	2
`haberman`	306	3	2
`ion`	351	32	2
`iris`	150	4	3
`liver`	341	6	2
`newthyroid`	215	5	3
`phoneme`	5,404	5	2
`svmguide1`	7,089	4	2
`vehicle`	846	18	4
`waveform`	5,000	21	3
`wdbc`	569	30	2
`wine`	178	13	3

BNCs. We retrieved 14 datasets from the UCI [27] and KEEL [9] repositories. We deleted the first variable in the `ion` dataset because it was discrete, i.e., it only took values 0 or 1. The values of the predictive variables were scaled to the domain $\Omega_{X_i} = [0, 1], i = 1, \ldots, n$. Table 8.7 shows the main features of the final datasets used in the experimentation.

NB [372] and TAN [199] classifiers were induced for each dataset. The probability of the class labels was modeled as a categorical distribution $p_C(c), c \in \Omega_C$. On the other hand, the conditional density of the predictive variables $\mathbf{X}$ given the class label $C = c$ was modeled using six different density estimation techniques, yielding twelve BNCs for comparison:

`NBMoP` and `TANMoP` use MoPs for non-parametric density estimation. The MoPs approximations $\varphi_{X_i|C}(x_i|c)$, $i = 1, \ldots, n$ were obtained using Algorithm 8.1 for NB classifiers. We considered different number of pieces $L_X = 1, \ldots, 10$ and orders $r_X = 2, \ldots, 5$, and we chose the MoP approximation with the highest BIC score (see Section 8.4.5). Similarly, the MoPs $\varphi_{X_iX_j|C}(x_i, x_j|c), i \neq j$ were approximated using Algorithm 8.2 for TAN classifiers. In TAN classifiers, we computed $\varphi_{X_i|C}(x_i|c)$ by marginalizing out X_j from $\varphi_{X_iX_j|C}(x_i, x_j|c)$. Note that we did not find explicit MoP approximations of the conditional densities $\varphi_{X_i|X_jC}(x_i|x_j, c)$ for TAN classifiers. Instead, we computed the values of the conditional densities by dividing the evaluation of the MoP $\varphi_{X_iX_j|C}(x_i, x_j|c)$ by the evaluation of the MoP $\varphi_{X_j|C}(x_j|c)$. We only considered two-dimensional MoPs $\varphi_{X_i|X_jC}(x_i|x_j, c)$ that had the same order and number of intervals for each dimension, i.e., $r_{X_i} = r_{X_j} = r_X$ and $L_{X_i} = L_{X_j} = L_X$. The values considered for these parameters were $r_X = 2, \ldots, 5$ and $L_X = 1, \ldots, 10$. The MoP approximation with highest BIC score was selected (see Section 8.4.5).

`NBKernel` and `TANKernel` use a Gaussian KDE technique as proposed by Pérez et al. [409]. The parameters and the densities of the one-dimensional and the multidimen-

sional Gaussian kernels were computed using the `ks` [154] and `KernSmooth` [509] R packages. A normal scale bandwidth was used for KDE.

`NBGauss` and `TANGauss` use Gaussian distributions to model the conditional densities as in a CLG network [408].

`NBFI` and `TANFI` use Fayyad and Irani's supervised discretization method [170] to discretize the continuous variables. ML estimates with Laplace correction were computed to fill in the CPTs in the NB and TAN classifiers.

`NBEF5` and `TANEF5` use an equal-frequency unsupervised discretization technique. Each variable is discretized into five bins. The parameters of the probability distributions were estimated using ML with Laplace correction.

`NBEF10` and `TANEF10` discretized each variable into ten equal-frequency bins, and the CPTs were filled in using the ML estimates with Laplace correction.

Once the probability distributions have been estimated, a new instance $\mathbf{x}$ is classified by applying the maximum a posteriori rule: $c^* = \arg\max_{c \in \Omega_C} p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|c)$, where $f_{\mathbf{X}|C}(\mathbf{x}|c)$ depends on the probability distribution used for modeling the predictive variables $\mathbf{X}$ and factorizes according to the graphical structure of the classifier (either NB or TAN).

Table 8.8 shows the mean accuracy achieved by each classifier in each dataset estimated using a stratified 10-fold cross-validation. The results show a clear example of the "no free lunch" theorem [520], as there is no algorithm that significantly outperforms the others in all the datasets. KDE-based classifiers (`NBKernel` and `TANKernel`) achieved the best results in five out of fourteen datasets, whereas the parametric Gaussian BNCs (`NBGauss` and `TANGauss`) and the proposed classifiers using MoPs (`NBMoP` and `TANMoP`) obtained the best result in three datasets. The null hypothesis of equal performance of all algorithms could not be rejected at a significance level $\alpha = 0.05$ using Friedman's test (p-value = 0.0524) [195]. On the other hand, Iman and Davenport's test [271] rejected the null hypothesis of equal performance of all algorithms (p-value = 0.0456). However, we found no significant differences when we studied all pairwise comparisons between algorithms [207]. Similarly, we found no significant differences between the algorithms when we analyzed NB classifiers and TAN classifiers separately. Most of the datasets included in the study had few instances for the different class values. This makes it difficult for the proposed methods to obtain MoPs that can model complex probability distributions, as we saw when fitting MoPs to samples from the `Mix2d` dataset (Section 8.5.1.5). Additionally, for small samples, the BIC score tends to select simple MoP models with fewer intervals.

`NBMoP`'s accuracy was higher than `NBKernel`'s accuracy in four datasets, whereas `NBKernel` outperformed `NBMoP` in nine datasets. However, we found no significant differences between the two methods when we considered all the datasets using a Friedman's and Iman-Davenpot. Additionally, we compared MoPs with KDE for each dataset independently. We applied a non-parametric paired Wilcoxon signed-rank test using the accuracy of the classifiers in the 10 folds of the cross-validation procedure. `NBMoP` significantly outperformed `NBKernel` in two

Table 8.8: Mean accuracy of the classifiers estimated using a stratified 10-fold cross-validation. The best result for each dataset is highlighted in boldface.

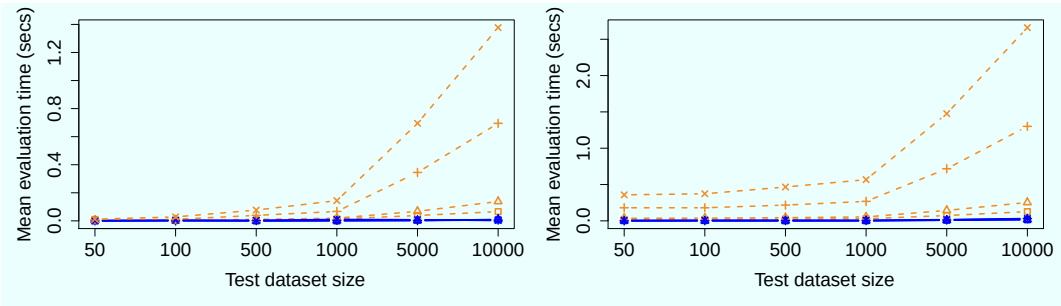
	NBMoP	NBKernel	NBGauss	NBFI	NBEF5	NBEF10
appendicitis	84.82	84.82	84.82	83.91	82.00	85.00
fourclass	85.49	83.87	75.41	78.18	76.91	83.41
glass	92.51	91.10	90.61	91.10	90.63	92.51
haberman	72.91	73.55	74.85	72.24	73.88	**75.85**
ion	86.06	90.87	81.17	89.16	88.59	88.87
iris	94.67	96.00	95.33	93.33	93.33	94.00
liver	62.76	67.45	57.18	57.75	63.94	61.29
newthyroid	94.87	**96.77**	**96.77**	94.89	95.41	95.87
phoneme	77.90	78.11	76.02	77.20	77.09	77.07
svmguide1	95.67	95.82	93.13	96.42	96.01	96.25
vehicle	64.06	60.51	46.21	61.22	58.63	63.23
waveform	81.08	80.70	80.90	80.78	80.82	80.64
wdbc	94.38	94.55	93.49	94.55	93.50	94.73
wine	96.60	98.33	98.33	98.33	98.33	96.67
	TANMoP	TANKernel	TANGauss	TANFI	TANEF5	TANEF10
appendicitis	82.91	81.91	79.09	84.91	**86.64**	80.91
fourclass	**98.84**	96.64	79.70	86.76	90.83	96.87
glass	90.17	92.47	92.49	92.03	91.13	**93.94**
haberman	**75.85**	74.18	74.84	72.24	69.97	73.52
ion	**92.29**	91.45	90.87	91.72	92.29	91.72
iris	88.00	**97.33**	**97.33**	93.33	94.00	91.33
liver	60.14	**70.69**	59.22	57.75	64.82	59.84
newthyroid	91.67	95.84	95.84	93.51	93.53	92.62
phoneme	80.53	80.31	77.87	80.46	80.64	**83.22**
svmguide1	95.70	95.97	93.75	**96.67**	96.23	96.39
vehicle	61.46	**77.90**	75.89	72.45	71.88	72.11
waveform	80.96	80.74	**82.28**	81.60	81.22	80.96
wdbc	94.02	95.25	95.43	94.37	94.91	**95.44**
wine	93.89	**100.00**	99.44	96.11	97.22	93.30

datasets (vehicle and waveform). On the other hand, TANMoP significantly outperformed TANKernel in only one dataset (fourclass), whereas TANKernel significantly outperformed TANMoP in four datasets (iris, liver, vehicle and wine). We could not find significant differences between MoP-based and KDE-based classifiers in most of the datasets. Therefore, we can conclude that NB and TAN classifiers using MoPs perform competitively against KDE-based NB and TAN classifiers.

BNCs with Gaussian densities and the three discretization algorithms yielded good performances. Discretization can help to reduce the noise in a dataset, specially when few training instances are available. We could not find significant differences between the BNCs using the three discretization algorithms. In fact, it has been shown that there is no discretization method that yields better BNCs in terms of accuracy for an extensive set of problems [190].

8.5.3 Comparison of evaluation times

We studied the evaluation time of the non-parametric density estimation techniques, i.e., MoPs and KDE. Other density estimation techniques such as discretization or assuming Gaussian densities were considered parametric alternatives and are expected to yield shorter evaluation times. Figure 8.9 shows the evaluation time of MoPs and KDE for two artificial datasets (see Table 8.1): the one-dimensional `Gauss` dataset and the two-dimensional `Gauss2d` dataset. Different training and test sizes were considered, and the evaluation times were averaged over 10 repetitions. Algorithms 8.1 and 8.2 were applied to learn the MoPs from the data and the BIC score was used to find appropriate values for the order and the number of intervals of the MoPs. We can see that the evaluation times for MoPs (solid lines) were almost constant, independently of the sizes of the training and test sets. On the contrary, the evaluation time for KDE (dashed lines) increased with both the training and test sizes. We can also see that the evaluation time for KDE increased from the one-dimensional (Figure 8.9(a)) to the two-dimensional scenario (Figure 8.9(b)). Additionally, MoPs are more efficient than KDE regarding storage. A MoP provides an explicit model of a probability density. Therefore, it only needs to store the coefficients of the polynomials and the limits of the intervals/hyperrectangles for each piece. On the other hand, KDE does not provide an explicit model, so it needs to save and analyze the complete training dataset to estimate the density of a new observation.



(a) One-dimensional `Norm` dataset (b) Two-dimensional `Norm2d` dataset

Figure 8.9: Comparison of evaluation time of MoPs (solid lines) with KDE (dashed lines) for: (a) the one-dimensional `Norm` dataset and (b) the two-dimensional `Norm2d` dataset (see Table 8.1). One line is shown for each training dataset size N: 50 (□), 100 (△), 500 (+) and 1000 (×). The mean evaluation time (in seconds) over ten repetitions is shown.

8.6 Conclusion

This chapter has presented a method for learning MoP approximations of the probability density underlying a dataset using B-spline interpolation. One-dimensional, multidimensional and conditional MoP approximations of probability densities were learned from data. A MoP was approximated as a linear combination of B-splines, since B-splines can be written as

MoPs, and MoPs are closed under multiplication and addition. The mixing coefficients of the linear combination were found using a ML approach. Thus we were able to perform model selection in a principled way by using a penalized likelihood criterion like the BIC score. Appropriate values for the order of the polynomials and the number of pieces of the MoPs were selected based on the BIC score of the MoPs. MoPs learned with B-splines have a number of advantages over other methods proposed in the literature, i.e., they are continuous, non-negative and integrate to one. These properties are important in some settings, e.g., for performing inference tasks. One-dimensional MoPs learned with B-spline interpolation were compared with MoPs using LIPs as proposed in [456]. Artificial datasets were used to evaluate the proposed method for learning MoP approximations. MoPs learned using B-spline interpolation outperformed MoPs learned with LIPs according to the Kullback-Leibler divergence, the mean squared error and the maximum absolute error between the MoPs and the true generating distributions. Also, the proposed method yielded MoPs with better generalization behavior according to the log-likelihood and BIC values computed in the test datasets. Conditional MoPs were compared with the approach proposed for learning MoTBFs in [316].

We also studied the use of MoPs as a non-parametric density estimation technique for BNCs for the first time. In particular, we studied and implemented two well-known BNCs: the NB classifier and the TAN classifier. NB and TAN classifiers using MoPs yielded competitive results against other state-of-the-art BNCs. MoPs offer some advantages over KDEs as non-parametric density estimators. First, MoPs provide an explicit model of the generating probability density. Second, the evaluation time for MoPs is shorter than for KDE. Therefore, BNCs using MoPs have faster classification times than those using KDE. Additionally, MoPs only need to store the model parameters (coefficients of the polynomials and limits of the intervals/hyperrectangles), whereas KDE has to save the complete training dataset. On the other hand, training time is longer for MoPs because the approach includes parameter estimation and model selection, although Equations (8.5) and (8.7) converge in few iterations [540]. Gaussian and discretization-based classifiers also yielded competitive results compared with the other methods. The parametric assumption of Gaussian densities may not hold in some settings. On the other hand, MoPs are a more flexible approach since they provide a non-parametric density estimation technique that can model any probability density without the need for discretization.

Future work will study the problem of finding the limits of the intervals/hyperrectangles A_{l_X} when they do not have the same width (non-uniform B-splines). Finding the best knot sequence given a dataset is expected to reduce the number of pieces necessary to find accurate MoPs. Some heuristics can be used to find a set of candidate points defining the limits of the intervals A_{l_X}, e.g., considering the local maxima, the local minima and the inflection points of the probability density function yielded good results for approximating with MTEs [435]. Then, a greedy search procedure could be used to split or merge adjacent intervals defined at these candidate points. Also, more complex methods can be found in the literature, for instance, knot density estimation [100], bootstrapping techniques [326], statistical testing

[474], regularization [391], Bayesian estimation [130, 138], etc.

Regarding the use of MoPs as a non-parametric density estimation technique in supervised learning problems, extensions to more complex BNCs will be considered, e.g., semi-naive Bayes classifiers, k-dependence BNCs, etc. Here, we followed a generative approach and computed ML estimates of the mixing coefficients of the linear combination of B-splines to build the MoPs that were used in the BNCs. However, the main goal in supervised learning problems is obtaining models that correctly classify the instances. Discriminative approaches to parameter fitting in BNCs look for parameters which maximize the classification accuracy or, alternatively, the conditional log-likelihood of the class variable given the predictive variables, e.g., see [77, 225]. We intend to investigate this discriminative approach for fitting BNCs with MoPs in the future.

Finally, we plan to perform a thorough comparison of the methods proposed for approximating with MoPs, MTEs and MoTBFs in hybrid BNs. These comparisons can be performed at different levels: quality of the approximations, modeling power, efficiency and computational complexity of the learning algorithms, discriminative power when used inside BNCs, etc. Also, BNs using MTEs have been applied to clustering [177, 205] and regression [175, 176, 377] problems. We would like to study and compare the use of MoPs for solving these different machine learning problems.

Chapter 9

Directional naive Bayes classifiers

9.1 Introduction

In Chapter 5 we highlighted the distinctive properties of directional data and the special techniques necessary to accurately manage them. Although directional data can be found in a lot of different domains, supervised classification problems including directional information as predictive variables have not been systematically studied by the machine learning research community. In fact, only 5 out of the 135 datasets for supervised learning available in the UCI Machine Learning Repository [27] include some variable measured in angles (see Section 9.3). To the best of our knowledge, the directional variables in those problems have been treated as linear continuous variables without taking into account the characterizing properties of the data.

In this chapter, we extend the NB classifier (see Section 3.4.2.1) for use with directional predictive variables. We study the decision functions of NB classifiers using von Mises distributions (see Section 5.2.2.1) or von Mises-Fisher distributions (see Section 5.3.2.1) to model directional data. We also consider hybrid scenarios with directional, linear and discrete predictive variables. The selective NB classifier [309] is adapted to work with these hybrid domains. We evaluate the proposed methods on a set of real problems and compare them with other BNCs that use Gaussian or discrete probability distributions for modeling the angular variables. A thorough analysis of the results is performed.

Classification problems using directional probability distributions have mainly been studied in the field of discriminant analysis. Morris and Laycock [378] studied the discriminant analysis of von Mises and Fisher distributions. Eben [155] analyzed the discriminant analysis of two von Mises distributions with unknown means and equal concentrations. Recently, discriminant analysis for von Mises-Fisher distributions was studied in [178], and misclassification probabilities for the von Mises distribution were estimated in two scenarios, i.e., considering populations with equal or different concentrations. Discriminant analysis has been studied for other directional distributions as well, e.g., Watson's, Selby's and Arnold's distributions in the sphere [159, 179]. In a related paper, SenGupta and Roy [449] proposed a classification rule based on the mean chord-length between an observation and two different

populations of angular data belonging to two different class labels. More recently, SenGupta and Ugwuowo [450] proposed a likelihood ratio test based on a bootstrapping approach to classify angular and linear data. These approaches show several differences to the one studied in this chapter. First, discriminant analysis focuses on the computation of misclassification probabilities. Here, we derive the decision functions of the NB classifiers and study them from a geometric point of view by analyzing the shape of the decision surfaces they induce. Second, these works only consider one predictive variable for classification. We study the decision functions for NB classifiers with two angular variables modeled with conditional (to the class) von Mises distributions. We also study NB classifiers with conditional von Mises-Fisher distributions. Additionally, we consider hybrid NB classifiers including linear, angular and discrete predictive variables at the same time. We also address the feature subset selection problem by adapting the selective NB classifier [309]. Finally, previous works only show the application of the techniques to one problem or dataset. In this paper, we perform an extensive evaluation of the proposed models on a set of real problems. This provides insights on the behavior of the directional NB classifiers and more general conclusions can be drawn.

Here, we only consider ML estimates of the parameters for the (conditional) von Mises and von Mises-Fisher probability densities. However, Bayesian parameter estimation for directional densities has received much interest, see e.g., [109, 231, 263, 353, 354].

The research included in this chapter has been published in López-Cruz et al. [337, 340].

Chapter outline

Section 9.2 introduces several extensions of the NB classifier including directional predictive variables, and their behavior is studied from a theoretical point of view. Section 9.3 includes the evaluation of these models using eight datasets and the statistical comparisons with other classifiers. Finally, conclusions and future research lines are discussed in Section 9.4.

9.2 Naive Bayes classifiers with directional predictive variables

In this section the von Mises naive Bayes (vMNB) classifier is introduced, which uses univariate von Mises distributions to model the conditional probability density functions of the angular variables. Next, the von Mises-Fisher naive Bayes (vMFNB) classifier is presented, where the conditional density functions of directional variables are modeled using multivariate von Mises-Fisher distributions. We derive the decision functions for each case and study the decision surfaces. Derivations of the decision functions and the surfaces that they induce are detailed in Appendices A and B. Hybrid scenarios with continuous and discrete predictive variables modeled using different probability distributions are a frequent occurrence in supervised classification. Therefore, we investigate the hybrid NB classifier in Sections 9.2.3 and 9.2.4, where the predictive variables are modeled using directional distributions and discrete or Gaussian distributions. Finally, the selective naive Bayes (SelNB) classifier is adapted to

work with directional distributions in Section 9.2.5.

9.2.1 The von Mises naive Bayes

In this section, we derive the decision surfaces of the vMNB, where the conditional probability densities of the predictive variables are modeled using von Mises distributions. First, in Section 9.2.1.1 we study the simplest approach where one predictive variable is considered. Then, we extend our analysis to the scenario where two predictive variables are used (Section 9.2.1.2).

9.2.1.1 vMNB with one predictive angular variable

We start with the simplest scenario, where vMNB has a binary class and only one predictive angular variable Φ.

Theorem 9.1. Let C be a binary class variable with values $\Omega_C = \{1, 2\}$. Let Φ be one predictive angular variable defined in the domain $\Omega(\Phi) = (-\pi, \pi]$, with conditional probability density functions modeled as von Mises distributions $M(\mu_{\Phi|c}, \kappa_{\Phi|c})$ for each class value $c \in \Omega_C$. Then, vMNB finds the two following decision angles that separate the class subregions

$$\begin{aligned} \phi' &= \alpha + \arccos(D/T), \\ \phi'' &= \alpha - \arccos(D/T), \end{aligned} \tag{9.1}$$

with the constants

$$\begin{aligned} D &= -\ln \frac{p_C(1) I_0(\kappa_{\Phi|2})}{p_C(2) I_0(\kappa_{\Phi|1})}, \\ \cos\alpha &= a/T, \\ \sin\alpha &= b/T, \\ T &= \sqrt{a^2 + b^2}, \\ a &= \kappa_{\Phi|1} \cos\mu_{\Phi|1} - \kappa_{\Phi|2} \cos\mu_{\Phi|2}, \\ b &= \kappa_{\Phi|2} \sin\mu_{\Phi|1} - \kappa_{\Phi|2} \sin\mu_{\Phi|2}. \end{aligned}$$

Proof. See Appendix A.1. □

Corollary 9.1. The vMNB classifier with a binary class and one predictive angular variable Φ is a linear classifier using the decision line

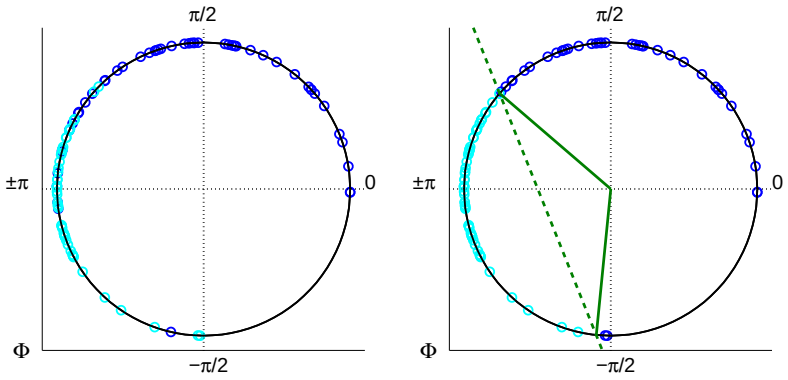
$$r(x, y) = (\kappa_{\Phi|1} \cos\mu_{\Phi|1} - \kappa_{\Phi|2} \cos\mu_{\Phi|2})x + (\kappa_{\Phi|2} \sin\mu_{\Phi|1} - \kappa_{\Phi|2} \sin\mu_{\Phi|2})y - D = 0,$$

where $(x, y) = (\cos\phi, \sin\phi)$ are the Cartesian coordinates in $\mathbb{R}^2$ of the point defined by the angle ϕ on the unit circle.

Proof. The proof is straightforward from Theorem 9.1 (see Appendix A.1). □

The vMNB classifier with one predictive circular variable modeled using von Mises conditional distributions divides the circle into two regions using two angles. Also, we can see vMNB as a linear classifier finding the line that goes through the points on the circumference defined by ϕ' and ϕ''. Angle α in (9.1) can be interpreted as a weighted average of the mean directions $\mu_{\Phi|c}$ for each class, using the values of the concentration parameters as weights. On the other hand, the length of the arc between the two angles (the distance that defines the size of the regions) is given by $\arccos(D/T)$, which depends on the concentrations, the mean directions and the prior probabilities of the class values. These prior probabilities are used in the logarithm in D. They influence the "size" of the class regions, moving the decision bounds so that more likely classes are given a larger subregion.

Figure 9.1a shows an example of a set of 100 points sampled from the conditional probability density distributions $\Phi|C = 1 \sim M(\pi/2, 2)$ and $\Phi|C = 2 \sim M(\pi, 5)$. The classes are considered equiprobable, i.e., $p_C(1) = p_C(2) = 0.5$. Figure 9.1b shows the class assigned to each angle by vMNB and the angles ($\phi' = 2.43$ and $\phi'' = -1.67$ radians) that define the class regions.



(a) (b)

Figure 9.1: True (a) and predicted (b) class for a set of 100 angles sampled from the conditional probability density distributions $\Phi|C = 1 \sim M(\pi/2, 2)$ and $\Phi|C = 2 \sim M(\pi, 5)$. Dark blue circles represent points for class $C = 1$ and light blue circles represent angles for class $C = 2$. The solid lines in (b) show the angles defining the bounds of each class region. The dashed line is the decision line induced by vMNB.

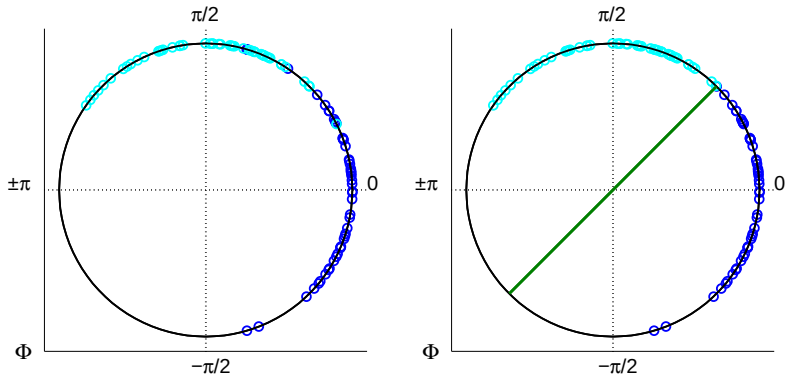
Particular cases To gain a thorough understanding of the classifier, we now study how these decision surfaces are defined for different values of parameters $\mu_{\Phi|c}$ and $\kappa_{\Phi|c}$. To study the decision bounds we consider that the classes are equiprobable, i.e., $p_C(1) = p_C(2) = 0.5$. This erases the influence of the prior probabilities of the class values.

Case 1: $\kappa_{\Phi|1} = \kappa_{\Phi|2}$ and $\mu_{\Phi|1} \neq \mu_{\Phi|2}$. When the two distributions share the same concentration value but have different mean directions, the decision angles are (see

Appendix A.1.1)

$$\phi' = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}),$$
$$\phi'' = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}) + \pi.$$

In this scenario, the decision surface is an axis that divides the circle into two semicircles (the angles are π radians apart). The axis goes through the center of the circle and is the bisector of the angle defined by the two mean directions. Figure 9.2a shows an example with a sample of 100 points drawn from the distributions $\Phi|C = 1 \sim M(0, 5)$ and $\Phi|C = 2 \sim M(\pi/2, 5)$. The classes are equiprobable a priori. vMNB finds an axis that forms an angle of $\pi/4$ with the horizontal axis and yields a semicircle for each class value (Figure 9.2b).



(a) (b)

Figure 9.2: True (a) and predicted (b) class for a sample of 100 angles where the conditional densities share the same concentration (Case 1). Dark blue circles represent points for class $C = 1$ and light blue circles represent angles for class $C = 2$. The green axis separates each class region in (b).

Case 2: $\kappa_{\Phi|1} \neq \kappa_{\Phi|2}$ and $\mu_{\Phi|1} = \mu_{\Phi|2} = \mu_\Phi$. vMNB finds the following angles when the mean directions are equal but the concentrations of the conditional distributions are different (see Appendix A.1.1)

$$\phi' = \mu_\Phi + \arccos \frac{D}{\kappa_{\Phi|1} - \kappa_{\Phi|2}},$$
$$\phi'' = \mu_\Phi - \arccos \frac{D}{\kappa_{\Phi|1} - \kappa_{\Phi|2}}.$$

The two angles are defined according to the shared mean direction μ_Φ, and the "spread" of the arc that they form is determined by the difference in the concentration parameters. The region including the mean direction always corresponds to the class with a

larger concentration. Figure 9.3a shows a set of 100 points sampled from the distributions $M(\pi/2, 2)$ and $M(\pi/2, 10)$. The two classes are equiprobable a priori. Figure 9.3b shows the classification provided by vMNB and the decision angles, which are both 0.47 radians away from the mean direction $\mu_\Phi = \pi/2$ (2.04 and 1.10 radians). The decision line is orthogonal to the mean direction μ_Φ and its position depends on the difference of the concentration values.

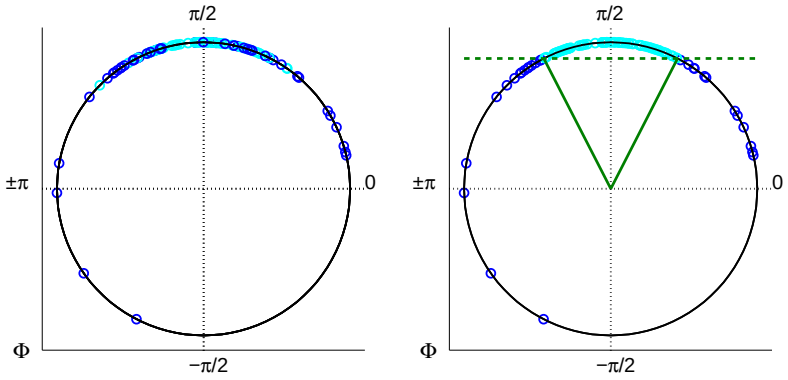


Figure 9.3: True (a) and predicted (b) class for a sample of 100 angles when the conditional densities share the same mean direction (Case 2). Dark blue circles represent points for class $C = 1$ and light blue circles represent angles for class $C = 2$. The solid lines in (b) show the angles defining each class region. The dashed line is the decision line induced by vMNB.

9.2.1.2 vMNB with two predictive angular variables

We now study the more complex scenario where two angular predictive variables Φ and Ψ are used in vMNB. The domain defined by the predictive variables is a torus $(-\pi, \pi] \times (-\pi, \pi]$.

Theorem 9.2. Let C be a binary class with values in $\Omega_C = \{1, 2\}$. Let Φ and Ψ be two angular variables defined in the domain $(-\pi, \pi]$. Let the conditional probability density functions of the variables Φ and Ψ be von Mises distributions $M(\mu_{\Phi|c}, \kappa_{\Phi|c})$ and $M(\mu_{\Psi|c}, \kappa_{\Psi|c})$. Then, the decision surface induced by the vMNB classifier is given by the 2-degree multivariate polynomials

$$\begin{aligned} clx + dly - az^2 + bz\sqrt{l^2 - z^2} + bLz + (aL + Dl)\sqrt{l^2 - z^2} + al^2 + DLl = 0, \\ clx + dly - az^2 - bz\sqrt{l^2 - z^2} + bLz - (aL + Dl)\sqrt{l^2 - z^2} + al^2 + DLl = 0, \end{aligned} \tag{9.2}$$

where (x, y, z) are the Cartesian coordinates in $\mathbb{R}^3$ of the points lying on the surface of the torus, and a, b, c, d, l, L and D are constants (see Appendix A.2).

The decision surfaces in (9.2) are quadratic in z, so vMNB is not a linear classifier when two predictive angular variables are used. The complexity of the classifier increases from the base scenario with one predictive variable (Section 9.2.1.1). This behavior differs from the

NB classifier with discrete variables, where the decision surfaces are always linear no matter the number of predictive variables.

We illustrate this scenario with an artificial example. Figure 9.4a shows a set of 1000 points sampled from the distributions $\Phi|C = 1 \sim M(\pi, 2)$ and $\Psi|C = 1 \sim M(-2\pi/3, 6)$ (shown in dark blue) and $\Phi|C = 2 \sim M(\pi/2, 5)$ and $\Psi|C = 2 \sim M(\pi, 3)$ (shown in light blue), and mapped into a torus. The two classes are equiprobable a priori. Figure 9.4b shows the classification provided by vMNB and the complex decision bounds induced by it, where we can see the non-linear behavior of the classifier.

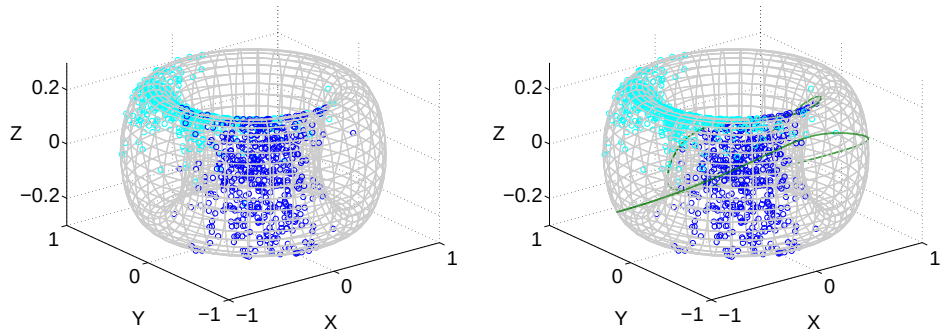


Figure 9.4: True (a) and predicted (b) class using vMNB for a sample of 1000 points. Points with $C = 1$ are shown in dark blue, whereas points with $C = 2$ are shaded light blue. The decision boundaries in (b) are drawn in green.

9.2.2 The von Mises-Fisher naive Bayes

The same approach can be used when the data in our problem are directional unit vectors in $\mathbb{R}^n$. These directional vectors can also be represented as points in the unit hypersphere $\mathbb{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n | \, \|\mathbf{x}\| = 1\}$ and modeled using the von Mises-Fisher distribution. Then, the classifier has only one n-dimensional predictive variable $\mathbf{X}$.

Theorem 9.3. Let C be a binary class variable with values $\Omega_C = \{1, 2\}$. Let $\mathbf{X}$ be a n-dimensional variable defined in the unit hypersphere $\mathbb{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n | \, \|\mathbf{x}\| = 1\}$. Let the conditional probability densities $\mathbf{X}|C = c$ follow a von Mises-Fisher distribution $M_n(\boldsymbol{\mu}_{\mathbf{X}|c}, \kappa_{\mathbf{X}|c})$. Then, vMFNB is a linear classifier yielding the decision hyperplane

$$(\kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}|2})^T\mathbf{x} + \ln \frac{p_C(1)(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{p_C(2)(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} = 0. \tag{9.3}$$

Proof. See Appendix B. □

Therefore, the decision surface in (9.3) is a hyperplane in $\mathbb{R}^n$ that divides the space into the two regions for classification. The intersection of the hyperplane and the hypersphere is

a circumference with the points that have the same posterior probability of being assigned to either class. The hyperplane can also be characterized by a non-zero normal vector and a point $\mathbf{x}_0$ belonging to the hyperplane. That characterization is easier to interpret. The hyperplane found by vMFNB is given by

$$(\kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}|2})^T(\mathbf{x} - \mathbf{x}_0) = 0. \tag{9.4}$$

Figure 9.5a shows an example in the sphere $\mathbb{S}^2$ of a set with 1000 points sampled from $\mathbf{X}|C = 1 \sim M_3((-1, 0, -0.2)^T, 10)$ (dark blue) and $\mathbf{X}|C = 2 \sim M_3((-0.5, -0.5, 1)^T, 20)$ (light blue). The classes are considered equiprobable a priori. If we replace the values of the parameters in the hyperplane expression (9.3), we get the plane $10x_2 - 22x_3 = -9.3069$. Alternatively, if we use the equation with the normal vector and the point (9.4), the plane that we get has the normal vector $(0, 10, -22)^T$ and contains the point $\mathbf{x}_0 = (0, 0, 0.4230)^T$. Figure 9.5b shows the classification given by vMFNB, the decision hyperplane and the circumference that bounds the class regions.

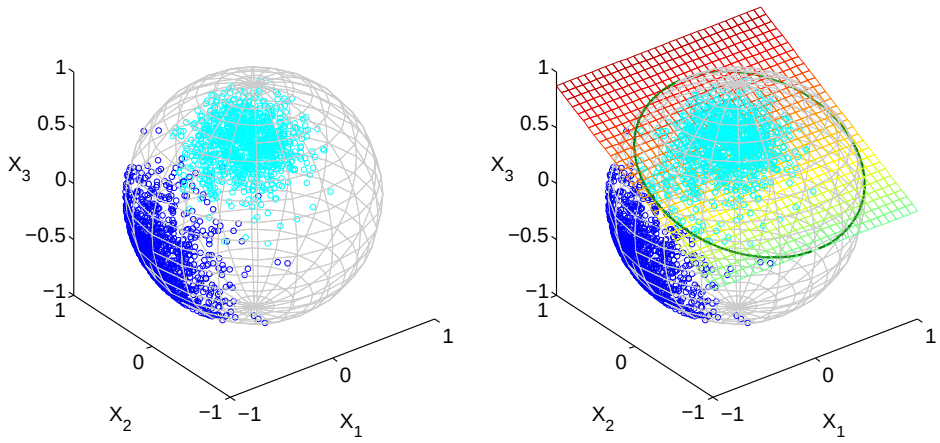


Figure 9.5: True class (a) and class predicted (b) using the von Mises-Fisher NB for a sample of 1000 points. Class $C = 1$ points are shown in dark blue, whereas class $C = 2$ data are drawn in light blue.

Figueiredo [178] also derived the decision function in (9.3). However, as far as we know, it is the first time that the geometric interpretation of the induced decision surface is studied at length, and the following special scenarios are analyzed.

9.2.2.1 Particular cases

We study the shape of the decision hyperplanes for some special cases when the conditional probability distributions share the value of one parameter. Like the analysis for the vMNB (Section 9.2.1.1), the classes are assumed to be equiprobable a priori.

Case 1: $\kappa_{\mathbf{X}|1} = \kappa_{\mathbf{X}|2}$ and $\boldsymbol{\mu}_{\mathbf{X}|1} \neq \boldsymbol{\mu}_{\mathbf{X}|2}$. When the concentration parameter values are the same but the mean directions are different, the hyperplane equation simplifies to (see Appendix B.1)

$$(\boldsymbol{\mu}_{\mathbf{X}|1} - \boldsymbol{\mu}_{\mathbf{X}|2})^T \mathbf{x} = 0. \tag{9.5}$$

Equation (9.5) defines a hyperplane that goes through the origin (center of the sphere), dividing it into two hemispheres. The plane goes through the "middle point" of the segment that contains the points in the hypersphere corresponding to the mean directions, like the bisector in vMNB. In fact, we can write the hyperplane equation as $(\boldsymbol{\mu}_{\mathbf{X}|1} - \boldsymbol{\mu}_{\mathbf{X}|2})^T(\mathbf{x} - \mathbf{0}) = 0$. Accordingly, vMFNB finds a hyperplane with the normal vector $(\boldsymbol{\mu}_{\mathbf{X}|1} - \boldsymbol{\mu}_{\mathbf{X}|2})^T$, which is the vector connecting the points in the hypersphere defined by the two mean directions. Additionally, the hyperplane contains the origin point ($\mathbf{0}$). In this case, since the plane goes through the center of the sphere, the intersection is a great circle (a.k.a. Riemannian circle), that is, one of the circles with the same radius as the sphere. The great circle and the hypersphere share the same center.

Figure 9.6a shows a set of 1000 points sampled from the distributions $\mathbf{X}|C = 1 \sim M_3((0,0,1)^T, 7)$ (dark blue) and $\mathbf{X}|C = 2 \sim M_3((0,1,0)^T, 7)$ (light blue). The classes have the same probability a priori. The classification provided by vMFNB can be seen in Figure 9.6b, where the decision hyperplane is given by the equation $-x_2 + x_3 = 0$.

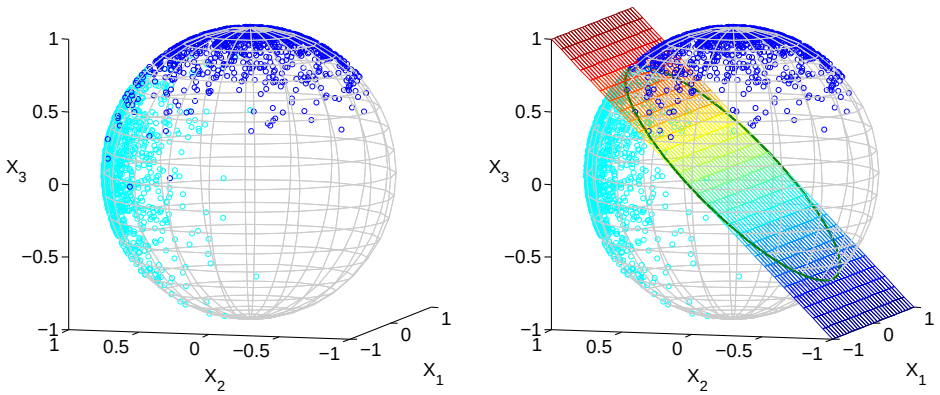


Figure 9.6: True class (a) and class predicted (b) when the conditional densities share the same concentration (Case 1). Class $C = 1$ points are shown in dark blue, whereas class $C = 2$ data are represented in light blue.

Case 2: $\kappa_{\mathbf{X}|1} \neq \kappa_{\mathbf{X}|2}, \boldsymbol{\mu}_{\mathbf{X}|1} = \boldsymbol{\mu}_{\mathbf{X}|2} = \boldsymbol{\mu}_{\mathbf{X}}$. In the scenario where the concentration parameters have different values but the mean directions are the same, vMFNB finds

the hyperplane (see Appendix B.1)

$$\boldsymbol{\mu}_{\mathbf{X}}^T(\mathbf{x}-\mathbf{x}_0)=0,$$
$$\text{with } \mathbf{x}_0 = \boldsymbol{\mu}_{\mathbf{X}}\frac{1}{\kappa_{\mathbf{X}|1}-\kappa_{\mathbf{X}|2}}\ln\frac{(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1}I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1}I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})}.$$

Therefore, vMFNB finds a hyperplane perpendicular to the mean direction and containing point $\mathbf{x}_0$. Point $\mathbf{x}_0$ is also located in the direction of the mean and its exact position depends on the values of the concentration parameters $\kappa_{\mathbf{X}|1}$ and $\kappa_{\mathbf{X}|2}$.

Figure 9.7a shows a set of 1000 points sampled from the distributions $\mathbf{X}|C=1 \sim M_3((-1,0,0)^T,20)$ (dark blue) and $\mathbf{X}|C=2 \sim M_3((-1,0,0)^T,5)$ (light blue). The two class values are equiprobable a priori. If we replace the values of the parameters in the hyperplane expression, we get the plane $x_1 = -0.9076$. Alternatively, if we use the equation with the normal vector and the point, the plane that we get has the normal vector $(-1,0,0)^T$ and contains the point $\mathbf{x}_0 = (-0.9076,0,0)^T$. Figure 9.7b shows the classification given by the vMFNB classifier, the decision hyperplane and the circumference that bounds the class regions.

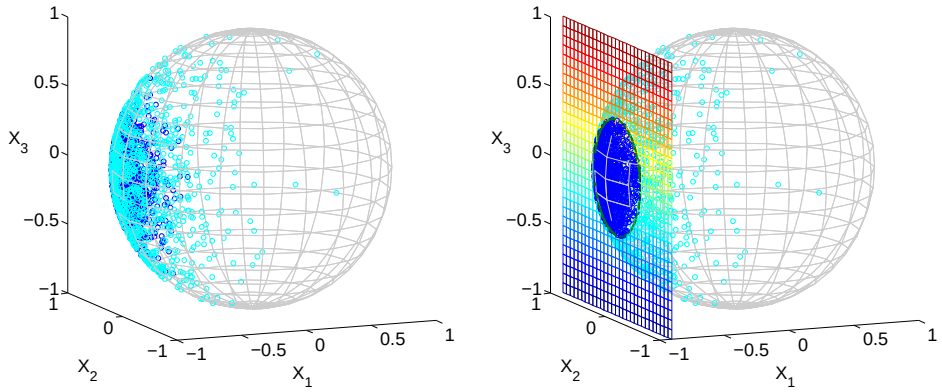


Figure 9.7: True class (a) and class predicted (b) when the conditional densities share the same mean direction (Case 2). Dark blue circles refer to class $C=1$ points and class $C=2$ data are drawn in light blue.

9.2.3 Hybrid Gaussian - von Mises-Fisher naive Bayes

A very interesting scenario arises when combining directional and non-directional data. This is a frequent situation when we can measure both the magnitude and the direction of a given phenomenon, e.g., the direction and the velocity of wind currents or the strength and orientation of a magnetic field. We study the hybrid NB classifier where the directional variable $\mathbf{X}$ is modeled using von Mises-Fisher distributions and the linear variable $\mathbf{Y}$ is modeled using

MG distributions. The conditional probability distributions of the predictive variables given the class value c are $\mathbf{X}|C = c \sim M_n(\boldsymbol{\mu}_{\mathbf{X}|c}, \kappa_{\mathbf{X}|c})$ and $\mathbf{Y}|C = c \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{Y}|c}, \boldsymbol{\Sigma}_{\mathbf{Y}|c})$.

The MG distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is defined by its two parameters: the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. The decision function $r(\mathbf{y})$ of a Gaussian NB [408] found by substituting this probability density function in (3.6) is:

$$\begin{aligned} r(\mathbf{y}) =& -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}|1})^T (\boldsymbol{\Sigma}_{\mathbf{Y}|1})^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}|1}) \\ &+ \frac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}|2})^T (\boldsymbol{\Sigma}_{\mathbf{Y}|2})^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}|2}) \\ &+ \ln \frac{p_C(1) \left|\boldsymbol{\Sigma}_{\mathbf{Y}|2}\right|^{1/2}}{p_C(2) \left|\boldsymbol{\Sigma}_{\mathbf{Y}|1}\right|^{1/2}}. \end{aligned} \tag{9.6}$$

Duda et al. [153] showed that the surfaces induced by that function are hyperplanes when $\boldsymbol{\Sigma}_{\mathbf{Y}|1} = \boldsymbol{\Sigma}_{\mathbf{Y}|2}$ and general hyperquadrics when $\boldsymbol{\Sigma}_{\mathbf{Y}|1} \neq \boldsymbol{\Sigma}_{\mathbf{Y}|2}$.

To compute the decision function for the hybrid Gaussian - von Mises-Fisher NB we have to substitute the von Mises-Fisher (5.8) and the Gaussian probability density functions in the decision function expression (3.6). Assuming conditional independence between $\mathbf{X}$ and $\mathbf{Y}$ given the class C, the decision function obtained after operating is the sum of two decision functions $r(\mathbf{x}, \mathbf{y}) = r(\mathbf{x}) + r(\mathbf{y})$, obtained in (9.3) and (9.6), but considering the prior probabilities $p_C(c)$ in only one of the components. Therefore, the shape of the surface induced by the function $r(\mathbf{x}, \mathbf{y})$ is determined by the most complex of the two components in the sum. We have shown that the decision surfaces defined by $r(\mathbf{x})$ are hyperplanes. Therefore, if the conditional probability distributions of the linear variable $\mathbf{Y}$ have the same covariance matrices, we have that the hybrid Gaussian - von Mises-Fisher NB finds a hyperplane to bound the class regions. On the other hand, if the covariance matrices are different, the decision surface is a general hyperquadric, ranging from simple hyperplanes to complex hyperhyperboloids [153]. We use an artificial example to illustrate this behavior.

The simplest model of this hybrid NB includes one circular variable $\mathbf{X} = (X_1, X_2)$ defined in the unit circumference $\mathbb{S}^1 = \left\{(x_1, x_2) \in \mathbb{R}^2 | x_1^2 + x_2^2 = 1\right\}$ and one linear variable Y defined in $\mathbb{R}$. The domain of the problem is the Cartesian product $\mathbb{S}^1 \times \mathbb{R}$, which defines a cylinder with unit radius. In this example the variable Y is 1-dimensional, so the covariance matrix is just the variance $\boldsymbol{\Sigma}_{\mathbf{Y}|c} = \sigma^2_{\mathbf{Y}|c}, c \in \{1, 2\}$.

9.2.3.1 Particular Cases:

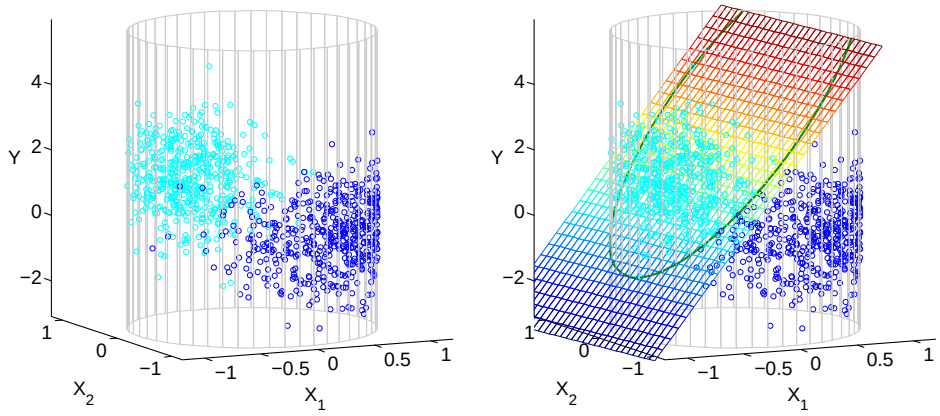
We analyzed the two cases described above for this model.

Case 1: $\sigma^2_{Y|1} = \sigma^2_{Y|2} = \sigma^2$. Substituting the probability density functions of the von Mises-Fisher and Gaussian distributions in the decision function (3.6) and arranging

all the terms, we get the following expression defining a hyperplane:

$$r(x_1, x_2, y) = (\kappa_{\mathbf{X}|1}\mu_{X_1|1} - \kappa_{\mathbf{X}|2}\mu_{X_1|2})x_1 + (\kappa_{\mathbf{X}|1}\mu_{X_2|1} - \kappa_{\mathbf{X}|2}\mu_{X_2|2})x_2 + \frac{\mu_{Y|1} - \mu_{Y|2}}{\sigma^2}y + \frac{\mu_{Y|2}^2 - \mu_{Y|1}^2}{2\sigma^2} + \ln\frac{p_C(1)I_0(\kappa_{\mathbf{X}|2})}{p_C(2)I_0(\kappa_{\mathbf{X}|1})}.$$

Figure 9.8a shows the true classification for 1000 points sampled using the distributions $\mathbf{X}|C = 1 \sim M_2((0.2, -0.8)^T, 5)$ and $Y|C = 1 \sim \mathcal{N}(0, 1)$ for points in class 1, and the distributions $\mathbf{X}|C = 2 \sim M_2((-0.8, -0.5)^T, 10)$ and $Y|C = 2 \sim \mathcal{N}(2, 1)$ for points with $C = 2$. Figure 9.8b shows the classes predicted by the hybrid Gaussian - von Mises-Fisher NB classifier and the hyperplane that separates the two class regions.



(a) (b)

Figure 9.8: True class (a) and class predicted (b) using the hybrid Gaussian - von Mises-Fisher NB classifier for a sample of 1000 points when the conditional Gaussian distributions share the same variance. Dark blue circles refer to class $C = 1$ points, whereas class $C = 2$ data are drawn in light blue.

Case 2: $\sigma_{Y|1}^2 \neq \sigma_{Y|2}^2$. In this scenario, the decision function obtained by the hybrid Gaussian - von Mises-Fisher NB is given by the following expression, which is quadratic for y:

$$r(x_1, x_2, y) = (\kappa_{\mathbf{X}|1}\mu_{X_1|1} - \kappa_{\mathbf{X}|2}\mu_{X_1|2})x_1 + (\kappa_{\mathbf{X}|1}\mu_{X_2|1} - \kappa_{\mathbf{X}|2}\mu_{X_2|2})x_2 + \frac{\sigma_{Y|1}^2 - \sigma_{Y|2}^2}{2\sigma_{Y|1}^2\sigma_{Y|2}^2}y^2 + \frac{\sigma_{Y|2}^2\mu_{Y|1} - \sigma_{Y|1}^2\mu_{Y|2}}{\sigma_{Y|1}^2\sigma_{Y|2}^2}y - \frac{\mu_{Y|1}^2}{2\sigma_{Y|1}^2} + \frac{\mu_{Y|2}^2}{2\sigma_{Y|2}^2} + \ln\frac{p_C(1)I_0(\kappa_{\mathbf{X}|2})\sigma_{Y|2}}{p_C(2)I_0(\kappa_{\mathbf{X}|1})\sigma_{Y|1}}.$$

Figure 9.9a shows a set of 1000 points sampled from the distributions $\mathbf{X}|C = 1 \sim M_2((0.2, -0.8)^T, 5)$ and $Y|C = 1 \sim \mathcal{N}(0, 0.25)$ for points in class 1, and the distributions

$\mathbf{X}|C = 2 \sim M_2((-0.8, -0.5)^T, 10)$ and $Y|C = 2 \sim \mathcal{N}(2, 4)$ for the points in the class 2. The classes are considered equiprobable a priori. The classification provided by the hybrid NB and the hyperquadratic decision surface that bounds the class regions are shown in Figure 9.9b.

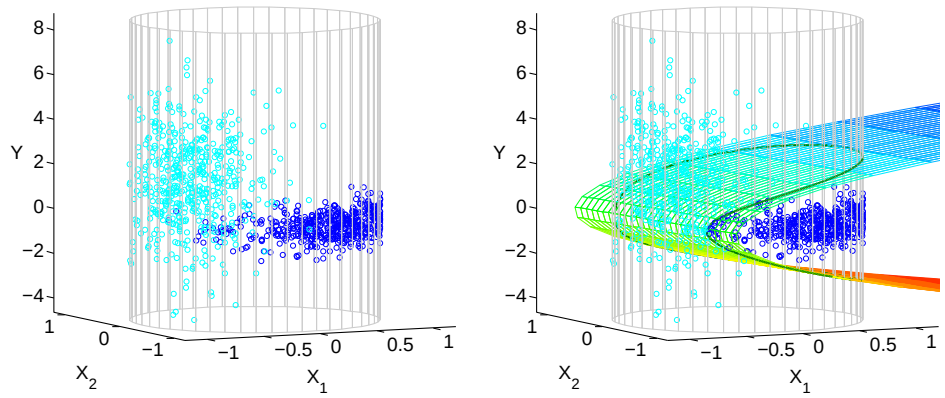


Figure 9.9: True class (a) and class predicted (b) using the hybrid Gaussian - von Mises-Fisher NB classifier for a sample of 1000 points when the conditional Gaussian distributions have different variances. Dark blue circles refer to class $C = 1$ points, whereas class $C = 2$ data are drawn in light blue.

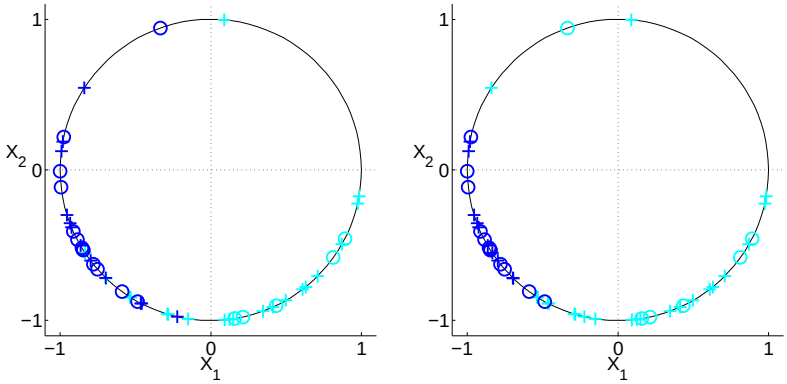
9.2.4 Hybrid discrete - Gaussian - von Mises-Fisher naive Bayes

Categorical data is also commonly found in different fields of science [5]. For example, binary variables can be used to indicate the presence or absence of a given trait in the phenomenon whose direction we are measuring. Discrete variables coding some qualitative aspect of the phenomenon can also be interesting for classification. Additionally, continuous variables with arbitrary distributions are usually discretized to make their analysis easier.

The NB classifiers presented above can be directly extended to the case including categorical predictive variables. Assuming that there are d discrete predictive variables $(X_1, \cdots, X_d)$, the classifier induces a set of decision surfaces, one for each possible combination of the values of the discrete variables. When analyzing a new instance $\mathbf{z}$, we first have to check the values of the discrete values to select the corresponding decision surface and use the values of the continuous variables for classification purposes.

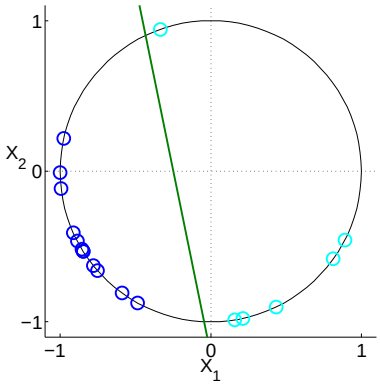
Adding discrete predictive variables would modify the independent term of the equation that specifies the decision surface, i.e., the probabilities of the discrete conditional distributions change the position of the decision surface but not its shape. Therefore, in the case of linear classifiers (vMNB, vMFNB and hybrid NB with equal covariance matrices), the decision hyperplanes found for every combination of values for the discrete predictors are all parallel to each other.

We use a simple artificial example of a NB classifier with two predictive variables to

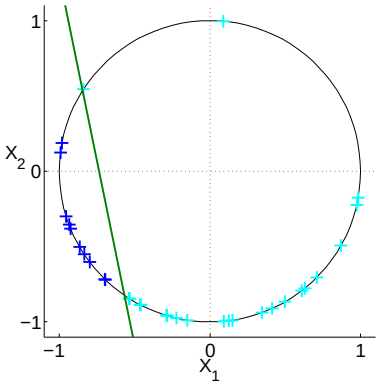


(a) True classification of the points.

(b) Predicted classification using the hybrid NB classifier.



(c) Predicted classification of the points and decision line for $Y = 1$.



(d) Predicted classification of the points and decision line for $Y = 2$.

Figure 9.10: True class and class predicted by the hybrid discrete - von Mises-Fisher NB classifier. Class $C = 1$ data are highlighted in dark blue, whereas class $C = 2$ data are highlighted in light blue. Circles are used to represent data with $Y = 1$ and crosses refer to data with $Y = 2$.

illustrate this point. We have a circular variable $\mathbf{X} = (X_1, X_2)$ defined in the unit circumference $\mathbb{S}^1 = \left\{(x_1, x_2) \in \mathbb{R}^2 | x_1^2 + x_2^2 = 1\right\}$ and one categorical variable Y that takes two values, e.g., $Y \in \{1, 2\}$. The class variable C is binary and its values are considered equiprobable. The conditional probability distributions of the predictive variables for class $C = 1$ are: $\mathbf{X}|C = 1 \sim M_2((0.2, -0.8)^T, 5)$ and $p_{Y|C}(1|1) = 0.15$. The conditional probability distributions for points with $C = 2$ are $\mathbf{X}|C = 2 \sim M_2((-0.8, -0.5)^T, 10)$ and $p_{Y|C}(1|2) = 0.6$.

A set of 50 points are drawn from those distributions and the true and predicted classifications are shown in Figures 9.10a and 9.10b, respectively. Figure 9.10c shows the points where $Y = 1$ and the decision line that bounds the class regions, whereas Figure 9.10d shows the same information for points $Y = 2$. The decision lines are clearly parallel. Note that the above analysis is also valid including linear MG distributions, although they have not been included in the artificial example for simplicity's sake.

9.2.5 Selective von Mises naive Bayes

NB classifiers are affected by redundant variables [309]. Finding good predictive variables can significantly increase the accuracy of NB. Langley and Sage [309] proposed the SelNB algorithm. SelNB finds the variables inducing the most accurate NB structure in a wrapper fashion. Pérez et al. [408] proposed a filter-wrapper approach to induce SelNB classifiers. First, the filter algorithm ranks the predictive variables using the MI between each variable and the class. Then each step of the wrapper algorithm induces a new classifier including the next predictive variable in the ranking. The algorithm uses the classification accuracy (computed with an inner 10-fold cross-validation procedure) to evaluate the models and selects the best classifier.

SelNB computes the MI between each predictive variable X_i and the class variable C. $MI(X_i, C)$ is the reduction of the entropy of the class given that we know the value of X_i. This measure represents the information that variable X_i gives about C. Therefore, higher values of MI relate to more informative variables. The MI between two variables X and Y is defined as

$$MI(X,Y) = \int_X \int_Y \rho_{X,Y}(x,y) \log \frac{\rho_{X,Y}(x,y)}{\rho_X(x)\rho_Y(y)} dxdy = \mathbb{E}_{(X,Y)} \left[\log \frac{\rho_{X,Y}(x,y)}{\rho_X(x)\rho_Y(y)} \right], \tag{9.7}$$

where ρ is a generalized probability function.

In supervised classification problems, we have to estimate $MI(X_i, C)$ from a set of data pairs (x_{ji}, c_j), $j = 1, \ldots, N$. When X_i is a discrete variable, an estimator of the MI in (9.7) is given by

$$MI(X_i, C) = \frac{1}{N} \sum_{j=1}^{N} \log \frac{p_{X_i,C}\left(x_{ji}, c^{(j)}; \widehat{\boldsymbol{\theta}}_{X_i,C}\right)}{p_{X_i}\left(x_{ji}; \widehat{\boldsymbol{\theta}}_{X_i}\right) p_C\left(c_j; \widehat{\boldsymbol{\theta}}_C\right)}, \tag{9.8}$$

where $\widehat{\boldsymbol{\theta}}_{X_i,C}$, $\widehat{\boldsymbol{\theta}}_{X_i}$ and $\widehat{\boldsymbol{\theta}}_C$ are the parameters of the probability distributions estimated from the counts in the dataset.

When the predictive variable X_i is continuous, we take an approach consistent with conditional independence assumptions and we model the conditional probability densities of $X_i|C = c$ as Gaussian or von Mises distributions, depending on the nature of the variable, i.e., linear or angular. Therefore, the marginal density of X_i is a mixture of Gaussian or von Mises distributions, respectively. Algorithm 9.1 shows the process for computing $MI(X_i, C)$.

Algorithm 9.1 (Estimation of $MI(X_i, C)$ with continuous X_i)

Inputs:

$\mathcal{D}_{X_i,C}$*: A dataset with* N *observations* $\mathcal{D}_{X_i,C} = \{(x_{1i}, c_1), \ldots, (x_{Ni}, c_N)\}$.

Output: The estimated value of $MI(X_i, C)$
Steps:

1. *Estimate the ML parameters* $\widehat{\boldsymbol{\theta}}_C$ *of the prior probability for each class value* $p_C(c) = N_c/N$, *where* N_c *is the number of instances in the dataset* $\mathcal{D}_{X_i,C}$ *belonging to class* $c \in \Omega_C$.

2. *Compute the ML estimates* $\widehat{\boldsymbol{\theta}}_{X_i|C}$ *of the parameters of the conditional density functions of* X_i *given each* $c \in \Omega_C$:

   If X_i *is a linear variable, fit a Gaussian probability density* $f_{X_i|C}(x|c;\widehat{\boldsymbol{\theta}}_{X_i|C})$ *where* $\widehat{\boldsymbol{\theta}}_{X_i|C} = \left\{\widehat{\mu}_{X_i|c}, \widehat{\sigma}^2_{X_i|c}\right\}_{c\in\Omega_C}$.

   If X_i *is an angular variable, fit a von Mises probability density* $f_{X_i|C}(x|c;\widehat{\boldsymbol{\theta}}_{X_i|C})$ *where* $\widehat{\boldsymbol{\theta}}_{X_i|C} = \left\{\widehat{\mu}_{X_i|c}, \widehat{\kappa}_{X_i|c}\right\}_{c\in\Omega_C}$.

3. *Sample* M *pairs of points* $\left(x^*_{ji}, c^*_j\right)$ *from the joint probability distribution* $\rho_{X_i,C}(x_i, c) = f_{X_i|C}(x_i|c;\widehat{\boldsymbol{\theta}}_{X_i|C})p_C(c;\widehat{\boldsymbol{\theta}}_C)$. *For each class value* $c \in \Omega_C$, *sample* $M\ \ p_C(c;\widehat{\boldsymbol{\theta}}_C)$ *instances from the density* $f_{X_i|C}(x_i|c;\widehat{\boldsymbol{\theta}}_{X_i|C})$ *and build the pairs* (x^*_{ji}, c^*_j).

4. *Compute the mutual information as*

$$\begin{aligned} MI(X_i, C) &= \frac{1}{M}\sum_{j=1}^{M} \log \frac{f_{X_i|C}\left(x^*_{ji}|c^*_j;\widehat{\boldsymbol{\theta}}_{X_i|C}\right) p_C\left(c^*_j;\widehat{\boldsymbol{\theta}}_C\right)}{f_{X_i}\left(x^*_{ji};\widehat{\boldsymbol{\theta}}_{X_i}\right) p_C\left(c^*_j;\widehat{\boldsymbol{\theta}}_C\right)} \\ &= \frac{1}{M}\sum_{j=1}^{M} \log \frac{f_{X_i|C}\left(x^*_{ji}|c^*_j;\widehat{\boldsymbol{\theta}}_{X_i|C}\right)}{\sum_{k\in\Omega_C} p_C(k;\widehat{\boldsymbol{\theta}}_C) f_{X_i|C}\left(x^*_{ji}|k;\widehat{\boldsymbol{\theta}}_{X_i|C}\right)}. \end{aligned} \tag{9.9}$$

The classifier learned by SelNB is a NB classifier which does not include all the predictive variables. This algorithm can discard irrelevant variables but still suffers from redundant variables. On the other hand, the wrapper algorithm proposed in [309] can discard both irrelevant and redundant variables. On the downside, however, it is less computationally efficient, since n^2 combinations of n predictive variables have to be tested in the worst-case scenario. The filter-wrapper algorithm uses a greedy heuristic to rank the variables according to the information they provide about the class. Accordingly, it has to test at most n classifiers. If the number of variables n is very large, we can limit the number of variables by setting $nmax < n$ in the wrapper step, and only $nmax$ subsets of variables are tested.

The complexity of the decision surfaces induced by SelNB depends on the number and the type of the variables selected in the final NB structure, as discussed in the previous sections.

9.3 Experiments

This section reports the results of the experimental evaluation of the classifiers presented in this chapter. Eight datasets were considered for evaluation (see Section 9.3.1). The performance of the different algorithms and the statistical comparison of the results are

Table 9.1: Datasets used in this study.

Dataset	# angular vars	# linear vars	# discrete vars	# class values	# instances
`Megaspores`	1	0	0	2	960
`Protein1`	2	0	0	2	49,676
`Protein10`	30	0	0	4	49,314
`Temperature`	1	1	1	3	8,753
`Auslan`	60	60	0	95	2,565
`MAGIC`	1	10	0	2	19,020
`Arrhythmia`	5	175	73	2	430
`Covertype`	2	8	44	7	100,000

included in Section 9.3.2. Section 9.3.3 illustrates the differences between using Gaussian and von Mises distributions to model angular data.

9.3.1 Dataset analysis and preprocessing

A thorough inspection of the datasets for supervised classification available in the UCI Machine Learning Repository [27] reported only five out of 135 datasets containing some variable measured in angles (bottom half of Table 9.1). We found no reference to these directional data having be given special treatment. For this reason we assume that they have been studied as linear continuous variables without taking into account their special properties. We omitted the Breast Tissue dataset [106, 284] from the study because it was not clear whether the "PhaseAngle" variable really represents an angle and how it was measured. Additionally, another four datasets not included in the UCI repository were considered for evaluation (top half of Table 9.1).

A description of the datasets used in this study follows:

9.3.1.1 UCI datasets

Australian sign language (`Auslan`): Identification of 95 Australian sign language signs using position (x, y, z) and orientation angles (roll, pitch, yaw) of both hands [287]. Therefore, 12 measurements are studied. According to [287], the bending measurements are not very reliable and they were omitted as predictive variables. This is a time series classification problem. The position and orientation of the hands are measured at different times, yielding approximately 54 data frames for each sign. We resampled a set of 10 evenly distributed frames and used them as predictive variables. According to the description, there are 95 different signs (class values), and each sign is repeated 27 times. However, the `her` sign only appears three times, whereas the `his-hers` sign appears 24 times. Therefore, we have assumed that they are the same sign and have considered them all as `his-hers` signs.

MAGIC gamma telescope (`MAGIC`): Discrimination of the images of hadronic showers initiated by primary gammas from those caused by cosmic rays in the upper atmosphere [48]. The images of the hadronic showers captured by the telescope are preprocessed and modeled as ellipses. The predictive variables describe the shape of

the ellipses. The dataset includes one angular variable that captures the angle of the major axis in the ellipse with the vector that connects the center of the ellipse with the center of the camera.

`Arrhythmia:` Identification of the presence and absence of cardiac arrhythmia from electrocardiograms (ECG). The original dataset has 16 class values: one for healthy items, 14 types of cardiac arrhythmias and one class value for unclassified items [232]. We erased the unclassified items and built a binary class (normal vs. arrhythmia). The predictive variables describe clinical measurements, patient data and ECG recordings. The angular variables describe the vector angles from the front plane of four ECG waves. We removed variable 14, which had more than 83% missing values, and used Weka's `ReplaceMissingValues` filter [236] to fill in the missing values of variables 11-13 and 15 with the mode. We also removed some non-informative discrete and continuous variables.

`Covertype:` Prediction of the kind of trees that grow in a specific area given some attributes describing the geography of the land [47]. The two angular variables describe the aspect (orientation) of the land from the true north and the slope of the ground. The original dataset has 581,012 instances and we used a Weka supervised resampling method (without replacement) to reduce the dimensionality of the dataset to 100,000 instances.

9.3.1.2 Other datasets

`Megaspores:` Classification of megaspores into two classes (their group in the biological taxonomy) according to the angle of their wall elements [304]. The dataset is an example included in Oriana software (see Section 5.4).

`Protein1:` Prediction of secondary structure including one aminoacid, using the dihedral angles (ϕ, ψ) of the residue as predictive information. We only considered α-helix and β-sheet structures, making the class binary. The data were retrieved from the protein geometry database [43].

`Protein10:` Prediction of secondary structure including one aminoacid, using the dihedral angles (ϕ, ψ) and the planarity angle (ω). We considered the three angles in ten consecutive residues. We classified the four most common structures: α-helices, β-sheets, bends and turns. The data were retrieved from [43].

`Temperature:` Prediction of the outdoor temperature from the season, wind speed and wind direction. We used hourly measurements from a weather station located in the city of Houston. Data for the year 2010 were retrieved, and we removed the hours with missing values for any of the four variables. The information was collected from the Texas commission on environmental quality website[1]. The class variable (outdoor

[1]The Texas commission on environmental quality website is available at: www.tceq.state.tx.us/

temperature) was measured in degrees Fahrenheit and discretized into the following three values: low ($T \leq 50$), medium ($50 < T < 70$) and high ($T \geq 70$).

9.3.2 Results

In this section we evaluate the performance of vMNB against other NB classifiers which ignore the angular nature of the data. We compared the following algorithms:

vMNB: NB classifier using Gaussian distributions for linear continuous variables and von Mises distributions for angular variables.

SelvMNB: Selective NB classifier, where the linear variables are modeled using Gaussian distributions and the angular variables are modeled using von Mises distributions.

GNB: Gaussian NB classifier where the probability density functions of all the continuous variables given the class values are modeled using Gaussian distributions.

SelGNB: Selective Gaussian NB classifier that uses Gaussian distributions for all the continuous predictive variables.

dNB: Discrete NB classifier where all the continuous variables are discretized using Fayyad and Irani's algorithm [170]. This classifier was run in Weka [236].

We use a stratified 10-fold cross validation technique to estimate the accuracy of the classifiers (see Section 2.3.2). The cross-validation procedure was run ten times independently. Therefore, 100 accuracy values are obtained. Table 9.2 shows the mean accuracy and the standard deviation for each dataset and each method. Table 9.3 shows the complexity of the final BNCs induced by the methods in the complete datasets averaged over ten independent runs, i.e., the number of parameters in the models, the number of predictive variables, the percentage of angular variables in the final classifier, and the elapsed time needed to learn the BNCs. We find that the performance of the classifiers using von Mises distributions for the angular predictive variables (vMNB and SelvMNB) is similar to or better than when Gaussian conditional probability distributions are used for those variables (GNB and SelGNB). dNB

Table 9.2: Mean accuracy and standard deviation of the BNCs evaluated on different datasets using ten runs of a stratified 10-fold cross validation.

Algorithm	vMNB	SelvMNB	GNB	SelGNB	dNB
`Megaspores`	76.50±3.56	76.50±3.56	**76.60**±3.58	**76.60**±3.58	75.22±3.37
`Protein1`	98.04±0.16	**98.39**±0.15	97.63±0.18	97.96±0.17	97.78±0.21
`Protein10`	83.98±0.55	86.14±0.54	80.77±0.60	82.11±0.48	**86.91**±0.50
`Temperature`	**74.08**±1.40	74.07±1.38	72.47±1.26	72.47±1.26	72.80±1.33
`Auslan`	64.47±3.01	81.72±2.80	64.39±3.03	**82.24**±2.44	78.24±2.81
`MAGIC`	72.75±0.92	75.26±0.82	72.68±0.92	74.92±0.88	**77.73**±0.81
`Arrhythmia`	76.52±6.63	78.19±6.09	76.47±6.56	78.17±6.16	**78.93**±6.30
`Covertype`	65.43±0.46	67.07±0.41	65.56±0.45	67.07±0.41	**68.49**±0.44

Table 9.3: Complexity analysis of the BNCs. For each dataset and each BNC, the table shows the number of parameters of the classifier (# par), the number of predictive variables (# vars), the percentage of angular variables out of the total number (# vars) of variables (% ang) and the elapsed time in seconds (time) used to learn the BNC. The results are averaged over ten runs. The complete datasets were used to learn the BNCs. We used Weka sofware to learn dNB, so the learning times are not comparable and have not been included.

	Megaspores	Protein1	Protein10	Temperature	Auslan	MAGIC	Arrhythmia	Covertype
				vMNB				
# par	5	9	243	23	22894	41	685	454
# vars	1	2	30	3	120	10	174	54
% ang	100	100	100	33.33	50	10	2.30	3.70
time	0.0006	0.0103	0.1494	0.0074	0.7274	0.0089	0.0496	0.8598
				SelvMNB				
# par	5	5	71	23	6687	13	363.40	20
# vars	1	1	8.50	3	34.70	3	90.60	1
% ang	100	100	100	33.33	58.98	33.33	1.60	0
time	0.1009	0.2375	9.4965	0.1113	159.7019	0.2860	8.8707	61.9942
				GNB				
# par	5	9	243	23	22894	41	685	454
# vars	1	2	30	3	120	10	174	54
% ang	100	100	100	33.33	50	10	2.30	3.70
time	0.0004	0.0052	0.0618	0.0029	0.4881	0.0075	0.0493	0.8449
				SelGNB				
# par	5	5	51.80	23	5832	13.40	469.20	20
# vars	1	1	6.10	3	30.20	3.10	117.10	1
% ang	100	100	100	33.33	60.26	32.50	2.57	0
time	0.0031	0.0300	4.3668	0.0206	127.4423	0.1692	8.3954	61.6814
				dNB				
# par	5	9	243	23	22894	41	685	454
# vars	1	2	31	3	120	10	174	54
% ang	100	100	96.77	33.33	50	10	2.30	3.70

using supervised discretization achieves competitive results against SelvMNB and SelGNB and yields the best results in four datasets (`Protein10`, `MAGIC`, `Arrhythmia` and `Covertype`). Note that the discretization algorithm can inherently perform some sort of feature selection by discretizing a variable in only one value. This could explain why dNB achieves such good results. Note also that dNB needs to estimate more parameters than SelvMNB and SelGNB in all the datasets but two (`Megaspores` and `Temperature`). For `Covertype`, SelvMNB and SelGNB achieved the same accuracy in all the folds. Neither algorithm selected either of the two angular variables (see Table 9.3), so SelvMNB and SelGNB induce exactly the same classifier for this problem and no significant differences can be found between them. The number of parameters in vMNB and GNB are the same because both Gaussian and von Mises distributions have two parameters and no feature subset selection is performed. However, GNB is slightly faster than vMNB because estimating the concentration of a von Mises density involves more operations than variance estimation for Gaussian densities. SelvMNB is also slower than SelGNB even when the number of selected variables is the same. Apart from having slower parameter estimation equations, the method used for sampling a von Mises density is computationally less efficient than the sampling algorithms for Gaussian densities. These sampling methods are used when computing the mutual information between each predictive variable and the class (see Algorithm 9.1). vMNB frequently outperforms GNB in those datasets with a higher percentage of angular variables, e.g., `Protein1`, `Protein10` or `Auslan`. This highlights the importance of using von Mises distributions for modeling angular data. SelvMNB and SelGNB included a similar percentage of angular variables in the final BNCs. In most scenarios, the same variables were selected by both SelvMNB and SelGNB. Therefore, when SelvMNB yields better results than SelGNB, it means that von Mises densities model the data in a better way than Gaussian densities (see Section 9.3.3).

Table 9.4 shows how each algorithm ranked on average across all datasets. SelvMNB is the highest-ranking algorithm, and we find that both vMNB and SelvMNB rank higher than their linear counterparts, GNB and SelGNB, respectively.

Table 9.4: Average ranking of the algorithms computed over all the datasets.

Algorithm	Average Ranking
SelvMNB	2.125
dNB	2.375
SelGNB	2.8125
vMNB	3.3125
GNB	4.375

Statistical methods for comparing algorithms over a set of problems were proposed in [129, 207]. We performed Nemenyi, Holm, Shaffer and Bergmann-Hommel post-hoc tests as computed in [207] to find statistical differences in the performance of all pairs of algorithms. Table 9.5 shows the adjusted p-values reported by these methods. Considering all the datasets, only the differences between the performance of GNB and SelvMNB are statistically significant (significance level $\alpha = 0.05$).

Table 9.5: Adjusted p-values of post-hoc tests when performing all pairwise comparisons between classifiers. Statistically significant results at $\alpha = 0.05$ are highlighted in bold.

H_1	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
SelvMNB $\neq$ GNB	**0.0443**	**0.0443**	**0.0443**	**0.0443**
GNB $\neq$ dNB	0.1141	0.1027	0.0685	0.0685
GNB $\neq$ SelGNB	0.4811	0.3849	0.2886	0.1924
vMNB $\neq$ SelvMNB	1.0000	0.9315	0.7985	0.7985
vMNB $\neq$ GNB	1.0000	1.0000	1.0000	0.7985
vMNB $\neq$ dNB	1.0000	1.0000	1.0000	0.7985
SelvMNB $\neq$ SelGNB	1.0000	1.0000	1.0000	1.0000
vMNB $\neq$ SelGNB	1.0000	1.0000	1.0000	1.0000
SelGNB $\neq$ dNB	1.0000	1.0000	1.0000	1.0000
SelvMNB $\neq$ dNB	1.0000	1.0000	1.0000	1.0000

Table 9.6: Results of a Wilcoxon signed-ranks test using the sorted difference in a 10-fold cross-validation averaged over 10 runs.

	vMNB vs. GNB		vMNB vs. dNB		GNB vs. dNB	
	H_1	p-value	H_1	p-value	H_1	p-value
`Megaspores`	$<$	0.2734	$>*$	0.0420	$>*$	0.0244
`Protein1`	$>*$	0.0010	$>*$	0.0010	$<*$	0.0068
`Protein10`	$>*$	0.0010	$<*$	0.0010	$<*$	0.0010
`Temperature`	$>*$	0.0020	$>*$	0.0098	$<$	0.1875
`Auslan`	$>$	0.3125	$<*$	0.0010	$<*$	0.0010
`MAGIC`	$>*$	0.0195	$<*$	0.0010	$<*$	0.0010
`Arrhythmia`	$>$	0.4375	$<$	0.1611	$<$	0.1377
`Covertype`	$<*$	0.0186	$<*$	0.0010	$<*$	0.0010
	SelvMNB vs. SelGNB		SelvMNB vs. dNB		SelGNB vs. dNB	
	H_1	p-value	H_1	p-value	H_1	p-value
`Megaspores`	$<$	0.2734	$>*$	0.0420	$>*$	0.0244
`Protein1`	$>*$	0.0010	$>*$	0.0010	$>*$	0.0098
`Protein10`	$>*$	0.0010	$<*$	0.0020	$<*$	0.0010
`Temperature`	$>*$	0.0020	$>*$	0.0098	$<$	0.1875
`Auslan`	$<$	0.2461	$>*$	0.0020	$>*$	0.0020
`MAGIC`	$>*$	0.0098	$<*$	0.0010	$<*$	0.0010
`Arrhythmia`	$>$	0.5098	$<$	0.4229	$<$	0.3477
`Covertype`	$\neq$	1.0000	$<*$	0.0010	$<*$	0.0010

Some datasets (see Table 9.1) include few angular variables (`Covertype`, `Arrhythmia`, `MAGIC`). Modeling these variables with a von Mises distribution or a Gaussian distribution is likely to have little impact on classifier accuracy. Therefore, it is worthwhile comparing algorithm performance on each dataset individually. Bouckaert [55] recommends using a t-test with a sorted runs sampling scheme to evaluate replicability of classifier learning experiments. He also states that this procedure yields an acceptable type I error and good power. We used a non-parametric alternative and applied a Wilcoxon signed-ranks test with the sorted runs

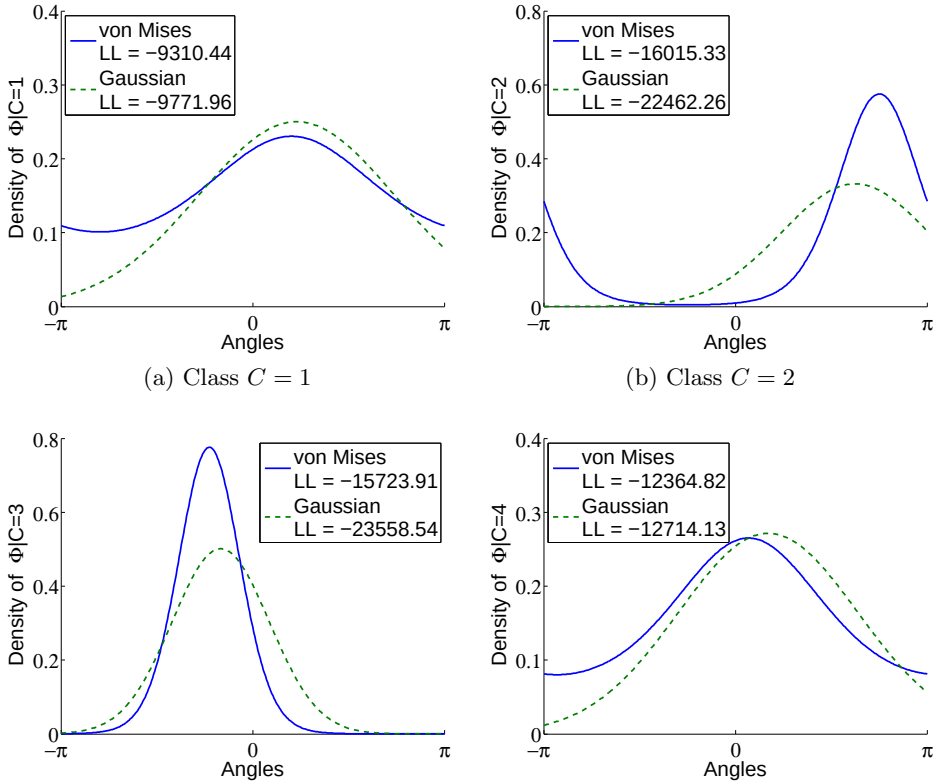
sampling scheme. Table 9.6 shows the p-values of the Wilcoxon signed-ranks test over the sorted difference of accuracies for a 10-fold cross-validation averaged over 10 runs. The null hypothesis is that the median of the averaged differences is zero, i.e., both algorithms perform similarly. The alternative hypotheses (H_1) were selected according to the results reported in Table 9.2. Statistically significant results at a significance level $\alpha = 0.05$ are highlighted with an asterisk ($*$). vMNB significantly outperformed GNB in four datasets (`Protein1`, `Protein10`, `Temperature` and `MAGIC`), whereas GNB only outperformed vMNB in the `Covertype` problem. We found no statistical differences between the two classifiers for the `Megaspores`, `Auslan` and `Arrhythmia` datasets. Similar results were found when comparing SelvMNB and SelGNB. However, SelGNB did not significantly outperform SelvMNB in any dataset, whereas SelvMNB outperformed SelGNB in four datasets. These two algorithms induced the same classifier for the `Covertype` dataset, so there were no statistical differences between the two methods for that dataset (p-value $= 1.0$ in Table 9.6). The dNB classifier with discretized predictive variables yielded very good results. dNB significantly outperformed vMNB in four datasets, whereas vMNB significantly outperformed dNB in three datasets. On the other hand, SelvMNB significantly outperformed dNB in four datasets, whereas dNB significantly outperformed SelvMNB in three datasets. vMNB and SelvMNB performed better against dNB than their linear counterparts, GNB and SelGNB, respectively.

9.3.3 Goodness-of-fit analysis

To understand why vMNB performs better, we illustrate the differences between using linear and angular distributions to model directional data. We took variable 11 in the `Protein10` dataset, which was selected by both SelvMNB and SelGNB as an important predictive variable for classification. `Protein10` has four class values. NB fits one conditional probability density for each class value c. Figure 9.11 plots the Gaussian (dashed lines) and the von Mises (solid lines) conditional distributions fitted from the data. We can see important mismatches for class values $C = 2$ and $C = 3$. Figure 9.11b shows how the Gaussian distribution ignores the periodicity of the data and yields a density of 3.97 10^{-5} for an angle of $-180°$ degrees, and a density of 0.2049 for $180°$ degrees. Therefore, Gaussian distributions yield two different densities for the same angle. On the other hand, the von Mises distribution in Figure 9.11c is more peaked and yields higher densities than the Gaussian distribution for values close to the mean.

The legends in Figure 9.11 include the log-likelihood of the models given the data: $LL = \ell\left(\mathcal{D}_{\Phi|c}|f_{\Phi|c}(\phi|c)\right)$. Von Mises distributions always yield higher log-likelihood than Gaussian distributions. In fact, the highest differences in the log-likelihood between von Mises and Gaussian distributions can be found for class values 2 and 3 (Figures 9.11(b) and (c), respectively).

Gaussian and von Mises distributions are very similar when the concentration of the values is high. However, using Gaussian distributions to model angles can negatively affect NB's behavior. We use an artificial example to illustrate this point. We generate a dataset with one angular predictive variable and a binary class with values $\Omega_C = \{1, 2\}$.



(a) Class $C = 1$ (b) Class $C = 2$

(c) Class $C = 3$ (d) Class $C = 4$

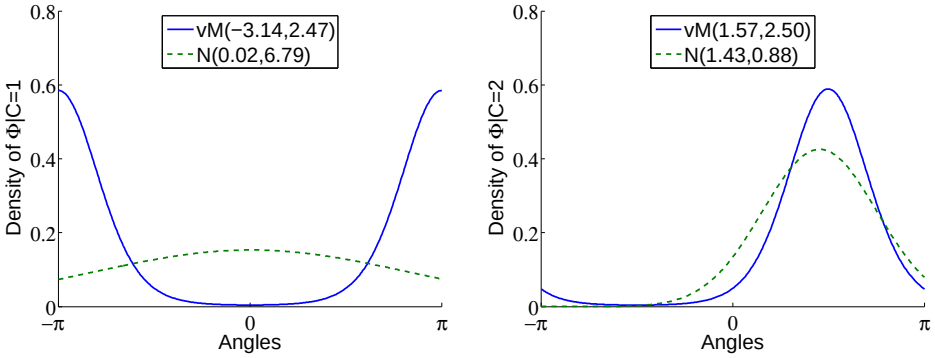
Figure 9.11: Von Mises (solid) and Gaussian (dashed) conditional distributions fitted for variable 11 in `Protein10` dataset.

The classes are equiprobable a priori. Instances from class $C = 1$ follow the distribution $\Phi|C = 1 \sim M(\pi, 2.5)$, whereas instances from class $C = 2$ follow the distribution $\Phi|C = 2 \sim M(\pi/2, 2.5)$. Figure 9.12 shows the conditional density functions of the von Mises and the Gaussian distributions fitted to a sample of 2,000 instances. Figure 9.12a shows that the Gaussian distribution ignores the periodicity of the data, overestimates the variance and incorrectly estimates the mean direction. This yields errors in NB's classification. For example, GNB classifies angle $\phi = \pi$ with class $C = 2$, whereas it should apparently belong to class $C = 1$ because the mean direction of the distribution that generates class $C = 1$ is $\mu_{\Phi|1} = \pi$. On the other hand, GNB labels the angle $\phi = 0$ with the class $C = 1$. The angle $\phi = 0$ is closer to the mean direction of the distribution with class $C = 2$ ($\mu_{\Phi|2} = \pi/2$), so it should be classified with $C = 2$.

9.4 Conclusion

Directional data can be found everywhere in science. Directional information has a number of properties that make it necessary to develop and use different techniques than the ones used with linear information.

In this chapter, we extended one of the simplest and best known models for classifica-



(a) Class $C = 1$ (b) Class $C = 2$

Figure 9.12: Von Mises (solid) and Gaussian (dashed) conditional distributions fitted for the artificial dataset.

tion, the NB classifier, to the case where directional data are used as predictive variables. Understanding the implications of the NB assumption and the theoretical properties of the classifier is the key to interpreting its behavior and establishing its problem-solving potential [407]. Therefore, we analyzed the decision functions of the NB classifiers using directional predictive variables and studied the surfaces induced by those decision functions at length for different values of the parameters. We also studied the more general scenarios where a hybrid NB classifier accounts for discrete, linear (Gaussian) and directional predictive variables.

We showed that the NB classifier with one directional predictive variable, using either the univariate von Mises or the multivariate von Mises-Fisher distribution, is a linear classifier. The decision surface induced by the classifier is a hyperplane (or a set of hyperplanes if more than two class values are considered) that separates the class regions. Therefore, it should be especially well suited for solving problems with linearly separable classes. When two angular predictive variables are considered, the vMNB classifier induces more complex quadratic decision surfaces. In the hybrid setting where von Mises-Fisher and Gaussian distributions are used to model the predictive variables, we showed that the complexity of the decision surfaces depends on the parameters of the Gaussian distribution. Thus, the decision surfaces are hyperplanes when the covariance matrices of the conditional predictive distributions are equal and hyperquadrics when they are not [153]. Artificial examples were used to illustrate the behavior of the different classifiers and to show the decision surfaces they induce.

NB performance is reduced when redundant predictive variables are used [309]. Therefore, we adapted the SelNB algorithm to the use of directional distributions.

We evaluated the vMNB classifier over eight datasets and compared it against the corresponding NB classifiers that use Gaussian distributions or discretization for modeling angular variables. SelvMNB was the best ranking algorithm. Statistical tests were performed to find significant differences in the performance of the classifiers. vMNB and SelvMNB performed similarly or better than the classifiers using linear distributions in all but one dataset.

The NB classifier's conditional independence assumption is quite restrictive and clearly limits the kind of problems that these models can solve. Several BNCs that relax the con-

ditional independence assumption have been proposed in the literature, e.g., the TAN [199], the seminaive Bayes [394], the k-dependence Bayesian classifier [438], etc. Extending these models to the use of directional variables is by no means trivial, since it has been shown that both marginal and conditional distributions cannot be von Mises distributions [352, 354]. Therefore, this is an open and interesting research field.

Directional data can be found in other machine learning scenarios, e.g., clustering and regression problems. Clustering with directional data has been extensively studied in recent papers, see e.g., [16, 30, 375]. Also, a lot of work is available on regression models where the target variable to predict is angular and the predictive variables are either angular [291] or linear [150, 182]. Regression models with spherical target and predictive variables have also been studied in [149, 428]. Recently, circular ordinal regression, where the target variable is discrete but defined in a circular ordered domain, has been approached in [135] using support vector machines. Directional information has also been used in neural networks, where Zemel et al. [535] proposed an extension of the Boltzmann machine with angular units.

On the other hand, BNs have also been applied to classical regression problems [194, 377]. Hybrid models that include different types of probability distributions have attracted much interest, and different approaches have been proposed [52, 274, 432, 458]. Directional distributions add yet another possibility to the range of distributions that can be considered in hybrid BNs. Hybrid probability distributions for modeling the joint density of angular and linear variables [280] could be used in hybrid BNs for regression. When several angular variables are included in the BN, the fact that we cannot model both marginal and conditional distributions as von Mises distributions is again a crucial problem in these models.

Directional statistics opens a number of interesting challenges and opportunities within machine learning research, particularly for PGMs. We hope that further research in this area and the implementation of more complex models will provide an excellent tool for solving difficult problems in a wide range of fields.

Part IV

CONTRIBUTIONS TO CONSENSUS ANALYSIS

Chapter 10

Consensus analysis for GABAergic interneuron classification

10.1 Introduction

As introduced in Chapter 6, the problem of classifying and naming neurons has been a topic of debate for over a hundred years. Nevertheless, a satisfactory consensus remains to be reached, even for restricted neuronal populations such as the GABAergic interneurons of the cerebral cortex. Over the last two decades, the amount of morphological, molecular, physiological and developmental data has grown rapidly, making classification harder rather than easier. A consistent neuronal classification and terminology would help researchers manage this multidisciplinary knowledge, and is needed for specialists in neuroscience subfields to establish and maintain effective communication and data sharing [413]. As in other domains of science, taxonomies can be empirical or scientific [269]: The aim of an empirical classification is the direct application of knowledge to utilitarian purposes. The scientific classification is the organization of existing knowledge, and its principles are methodical guides to further investigation.

In spite of the many studies performed since the original findings of Santiago Ramón y Cajal, we still lack a catalog of neuron types and names that is accepted by the general scientific community. Recognizing this problem, the International Neuroinformatics Coordinating Facility has recently established a Neuron Registry within the Program on Ontologies of Neural Structures[1], with the aim to identify known neuron types on the basis of their defining properties [238]. A collation of terms referring to neuron types is available as part of the Neuroscience Information Framework from NeuroLex[2] [320].

A milestone towards a future classification of the GABAergic interneurons in the cerebral cortex (neocortex, hippocampus, and related areas) was the standardization of the nomenclature of their properties [413]. However, at that time it was not possible to identify a

[1] Available at http://incf.org/core/programs/pons
[2] Available at http://neurolex.org/wiki/Category:Neuron

set of anatomical traits that unambiguously define an interneuron class. In this chapter, a limited number of neuron types and morphological properties were selected based on studies performed over the years in many laboratories. Then, a group of 48 experts in the field were asked to individually classify 320 cortical interneurons based on their most prominent morphological features. Finally, we extensively analyzed the level of agreement between experts when classifying the selected set of interneurons.

The research included in this chapter has been published in DeFelipe et al. [123].

Chapter outline

Section 10.2 introduces the interneuron classification problem and explains the data acquisition process. The preliminary analyses of the raw data are reported in Section 10.3. The agreement between experts when classifying the interneurons is quantified and analyzed in Section 10.4. Section 10.5 reports the results of the clustering of the interneurons attending to the classifications of the experts. The behaviors of the experts when classifying the interneurons are modeled and compared using BNs in Section 10.6. Supervised learning models are induced from data in Section 10.7. Finally, the chapter ends with discussion and future work in Section 10.8.

10.2 Interneuron classification by a set of experts

We selected $N = 320$ cortical GABAergic interneurons from different species: cat, human, monkey, mouse, rabbit and rat. Three-dimensional reconstructions of 241 of those interneurons were retrieved from http://www.neuromorpho.org [23], whereas the rest were scanned from relatively old papers with no data on the three-dimensional distribution of their dendrites and axons. A set of 48 experts were asked to classify each one of the neurons according to their most prominent morphological features. A web application[3] was specifically built to display the neuronal morphologies for the participants and to retrieve their classifications. Two-dimensional projections of all the neurons were available. Additionally, a three-dimensional visualization applet based on Cvapp software [74] was provided for the neurons taken from NeuroMorpho.org, which experts could use to navigate, rotate and zoom the neuronal morphologies. Figure 10.1 shows a screenshot of the web application. Additional data about the location of the neuron, such as the cortical area, the layer and the thickness of the layer were included when available. Other web application features included a help page with instructions and definitions of the neuronal types, and a search engine which showed other neurons previously classified by the expert in the same category. Each expert was administered the form in Figure 10.1 for each neuron.

The experts who participated in the experiment were asked to classify the neurons according to six axonal features describing the main morphological characteristics of the neurons:

[3] Available at http://cajalbbp.cesvima.upm.es/gardenerclassification/

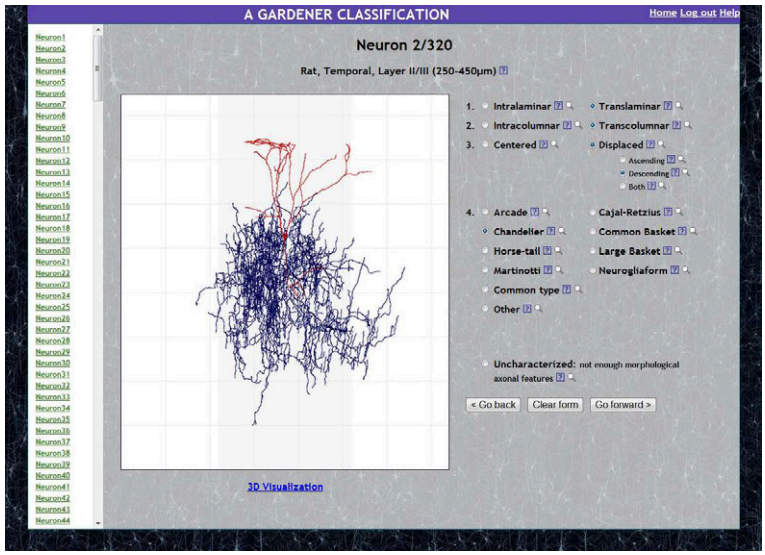


Figure 10.1: Web application showing one of the 320 neurons to be classified by each expert.

1. Feature 1 (F1) refers to the distribution of the interneuron axonal arborization relative to cortical layers (Figure 10.2(a-d)). Within this feature, two categories are proposed: `Intralaminar`, which refers to interneurons with axonal arbors distributed predominantly in the layer of the parent soma (Figures 10.2(a, b)); and `Translaminar`, which refers to interneurons with axonal arbors distributed mainly above and/or below the cortical layer of the parent soma (Figures 10.2(c, d)).

2. Feature 2 (F2) refers to the distribution of the axonal arborization relative to the size of cortical columns, from a broad anatomical point of view (Figure 10.2(e-h)). We have arbitrarily set the size of a cortical column at a diameter of 300µm, a value that remains rather similar across several species and cortical areas for many of these structures [350, 380] (see Section 6.3). Within this feature, two categories are considered: `Intracolumnar`, which refers to interneurons with axonal arbors primarily distributed at a distance from the parent soma that does not exceed 300µm in the horizontal dimension (Figures 10.2(e, g)); and `Transcolumnar`, which refers to interneurons with horizontal axonal collaterals exceeding a distance of 300µm from the parent soma in the horizontal dimension (Figures 10.2(f, h)).

3. Feature 3 (F3) refers to the relative location of the axonal and dendritic arbors (Figure 10.2(i-p)). Within this feature, the following categories are distinguished: `Centered`, which refers to interneurons whose dendritic arbor is located mostly in the center of the axonal arborization (Figure 10.2(i-l)); and `Displaced`, which refers to interneurons whose dendritic arbor is shifted with respect to the axonal arborization (Figure 10.2(m-p)).

4. Feature 4 (F4): If a neuron is categorized as being both `Translaminar` (for the first axonal feature) and `Displaced` (for the third axonal feature), it can be further dis-

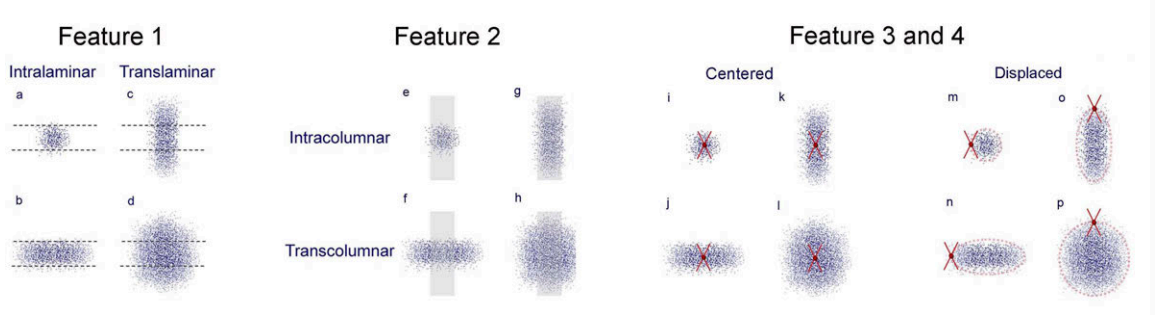


Figure 10.2: Schematics of the morphological Features 1 to 4. Dashed horizontal lines indicate the extent of the layer. Vertical grey shadows indicate the extent of the column. Axonal arborization is represented by blue dots. Soma and dendritic arborization are represented in red. Possible variations on the relative position of the somata with respect to the axonal arborization of displaced neurons are represented by red dotted ovals.

tinguished into the following categories [413]: `Ascending`, which refers to interneurons whose axonal arborization is distributed mostly towards the cortical surface; `Descending`, which refers to interneurons whose axonal arborization is distributed mostly towards the white matter (Figure 10.2(o, p)); or `Both`, which refers to interneurons whose axonal arborization is distributed towards both the cortical surface and the white matter.

5. Feature 5 (F5) refers to a limited number of cell types (Figure 10.3) defined for classification on the basis of recognizable morphological characteristics and the common usage of their name in the literature [412].

   (a) `Common type` denotes neurons with somata in layers I-VI, multipolar, bipolar or bitufted dendritic arbors, and axon collaterals without any apparent target or orientation preference.

   (b) `Horse-tail` cells denote neurons with somata mostly in layers II-III, multipolar, bitufted or bipolar dendrites, and axons forming tightly intertwined bundles of long descending vertical collaterals.

   (c) `Chandelier` cells denote neurons with somata in layers II-VI, multipolar or bitufted dendritic arbors, and terminal axon branches that form short vertical rows of boutons resembling candlesticks. These interneurons are also referred to as axo-axonic cells as they synapse on the axonal initial segment of their pyramidal targets.

   (d) `Martinotti` cells denote neurons with somata in layers II-VI, multipolar, bitufted or bipolar dendrites, and ascending axons that give rise to two axonal arbors, one near the soma and another at a variable distance above. This second plexus may be dense (axonal tuft) or diffuse, and it can be either in the same layer as the soma of origin or in the layers above (ascending axons can travel from layer VI to layer I).

(e) `Common basket` cells denote neurons with somata in layers II-VI, multipolar or bitufted dendritic arbors and axon collaterals that have numerous curved pre-terminal axon branches.

(f) `Arcade` or willow cells denote neurons with somata in layers II-VI, multipolar or bitufted dendrites, and axons that give rise to axonal arcades, with predominantly vertical arbors and relatively long descending collaterals.

(g) `Large basket` cells denote neurons with somata in layers II-VI, multipolar or bitufted dendrites, and horizontally oriented axon collaterals that can reach a length of several hundred micrometers. These collaterals show numerous curved pre-terminal axon branches that innervate the somata and proximal dendrites of neurons. Frequently, these cells display one or several long descending axonal branches.

(h) `Cajal-Retzius` cells denote neurons with an axon plexus that is restricted to layer I and long dendrites with ascending branchlets to the pia. These neurons are not present in adult neocortex and in rodents persist only during the two first postnatal weeks [89, 357]. Proper Cajal-Retzius cells do not contain GABA or express GABA-synthesizing enzymes GAD65/GAD67 [256, 369]. There are also GABAergic neurons with somata in layer I and prominent long horizontal axon collaterals and/or dendrites [369], and these are often also named Cajal-Retzius neurons in the developing neocortex, in spite of their different molecular characteristics from proper Cajal-Retzius neurons [256]. Given the purely morphological nature of the present study, most of the authors practically considered any GABAergic neuron in layer I with horizontally oriented axonal arborization as putative Cajal-Retzius cells.

(i) `Neurogliaform` cells denote neurons with somata in layers I-VI, with multipolar dendritic arbors, and are characterized by very small and dense local axonal arborization around the parent cell body.

(j) Also, we added the option `Other` to label any given neuron with an alternative name in case the expert considered another term more appropriate.

6. Feature 6 (F6): Interneurons that are uniquely characterized by peculiar morphological features can often be easily recognized, even when their axon is rather incompletely labeled. However, in many other cases the axon needs to be fully labeled and reconstructed in order to distinguish the neuronal identity unequivocally. Thus, although it is not always necessary to visualize the full axonal and dendritic arborization to distinguish a given neuron, this is the preferred situation. Pragmatically, "sufficiently complete" labeling simply means "clear enough" to allow for the identification of a given morphological type. We use the term `Characterized` to refer to that situation. When an insufficient number of morphological axonal features are visualized for a given interneuron (because of incomplete staining, tissue slicing, etc.), we propose that the cell should be deemed an anatomically `Uncharacterized` interneuron.

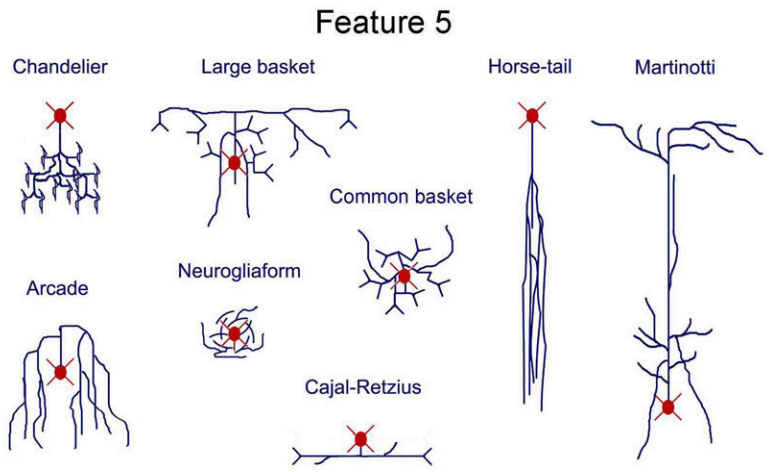


Figure 10.3: Schematics of the interneuron types. Axonal arborization is shown in blue. Soma and dendritic arborization are shown in red. Interneuron types `Common type` and `Other` are not shown.

10.3 Analysis of raw data

Forty-two out of the 48 experts classified the 320 neurons included in the experiment, and only data from these 42 experts are considered in the remainder of the analysis.

First, we performed a preliminary exploratory analysis of the raw data to study how the votes of the experts were distributed for the different features. We assessed the relative frequency of each category in the experiment, i.e., of each possible value for each feature (Figure 10.4). Less than 10% of neurons were rated as `Uncharacterized`. Thus, the vast majority of the neurons in the experiment were considered as `Characterized`. The most frequently assigned categories of Features 1 to 3 proposed in this study were `Translaminar`, `Intracolumnar`, and `Displaced`. The categories `Ascending` and `Descending` received a similar percentage of the ratings, whereas fewer neurons were assigned to the category `Both`.

We then assessed the frequency with which interneurons were assigned to specific interneuron types (F5 in Figure 10.4). The most commonly assigned interneuron types were `Common type`, `Common basket` and `Large basket`. The interneuron types `Martinotti`, `Horse-tail` and `Neurogliaform` received an intermediate percentage of ratings, whereas `Chandelier` and `Arcade` received the lowest percentage of ratings. Only three cells were classified as `Cajal-Retzius` by six experts, whereas the remaining experts classified these neurons as `Uncharacterized`, `Common type`, `Common basket`, `Large basket`, `Martinotti` or `Other`.

We compared the categories assigned by the individual experts for each one of the six features (Figure 10.5). Some experts were "outliers" in terms of their selections, e.g., one expert categorized all the neurons as `Intralaminar` in Feature 1 (last expert in Figure 10.5(a)), and the same rater categorized almost all the neurons as `Centered` in Feature 3 (last expert in Figure 10.5(c)). Regarding Feature 5, high bars in Figure 10.5(e) indicate that a high number of experts selected a particular category for a particular neuron. On the other hand, short

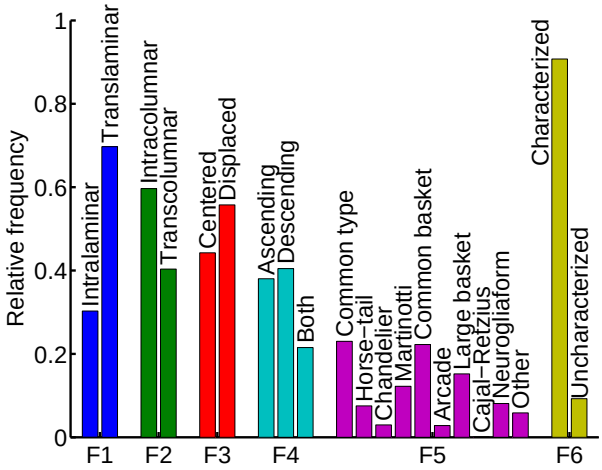


Figure 10.4: Relative frequency of each category for each feature (F1 to F6), i.e., the number of times a category was selected divided by the total amount of ratings for the feature.

bars for a particular category and a particular neuron indicate that the corresponding neurons received very few votes in that neuron type. For example, it is possible to distinguish seven high bars for the `Chandelier` category indicating that experts agreed when assigning this particular category for those specific neurons. With regard to Feature 6 (Figure 10.5(f)), the majority of the experts considered that most of the neurons were `Characterized` and tried to classify them. Indeed, 35 out of 42 experts (83.33%) characterized more than 280 neurons, whereas two experts characterized less than 200 neurons (first two experts in Figure 10.5(f)).

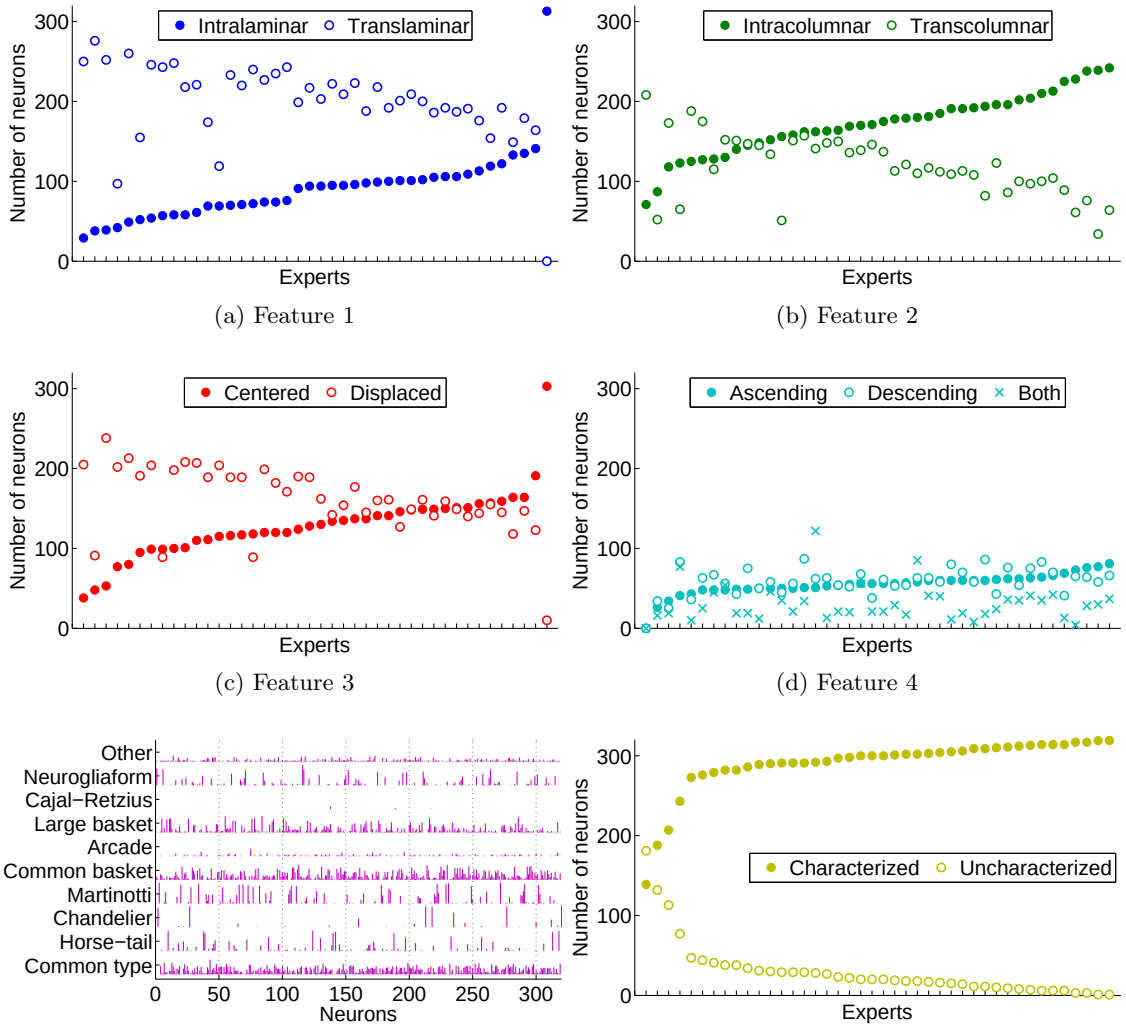
Finally, we checked whether the names given to the 79 neurons that were scanned from original publications were maintained in the present experiment by the authors of those publications. Interestingly, the authors were frequently inconsistent for certain neurons. For example, some neurons named `Neurogliaform` cells in the publication were classified as `Uncharacterized` in the current experiment by the same author.

10.4 Experts' agreement values

We computed statistical measures of inter-expert agreement to analyze the degree of concordance between the ratings given by the experts (see Chapter 4). The goal was to quantify the agreement among experts for each feature independently. We studied the agreement for both features and categories using the two most studied agreement indices: Fleiss' pi and Cohen's kappa indices.

Figure 10.6 shows the values of the observed agreement (Section 4.2.1) and the chance-corrected Fleiss'pi values (crosses) for each one of the six features (Section 4.2.2.2). We found a high level of observed agreement between experts in the classification of neurons according to Features 1-4 and 6 (observed agreement values exceeding 0.7 in Figure 10.6). The lowest level of inter-expert agreement (below 0.5) was found for Feature 5.

After correcting for chance agreement, the highest chance-corrected Fleiss' pi inter-expert



(a) Feature 1 (b) Feature 2

(c) Feature 3 (d) Feature 4

(e) Feature 5 (f) Feature 6

Figure 10.5: Graphical representation of the ratings given to the different categories of the six features by the 42 experts who completed the experiment. Experts are sorted in ascending order (in the horizontal axis) based on the number of votes of the categories `Intralaminar` (a), `Intracolumnar` (b), `Centered` (c), `Ascending` (d) and `Characterized` (f). A vertical bar is shown in (e) for each neuron and each category, representing the number of experts who selected that category for that neuron. High bars (e.g., for categories `Chandelier`, `Horse-tail` and `Martinotti`) show high agreement when classifying the neurons in these neuronal types, whereas short bars (e.g., for categories `Common type`, `Common basket`, `Large basket`, `Other`, `Arcade`, etc.) represent low agreement.

agreement was found for Feature 4 (Figure 10.6). Features 1, 2, and 3 yielded intermediate chance-corrected agreement values, whereas Features 5 and 6 had low agreement. The difference between observed agreement and Fleiss' pi index was particularly high for Feature 6, i.e., for the decision on whether or not a neuron could be characterized. This feature had the highest observed agreement and the lowest Fleiss' pi value. This was due to the fact that the

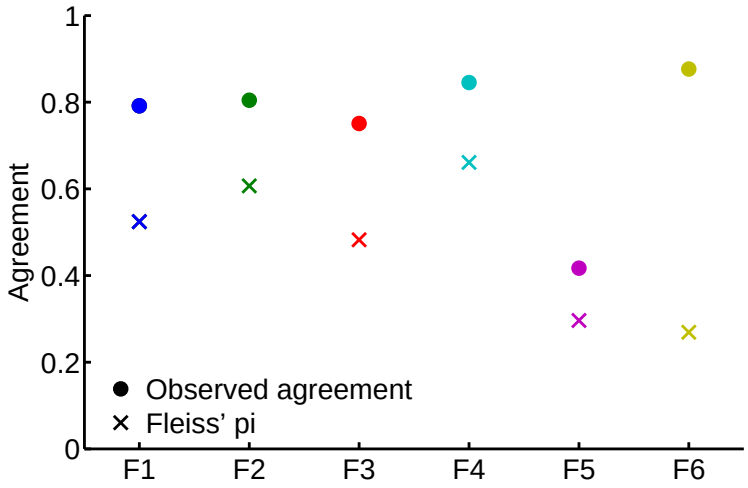
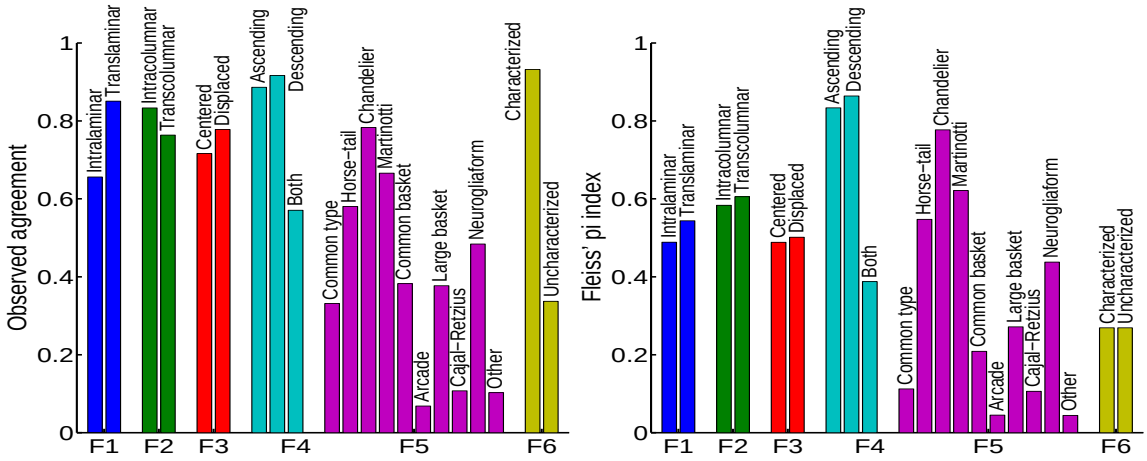


Figure 10.6: Expert's feature agreement values.

category prevalence of this feature was very unbalanced, such that `Characterized` neurons were much more frequent than `Uncharacterized` ones, reducing the values of the agreement measures (see Section 4.2.2). A permutation test reported statistically significant differences from chance agreement (uncorrected p-value < 0.0001) for all the features (see Section 4.3).

Figure 10.7 shows the observed agreement and Fleiss' pi values for each category of every feature (Section 4.2.3). The observed agreement for all categories in Features 1 to 3 was high (Figure 10.7(a)), whereas Fleiss' pi values were lower (Figure 10.7(b)). We also observed a high agreement for categories `Ascending` and `Descending` in Feature 4, whereas agreement was lower for category `Both`. Regarding Feature 5, we found that the `Chandelier` category yielded the highest consensus, with similar values for Fleiss' pi (Figure 10.7(b)) and the observed agreement (Figure 10.7(a)). The level of agreement was also high or medium for `Martinotti`, `Horse-tail` and `Neurogliaform` cells, whereas it was lower for the rest of the proposed interneuron types (`Large basket`, `Common basket`, `Common type`, `Cajal-Retzius`, `Arcade` and `Other`). As in the above agreement analysis for Feature 6, `Characterized` and `Uncharacterized` yielded a low level of chance-corrected Fleiss' pi agreement (Figure 10.7(b)). Some experts tried to characterize all the neurons whereas other experts frequently categorized them as `Uncharacterized` (see Figure 10.5(f)). The differences in experts' biases and the unbalanced prevalence of the two categories explain the very low Fleiss' pi values for both `Characterized` and `Uncharacterized` categories. The values of the observed agreement and the category-specific Fleiss' pi indices for all the categories of all the features significantly differed from chance agreement according to a permutation test (uncorrected p-value < 0.0001).

In a separate analysis, we tried to identify possible outliers in the group of experts by studying the influence of every one of the experts in the chance-corrected Fleiss' pi index computed for each feature (Figure 10.8(a)). Additionally, we also removed groups of three experts in all possible combinations to further identify possible sets of experts contributing

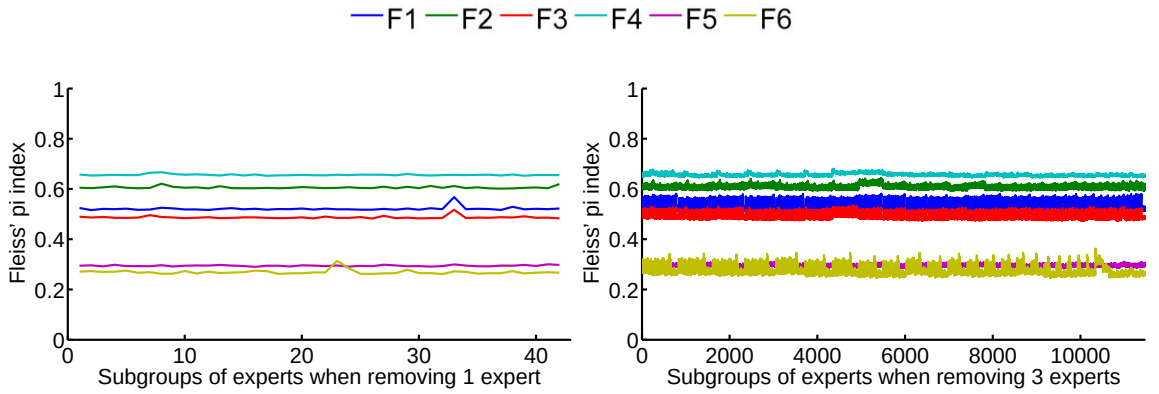


(a) Observed agreement (b) Fleiss' pi

Figure 10.7: Experts' agreement values for each category of each feature.

to low Fleiss' pi index values (Figure 10.8(b)). Agreement increased for Features 1 and 3 when expert 33 was removed (as revealed by the small peak in the blue and red curves). This is consistent with the different selection of categories by this expert for these features (see rightmost expert in Figures 10.5(a) and (c)). Similarly, removing expert 23 increased the agreement for Feature 6, as shown by the peak in the ochre curve. The peaks in Figure 10.8(b) corresponded to the subgroups of experts excluding expert 33 in Features 1 and 3, and expert 23 in Feature 6. For instance, this means that expert 33 selected categories for Features 1 and 3 in a different way than the rest of the experts. The agreement for Features 2, 4 and 5 did not vary when one or three experts were removed. The largest difference in Fleiss' pi index corresponded to the scenario where experts 23, 24 and 29 were removed in Feature 6. In this case, the agreement increased from 0.269 (when the 42 experts were considered) to 0.3628 (when 39 experts were considered). However, we did not remove any expert from the remainder of the analysis since there was not an expert (or a group of three experts) whose removal produced statistically significant Fleiss' pi index differences for all features.

Next, we investigated whether the Fleiss' pi values increased or decreased when merging two categories of Feature 5. The rationale for this was to study possible overlapping between interneuron types. Table 10.1 shows the values obtained in the analyses where a particular category (rows) was merged with another category (columns). The reference value obtained when all the interneuron types were considered as different categories was 0.2963 (purple cross in Figure 10.6). Thus, Fleiss' pi values above this number will indicate categories that were confused with each other. Merging category `Martinotti` with any other category decreased Fleiss' pi value, with one exception, namely when categories `Martinotti` and `Other` were merged. The lowest Fleiss' pi value (0.2645) in Table 10.1 was achieved when category `Martinotti` was merged with category `Common basket`. The Fleiss' pi value was also lower in all analyses in which category `Chandelier` was merged with any other category. These results suggest that these interneuron types are well defined and easily distinguishable. Extending



(a) (b)

Figure 10.8: Fleiss' pi values for all the groups of experts obtained (a) when removing one expert (42 possible subgroups) and (b) when three experts were removed (11480 possible subgroups).

Table 10.1: Fleiss' pi index values when a category of Feature 5 is merged with another category. The numbers in columns refer to the number of the interneuron types in rows.

	2	3	4	5	6	7	8	9	10. `Other`
1. `Common Type`	0.2973	0.2891	0.2876	0.3444	0.3040	0.3187	0.2970	0.2836	0.3158
2. `Horse-tail`		0.2937	0.2854	0.2790	0.2969	0.2844	0.2962	0.2862	0.3102
3. `Chandelier`			0.2910	0.2922	0.2959	0.2909	0.2963	0.2944	0.2945
4. `Martinotti`				0.2645	0.2941	0.2916	0.2961	0.2803	0.2984
5. `Common basket`					0.3006	0.3170	0.2959	0.3259	0.2977
6. `Arcade`						0.3003	0.2963	0.2953	0.2982
7. `Large basket`							0.2973	0.2839	0.2961
8. `Cajal-Retzius`								0.2962	0.2965
9. `Neurogliaform`									0.2952

this analysis beyond pairwise merges, the Fleiss' pi value was lowest (0.2312) when categories `Horse-tail`, `Martinotti` and `Common basket` were all merged together into a single category (not shown).

The highest Fleiss' pi value (0.3444) was achieved when categories `Common type` and `Common basket` were merged, revealing ill-defined neuron types. Fleiss' pi value also increased when merging categories `Common basket` and `Neurogliaform` (0.3259), `Common type` and `Large basket` (0.3187), and `Common basket` and `Large basket` (0.3170). When we considered combinations of three neuronal types, the highest Fleiss' pi value (0.4110) was achieved when categories `Common type`, `Common basket`, and `Large basket` were merged into a single category (not shown).

Furthermore, the above results were confirmed in a separate analysis using Cohen's kappa index. This index is defined for scenarios with two experts and two categories (see Section 4.2.2.1). Additionally, two of its variants were computed (see Section 4.2.2.1): the ratio between Cohen's kappa and its maximum value taking into account fixed marginals ($\kappa/\kappa_{\max}$), and the $PABA\kappa$ index. Thus, we assessed the level of agreement between all possible pairs of experts resulting in a comparison of each expert with all the other 41 experts (Figures 10.9,

10.10 and 10.11). For example, the first blue box in Figure 10.9(c) summarizes the Cohen's kappa values between the first expert and the other 41 experts regarding the categorization of a neuron as `Chandelier`. Thus, this high-valued box means that this expert categorized as `Chandelier` the same neurons as the majority of the remaining experts. Also, we can conclude that there was a high agreement between experts for category `Chandelier`, since all boxplots (excluding expert #20) showed high Cohen's kappa index values. Other categories showing high agreement were `Martinotti` and `Horse-tail`. In contrast, a low level of agreement was found for `Common type` (Figures 10.9(a)), `Common basket` (Figures 10.9(e)), and `Large basket` cells (Figures 10.9(g)), as reflected by the low values of the boxplots. The agreement values of $PABA\kappa$ and the ratio $\kappa/\kappa_{\max}$ were similarly low for these categories (Figures 10.10 and 10.11). However, `Arcade` and `Other` categories yielded low agreements for Cohen's kappa (Figures 10.9(f) and (j)) and the ratio between Cohen's kappa and its maximum value (Figures 10.11(f) and (j)), whereas the agreement values of $PABA\kappa$ were high (Figures 10.10(f) and (j)). The low agreement found in these two categories is probably due to the low number of votes assigned by the experts to these categories. In fact, `Arcade` was the second category (after `Cajal-Retzius`) with fewest votes. Since $PABA\kappa$ corrects for the differences in the number of votes, it yields much higher values (Figure 10.10) than Cohen's kappa or the ratio between Cohen's kappa and its maximum value. Similar conclusions can be drawn for category `Other`.

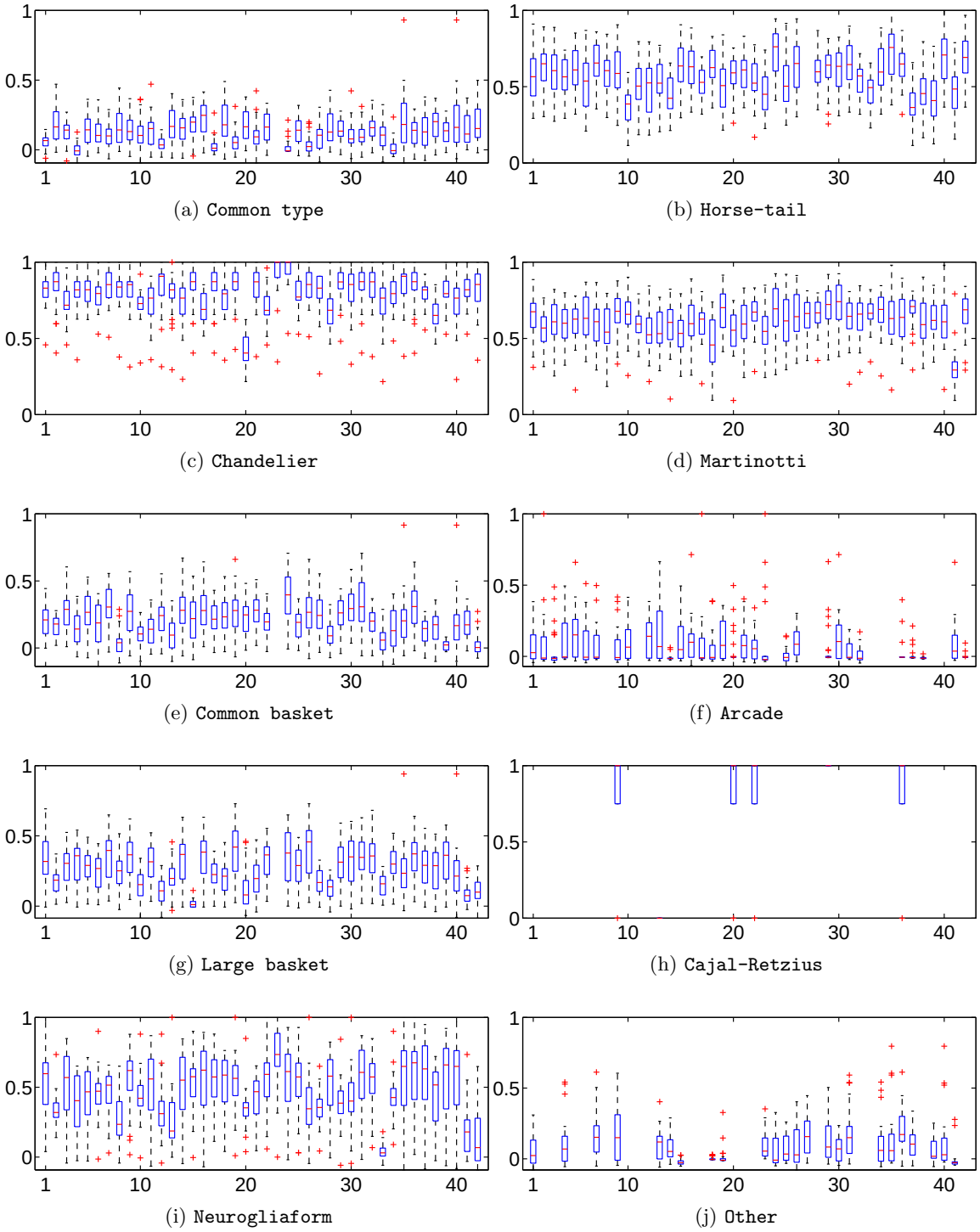
10.5 Neuron clustering

We ran clustering algorithms to find groups of neurons at two levels: neuron clustering for each feature independently and neuron clustering for all features simultaneously. These algorithms find clusters of neurons with similar properties. Then, we studied whether or not all the neurons in a cluster were assigned the same categories by the experts. The goal was to check whether or not the experts' votes for a given feature could separate neurons into clear groups.

10.5.1 Neuron clustering for each feature

First, we wanted to generate clusters for the set of N neurons considering each feature independently. For a given feature, we used the category assigned for each expert to each neuron (category value $q = 1, \ldots, Q$) as information for the clustering. Therefore, the dataset used for the clustering algorithm had $N = 320$ instances (neurons), where each instance is an n-dimensional vector ($n = 42$ experts).

We applied the K-modes algorithm [268], an extension of the K-means algorithm [347] that manages categorical data. Algorithm 10.1 sketches the main steps of the K-means algorithm, which are the same as in the K-modes algorithm. The goal of the K-means algorithm is to find the K cluster centers $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ that minimize a measure of dissimilarity, where $K > 1$ is a parameter of the algorithm indicating the number of clusters. For Features 1-3 and 6 a number of clusters $K = 2$ was used. For Feature 4, three clusters



(a) Common type (b) Horse-tail

(c) Chandelier (d) Martinotti

(e) Common basket (f) Arcade

(g) Large basket (h) Cajal-Retzius

(i) Neurogliaform (j) Other

Figure 10.9: Boxplots showing Cohen's kappa values for each pair of experts when comparing one category against all other categories in Feature 5. For example, the first box in panel (a) shows the agreement between the first expert (X-axis) and the rest of the experts for category Common type.

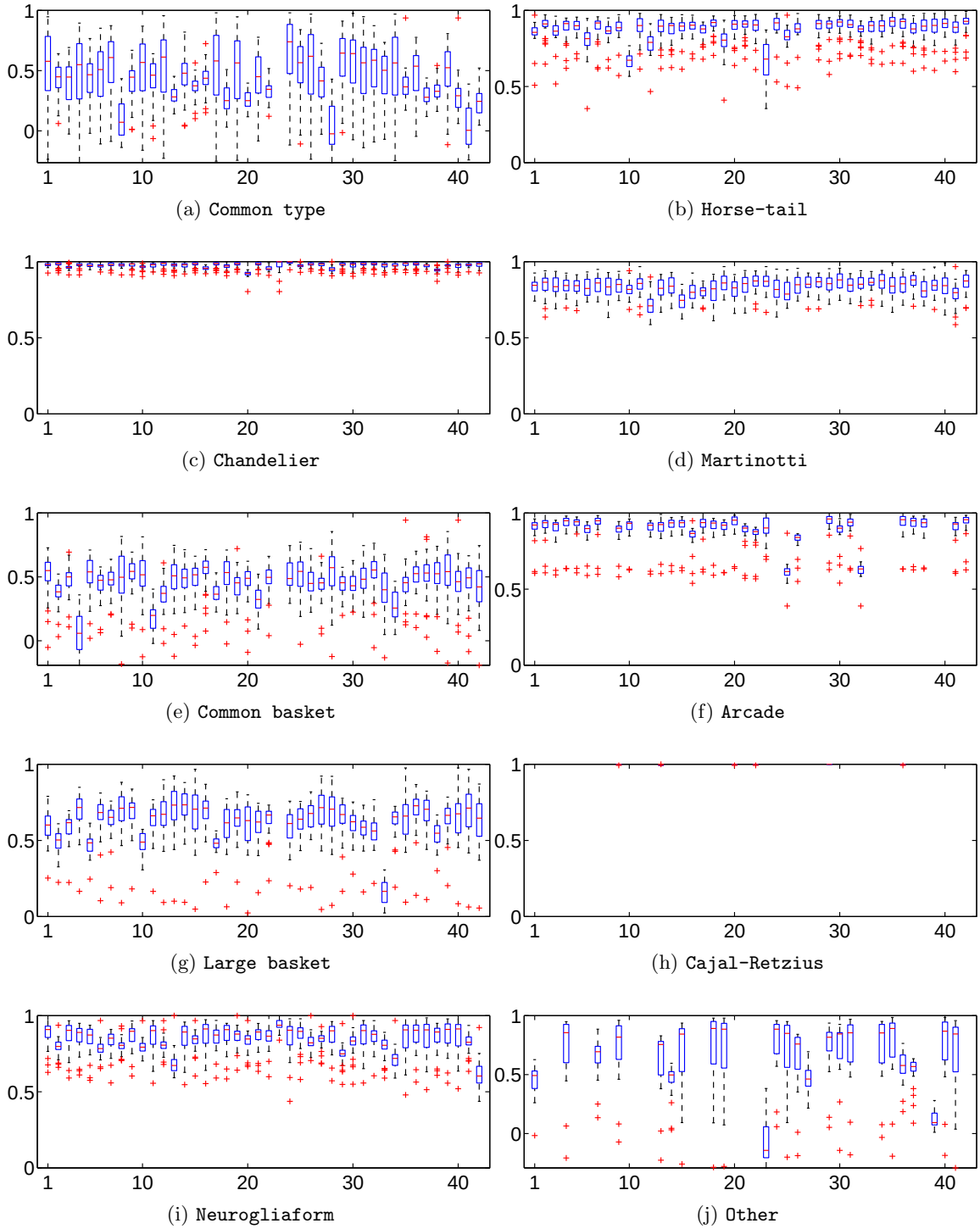
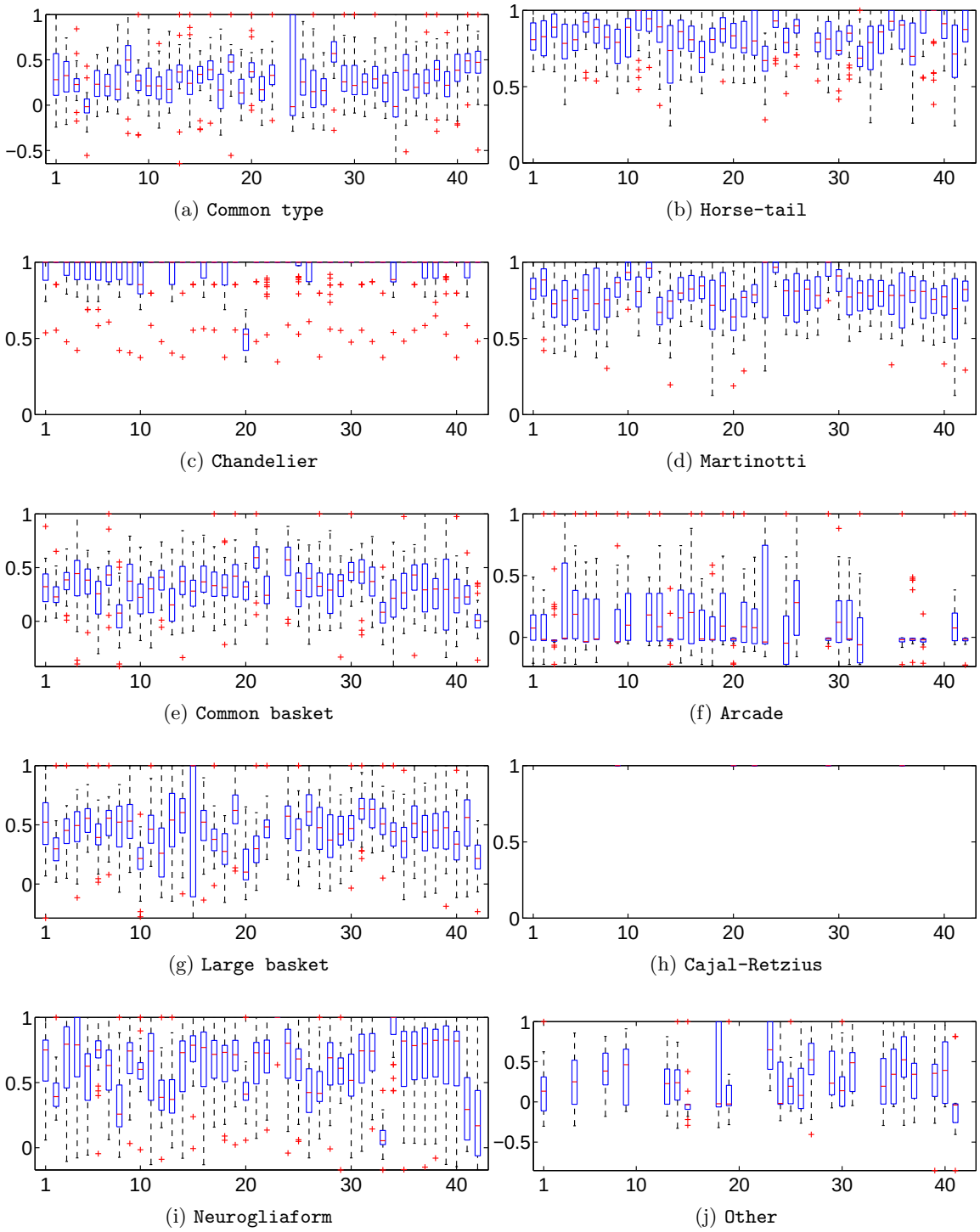


Figure 10.10: Boxplots showing Cohen's Prevalence-Adjusted Bias-Adjusted kappa ($PABA\kappa$) values for each pair of experts when comparing one category against all other categories in Feature 5. For example, the first box in panel (a) shows the agreement between the first expert (X-axis) and the rest of the experts for category `Common type`.



(a) Common type (b) Horse-tail

(c) Chandelier (d) Martinotti

(e) Common basket (f) Arcade

(g) Large basket (h) Cajal-Retzius

(i) Neurogliaform (j) Other

Figure 10.11: Boxplots showing the ratio between Cohen' kappa and its maximum value given fixed marginal frequencies for each pair of experts when comparing one category against all other categories in Feature 5. For example, the first box in panel (a) shows the agreement between the first expert (X-axis) and the rest of the experts for category `Common type`.

($K = 3$) were selected. Different numbers of clusters (six to ten) were analyzed for Feature 5. The clearest results from a biological interpretation point of view were obtained with $K = 8$. A neuron is assigned to the cluster with the closest center. Therefore, the fitness function to minimize is the sum of the distance of each item to the center of its cluster.

For categorical data, K-modes uses the Hamming distance to measure the distance between two items (neurons) or between an item and a cluster center $d(\mathbf{x}_i, \mathbf{c}_k)$. The set of cluster centers $\mathbf{C}$ is found by computing the modes of the items belonging to the cluster (`Combine` operation in step 2.(c)). Ties when computing the modes or when assigning items to clusters are broken randomly. In our implementation, the algorithm stopped when no change in the cluster centers occurred or when the fitness function had the same value for 100 consecutive iterations.

Algorithm 10.1 (K-means clustering algorithm)
Inputs:

> $\mathcal{D}_{\mathbf{X}}$: *A dataset with* N *observations* $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_j = (x_{j1}, \ldots, x_{jn})$, $j = 1, \ldots, N$.
>
> K: *The number of clusters*

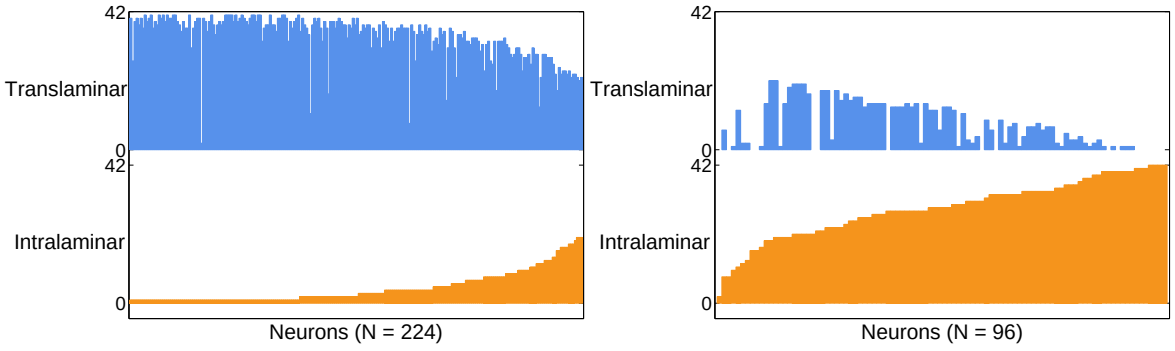
Output: The set of cluster centers $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
Steps:

1. *Initialize the* K *cluster centers* $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ *to* K *random observations in* $\mathcal{D}_{\mathbf{X}}$ *without replacement.*
2. *While the cluster centers* $\mathbf{C}$ *change*
   (a) *For each observation* $\mathbf{x}_j$, $j = 1, \ldots, N$, *compute the dissimilarity between* $\mathbf{x}_j$ *and each cluster center* $\mathbf{c}_k$: $d(\mathbf{x}_j, \mathbf{c}_k)$.
   (b) *Assign each observation* $\mathbf{x}_j$ *to cluster* $k_j^* \in \{1, \ldots, K\}$ *with the closest center:* $k_j^* \leftarrow \arg\min_{k=1,\ldots,K} d(\mathbf{x}_j, \mathbf{c}_k)$.
   (c) *Recompute the cluster centers* $\mathbf{C}$ *from the observations in each cluster:* $\mathbf{c}_k \leftarrow$ `Combine`$(\{\mathbf{x}_j \in \mathcal{D}_{\mathbf{X}} | k_j^* = k\})$.
3. *Return the cluster centers* $\mathbf{C}$.

The K-means algorithm iteratively minimizes the sum of the distances of each observation to its cluster center: $J(\mathcal{D}_{\mathbf{X}}, \mathbf{C}) = \sum_{j=1}^{N} d(\mathbf{x}_j, \mathbf{c}_{k_j^*})$. K-means (and also K-modes) is guaranteed to find a local minimum of $J(\mathcal{D}_{\mathbf{X}}, \mathbf{C})$. Therefore, they can yield suboptimal solutions if it gets stuck in local minima. To avoid this situation, the algorithm was run several times (25 in our case) with different initial values for the cluster centers in step 1 of Algorithm 10.1. The best result (minimum fitness function value) is reported.

For Feature 1, the K-modes algorithm (with $K = 2$) separated neurons into one cluster of neurons mainly categorized by the experts as `Translaminar` (Figure 10.12(a)), and

another cluster of neurons mainly categorized as `Intralaminar` (Figure 10.12(b)). The vertical bars in the graphs show the number of experts who selected each category for each neuron in the cluster. However, note that the K-modes algorithm does not use this summarized information. Instead, it clusters the neurons using the category selected by each expert individually. Similarly, single clusters were easily identified for every category of Feature 2 (Figure 10.13) and 3 (Figure 10.14). Regarding Feature 4, the K-modes algorithm ($K = 3$) found two clusters of neurons mainly categorized by the experts as `Ascending` (Figure 10.15(a)) and `Descending` (Figure 10.15(b)), respectively. However, the third cluster (Figure 10.15(c)) contains neurons categorized by the experts as `Ascending`, `Descending` or `Both`, showing confusion about the `Both` category. With respect to Feature 5, the K-modes algorithm ($K = 8$) identified individual clusters containing neurons mainly categorized by the experts as `Martinotti` (Figure 10.16(a)), `Horse-tail` (Figure 10.16(b)), `Chandelier` (Figure 10.16(f)) or `Neurogliaform` (Figure 10.16(g)). However, category `Neurogliaform` was sometimes confused with categories `Common type` and `Common basket` (Figures 10.16(c), (e) and (g)). Other clusters included neurons that the experts categorized as `Common type`, `Common basket`, and `Large basket` (Figure 10.16(d), (e) and (h)). Thus, the K-modes clustering algorithm identified clusters where these three interneuron types were intermingled. The algorithm also showed that the `Arcade` category appeared distributed in all clusters, although this category was more frequent in clusters in which `Common type`, `Common basket`, and `Large basket` categories were also frequent (Figure 10.16(h)). As for Feature 6, the K-modes ($K = 2$) identified a cluster with neurons mainly categorized as `Characterized` (Figure 10.17(a)), whereas Figure 10.17(b) contains neurons categorized as either `Characterized` or `Uncharacterized` by different experts, showing disagreements for these neurons.



(a) (b)

Figure 10.12: Clusters of neurons obtained with the K-modes algorithm ($K = 2$) for Feature 1. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Intralaminar` category for clarity. Panels (a) and (b) clearly correspond to `Translaminar` and `Intralaminar` categories, respectively. The number of neurons (N) for each cluster is shown at the bottom of each panel.

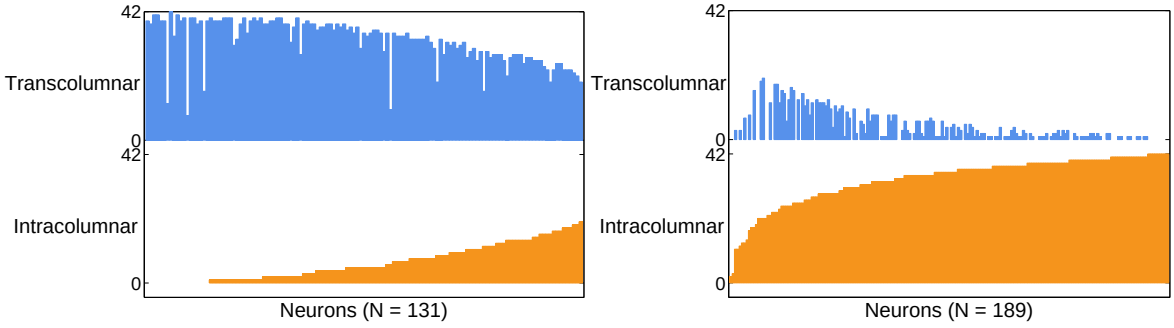


Figure 10.13: Clusters of neurons obtained with the K-modes algorithm ($K = 2$) for Feature 2. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Intracolumnar` category for clarity. Panel (a) clearly corresponds to neurons mainly categorized as `Transcolumnar`, whereas panel (b) clearly corresponds to `Intracolumnar`.

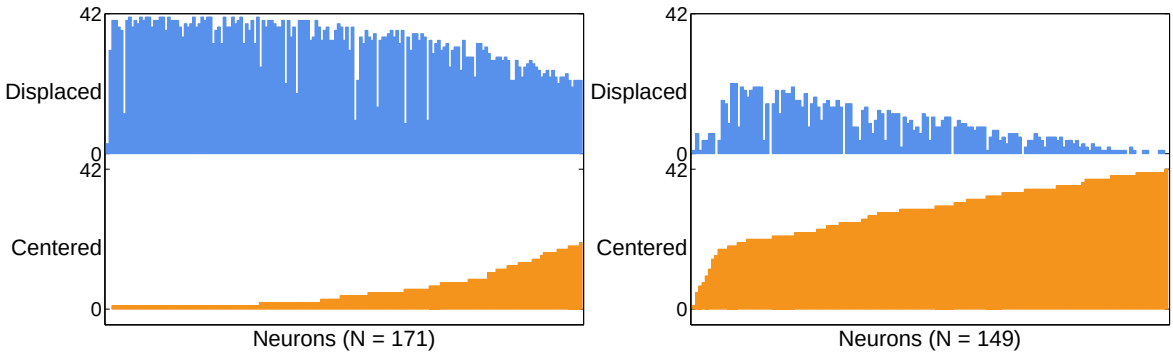


Figure 10.14: Clusters of neurons obtained with the K-modes algorithm ($K = 2$) for Feature 3. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Centered` category for clarity. Panels (a) and (b) clearly correspond to `Displaced` and `Centered` categories, respectively.

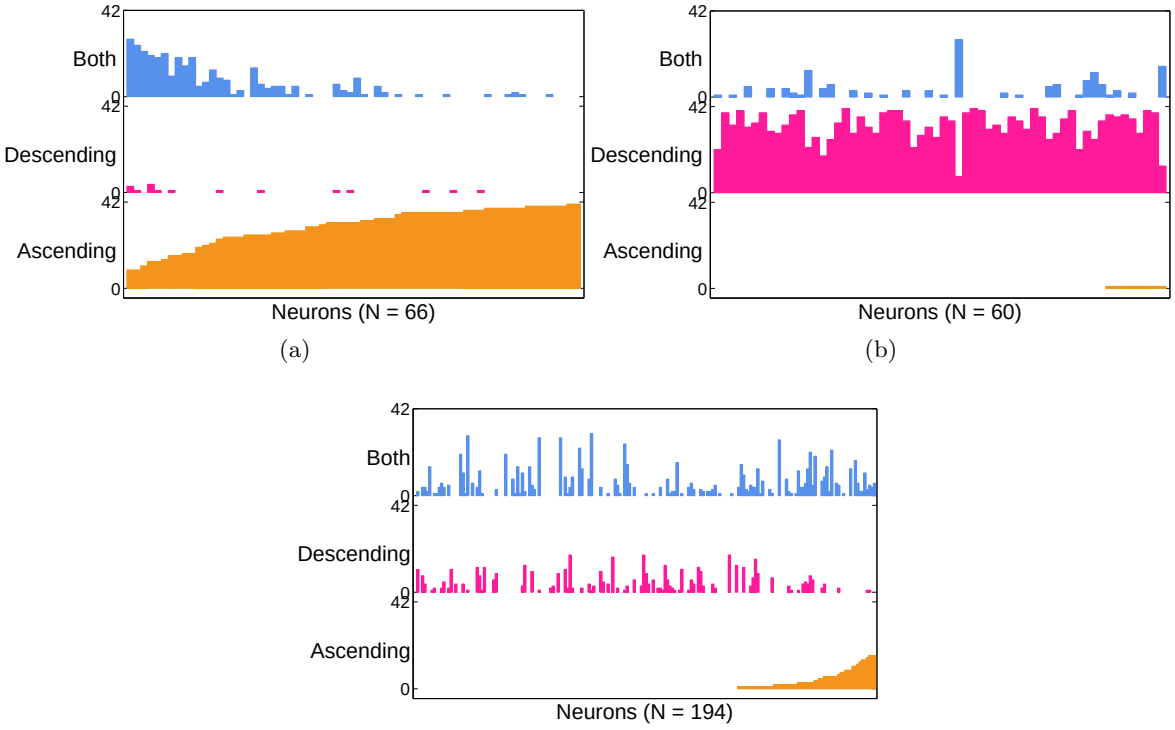


Figure 10.15: Clusters of neurons obtained with the K-modes algorithm ($K = 3$) for Feature 4. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Ascending` category for clarity. Panels (a) and (b) correspond to neurons mainly categorized as `Ascending` and `Descending`, respectively. Panel (c) shows neurons where different experts disagreed, categorizing them as `Ascending`, `Descending` or `Both`.

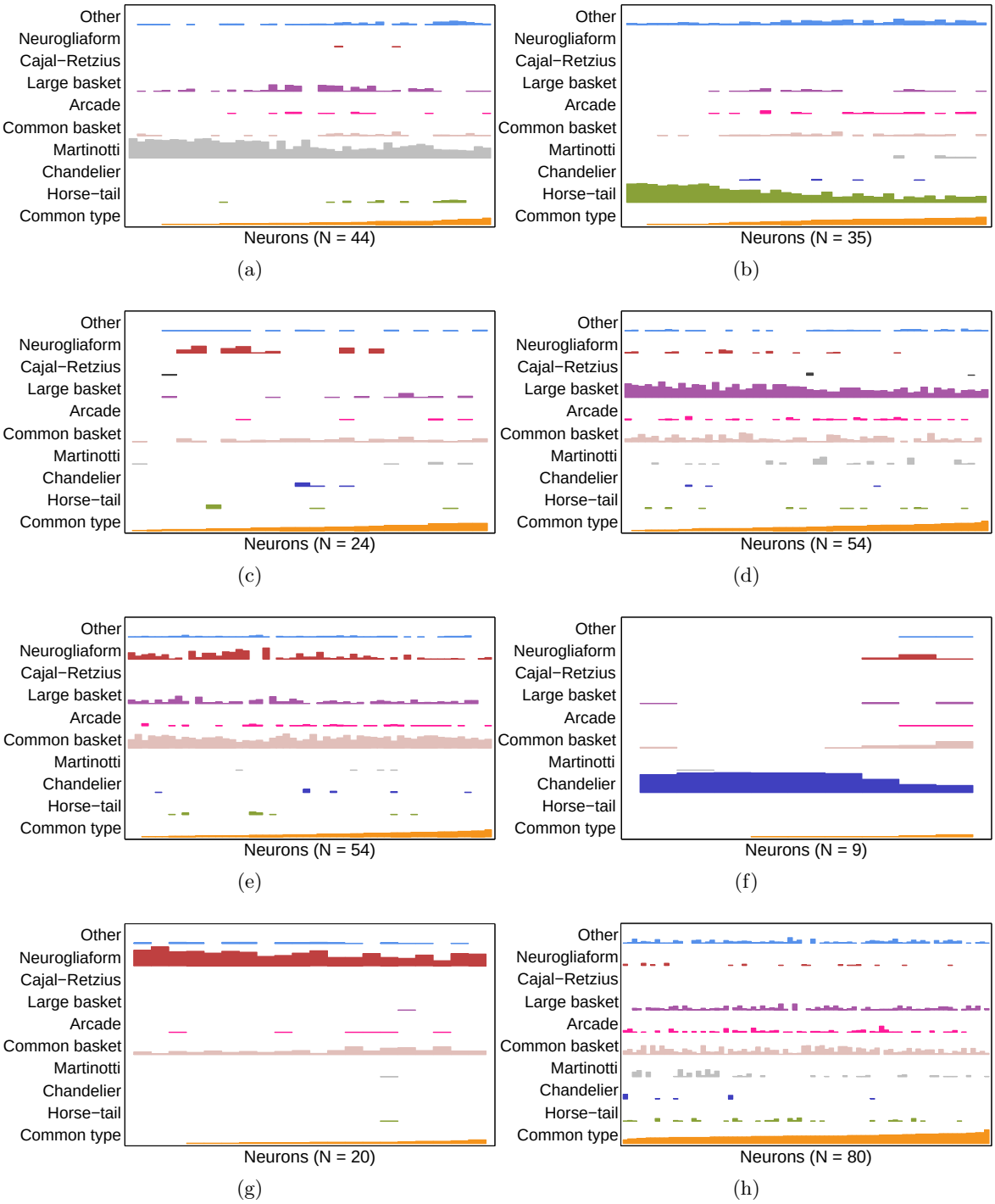
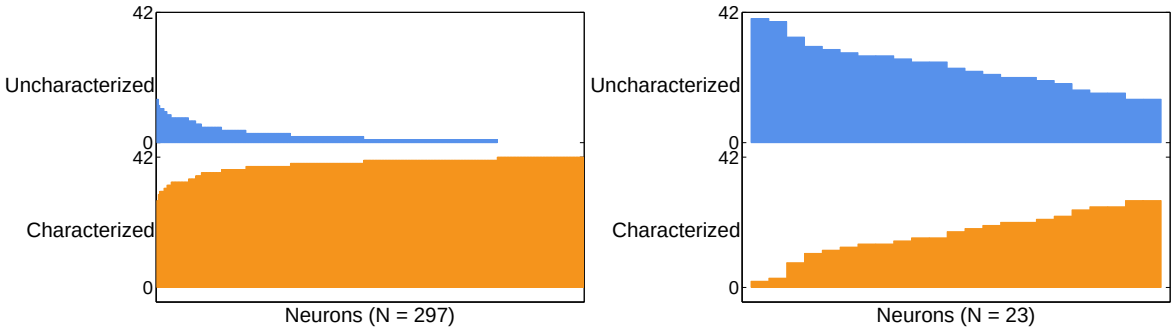


Figure 10.16: Clusters of neurons obtained with the K-modes algorithm ($K = 8$) for Feature 5. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Common type` for clarity. Panels (a) and (f) show clusters of neurons clearly corresponding to `Martinotti` and `Chandelier` cells, respectively. Other panels (e.g., (e)) show clusters of neurons that did not correspond to a single category.



(a) (b)

Figure 10.17: Clusters of neurons obtained with the K-modes algorithm ($K = 2$) for Feature 6. Vertical bars show the number of experts who selected each category for each neuron in the cluster. Neurons have been sorted in ascending order by the number of votes in the `Characterized` category for clarity. Panel (a) contains neurons mainly categorized as `Characterized`, whereas panel (b) contains neurons where different experts disagreed, categorizing them as either `Characterized` or `Uncharacterized`.

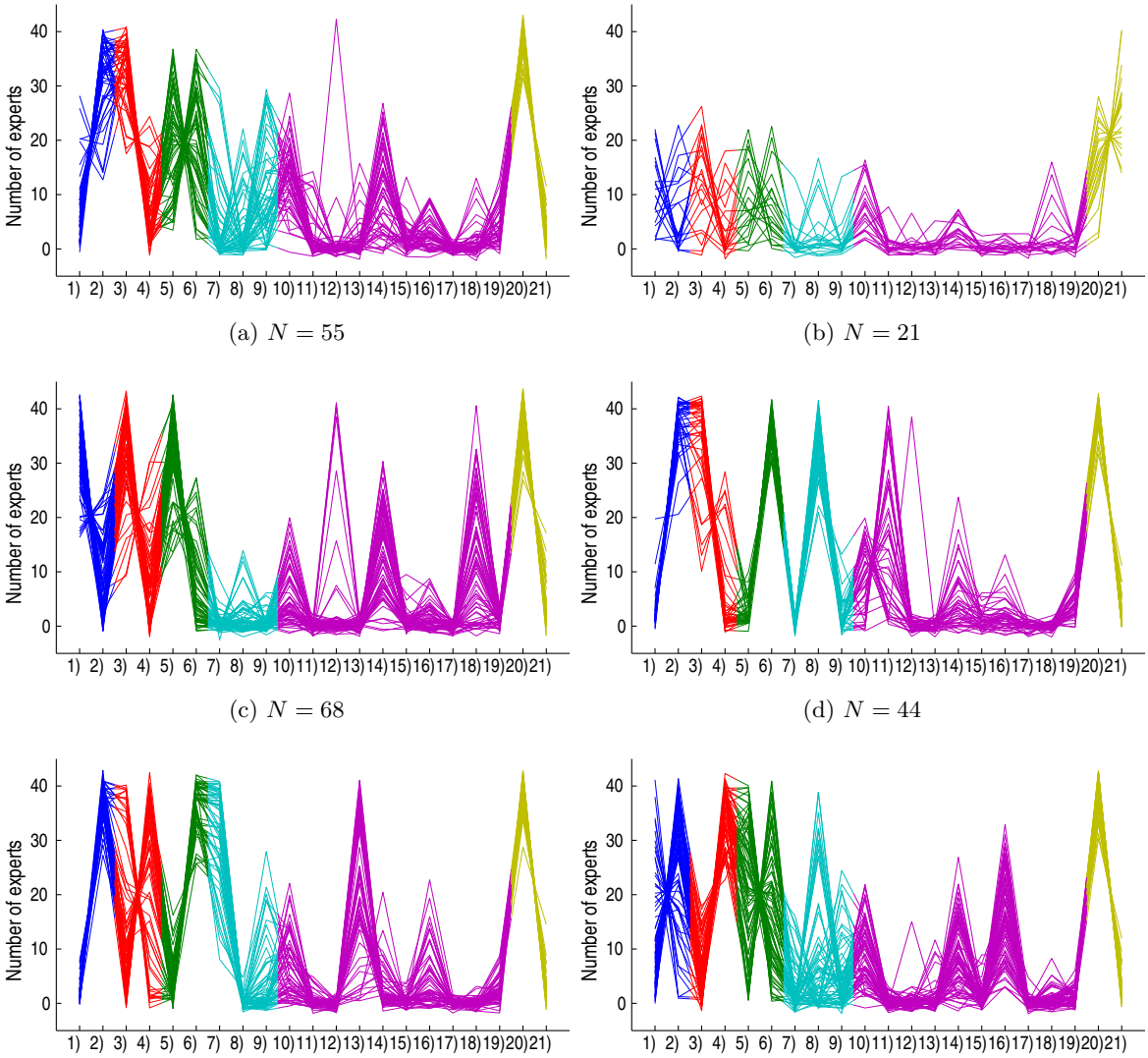
10.5.2 Neuron clustering for all the features

We wanted to generate clusters of cells taking into account the agreement of the experts in all the features at the same time. For every neuron, we computed the number of experts that assigned the neuron to each category of every feature. Therefore, the dataset used in the clustering algorithm had $N = 320$ instances (neurons), and each instance was an n-dimensional vector ($n = 21$), corresponding to all the categories of the six features: `Intralaminar`, `Translaminar`, `Intracolumnar`, `Transcolumnar`, `Centered`, `Displaced`, `Ascending`, `Descending`, `Both`, `Common type`, `Horse-tail`, `Chandelier`, `Martinotti`, `Common basket`, `Arcade`, `Large basket`, `Cajal-Retzius`, `Neurogliaform`, `Other`, `Characterized`, and `Uncharacterized`.

We used the K-means algorithm to cluster cells according to the number of votes each neuron had in each category. Different numbers of clusters (six to ten) were analyzed. The clearest results for biological interpretation were obtained with $K = 6$. For continuous data, K-means uses Euclidean distance to compute the distance between every two items $d(\mathbf{x}_i, \mathbf{x}_j)$, $i \neq j$. Every time step 2.(c) in Algorithm 10.1 is performed, K-means computes the cluster centers as the centroid of the items in the cluster (`Combine` operation).

The algorithm was run 25 times with different initial values for the cluster centers to avoid local optima, similarly to K-modes, and the best result was shown. The clusters were illustrated using parallel coordinate diagrams [514]. Each line represents one neuron in the cluster and its height shows the number of experts who selected each category for that neuron. A small amount of noise drawn from a normal distribution $\mathcal{N}(0, 0.75)$ was added to the values to ensure that all lines were visible.

Figure 10.18 represents the clusters obtained in the analysis. We found some clusters containing neurons with clearly identified categories. For example, Figure 10.18(c) shows a cluster of neurons that were clearly categorized as `Intralaminar`, `Intracolumnar`, `Centered`, and `Characterized`. Furthermore, some of these neurons were mainly categorized as either `Common type`, `Chandelier`, `Common basket` or `Neurogliaform`. Similarly, Figure 10.18(e) shows neurons that were mainly categorized as `Translaminar`, `Transcolumnar`, `Displaced`, `Ascending`, `Martinotti`, and `Characterized`. On the other hand, Figure 10.18(a) shows a cluster of neurons mainly categorized as `Translaminar` and `Intracolumnar`, but were not clearly categorized for the rest of the features. The cluster in Figure 10.18(d) includes neurons mainly categorized as `Characterized`, `Translaminar`, `Intracolumnar`, `Displaced`, `Descending` and `Horse-tail`. Figure 10.18(f) shows a cluster with low agreement, where categories `Common type`, `Common basket` and `Large basket` received most of the votes. Finally, Figure 10.18(b) shows a cluster of neurons showing no clearly identified categories, corresponding mainly to `Uncharacterized` neurons.



(a) $N = 55$ (b) $N = 21$

(c) $N = 68$ (d) $N = 44$

(e) $N = 60$ (f) $N = 72$

Figure 10.18: Clusters of neurons considering all features simultaneously. (a-f) Parallel coordinates diagrams of clusters of neurons obtained with the K-means algorithm ($K = 6$) considering all the features at the same time. Each line represents one neuron, showing the number of experts who selected each category of every feature when classifying that neuron. For example, panel (b) shows a cluster where the majority of neurons were categorized by many experts as `Translaminar` (blue), `Transcolumnar` (red), `Displaced` (green), `Ascending` (light blue), `Martinotti` (purple) and `Characterized` (ochre). The categories are numbered: 1) `Intralaminar`, 2) `Translaminar`, 3) `Intracolumnar`, 4) `Transcolumnar`, 5) `Centered`, 6) `Displaced`, 7) `Ascending`, 8) `Descending`, 9) `Both`, 10) `Common type`, 11) `Horse-tail`, 12) `Chandelier`, 13) `Martinotti`, 14) `Common basket`, 15) `Arcade`, 16) `Large basket`, 17) `Cajal-Retzius`, 18) `Neurogliaform`, 19) `Other`, 20) `Characterized`, and 21) `Uncharacterized`.

10.6 Bayesian networks for modeling experts' opinions

A BN was learned for each one of the $R = 42$ experts who completed the experiment. The goal was to build a model which captures how each expert understands the values of the morphological attributes and their relationships. The graphical representation of the BN structures offers a compact and easy way for the experts in the domain to interpret their models. The BNs were learned independently for each expert, so they do not capture whether or not the experts classified the same neurons in the same way. However, since the experts classified the same set of interneurons, we can use the BNs to systematically analyze their opinions and behaviors. One would expect that if two experts differed in their opinions (as encoded in their BNs), then they would also classify the neurons differently.

Therefore, one dataset for each expert was generated with the classifications provided in the experiment. The resulting dataset had $N = 320$ observations (the number of interneurons in the experiment) and $n = 6$ variables, which corresponded to the features that the experts were asked to classify. Some restrictions on different combinations of feature values were imposed in the experiment design, e.g., selecting `Uncharacterized` in the first feature disabled all the other variables. Therefore, when a neuron was classified as `Uncharacterized`, the values for the other variables were empty. Similarly, Feature 4 was only available for classification when `Translaminar` and `Displaced` were selected for Features 1 and 3, respectively. To build the dataset for each expert, we filled in incomplete observations with a new category named `Dummy`. Therefore, for each expert, we had a dataset with categorical variables $X_i, i = 1, \ldots, 6$ with r_i values, respectively:

X_1 ($r_1 = 3$): `Intralaminar`, `Translaminar`, `Dummy`.

X_2 ($r_2 = 3$): `Intracolumnar`, `Transcolumnar`, `Dummy`.

X_3 ($r_3 = 3$): `Centered`, `Displaced`, `Dummy`.

X_4 ($r_4 = 4$): `Ascending`, `Descending`, `Both`, `Dummy`.

X_5 ($r_5 = 11$): `Common type`, `Horse-tail`, `Chandelier`, `Martinotti`, `Common basket`, `Arcade`, `Large basket`, `Cajal-Retzius`, `Neurogliaform`, `Other`, `Dummy`.

X_6 ($r_6 = 2$): `Characterized`, `Uncharacterized`.

We used the data provided by each expert in the experiment to learn a BN which encoded the conditional independence relationships between the variables for that expert. The BNs were learned from data using a score+search approach (see Section 3.3.2). The GTT algorithm (see Algorithm 3.2 on page 32) was used to search the space of DAGs, and the K2 score (Equation (3.5) on page 30) was used to evaluate the candidate network structures. The parameters of the BNs were fitted using maximum likelihood estimates with Laplace correction. We did not allow any variable to be a parent of variable X_6, corresponding to Feature 6. This restriction encoded the knowledge that the decision of classifying a neuron

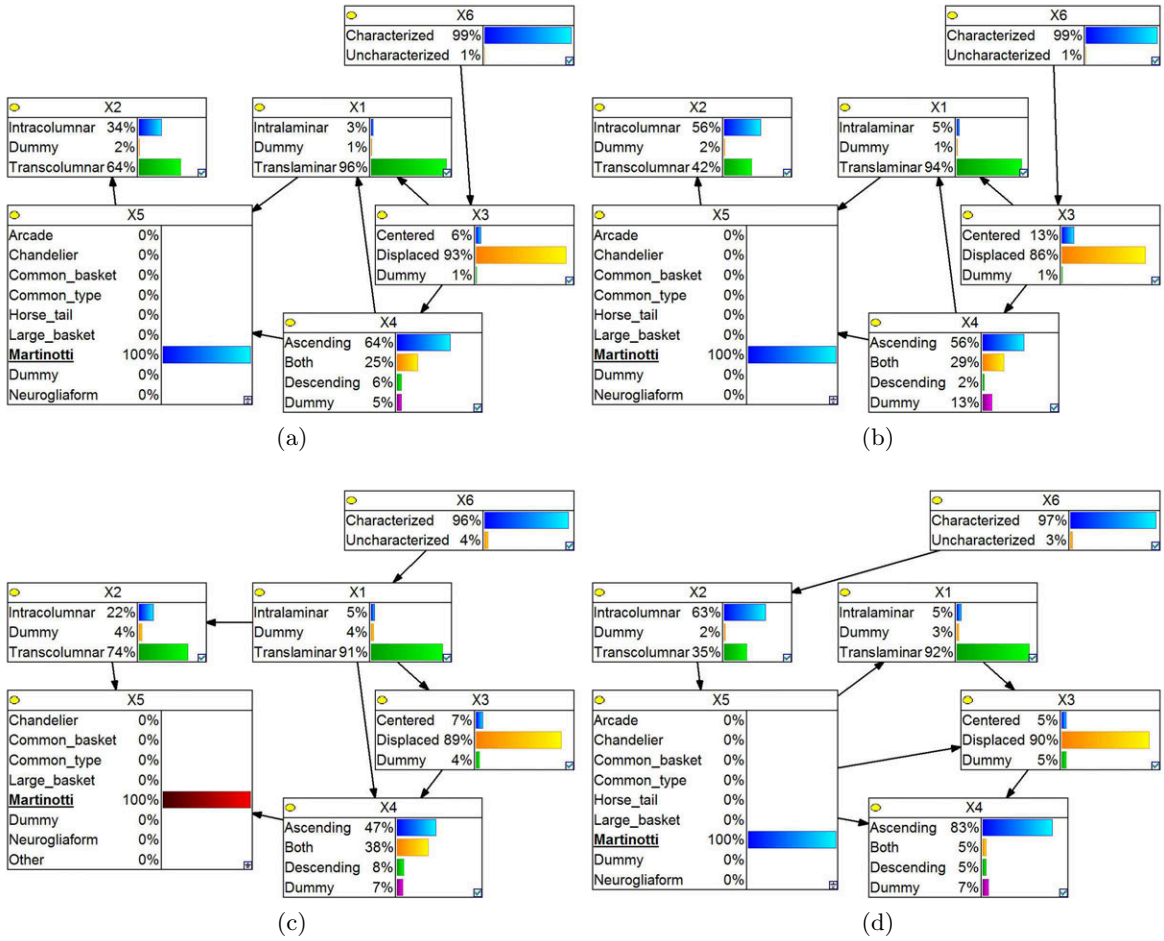


Figure 10.19: BNs for experts 16 (a), 17 (b), 27 (c) and 32 (d). `Martinotti` was selected in Feature 5 and the probabilities were propagated through the four BNs. Bar charts show the propagated probabilities of the remaining features conditioned on the `Martinotti` category.

as `Characterized` or `Uncharacterized` should be taken before classifying all the other features (modeled with variables X_1 to X_5). We limited the complexity of the BNs by imposing a maximum of three parents for each variable. This allowed us to control the size of the conditional probability distributions and to compute robust estimates of their parameters. However, this was not a very restrictive constraint since only 5 out of $6 \times 42 = 252$ variables in all the BNs had three parents.

As an example, Figures 10.19 and 10.20 show four BNs corresponding to four different experts. The BNs for experts 16 (Figure 10.19(a)) and 17 (Figure 10.19(b)) had the same structure and similar propagated probabilities when these experts assigned a neuron as a `Martinotti` cell in X_5. The greatest difference between the two networks occurred in X_2, where the probabilities of `Intracolumnar` and `Transcolumnar` were respectively 0.34 and 0.64 for expert 16, and 0.56 and 0.42 for expert 17. The BNs for experts 27 (Figure 10.19(c)) and 32 (Figure 10.19(d)) had a different structure from each other and from experts 16 and

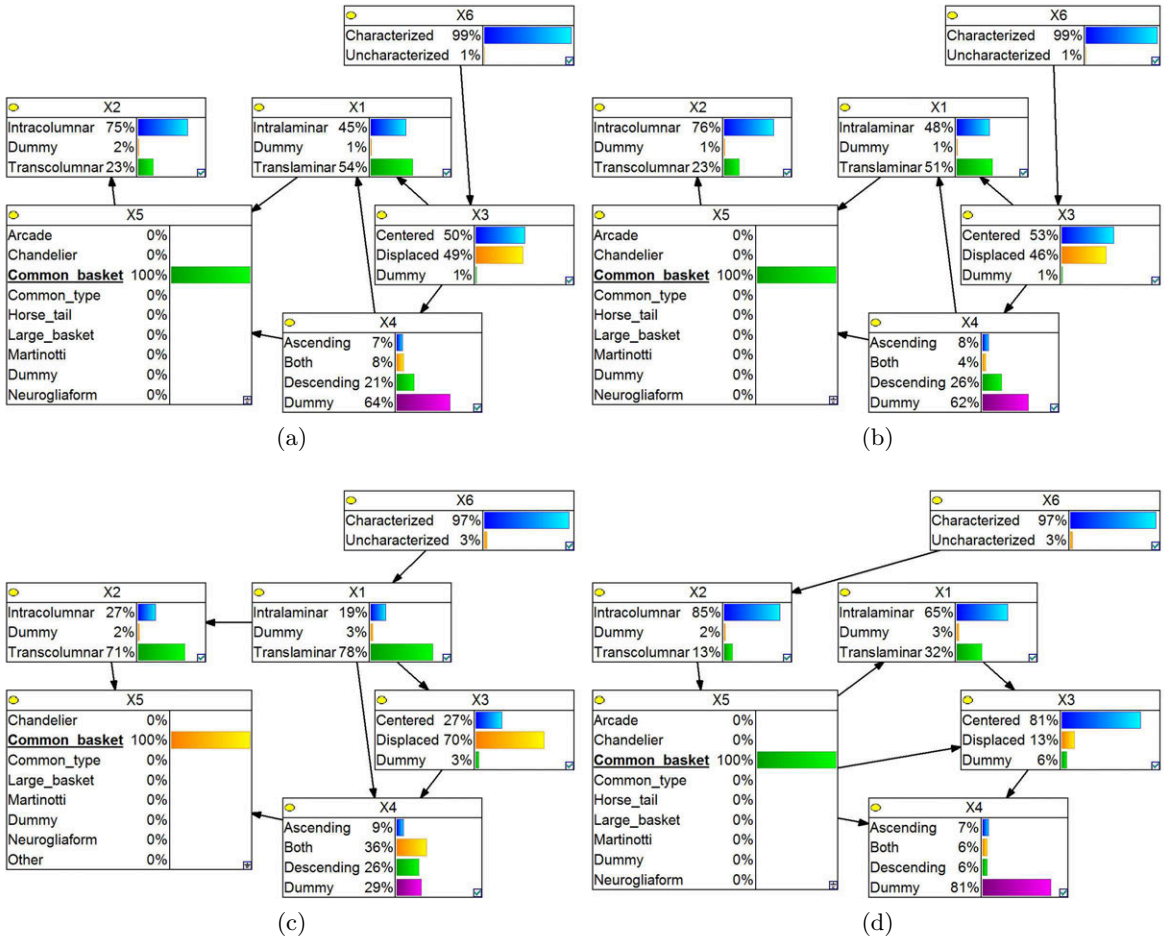


Figure 10.20: BNs for experts 16 (a), 17 (b), 27 (c) and 32 (d). `Common basket` was selected in Feature 5 and the probabilities were propagated through the four BNs. Bar charts show the propagated probabilities of the remaining features conditioned on the `Common basket` category.

17. However, the probabilistic reasoning on X_1 and on X_3 when the four experts considered a neuron as a `Martinotti` cell was similar, e.g., these four experts agreed (assigning probabilities higher than 0.86) that `Martinotti` cells were `Translaminar` and `Displaced`. Differences between the experts could also be identified in the BNs, e.g., X_5 in Figure 10.19(c) did not include as possible categories `Arcade` or `Horse-tail` cells, but included category `Other`. That means that expert 27 did not categorize any neuron as `Arcade` or `Horse-tail`.

We also used BNs to analyze the disagreements between experts about the classification of interneuron types. Figure 10.20 shows the BNs for the same four experts when `Common basket` was selected as evidence in X_5 and the probabilities were propagated. The posterior probabilities for expert 16 (Figure 10.20(a)) and expert 17 (Figure 10.20(b)) were similar but they were different for expert 27 (Figure 10.20(c)) and expert 32 (Figure 10.20(d)). For example, regarding X_1, the probability of `Translaminar` was 0.78 in Figure 10.20(c) and

0.32 in Figure 10.20(d). With respect to X_2, the probability of a `Common basket` being `Transcolumnar` was 0.71 in Figure 10.20(c), whereas in the other three BNs the probability was below 0.24. For X_3, `Centered` was the most probable value in Figure 10.20(d) and `Displaced` had the highest probability in Figure 10.20(c). Also, Figure 10.20(c) shows a higher probability for the category `Both` in X_4 than the other BNs.

The analysis of the 42 BN structures is summarized in Figure 10.21, including frequent relationships (high numbers) and rare relationships between features. X_1, X_3 and X_4 appeared frequently related. This could be explained by the fact that categories `Ascending`, `Descending` and `Both` are associated to categories `Translaminar` and `Displaced`, describing the vertical orientation of the neuron. X_5 was frequently linked to X_1, X_2, and X_4 in more than half of the BN structures. Therefore, these three features (laminar, columnar, and ascending/descending) are identified in this analysis as relevant when describing morphological properties of interneuron types (X_5).

	X_1	X_2	X_3	X_4	X_5	X_6
X_1		4	32	30	30	17
X_2			0	0	38	17
X_3				41	14	14
X_4					29	2
X_5						10
X_6						

Figure 10.21: Number of BNs out of 42 that include the possible (undirected) edge between the nodes in the corresponding row and column. The presence of an edge in the BN indicates that the choices of categories in those features by that expert are related. Frequency of relationships is highlighted with a gradient of color shades from red (most frequent) to white (non-existent or rare).

10.7 Supervised classification of interneurons

We aimed to build a model that could automatically classify the neurons in each of the six features on the basis of a set of morphological measurements of the digital 3D reconstructions. From the total of 320 neurons, we used the 241 neurons for which digital 3D morphological reconstructions were available. We loaded the neurons in Neurolucida Explorer software and performed the branched structure, convex hull, Sholl, fractal, vertex and branch angle analyses. These analyses were conducted for the complete neuronal morphology as well as separately for the dendritic and axonal arbors. These analyses yielded a set of 2,886 morphological measures of each neuron, including:

General information about the dendrites and the axons, e.g., the number of endings, the number of nodes (branching points), the total length and the mean length of each dendritic arbor, etc.

Morphometric measures of the soma such as the area, aspect ratio, compactness, con-

vexity, contour size (maximum and minimum feret), form factor, perimeter, roundness and solidity.

The total, mean, median and standard deviation of the length of the segments belonging to dendritic arbors and axons independently. Also, we performed these analyses dividing the segments by their centrifugal order from the soma.

Number of nodes and segments of the complete dendrites and axons, and number of nodes and segments measured by centrifugal order.

Convex hull analysis. We performed 2D and 3D convex hull analysis of the dendrites and the axon independently to obtain measures of the area, perimeter, volume and surface of the neuronal morphology.

Sholl analysis. We computed the number of intersections in concentric spheres centered at the soma with increasing radii of 20 μm. We also used the number of endings, nodes and the total length of the segments included in those spheres.

Fractal analysis. We computed the fractal dimension for the dendrites and the axon independently using the box-counting method [351]. The fractal dimension is a quantity that indicates how completely the neuron fills space.

Vertex analysis of the connectivity of the nodes in the branches to describe the topological and metric properties of the arbors. We used the number of nodes of each one of the three types: V_a (branching points where the two child segments end), V_b (branching points where one of the child segments end) and V_c (branching points where the two child segments bifurcate). We also used the ratio V_a/V_b and computed the number of nodes of each type by centrifugal order.

Branch angle analysis. We used planar, local and spline angles that measure the direction of the branches at different levels. We computed the mean, standard deviation, and median of the three angles for dendrites and axon individually. Additionally, we computed the mean, standard deviation, and median of the angles of the segments grouped by centrifugal order.

Many variables were measured according to the centrifugal order of the segments they belonged to. Since neurons have different maximum centrifugal order, length, etc., each neuron had a different number of computable variables. For example, one neuron might have dendrites with a maximum centrifugal order of 9 and another neuron could have dendrites with a maximum centrifugal order of 5. Variables that measured neuron morphology at orders 6, 7, 8 and 9 were not computable in the second neuron, so we set those values to 0 to be manageable by the algorithms. Variables concerning the complete neuron morphology are not affected by this issue, since they were obtained from the data directly coming from the 3D reconstructions. For each one of the features in the experiment, we had to assign a single "true category" (true class label) to each neuron. We used the most frequently occurring

value in the 42 assignments made by the experts who completed the experiment, i.e., we applied a simple majority vote to assign a class label to each neuron for each feature. Using this approach, there were no neurons categorized as `Arcade`, `Cajal-Retzius` or `Other` by the majority of the experts.

We applied the following 10 classification algorithms available in Weka [236] using their default parameters [519] (see Section 2.3.1 for a brief description of the supervised learning approaches).

`NB`: Naive Bayes classifier, where the conditional distributions of the continuous variables given the class values are modeled using Gaussian distributions [408].

`NBdis`: Discrete naive Bayes classifier [372]. The continuous variables are discretized using Fayyad and Irani's supervised discretization technique [170].

`RBFN`: Artificial neural network for classification tasks with one single hidden layer that uses Gaussian radial basis functions as activation functions [45].

`SMO`: Support vector machine with polynomial kernels implementing the sequential minimal optimization algorithm [293, 416].

`IB1`: Nearest neighbor classifier [7].

`IB3`: Nearest neighbor classifier using 3 neighbors.

`JRip`: Rule induction technique using RIPPER algorithm [95].

`J48`: Classification tree using C4.5 algorithm [421].

`RForest`: Classification technique using a set of random tree classifiers [64].

`RTree`: Classification tree that chooses the variables at each node randomly.

Additionally, two variable selection methods were studied:

`GainRatio`: A univariate filter algorithm that ranks the predictive variables according to their information gain ratio [420] with the class label and keeps the best 500 variables.

`CfsSubset`: This algorithm tries to find a subset of predictive variables that is highly correlated with the class, but has low intercorrelation between the predictive variables. It starts with an empty subset and iteratively adds the variable that yields a subset with the highest correlation value. The correlation measures the symmetric uncertainty of each variable in the subset with the class (to maximize), and adjusts it to take into account the symmetric uncertainty between the predictive variables (to minimize). The symmetric uncertainty is a measure of correlation based on the marginal entropies and the joint entropies between pairs of variables [237].

The accuracy of the classifiers was estimated using the leave-one-out technique (see Section 2.3.2). Additionally, we performed an exact binomial test to test the hypothesis that the number of correctly classified neurons is greater than that expected with a base classifier always assigning the class with maximum prior probability. To estimate the number of correctly classified neurons, we multiplied the accuracy reported by the leave-one-out technique by 241. The null hypothesis is that the number of correctly classified neurons matches 241 times the maximum prior probability. The alternative hypothesis is that the number of correctly classified neurons is higher than 241 times the maximum prior probability. Statistical significance was established when the p-values were smaller than the significance level $\alpha = 0.05$.

These classification algorithms were applied in three different settings. Section 10.7.1 reports the results for the classifiers for each feature independently. The results of the binary classifiers for each category in Feature 5 are shown in Section 10.7.2. Finally, we learned classifiers merging confusing interneuron types in Section 10.7.3.

10.7.1 Classifiers for each feature independently

Each one of the features in the experiment was considered independently. The number of class values was the same as the number of categories in the features, i.e., two class values for Features 1-3 and 6; and three class values for Feature 4. There were no neurons classified as `Arcade`, `Cajal-Retzius` or `Other`, so the classifiers for Feature 5 had 7 class values.

Table 10.2 shows the accuracy of the classifiers. The classifiers were able to distinguish whether or not a neuron was `Characterized`, as the best result in accuracy is 99.17% (2 neurons misclassified). The best performing classifiers for Feature 1 and Feature 2 yielded an accuracy over 80%, whereas the best result for Feature 3 was 73.86%. The accuracy of the classifiers was below 70% for both Feature 4 and Feature 5. One explanation for the low accuracy for Feature 5 is that the class labels were not very reliable because the experts frequently disagreed when classifying the neurons in this feature. However, it is also possible that the interneuron classes could not be distinguished using the set of morphological measurements included in the study. Moreover, according to majority votes, the number of neurons assigned by the experts to the different interneuron types were unbalanced, with only three `Chandelier` cells and four `Neurogliaform` cells, but as many as 77 `Common type` cells and 68 `Common basket` cells. This makes it especially difficult for the classifiers to distinguish the least frequent neuronal types. Surprisingly, the classifiers achieved the lowest accuracy for Feature 4. This may be explained by the same two factors indicated above: the `Both` category was confusing to the experts, so the neurons might have been assigned to the wrong category. Also, there may be no morphological variables that capture the orientation of the axon. To test the significance of these results, we computed the category with maximum prior probability for the classifier induced for each feature independently:

Feature 1: 0.7718 (achieved at `Translaminar`)

Feature 2: 0.5187 (`Transcolumnar`)

Table 10.2: Accuracy (%) of the classifiers trained for each feature independently using ten different classification algorithms (in columns) and three variable selection methods (in rows): `NoFSS` (no feature subset selection, i.e., all variables selected), `GainRatio`, and `CfsSubset`. The highest accuracy for each feature and variable selection method is highlighted in bold. Additionally, the overall highest accuracy for each feature is shaded in gray. A binomial test was used to check whether or not the classifiers outperformed a base classifier always selecting the category with maximum prior probability. Asterisks indicate a p-value < 0.05.

	NB	NBdisc	RBFN	SMO	IB1	IB3	JRip	J48	RForest	RTree
Feature 1: `Intralaminar` vs. `Translaminar`										
NoFSS	57.68	58.51	77.59	82.16∗	72.20	73.44	82.57∗	**85.48∗**	82.16∗	75.93
GainRatio	64.73	54.36	79.67	82.99∗	69.71	75.93	83.82∗	**85.48∗**	84.23∗	79.67
CfsSubset	75.93	75.10	81.33	84.23∗	73.86	80.08	**84.65∗**	80.08	82.16∗	80.08
Feature 2: `Intracolumnar` vs. `Transcolumnar`										
NoFSS	59.75∗	62.66∗	52.28	**75.52∗**	57.68∗	65.56∗	74.27∗	68.46∗	66.39∗	58.09∗
GainRatio	66.39∗	63.07∗	53.11	**76.35∗**	64.32∗	65.98∗	75.52∗	68.88∗	70.12∗	65.98∗
CfsSubset	72.61∗	65.56∗	76.76∗	**81.33∗**	73.86∗	73.03∗	74.69∗	70.54∗	76.35∗	69.29∗
Feature 3: `Centered` vs. `Displaced`										
NoFSS	62.24	53.94	54.77	**68.88∗**	64.73∗	68.05∗	66.80∗	67.63∗	68.46∗	62.24
GainRatio	64.73∗	**73.03∗**	65.98∗	70.54∗	65.56∗	71.37∗	70.54∗	66.39∗	72.20∗	68.46∗
CfsSubset	68.88∗	**73.86∗**	70.54∗	73.03∗	65.15∗	68.05∗	63.90∗	71.78∗	68.46∗	65.15∗
Feature 4: `Ascending` vs. `Descending` vs. `Both`										
NoFSS	34.44	27.80	44.40∗	49.38∗	41.91	38.59	33.61	**54.36∗**	40.25	37.76
GainRatio	43.57∗	33.20	43.98∗	**49.79∗**	41.91	42.32	43.57∗	46.89∗	45.64∗	42.74
CfsSubset	47.30∗	51.87∗	47.30∗	58.51∗	47.30∗	52.28∗	48.13∗	42.32	**60.17∗**	47.30∗
Feature 5: Interneuron type (7 classes)										
NoFSS	56.02∗	19.09	45.23∗	**58.51∗**	50.62∗	53.94∗	50.62∗	47.72∗	52.28∗	40.25∗
GainRatio	60.17∗	26.14	58.92∗	**62.24∗**	49.79∗	51.87∗	48.55∗	43.15∗	58.09∗	43.98∗
CfsSubset	61.00∗	43.57∗	**61.41∗**	60.58∗	58.09∗	56.85∗	53.94∗	49.38∗	56.85∗	51.45∗
Feature 6: `Characterized` vs. `Uncharacterized`										
NoFSS	77.18	88.38	95.85	**97.93∗**	97.51	97.51	**97.93∗**	97.51	96.27	95.85
GainRatio	**98.34∗**	73.86	97.51	96.68	97.10	97.51	97.93∗	97.93∗	97.51	**98.34∗**
CfsSubset	97.51	89.63	96.27	97.10	95.44	95.02	97.93∗	96.27	97.51	**99.17∗**

Feature 3: 0.5809 (`Displaced`)

Feature 4: 0.3817 (`Ascending`)

Feature 5: 0.3195 (`Common type`)

Feature 6: 0.9544 (`Characterized`)

For every feature, the best classifier in Table 10.2 significantly outperformed a base classifier which always selects the class with maximum prior probability according to an exact binomial test (see asterisks in Table 10.2).

To further analyze the results for Feature 5, Table 10.3 shows the confusion matrix of the best performing algorithm (`SMO`), which achieved an accuracy of 62.24% (Table 10.2). The confusion matrix shows the performance of an algorithm by displaying the number of neurons of each true category (rows) matched to the categories predicted by the classifier (columns). Some `Martinotti` cells were wrongly classified as `Common type` (9 cases), `Large basket` (4) and `Chandelier` (1). This was similar to the results shown by the clustering algorithms. `Horse-tail` cells were wrongly classified as `Common type` cells. Also, the classifier often

Table 10.3: Confusion matrix for the `SMO` classifier and the `GainRatio` variable selection method using Feature 5 data. The numbers in columns refer to the number of the interneuron types in rows. The main diagonal of the matrix (shaded) indicate the number of correctly classified neurons, whereas non-zero values outside the main diagonal show the number of wrongly classified neurons.

True class	Predicted class						
	1	2	3	4	5	6	7
1. `Common type`	55	1	0	5	11	5	0
2. `Horse-tail`	7	5	0	2	0	0	0
3. `Chandelier`	0	0	1	0	2	0	0
4. `Martinotti`	9	0	1	24	0	4	0
5. `Common basket`	15	1	0	0	49	3	0
6. `Large basket`	11	0	0	3	7	16	0
7. `Neurogliaform`	0	0	0	0	4	0	0

confused `Common type`, `Common basket` and `Large basket` neuron types (Table 10.3). The four `Neurogliaform` cells and two out of the three `Chandelier` cells were wrongly classified as `Common basket`.

10.7.2 Binary classifiers for each interneuron type

We induced a binary classifier (with two class values) to identify each category in Feature 5 versus all the other categories merged together. The goal was to check whether a particular interneuron type could be distinguished from all the other interneuron types. Neurons classified as `Chandelier` (3 neurons) or `Neurogliaform` (4 neurons) were very rare. Therefore, we did not induce binary classifiers for these two categories, because the class values would be too unbalanced for the classifiers to find the characterizing properties of these interneuron types. Table 10.4 shows the accuracies of the binary classifiers for each category. The classifiers for `Horse-tail` and `Martinotti` cells achieved high accuracies, whereas the classifiers for `Common type`, `Common basket`, and `Large basket` cells yielded lower accuracies. The maximum prior probabilities for these binary classifiers were:

`Common type` vs. the rest: 0.6805 (achieved at the rest)

`Horse-tail` vs. the rest: 0.9419 (the rest)

`Martinotti` vs. the rest: 0.8423 (the rest)

`Common basket` vs. the rest: 0.7178 (the rest)

`Large basket` vs. the rest: 0.8465 (the rest)

The induced classifiers were not able to significantly outperform the base classifier for `Horse-tail` and `Large basket` categories. Few neurons were categorized as `Horse-tail` by the majority of the experts, so it was difficult to induce classifiers able to distinguish this

Table 10.4: Accuracy (%) of the binary classifiers (in columns) induced for the categories in Feature 5 and two variable selection methods (in rows). Each classifier tried to identify whether a neuron belonged to a particular category vs. all other categories, and this was repeated for each category separately. The best results for each category and variable selection method are highlighted with bold face. The highest accuracy for a given category is shaded in gray. A binomial test was used to check whether or not the classifiers outperformed a base classifier always selecting the category with maximum prior probability. Asterisks indicate a p-value < 0.05.

	NB	NBdisc	RBFN	SMO	IB1	IB3	JRip	J48	RForest	RTree
					`Common type` vs. the rest					
NoFSS	61.83	54.36	71.37	70.95	69.29	75.52∗	77.59∗	76.76∗	**78.84∗**	68.88
GainRatio	67.22	63.49	75.10∗	69.71	71.37	74.27∗	75.10∗	**77.18∗**	75.10∗	68.88
CfsSubset	74.69∗	64.32	75.10∗	74.27∗	76.76∗	**78.84∗**	71.78	69.29	70.95	68.88
					`Horse-tail` vs. the rest					
NoFSS	91.70	51.87	93.36	94.19	90.87	94.19	92.53	90.87	**94.61**	92.53
GainRatio	86.31	88.38	90.46	94.61	92.95	94.19	92.12	90.87	**95.02**	93.78
CfsSubset	92.53	72.61	93.36	**95.02**	94.61	93.36	92.12	93.78	93.78	94.19
					`Martinotti` vs. the rest					
NoFSS	84.23	65.56	82.99	**88.80∗**	85.48	86.72	82.57	84.23	85.48	83.82
GainRatio	84.65	67.63	81.33	**88.38∗**	84.65	87.14	84.23	80.91	85.89	83.40
CfsSubset	85.89	77.18	86.31	87.97	87.97	**90.46∗**	84.65	84.23	87.55	85.48
					`Common basket` vs. the rest					
NoFSS	68.46	54.77	71.78	**79.25∗**	77.18∗	78.01∗	78.84∗	77.18∗	78.42∗	76.76∗
GainRatio	72.61	51.87	75.52	79.25∗	76.76∗	77.59∗	76.35	74.27	**83.40∗**	78.42∗
CfsSubset	78.01∗	78.84∗	80.91∗	**81.33∗**	77.59∗	77.18∗	79.25∗	74.69	80.91∗	79.25∗
					`Large basket` vs. the rest					
NoFSS	54.77	67.63	84.65	80.50	83.40	**85.06**	83.40	79.67	82.57	74.69
GainRatio	70.95	66.39	84.23	80.50	79.25	80.08	**84.65**	81.74	82.99	79.67
CfsSubset	81.74	59.34	81.33	82.99	80.91	82.57	84.23	82.57	**84.65**	80.08

category, even though it was easily distinguishable for the experts. This limitation should vanish when more data become available. On the other hand, neurons categorized as `Large basket` were difficult to distinguish for both experts and supervised classifiers.

10.7.3 Classifiers merging interneuron types

Following the agreement results observed in the previous analyses, we decided to check whether the classification algorithms performed better when interneuron types that are difficult to distinguish were merged into one category. Therefore, we trained classifiers after having merged the categories corresponding to `Common type`, `Common basket` and `Large basket` into a single category, as these three interneuron types were frequently confused with each other. The rest of the categories were considered individually. Table 10.5 shows the accuracy of these supervised classifiers. When merging the three categories (`Common type`, `Common basket`, and `Large basket`) into one single category, the accuracy of the best classifier increased from 62.24% (Table 10.2) to 83.40%. When we only merged `Common type` and `Common basket` cells, the best classifier accuracy was 73.86%. When only merging `Common type` and `Large basket`, the best classifier accuracy was 69.29%. Lastly, only merging `Common basket` and `Large basket` cells resulted in the best accuracy among classifiers of 70.12%. These re-

Table 10.5: Accuracy (%) of the classifiers (in columns) for Feature 5 in an analysis in which `Common type`, `Common basket`, and `Large basket` or pairs among them were merged into one category. Two variable selection methods are used (in rows). The best results for each combination of categories and variable selection method are highlighted with bold face. The highest accuracy for a given combinations of categories is shaded in gray. A binomial test was used to check whether or not the classifiers outperformed a base classifier always selecting the category with maximum prior probability. Asterisks indicate a p-value < 0.05.

	NB	NBdisc	RBFN	SMO	IB1	IB3	JRip	J48	RForest	RTree
`Common type` + `Common basket` + `Large basket` vs. each neuron type										
NoFSS	79.25	20.75	78.42	**82.57**∗	74.69	79.67	77.18	69.29	79.25	70.12
GainRatio	77.18	32.37	73.86	80.91∗	77.59	**82.16**∗	73.44	73.44	78.84	73.03
CfsSubset	80.91∗	50.21	80.91∗	80.91∗	80.50∗	**83.40**∗	74.27	75.10	**83.40**∗	74.69
`Common type` + `Common basket` vs. each neuron type										
NoFSS	**67.22**∗	21.99	58.51	66.80∗	60.58	66.80∗	58.51	53.53	62.66	59.75
GainRatio	64.32	27.39	63.49	**73.03**∗	63.07	65.56∗	61.83	51.45	69.71∗	61.83
CfsSubset	68.88∗	39.83	68.88∗	73.03∗	70.12∗	**73.86**∗	64.73	63.90	69.71∗	63.07
`Common type` + `Large basket` vs. each neuron type										
NoFSS	60.17∗	19.92	49.38	**64.73**∗	57.26∗	59.75∗	56.02∗	51.45	59.34∗	51.45
GainRatio	64.73∗	27.80	59.75∗	**68.46**∗	59.34∗	63.07∗	55.60	50.62	64.32∗	54.36∗
CfsSubset	65.56∗	39.83	69.71∗	64.73∗	65.98∗	**69.29**∗	57.68∗	59.75∗	64.73∗	54.36∗
`Common basket` + `Large basket` vs. each neuron type										
NoFSS	60.17∗	16.60	54.36∗	**65.56**∗	58.09∗	61.00∗	55.60∗	52.70∗	62.66∗	54.36∗
GainRatio	61.41∗	51.45∗	65.98∗	64.32∗	61.00∗	65.56∗	55.60∗	49.79∗	**67.63**∗	53.53∗
CfsSubset	66.39∗	41.08	68.46∗	**70.12**∗	64.73∗	66.80∗	52.70∗	58.09∗	68.46∗	55.19∗

sults suggest that `Common type`, `Common basket`, and `Large basket` are not well-defined categories. For all these experiments, the induced classifiers significantly outperformed the base classifiers using the maximum prior probabilities:

`Common type` + `Common basket` + `Large basket` vs. each neuron type: 0.7552 (at `Common type` + `Common basket` + `Large basket`)

`Common type` + `Common basket` vs. each neuron type: 0.6017 (`Common type` + `Common basket`)

`Common type` + `Large basket` vs. each neuron type: 0.4730 (`Common type` + `Large basket`)

`Common basket` + `Large basket` vs. each neuron type: 0.4357 (`Common basket` + `Large basket`)

10.8 Conclusion

In this chapter we empirically and quantitatively demonstrated that the utilitarian approach to neuron classification is problematic at this time, confirming the impression that different researchers use their own, mutually inconsistent schemes for classifying neurons based on morphological criteria. Many ambiguities are independent of the relative reconstruction quality and completeness of the tested neurons. A striking indication of the problem is that,

in several cases, experts assigned different names to a neuron in this classification experiment and in their own original publication from which that same neuron was taken. The analyses of inter-expert agreement, application of BNs, and different clustering and classification algorithms revealed readily distinguishable interneuron types and apparently confusing interneuron names. High-consensus terms included `Chandelier` and `Martinotti` cells, indicating that these are more easily identifiable interneuron types. Low-consensus terms included `Arcade`, `Common basket`, `Large basket` and `Cajal-Retzius` cells, suggesting that these are potentially less useful names. Researchers generally agreed on specific morphological features, such as `Ascending` vs. `Descending` and `Intracolumnar` vs. `Transcolumnar` axonal arbor.

It should be kept in mind that the present analysis is limited to neurons from a small number of species, representing mammals commonly used in brain research. These include: rabbit, rat, mouse, cat, human and monkey. Although the results from our analysis may be consistent among these mammalian orders, the level of inter-expert agreement was not compared between species. Furthermore, the selection of interneurons from these species does not cover the likely variability of interneuronal morphologies among all mammalian families. In fact, except for the cat, the species in our study all belong to only one mammalian superorder. Although several "canonical" neuronal morphologies are doubtlessly common to all placental mammals, some species depart from the commonly observed neuron types [258, 460]. Future inclusion of other species in the study will allow detailed analysis of evolutionary conservation and species-specific neuron types.

The data collected for this research and the classifications provided by the experts for each interneuron may be used in very different ways. BNs are the representation of choice for uncertainty in artificial intelligence. BN fusion and aggregation is a topic of intense research. In Chapter 11 we propose a method for learning a consensus BN that accurately represents the opinions of a group of experts, and apply it to this data set of interneurons. Also, in this chapter we used majority voting to aggregate the opinions of the experts and assign a single category (class label) to each interneuron. However, more flexible approaches like the ones mentioned in Section 2.5 could be used to analyze these data, e.g., semi-supervised learning, partially supervised learning, etc. In Chapter 12 we propose a method for learning BNCs when the true class label of the training instances is not known. Instead, for each training instance, we have a vector with counts that model the number of experts who selected each class label, and we use that information for learning the BNCs.

Chapter 11

Bayesian network modeling of the consensus between experts

11.1 Introduction

In Chapter 6 we highlighted the extreme morphological, molecular and electrophysiological variability of neuronal cells [140, 413, 436, 472]. Neuronal morphology is a key feature in the study of brain circuits, as it is highly related to information processing and functional identification. However, except for some special cases, this variability makes it hard to find a set of features that unambiguously define a neuronal type [413]. In Chapter 10 we saw that the classification of GABAergic interneurons in particular has been a challenge for some time.

In this chapter, we present a methodology for building a BN that models the opinions of a group of experts when performing a classification experiment. In Section 10.6, a BN was learned for each one of the 42 experts, representing his/her behavior in the classification task. Here, we retrieve those BNs and run a clustering algorithm to find groups of experts with similar behaviors. Then, a representative BN is induced for each cluster of experts. Expert behavior when classifying the set of interneurons was extremely variable. Therefore, experts with similar behaviors have to first be clustered and then combined. Otherwise, combining all experts behaviors into a single consensus model would presumably hide some of these differing behaviors [164, 223, 224]. In this way, we explicitly model each group of similar experts as a representative BN for the cluster. The final consensus model is a Bayesian multinet [213] encoding a mixture of BNs [368, 487], where each component is the BN which represents the opinions of a cluster of experts. A similar idea was proposed for case-based BNs [270, 440], where the authors clustered the observations before learning a BN which captures the different properties of each cluster. Bayesian multinets are a kind of asymmetric BN which allows to model different statistical (in)dependencies between the variables for different values of a distinguished variable. Bayesian multinets can capture local differences between variables and model the problem domain more closely, allowing for sparser models

and more robust parameter estimation. For instance, they have been shown to outperform other BN models in some supervised learning problems [84].

We apply the proposed methodology to the problem of the morphological classification of GABAergic interneurons from the cerebral cortex (see Chapter 10). The final model is studied at length to validate the proposed methodology and to gather useful knowledge for neuroscience research. The resulting consensus Bayesian multinet is used to analyze the behavior of a set of experts and to reason about the underlying classification task. The representative BNs for each cluster are compared to find similarities and differences between groups of experts and to identify different behaviors or currents of opinion. Also, we use the consensus model to reason about the task the experts were asked to perform. For instance, we introduce some evidence into the consensus Bayesian multinet and infer "agreed" answers to those queries. These "agreed" answers are compared to those obtained by each representative BNs to find clusters of experts with outlying behaviors against experts with moderate opinions.

The methodology presented in this study can be applied to a wide range of scientific fields. For instance, in a medical setting, it may be interesting to model and analyze the different opinions of a group of physicians regarding the diagnosis, prognosis or the most appropriate treatment for a given disease. Another example can be found in a risk assessment scenario, where different people could have different opinions on a given matter depending on their personal preferences, risk perception, etc. The process of obtaining the opinions of different experts on a given task (here, the morphological classification of interneurons) is challenging because it can be difficult, costly and time-consuming. However, new Internet tools and crowd-sourcing techniques have alleviated some of these problems, and obtaining classification data from different experts is now affordable for a lot of problems [141]. The proposed methodology is not restricted to the scenario where class labels are given by experts. For instance, in the medical setting, different diagnostic tests can yield conflicting outputs on the condition of a given patient. Also, in the field of web mining, several class labels can be automatically assigned to an object (image, video, document, etc.) based on, e.g., titles, captions or user comments [426].

The research included in this chapter is being published in López-Cruz et al. [342].

Chapter outline

The chapter is organized as follows. Section 11.2 details the proposed methodology for building a consensus Bayesian multinet which models experts' opinions. Section 11.3 includes the evaluation of the model and the biological interpretation of the results. Finally, Section 11.4 ends with conclusions and suggestions for future work.

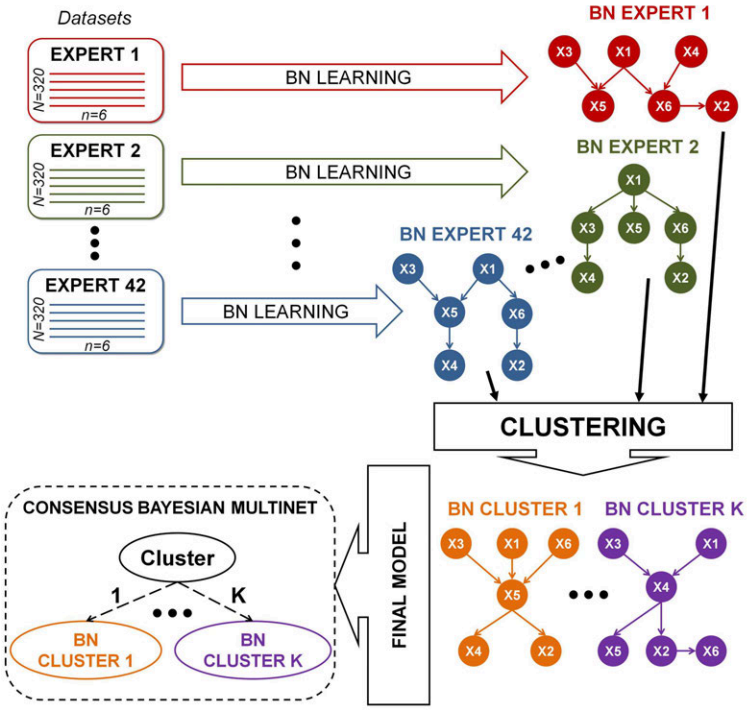


Figure 11.1: Proposed methodology for building a consensus Bayesian multinet which represents the behavior of a set of experts.

11.2 A methodology for inducing a consensus Bayesian multinet from a set of expert opinions

In this section, we detail the process for obtaining a Bayesian multinet representing the consensus among the experts who completed the classification experiment. Figure 11.1 visually represents the whole methodology, which can be summarized in three main steps:

1. Learn one BN for each expert using the classifications provided in the experiment.

2. Cluster the BNs into groups and induce a new representative BN for each cluster, which models the opinions of the experts in the cluster.

3. Combine the representative BNs of each cluster into one consensus Bayesian multinet.

The following sections describe each step in the previous methodology. In Section 11.2.1 we retrieve the BNs representing the experts behavior in the classification experiment. Section 11.2.2 explains how to discover groups of similar BNs by applying clustering algorithms and how to induce a representative BN for each group. In Section 11.2.3, the final consensus Bayesian multinet model is built from the representative BNs of each cluster.

11.2.1 Bayesian network modeling of each expert's behavior

In Section 10.6, one dataset for each expert was generated with the classifications provided in the experiment. The resulting dataset had $N = 320$ observations (the number of interneurons in the experiment) and $n = 6$ categorical variables, which corresponded to the features that the experts were asked to classify. Then, a BN was learned for each one of the $N_e = 42$ experts who completed the experiment. The goal was to build a model which captures how each expert understands the values of the morphological attributes and their relationships. Having an individual BN for each expert makes it easier to analyze and validate the representative BNs for each cluster and the final consensus Bayesian multinet, because the inputs (BNs) and the output (Bayesian multinet) share the same representation.

11.2.2 Clustering of Bayesian networks

The experiment was designed to find groups of BNs corresponding to experts with similar behaviors. In this section, we detail the process of finding groups of BNs which define similar JPDs and inducing a representative BN for each cluster. To the best of our knowledge, the problem of clustering BNs had not been studied before. Note that this is not the same problem as using BNs to cluster data [398, 415] or clustering variables in BN learning for high-dimensional problems [81, 286]. A BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ has two main components (see Section 3.3): the graphical part and the probabilistic part. Therefore, we could consider clustering at the two levels:

Clustering of BN structures: The graphical component $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ of a BN is a DAG which encodes the conditional (in)dependence relationships between the variables in the problem domain. Therefore, we could use existing approaches for clustering graphs [206, 430] and, in particular, clustering DAGs [217] to find groups of structurally similar BNs. Another approach could be to list the conditional independence relationships encoded in a BN and then apply a clustering algorithm to group BNs which share the same set of conditional independences.

Clustering of BN probabilities: The probabilistic component in a BN contains the parameters $\boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}$ of the conditional probability distributions of each variable X_i given its parents $\mathbf{Pa}(X_i)$, $i = 1, \ldots, n$. Clustering of probability distributions has not received much attention in the statistics and machine learning fields. The approaches in [220, 496] cannot be directly applied to our problem because $\mathcal{B}$ includes several (conditional) probability distributions: one probability distribution for each variable given its parents' values. Comparing the conditional probability distributions $\rho_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}(x_i); \boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}\right)$ of the same variable in two different BNs is challenging because each variable can have a different number of parents, and the set of parents may be different. Therefore, the conditional probability distributions cannot be directly compared. A simple approach, which could also be useful in problem domains with a lot of variables, would be to compute the marginal probability distribution $\rho_{X_i}\left(x_i; \boldsymbol{\theta}_{X_i}\right)$

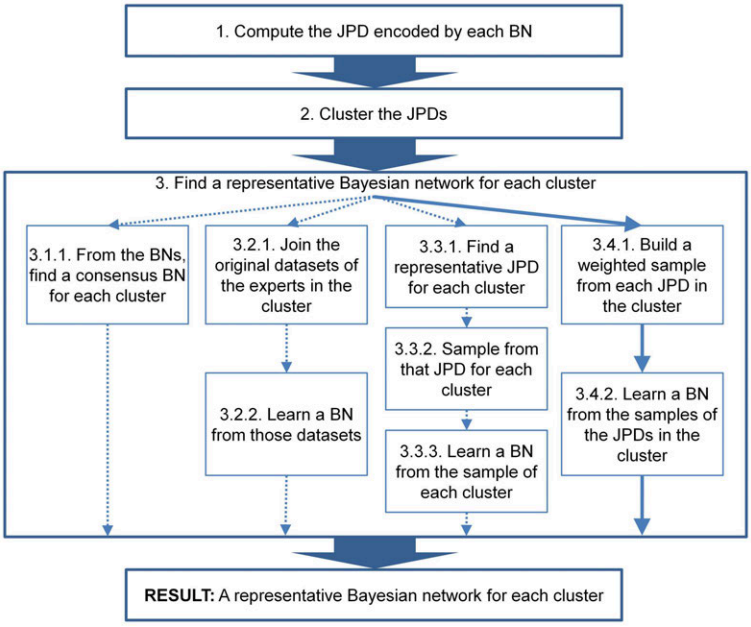


Figure 11.2: Procedure for clustering BNs. In step 3, the solid line represents the proposed workflow for inducing a representative BN for each cluster, whereas the dashed lines show alternative ways of achieving this goal.

for each variable X_i in each BN and to cluster the BNs based on these marginal distributions.

Here, we propose clustering the BNs based on the JPDs that they encode. Therefore, our approach is included in the second group of techniques. Figure 11.2 outlines the proposed methodology, which can be summarized in three steps. First, the JPD encoded by each BN is computed. These JPDs also model the experts' behavior in the experiment. Second, groups of similar experts/BNs are found by clustering their corresponding JPDs. Third, a representative BN is induced for each cluster, which represents the common behavior of the experts in the cluster. The following sections detail each one of these three steps.

11.2.2.1 Computation and preprocessing of the joint probability distributions

For each expert, we computed the JPD over the six variables encoded by the BN learned in the previous step. Not all the experts selected all the possible values when completing the experiment, e.g., some experts did not classify any neuron as `Arcade`, `Cajal-Retzius` or `Other` in variable X_5. Therefore, not all the BNs contained all the values for all the variables. However, we wanted all the JPDs to have the same number of values for the purposes of comparison. Therefore, we completed the conditional probability tables in the BNs learned with GeNIe using ML estimates with Laplace correction, so that all the BNs had all the values for all the variables. Then, the JPD over all the variables encoded by each BN was computed by multiplying the conditional probability distributions $\rho_{X_i|\mathbf{Pa}_{X_i}}\left(x_i|\mathbf{pa}(x_i);\boldsymbol{\theta}_{X_i|\mathbf{Pa}(X_i)}\right)$, as in Equation (3.1). The resulting JPD had $2 \times 3 \times 3 \times 3 \times 4 \times 11 = 2{,}376$ values. However,

most of these values corresponded to inadmissible combinations of the values of the variables (see Section 10.6). For example, when `Uncharacterized` was selected, all the other variables should have the value `Dummy`, and any other combination of values was not valid. Similarly, variable X_4 could only take a value different from `Dummy` when $X_1 =$ `Translaminar` and $X_3 =$ `Displaced`. We erased the values in the JPDs corresponding to these forbidden combinations. The resulting JPDs had 121 values each.

11.2.2.2 Clustering of joint probability distributions

We approached the problem of finding groups of similar BNs by clustering the JPDs obtained in the previous step (Section 11.2.2.1). We generated a dataset with $N_e = 42$ observations and $r = 121$ variables, where each observation (row) was a JPD corresponding to the BN of each expert and each variable (column) was a value of the JPD. There are three main paradigms which can be used for clustering (see Section 2.4.1): hierarchical, partitional and probabilistic clustering.

Hierarchical and partitional paradigms are the classical approaches to clustering. In both hierarchical and partitional approaches, the number of clusters to be generated is a free parameter that has to be set by the expert. Also, an appropriate distance measure has to be chosen depending on the nature of the data. Probabilistic clustering deals with the problem of fitting an FMM [367]. Probabilistic clustering generates an explicit probabilistic model which describes the data. Also, an appropriate number of clusters can be found using statistically sound techniques. Since each of our observations is a JPD, the Dirichlet distribution [62] could be a suitable choice of a probability density function for each component. However, the low number of observations ($N_e = 42$) over the number of variables ($r = 121$) ruled out the use of this approach, because it is difficult to obtain accurate estimators of an FMM with so few data.

Here, we chose to adapt the classical K-means algorithm [347] to characterize properties of our data. Algorithm 10.1 (see page 182) shows a general outline of the algorithm. The algorithm alternates two steps. First, the observations are assigned to the cluster with the closest center. Second, the cluster centers are recomputed taking into account only the observations in the clusters. A similar approach was used in [223, 224] in the context of decision making in influence diagrams. Also, Etminani et al. [164] aggregate probability distributions by clustering them first and averaging them afterwards. In order to apply the K-means algorithm to the problem of clustering JPDs we have to choose a suitable dissimilarity measure $d(\mathbf{x}_j, \mathbf{c}_k)$ and a method for computing the cluster centers from the observations in the cluster (`Combine` function in step 2.(c) of Algorithm 10.1).

Dissimilarity measures for probability distributions In general, our choice of a dissimilarity measure $d(\mathbf{x}_j, \mathbf{c}_k)$ should be, at least, symmetric. Therefore, one could consider using the symmetric KL measure,

$$d_{KL}(\mathbf{x}_1, \mathbf{x}_2) = KL(\mathbf{x}_1, \mathbf{x}_2) + KL(\mathbf{x}_2, \mathbf{x}_1),$$

where $KL(\mathbf{x}_1, \mathbf{x}_2)$ is the KL [305] from an empirical probability distribution $\mathbf{x}_1$ to the true distribution $\mathbf{x}_2$

$$KL(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{r} x_{1i} \log \frac{x_{1i}}{x_{2i}},$$

where r is the number of values of the probability distribution $\mathbf{x}_j$, and x_{ji} is the probability of the ith value in the probability distribution $\mathbf{x}_j$. One disadvantage of the KL is that it is not upper bounded. However, other measures can be considered, such as the Jensen-Shanon divergence,

$$d_{JS}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2} KL(\mathbf{x}_1, \mathbf{m}) + \frac{1}{2} KL(\mathbf{x}_2, \mathbf{m}), \tag{11.1}$$

where $\mathbf{m}$ is the mean probability distribution $\mathbf{m} = 0.5(\mathbf{x}_1 + \mathbf{x}_2)$. The Jensen-Shannon divergence has a number of interesting properties [333]: it is symmetric, its square root is a metric and it is bounded $0 \leq d_{JS} \leq 1$. Therefore, we chose d_{JS} as the dissimilarity measure for the K-means algorithm. Additionally, the fact that d_{JS} is a bounded measure was also useful when computing the representative BN for each cluster (Section 11.2.2.3).

Combination of probability distributions Two main methods can be found in the literature to compute an average probability distribution $\overline{\mathbf{x}}$ from a set of probability distributions [214]: the linear combination pool (LinOp) and the logarithmic combination pool (LogOp). If we have N_k probability distributions $\{\mathbf{x}_1, \ldots, \mathbf{x}_{N_k}\}$ in a cluster, the linear combination pool is defined as the weighted arithmetic mean

$$\overline{\mathbf{x}}_{LinOp} = \sum_{j=1}^{N_k} \omega_j \mathbf{x}_j, \tag{11.2}$$

where $\sum_{j=1}^{N_k} \omega_j = 1$ and $\omega_j > 0$ is the weight for the probability distribution $\mathbf{x}_j$. The logarithmic combination pool is defined as the weighted geometric mean

$$\overline{x}_{iLogOp} = \frac{\prod_{j=1}^{N_k} x_{ji}^{\omega_j}}{\sum_{v=1}^{r} \prod_{j=1}^{N_k} x_{jv}^{\omega_j}}. \tag{11.3}$$

Genest and Zideck [214] give a number of reasons for choosing LogOp over LinOp, the most compelling being that it is externally Bayesian, i.e., it can be derived from joint probabilities [49]. Also, it is known that LinOp does not preserve independences [530], i.e., combining probability distributions which share a common independence does not guarantee that the resulting distribution will be equally independent. Heskes [254] showed that using LogOp is equivalent to finding the probability distribution $\mathbf{x}$ which minimizes the weighted sum of the KL values to each probability distribution $\mathbf{x}_j$

$$\overline{\mathbf{x}}_{LogOp} = \arg\min_{\mathbf{x}} \sum_{j=1}^{N_k} \omega_j KL(\mathbf{x}, \mathbf{x}_j).$$

Therefore, we chose LogOp as a combination method for computing the cluster centers in the K-means algorithm (step 2.(c) of Algorithm 10.1). All the experts were considered as equals, so the weights ω_j were all set to $1/N_k$ for each cluster.

11.2.2.3 Finding a representative Bayesian network for each cluster

Once the JPDs have been clustered and K cluster centers (JPDs) have been obtained, the next step is to induce a BN which represents the common features of the corresponding BNs (and experts) in the cluster. Step 3 in Figure 11.2 shows four possible approaches for finding a representative BN for each cluster. In the following, we discuss the four approaches for performing this task, we review the works related to each one and analyze their advantages and disadvantages for modeling experts' opinions on the problem of the morphological classification of GABAergic interneurons.

The first approach consists of directly combining a set of BNs into a single representative one (Figure 11.2, step 3.1.1). Learning BNs from a set of expert opinions has been a recurrent interest in the field. However, Pennock and Wellman [406] showed that even when the BNs share the same structure, there is no way of combining the parameters to preserve that structure. They proposed a methodology for combining both the BN structures and the parameters. The algorithm finds a common network structure by transforming the DAGs into moral graphs, performing the union of the edges and transforming the resulting moral graph back into a DAG. The CPTs are combined by applying the LogOp combination pool of Equation (11.3). This approach is expected to yield highly connected BNs because of the union of the edges of the moral graphs. Therefore, the conditional probability distributions will have a lot of parameters and their estimates will not very robust when there are few training instances (in our scenario, 320 neurons). Sagrado and Moral [437] studied the theoretical properties of BNs obtained by performing either the intersection or the union of the arcs of the network structures, and proposed ways for finding the consensus BN structure. However, they left the combination of the CPTs as a matter for future research. Zhang et al. [537] built on the work by Sagrado and Moral [437] and proposed a score+search method for fusing the BN structures. However, they applied Bayesian inference not data to combine the parameters of the BNs and to compute the scores of the network structures. Peña [396] derived a correction of the algorithms proposed by Matzkevich and Abramson [363, 364] for finding the consensus BN structure with a minimum number of parameters. It represents only the common independences appearing in all the BN structures. He outlined some ideas for combining the parameters of the BNs, but this issue was mainly left for future research. Later, Etminani et al. [164] clustered the experts and aggregated the probability distributions of the cluster with the highest number of observations, following a "democratic majority" approach. On the other hand, we did not discard the probability distributions in small clusters ("minorities"). Instead, all the differing opinions were included in the consensus mixture of BNs. Finally, other methods for BN aggregation have been proposed in the context of model averaging (for a review, see Section 4.13 in [108]). These methods combine the probabilities inferred with a set of BNs but they do not obtain a single representative BN which models the

opinions of a set of experts. In the neuron classification problem, obtaining the representative BN explicitly was important because the experts would like to analyze and interpret these models and not only their outputs.

The second approach deals with the problem of learning a consensus BN from data. Maynard-Reich II and Chajewska [365] assumed that the differences between experts are the result of observing different subsets of data. This is related to the problem of learning BNs from distributed datasets, e.g., see [83]. In our experiment, however, all the experts classified the same 320 interneurons, so this assumption did not apply. Steps 3.2.1 and 3.2.2 show another possibility which conformed to our problem: joining the original datasets for each expert in the cluster and learning a BN from this cluster's dataset. We could consider different degrees of membership of each expert to his cluster by only including a subset of interneurons from his dataset in the cluster's dataset. However, there were some neuronal morphologies which did not appear frequently in the data. Therefore, this approach could erase some important information about the experts.

The third approach is based on sampling the JPDs and learning a BN from the generated data as explained in Section 11.2.1 (Figure 11.2, steps 3.3.1 to 3.3.3). First, we compute a representative JPD for each cluster, then we sample the JPD to obtain a dataset and, finally, we learn a BN from that dataset. Again, one could consider using the LinOp (Equation (11.2)) or the LogOp (Equation (11.3)) combination pools for computing the representative JPD, and different weights could be applied to each expert's JPD. However, if the cluster center JPD does not accurately represent all the experts in the cluster, the resulting representative BN for the cluster would not model all the experts' opinions either.

Here, we implemented another approach based on proportional sampling of the individual JPDs of each expert (Figure 11.2, steps 3.4.1 and 3.4.2). The goal was to obtain a sample of data for each cluster k, taking into account the dissimilarity between each JPD and the cluster center $\mathbf{c}_k$ to decide the number of samples to draw from each JPD. The fact that $d_{JS}(\mathbf{x}_j, \mathbf{c}_k)$ (Equation (11.1)) is upper bounded facilitates the computation of these expert degrees of membership. For a given cluster k, we found the JPDs included in the cluster and computed a degree of membership μ_j for each one as

$$\mu_j = \frac{1 - d_{JS}(\mathbf{x}_j, \mathbf{c}_k)}{\sum_{j=1}^{N_k} (1 - d_{JS}(\mathbf{x}_j, \mathbf{c}_k))}.$$

Then, to obtain a sample with size M for cluster k, $\mu_j \times M$ observations were drawn from each JPD $\mathbf{x}_j$ in cluster k. Finally, both the structure and the parameters of the representative BN were learned (see Sections 10.6 and 11.2.1) from that sample of size M obtained for each cluster.

This approach tries to avoid some of the disadvantages of the other three approaches. The learning algorithm allows to fully specify the BNs as opposed to the methods in the first approach (step 3.1.1), which can have problems when computing the parameters of the conditional probability distributions. An advantage of this method over the second approach (steps 3.2.1 and 3.2.2) is that our approach uses the BNs themselves (through their JPDs)

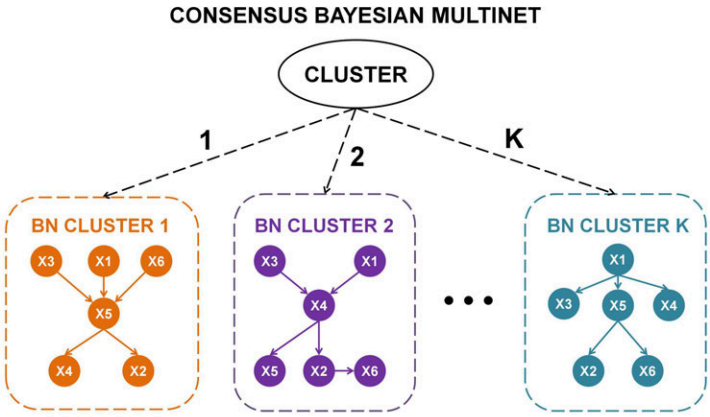


Figure 11.3: Finite mixture of BNs represented as a Bayesian multinet with the cluster variable as the distinguished variable.

to compute the representative BN for the cluster. The second approach, on the other hand, assumes that the BNs were learned from data and that experts' data is still available. This may not be the case in some scenarios where BNs are elicited from experts' knowledge and not induced from data. Finally, as opposed to the third approach, we consider each BN in the cluster individually through its JPD while taking into account different degrees of membership to the cluster.

11.2.3 Building the consensus Bayesian network

The final step in the methodology (see Figure 11.1) deals with the problem of building a PGM that represents all the experts who participated in the experiment and also takes into account their differing behaviors. We modeled the whole problem as a finite mixture of BNs [487]

$$p_{\mathbf{X}}(\mathbf{x}) = \sum_{k=1}^{K} p_C(k;\boldsymbol{\theta}_C) p_{\mathbf{X}|C}(\mathbf{x}|k,\mathcal{G}_k,\boldsymbol{\theta}_{\mathbf{X}|k}), \tag{11.4}$$

where $p_C(k;\boldsymbol{\theta}_C)$ was set to the proportion of experts in the kth cluster (N_k/N_e), and each component $p_{\mathbf{X}|\mathbf{C}}(\mathbf{x}|k,\mathcal{G}_k,\boldsymbol{\theta}_{\mathbf{X}|k})$ was the representative BN for the kth cluster with structural component $\mathcal{G}_k$ and probabilistic component $\boldsymbol{\theta}_{\mathbf{X}|k}$. Finite mixtures of BNs form a kind of Bayesian multinet [213] with a distinguished variable C which represents the cluster variable. In principle, the cluster variable C is hidden but we found it previously by clustering the BNs (Section 11.2.2). Figure 11.3 is a diagram of the final consensus Bayesian multinet.

11.3 An application to interneuron classification

This section includes the results corresponding to one run of the whole process as described in Section 11.2 (see Figure 11.1). First, one BN was learned for each one of the 42 experts who completed the experiment (Section 11.2.1). Then we clustered the BNs following the

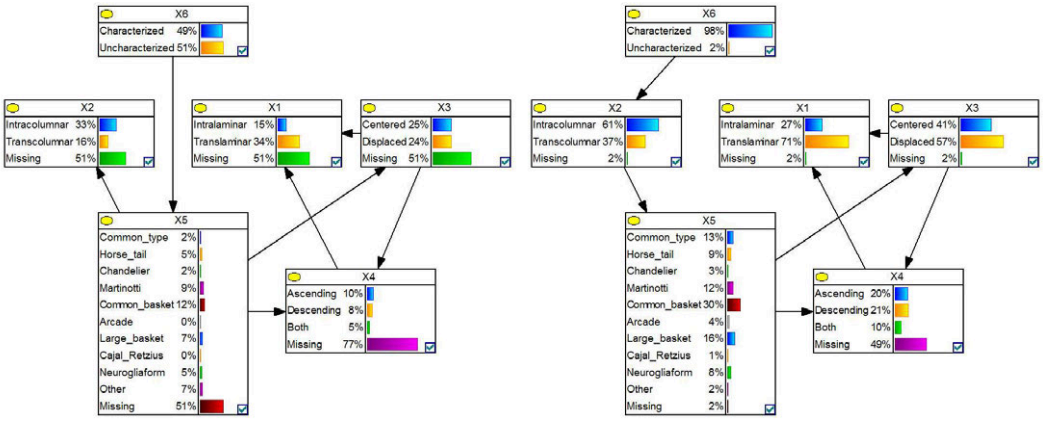
procedure described in Section 11.2.2. We started the process by computing the JPD encoded by each BN and generating a data matrix with dimensions 42×121, where each row was a JPD corresponding to an expert and each column corresponded to a value of the JPD, i.e., a combination of possible values of the variables in the experiment. We used the K-means algorithm with Jensen-Shanon distance (Equation (11.1)) and the LogOp combination pool (Equation (11.3)) to cluster the JPDs. We used $K = 6$ clusters because we were thus able to find distinguishable clusters with characterizing properties. We used proportional sampling to get a dataset for each cluster, and a representative BN was learned from that sample using GeNIe. Finally, a consensus PGM was built as a finite mixture of BNs represented with a Bayesian multinet (Section 11.2.3). In the consensus Bayesian multinet, the cluster variable was the distinguished variable and each component of the mixture was the representative BN for a cluster (see Figure 11.3).

In the following sections, we analyze the results by studying the consensus Bayesian multinet at different levels. Figure 11.4 shows the representative BNs learned for each cluster of experts. These BNs can be downloaded in GeNIe format from the supplementary material website[1]. First, the BNs for each expert learned with the GTT algorithm were compared with other algorithms for learning BN structures from data (Section 11.3.1). Then, we tried to characterize each one of the clusters by studying the marginal probabilities of their representative BNs (Section 11.3.2). Also, a structural analysis of the BNs was performed to validate the results and to find agreements and differences between clusters (Section 11.3.3). We extracted agreed definitions of the different neuronal types proposed in the experiment by performing inferences in both the consensus Bayesian multinet and the representative BNs for each cluster (Section 11.3.4). A principal component analysis was performed to visually inspect a low-dimensional representation of the clusters (Section 11.3.5). Finally, we looked for possible currents of opinion by studying correlations between the clusters and the geographical location of the experts' workplace (Section 11.3.6).

11.3.1 Validation of the Bayesian network structure learning algorithm

We studied the influence of the structure learning algorithm when finding the BNs for each expert (see Section 11.2.1). We compared the BNs learned with the GTT algorithm (Algorithm 3.2) with other four algorithms for learning BN structures available in the `bnlearn` package [446] for R statistical software [422]: a hill-climbing algorithm (HC), a tabu search algorithm (TA), a max-min algorithm (MM) and the 2-phase restricted search max-min algorithm (RS). HC and TA are score + search algorithms, whereas MM and RS are hybrid algorithms combining score + search with constraint-based approaches. 100 restarts were computed for the hill-climbing algorithm and the best scoring network structure was returned. Additionally, we considered two scoring functions: K2 [97] and BIC [443] (see Section 3.3.2.1). Thus, for each expert, we learned eight BN structures using the four algorithms and the two scoring functions. ML estimates of the parameters with Laplace correction were

[1] Available at http://cig.fi.upm.es/index.php/members/138-supplementary-material/



(a) BN for cluster 1 (3 experts)

(b) BN for cluster 2 (15 experts)

(c) BN for cluster 3 (4 experts)

(d) BN for cluster 4 (12 experts)

(e) BN for cluster 5 (7 experts)

(f) BN for cluster 6 (1 expert)

Figure 11.4: Network structures and marginal probabilities of the representative BNs for each cluster. Each one of the BNs corresponds to a component in the finite mixture of BNs that builds up the consensus Bayesian multinet.

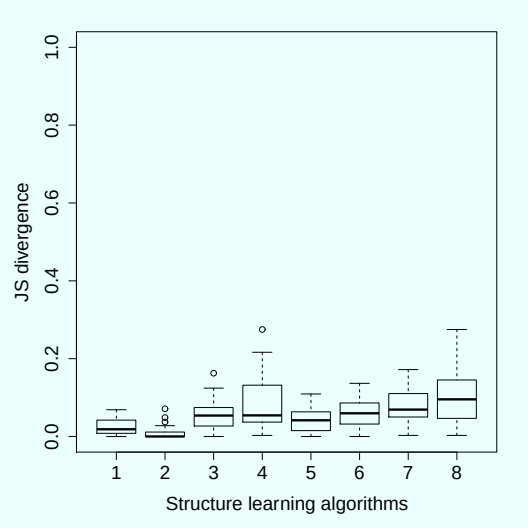


Figure 11.5: Comparison between the GTT algorithm and eight algorithms for learning the BN structures: 1) HC-K2, 2) TA-K2, 3) MM-K2, 4) RS-K2, 5) HC-BIC, 6) TA-BIC, 7) MM-BIC and 8) RS-BIC. Each boxplot summarizes the 41 Jensen-Shanon divergence values (42 experts minus expert #33) between the JPDs of the BNs obtained with the GTT algorithm and the JPDs obtained with each one of the eight alternative methods.

computed for filling in the CPTs. The JPD encoded by each BN was computed and simplified to a 121-dimensional JPD as explained in Section 11.2.2.1. The Jensen-Shanon divergence (Equation (11.1)) between the JPD corresponding to the BN learned with the GTT algorithm and the eight alternative structure learning methods was computed. The structure learning algorithms could not be applied for expert #33 because he/she classified all the neurons as $X_1 =$ `Intralaminar` and the algorithms could not handle variables with only one value.

Figure 11.5 shows boxplots of the Jensen-Shanon divergence values (Y axis) between the GTT algorithm and the other eight algorithms (X axis) obtained for the 41 experts (42 minus expert #33). Note that the JS divergence is both lower and upper bounded: $0 \leq d_{JS} \leq 1$. We can see that the JS divergence yielded very low values, being almost all of them below 0.2. On the one hand, the TA-K2 algorithm (second boxplot in Figure 11.5) yielded the lowest JS divergence values. On the other hand, the RS algorithm (fourth and eighth boxplots in Figure 11.5) learned BNs which yielded JPDs differing the most compared to those obtained with the GTT algorithm. As expected, we can see that algorithms using K2 scoring function yielded lower JS divergences than those using BIC, because the GTT algorithm also used the K2 scoring function. We concluded that the algorithm used for learning the BN structures did not have an important influence in the proposed methodology because we used the JPDs for clustering the BNs and they were similar regardless of the applied algorithm.

11.3.2 Cluster labeling and analysis of the probability distributions

We identified differences between the groups of experts by studying the marginal (or prior) probabilities in the representative BNs for each cluster (see Figure 11.4). We used these

marginal probabilities to characterize each group of experts and we interpreted these differences as different approaches when classifying the neurons:

Cluster 1 (including three experts) represented experts who considered that half of the neurons in the experiment did not have enough reconstructed axonal processes for it to be feasible to actually try to classify them. Thus, they assigned the neurons to the `Uncharacterized` category in X_6 (probability 0.51). The probability of `Uncharacterized` was much lower in all the other BNs ($\leq$ 0.07). In fact, the combination of values of the variables with higher probability (mode) corresponded to X_1 = `Dummy`, X_2 = `Dummy`, X_3 = `Dummy`, X_4 = `Dummy`, X_5 = `Dummy` and X_6 = `Uncharacterized`.

Cluster 2 included 15 experts with a coarse classification scheme. In this BN, most of the neurons were classified as `Common basket` (0.30). The mode of the JPD encoded in the representative BN was X_1 = `Intralaminar`, X_2 = `Intracolumnar`, X_3 = `Centered`, X_4 = `Dummy`, X_5 = `Common basket` and X_6 = `Characterized`.

Cluster 3 (including four experts) represented experts who stuck to the fine-grained classification scheme proposed in the experiment and tried to distinguish between the different neuronal types, including the difficult ones such as `Common basket`, `Common type`, `Large basket` and `Arcade` cells. Experts in this cluster found more `Arcade` cells (0.07) than the experts in the other clusters. In this cluster, `Common type` (0.17), `Common basket` (0.14) and `Large basket` (0.20) cells had similar probabilities. The mode of the JPD encoded by the representative BN was the same as in cluster 2.

Similarly to cluster 2, experts in cluster 4 (including 12 experts) showed a less detailed classification scheme than those in clusters 3 or 5. However, experts in cluster 4 assigned a high probability to the `Common type` class (0.40), whereas the most likely neuronal type in cluster 2 was `Common basket`. Accordingly, the mode of the JPD of the representative BN was X_1 = `Translaminar`, X_2 = `Intracolumnar`, X_3 = `Centered`, X_4 = `Dummy`, X_5 = `Common type` and X_6 = `Characterized`.

Cluster 5 represented a group of seven experts with a detailed classification scheme, since they distinguished between `Common type`, `Common basket` and `Large basket` cells. However, the experts did not seem to agree with the nomenclature included in the experiment or found it incomplete. This was observed in the high probability of the category `Other` (0.22) in X_5, where they could propose an alternative name for that class of neurons. Interestingly, the mode of the JPD of the representative BN for this cluster was X_1 = `Dummy`, X_2 = `Dummy`, X_3 = `Dummy`, X_4 = `Dummy`, X_5 = `Dummy` and X_6 = `Uncharacterized`. In fact, we can see that this cluster assigned the second highest probability to `Uncharacterized` in all the clusters.

Cluster 6 included only one expert with a remarkably different behavior than the other experts. This expert did not classify any neuron as `Translaminar` in X_1, so the probability of that value in the representative BN is almost 0. Also, this expert assigned a

very high probability to `Centered` in X_3 (0.96). Therefore, X_4 was disabled for all the neurons (recall that X_4 was only available when `Translaminar` and `Displaced` were set as values in X_1 and X_3, respectively). Therefore, X_4 had a constant `Dummy` value in Figure 11.4(f). The conclusions of the analysis of the mode of the JPD were the same, as the combination of values of the variables with highest probability was X_1 = `Intralaminar`, X_2 = `Transcolumnar`, X_3 = `Centered`, X_4 = `Dummy`, X_5 = `Large basket` and X_6 = `Characterized`.

11.3.3 Analysis of the Bayesian network structures

Similarities in the behaviors of all the group of experts were identified by analyzing the representative BN structures. Variables X_2 and X_5 were the only two variables which were directly related in all the BNs. Variable X_2 describes the neuronal morphology in the horizontal dimension. This feature encodes whether or not the axonal arborization of the neuron extends more than 300 μm from the soma. This means that the interneuron contacts with neurons inside and outside its cortical column, so we could conclude that some neuronal types mainly connect with other neurons from the same cortical column, whereas other neuronal types connect additionally with neurons from different cortical columns.

Variables X_1, X_3 and X_4 were related in all but one BN, the one corresponding to cluster 6. Also, there was an edge between X_4 and X_5 in all the BNs but the one for cluster 6. Note that cluster 6 contained only the outlying expert 33. Variables X_1, X_3 and X_4 are mainly related to the neuronal morphology in the vertical dimension. These relationships could determine whether a given neuronal type sends the information to other neurons in the same cortical layer or in different (either upper or lower) layers. We also analyzed the Markov properties of the representative BN structures to identify conditional independence relationships between the variables. X_2 was conditionally independent of variables (X_1, X_3, X_4) given the value of X_5 and X_6. Therefore, the morphological properties of GABAergic interneurons in the horizontal and vertical dimensions seemed to be independent given the neuronal type.

11.3.4 Finding agreed definitions for neuronal types using inference in Bayesian networks

The representative BNs were used to infer the main properties of the different neuronal types in X_5 by setting evidence in some variables and updating the probabilities in the unobserved variables. We studied the propagated probabilities and identified differences and similarities between clusters. Cluster 6 corresponded to an outlier expert which has already been analyzed, so we focused on the other five clusters. First, the main morphological properties of the neuronal types were found by setting every value in X_5 as evidence and propagating the probabilities using the clustering algorithm [277, 323] in GeNIe:

`Martinotti` cells were defined as `Translaminar` ($\geq$ 0.94), `Displaced` ($\geq$ 0.83) and `Ascending` ($\geq$ 0.57) cells. Experts in cluster 5 classified these neurons as mostly

`Transcolumnar` (0.73), whereas they were classified in clusters 1, 2, 3 and 4 as either `Intracolumnar` or `Transcolumnar` with similar probabilities.

`Horse-tail` cells seem to have a common and easily recognizable morphology, since the most likely values achieved high probabilities in all the clusters: `Translaminar` (≥ 0.92), `Intracolumnar` (≥ 0.80), `Displaced` (≥ 0.88) and `Descending` (≥ 0.50).

`Chandelier` cells seemed to be mainly `Intracolumnar` (≥ 0.72). However, they were classified as either `Intralaminar` or `Translaminar` and `Centered` or `Displaced` in different clusters. Clusters 2 and 4 assigned a higher probability to `Translaminar`, cluster 3 assigned a higher probability to `Intralaminar` and the probabilities were almost uniform in the X_1 variable in clusters 1 and 5. `Centered` received a higher probability in cluster 3, whereas the probabilities were more uniform in the other clusters.

`Neurogliaform` cells were defined as mainly `Intracolumnar` (≥ 0.83). Experts in clusters 3, 4 and 5 classified them as `Intralaminar` (≥ 0.76), whereas experts in clusters 1 and 2 assigned more uniform probabilities in variable X_1. For experts in cluster 5, `Neurogliaform` cells could be either `Centered` or `Displaced`, whereas `Centered` was more likely in all the other clusters (≥ 0.75).

`Common type` cells were characterized as `Translaminar` (≥ 0.62) cells. Experts in clusters 4 and 5 classified them as either `Intracolumnar` or `Transcolumnar`, whereas experts in clusters 1, 2 and 3 selected `Intracolumnar` as the most likely value (≥ 0.66).

The properties for `Common basket` cells could not be easily identified. Experts in cluster 2 and 4 classified most of them as `Translaminar` (≥ 0.63), cluster 3 assigned the highest probability to `Intralaminar` (0.82), whereas in the other clusters they were classified as either `Translaminar` or `Intralaminar`. `Intracolumnar` was always more likely than `Transcolumnar`, although the differences in the probability values greatly varied in the clusters. We also found major disagreements in X_4: Clusters 1 and 3 assigned `Centered` with a high probability (≥ 0.86), whereas the probabilities of `Centered` and `Displaced` were similar in the other clusters.

`Large basket` cells were characterized as `Translaminar` (≥ 0.58) and `Transcolumnar` (≥ 0.63) cells. Clusters 1 and 3 defined them as mainly `Centered` (≥ 0.74), cluster 5 assigned a higher probability to `Displaced` (0.6), whereas in the other clusters `Centered` and `Displaced` had more uniform probabilities.

`Arcade` cells were frequently classified as `Translaminar` (≥ 0.65), `Intracolumnar` (≥ 0.55) and, when `Translaminar` and `Displaced` were selected, as `Descending` cells.

Most of the neurons classified as `Other` were characterized as `Translaminar` (≥ 0.62). `Intracolumnar` was more likely than `Transcolumnar` in all the clusters. `Displaced` had a higher probability than `Centered` in all the clusters, except for cluster 6. However, the differences between the probabilities of these values greatly varied from cluster to

cluster. Cluster 3 yielded a high probability for `Both` category in X_4 (0.50), whereas cluster 1 assigned a greater probability to `Descending` (0.38). The probabilities in X_4 were more uniform in the other clusters.

Setting evidence in the other variables also highlighted some differences between groups of experts. For example, setting `Intralaminar` as evidence in X_1 yielded `Common basket` as the most likely value for X_5 in all the BNs, except for the one corresponding to cluster 4, where `Common type` and `Neurogliaform` got higher probabilities. Setting `Translaminar` as evidence in X_1 yielded very different propagated probabilities in the clusters. When setting `Intracolumnar` as evidence in X_2, the most likely values in X_5 were `Common basket` (clusters 1 and 2), `Common type` (clusters 3 and 4) and `Other` (cluster 5).

The consensus Bayesian multinet was used to perform inferences taking into account all the representative BNs at the same time. The probability of a given query was computed using the finite mixture of BNs expression (Equation (11.4)). Table 11.1 shows the conditional probabilities of each variable given the neuronal type in X_6. We used these conditional probabilities to infer a set of agreed definitions for some neuronal types:

`Martinotti` cells were usually classified as `Translaminar`, `Displaced` and `Ascending`.

`Horse-tail` cells were commonly defined as `Translaminar`, `Intracolumnar`, `Displaced` and `Descending` neurons.

A common feature of `Chandelier` neurons was that they were `Intracolumnar`.

`Neurogliaform` cells were mainly `Intralaminar`, `Intracolumnar` and `Centered` cells.

`Common type` cells were primarily `Translaminar`.

`Large basket` neurons were characterized as `Translaminar` and `Transcolumnar`.

`Arcade` neurons were usually classified as `Translaminar`.

Neurons classified as `Other` were commonly classified as `Translaminar` and `Intracolumnar` cells.

11.3.5 Clustering visualization with principal component analysis

The clusters obtained with K-means were visually inspected using a representation in a lower-dimensional space. The goal was to obtain a three-dimensional representation that approximates the 121-dimensional JPDs and check whether or not the clusters were visually distinguishable. A principal component analysis was performed, and the three principal components (PCs) which account for the highest proportion of variance (67.14%) were studied [403]. Figure 11.6 plots the values of the JPDs for each expert in the transformed three-dimensional space. Different symbols and colors were used to show the cluster assigned by the K-means algorithm to each expert. Two-dimensional projections were also included for

Table 11.1: Conditional probabilities of each variable given the neuronal type (X_5), computed with the consensus Bayesian multinet. The largest value for each conditional probability distribution is highlighted in boldface.

	Common type	Horse-tail	Chandelier	Martinotti	Common basket	Arcade	Large basket	Cajal-Retzius	Neurogliaform	Other	Dummy
					Conditional probabilities $p_{X_1\|X_5}(x_1\|x_5)$						
Intralaminar	0.2847	0.0720	0.4270	0.0632	0.4477	0.2863	0.2642	0.2947	**0.6806**	0.2423	0.0072
Translaminar	**0.7136**	**0.9254**	**0.5671**	**0.9350**	**0.5511**	**0.7057**	**0.7342**	**0.6817**	0.3170	**0.7491**	0.0081
Dummy	0.0017	0.0026	0.0059	0.0018	0.0012	0.0080	0.0016	0.0236	0.0024	0.0086	**0.9847**
					Conditional probabilities $p_{X_2\|X_5}(x_2\|x_5)$						
Intracolumnar	**0.6190**	**0.8639**	**0.7903**	0.4001	**0.6862**	**0.6365**	0.1874	0.4687	**0.8589**	**0.7242**	0.0030
Transcolumnar	0.3802	0.1346	0.2065	**0.5990**	0.3132	0.3579	**0.8117**	**0.5165**	0.1398	0.2716	0.0030
Dummy	0.0008	0.0015	0.0032	0.0009	0.0006	0.0056	0.0009	0.0148	0.0013	0.0042	**0.9940**
					Conditional probabilities $p_{X_3\|X_5}(x_3\|x_5)$						
Centered	0.4293	0.1088	**0.5292**	0.1151	**0.6075**	0.4078	**0.5126**	0.3833	**0.7524**	0.3410	0.0052
Displaced	**0.5696**	**0.8893**	0.4668	**0.8837**	0.3917	**0.5856**	0.4862	**0.5997**	0.2459	**0.6540**	0.0055
Dummy	0.0011	0.0019	0.0040	0.0012	0.0008	0.0066	0.0012	0.0170	0.0017	0.0050	**0.9893**
					Conditional probabilities $p_{X_4\|X_5}(x_4\|x_5)$						
Ascending	0.1623	0.1244	0.0950	**0.6479**	0.1008	0.1290	0.1630	0.1852	0.0400	0.1859	0.0036
Descending	0.2169	**0.6439**	0.1961	0.1103	0.1270	0.2606	0.1762	0.2259	0.0369	0.2106	0.0036
Both	0.1296	0.1119	0.0754	0.1187	0.0790	0.1311	0.1096	0.1553	0.0302	0.2252	0.0036
Dummy	**0.4912**	0.1198	**0.6335**	0.1231	**0.6932**	**0.4793**	**0.5512**	**0.4336**	**0.8929**	**0.3783**	**0.9892**
					Conditional probabilities $p_{X_6\|X_5}(x_6\|x_5)$						
Characterized	**0.9989**	**0.9981**	**0.9962**	**0.9988**	**0.9992**	**0.9937**	**0.9988**	**0.9835**	**0.9983**	**0.9950**	0.0115
Uncharacterized	0.0011	0.0019	0.0038	0.0012	0.0008	0.0063	0.0012	0.0165	0.0017	0.0050	**0.9885**

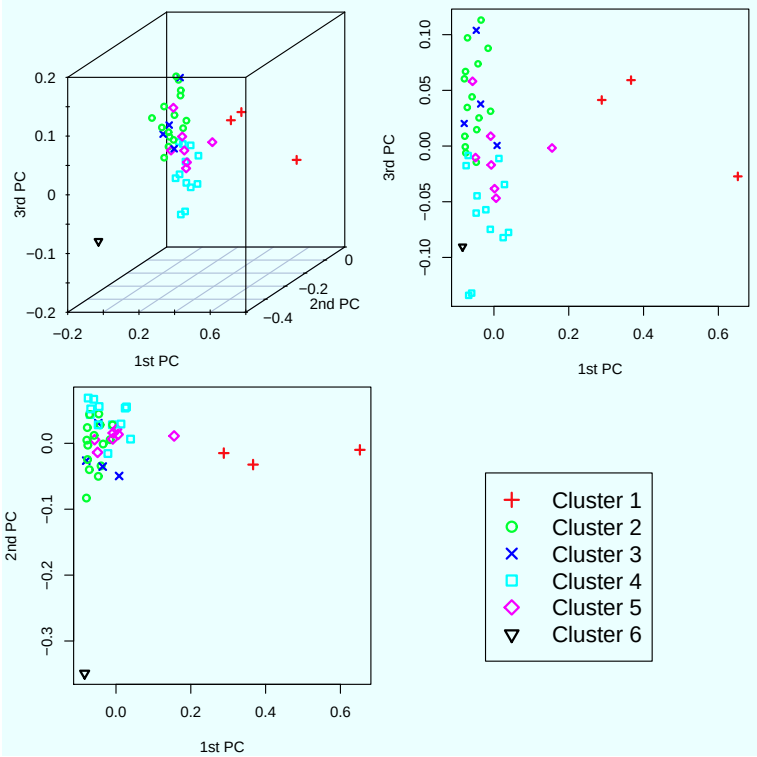


Figure 11.6: Visualization of the clusters computed with K-means ($K = 6$) in three and two-dimensional spaces obtained with principal component analysis. The three-dimensional coordinates of the experts correspond to the values of the three principal components with highest proportion of variance.

ease of interpretation. Also, we studied the weights associated with each JPD value in each one of the PCs:

The first PC, which accounted for 47.32% of the variance, distinguished the experts in cluster 1 from the other clusters. In this PC, the value of the JPD with highest (absolute) weight was $X_1 =$ `Dummy`, $X_2 =$ `Dummy`, $X_3 =$ `Dummy`, $X_4 =$ `Dummy`, $X_5 =$ `Dummy`, $X_6 =$ `Uncharacterized` (weight = 0.9828). The second weight with the largest absolute value had a value equal to -0.06119. This PC primarily separated experts with different behaviors when classifying the neurons as either `Characterized` or `Uncharacterized` in variable X_6. Therefore, the three experts in cluster 1 (Figure 11.4(a)), which classified a lot of neurons as `Uncharacterized`, were easily distinguished using this PC.

The second PC distinguished the outlying expert in cluster 6 and accounted for 10.74% of the variance. This PC yielded the largest weight (in absolute terms) for the value of the JPD corresponding to $X_1 =$ `Intralaminar`, $X_2 =$ `Transcolumnar`, $X_3 =$ `Centered`, $X_4 =$ `Dummy`, $X_5 =$ `Large basket`, $X_6 =$ `Characterized` (weight = -0.7385). Figure 11.4(f) shows that the representative BN of the outlying expert in cluster 6 had a very high probability (0.46) for `Large Basket` cells. Therefore, this PC separated the expert in cluster 6 from the rest of the clusters.

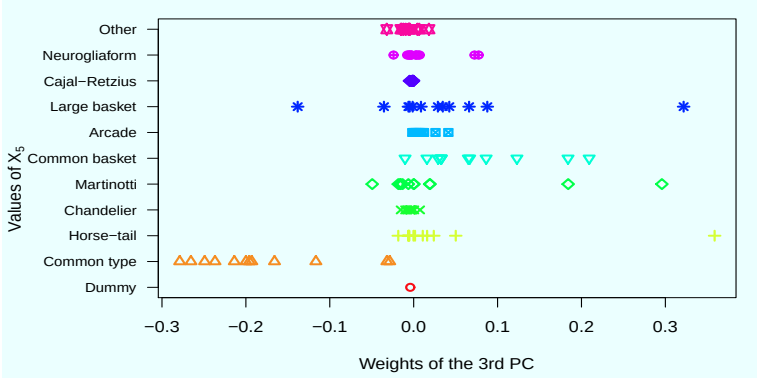


Figure 11.7: Weights of the third principal component according to the value of the variable X_5 in the JPD.

The third PC accounted for 9.08% of the variance and could not easily separate the rest of the clusters. However, this PC seemed to be able to distinguish between experts in cluster 2 and experts in cluster 4. These two clusters contained experts with two different behaviors. Figure 11.4(d) shows that the experts in cluster 4 classified most of the neurons as `Common type` (0.40), whereas the experts in cluster 2 (Figure 11.4(b)) classified most of the neurons as `Common basket` (0.30). Clusters 3 and 5 were less distinguishable because the probability was more uniformly distributed across the values in X_5 (Figures 11.4(c) and (e)). The weights in the third PC were also harder to interpret. However, cluster 4 and cluster 2 could be distinguished. All the values of the JPD with X_5 = `Common type` had weights smaller or equal than -0.02859, whereas all the values with X_5 = `Common basket` had weights greater or equal than -0.0102 (see Figure 11.7). Therefore, the set of values with X_5 = `Common type` (cluster 4) and X_5 = `Common basket` (cluster 2) were disjoint according to the third PC.

We concluded that the behavior of experts in clusters 1 and 6 was remarkably different from the behavior of the rest of the experts. The K-means algorithm was able to identify those characterizing behaviors and generated two different clusters for them. Additionally, differences between the experts in clusters 2 and 4 were also correctly identified. The differences between clusters 3 and 5 were more subtle and it was difficult to find a three-dimensional representation of the JPDs which separated these experts.

11.3.6 Geographical identification of the clusters

We studied possible correlations between the experts' workplace and the cluster they were assigned to. The goal was to try to identify different approaches or currents of opinion regarding interneuron classification in different regions, cities or laboratories in the world. We studied the statistical significance of some of the groups of experts according to the country or the city where they worked. A bootstrapping approach was used, where a sample of experts was selected without replacement and we estimated the probability of some of

them belonging to the same clusters. The sampling procedure was repeated 100,000 times for different sample sizes. We could not find any statistically significant result using a significance level of $\alpha = 0.05$. Therefore, we concluded that there is no geographical correlation between the experts and the cluster they were assigned to.

11.4 Conclusion

In this chapter, we have presented a methodology for building a consensus Bayesian multinet that represents the opinions of a set of experts. The methodology can be summarized in three steps. First, a BN was learned for each expert. Second, the BNs corresponding to experts with similar behaviors were clustered. Third, a consensus Bayesian multinet was built which models the behavior of all experts. To the best of our knowledge, the problem of clustering BNs had not been studied in the literature before. Therefore, we also addressed an interesting problem in BN research. Our proposal consisted of computing the JPD encoded by each BN and applying a partitional clustering approach to find groups of similar JPDs. The K-means algorithm with logarithmic combination pool and Jensen-Shanon divergence was used to cluster the JPDs. Then, a representative BN was induced for each cluster by proportional sampling of the JPDs in the cluster and applying a BN learning algorithm on the generated dataset. The final model was a consensus Bayesian multinet which encoded a finite mixture of BNs, where each component was the representative BN for a cluster of experts.

We applied the proposed methodology to a problem of modeling experts' opinions when classifying a set of cortical GABAergic interneurons based on the morphological features of their reconstructions (see Chapter 10). This is a difficult task because neuroscientists do not have a set of commonly agreed definitions which clearly distinguish the different neuronal types [413]. The consensus Bayesian multinet built in this work was analyzed to gain some insights into the problem of classifying GABAergic interneurons. We analyzed the representative BN structures to identify common conditional independence relationships between the groups of experts, which had a direct biological interpretation. We also managed to find some easily distinguishable clusters of experts, which behaved differently from the rest of the clusters. By studying the marginal probabilities in the representative BNs, we were able to identify different approaches to neuron classification. This highlighted the importance of clustering the experts before building the consensus model. Directly combining experts with such differing behaviors would presumably hide some of these opinions, and the final model would not represent all the experts thoughts. Additionally, we performed inference with the model to provide agreed definitions of the neuronal types.

We proposed many different approaches that could be considered for performing each one of the tasks and motivated our decisions in each step. One advantage of the proposed methodology is that it is data independent, i.e., we only used experts' data for learning the initial BNs in Section 10.6, and the clustering algorithm and the construction of the consensus Bayesian multinet are only based on these BNs. Therefore, the methodology can

still be applied when experts' data is not available, e.g., when BNs are elicited from experts' knowledge.

Future work includes the application of different techniques for clustering JPDs. In particular, model-based clustering using FMMs was discarded because of the low number of observations (42 experts) and the high dimensionality of the JPDs (121 values). We discussed the possible use of finite mixtures of Dirichlet distributions [62] as the most straightforward model for clustering probability distributions. However, the Dirichlet distribution has some constraints, e.g., its covariance matrix is strictly negative so it cannot model positive correlations between variables. Finite mixtures of generalized Dirichlet distributions [60] overcome some of these constraints. However, the generalized Dirichlet distribution has more parameters than the Dirichlet distribution, so the problem of high-dimensionality combined with few data is even more challenging. L_1-regularization approaches in finite mixture modeling [392, 523] could be used to perform feature subset selection and reduce data dimensionality. Another point for future research is the use of different techniques for finding a representative BN for each cluster. Some of the possibilities for achieving this goal were discussed in Section 11.2.2.3. Finding a consensus BN from a set of BNs which represent different experts' opinions has been a recurring interest in the field [363, 396, 406, 437, 537]. Combining these methods with the proposed clustering approach and studying the differences and similarities in the representative BNs obtained for each cluster in a real problem could give some insights into the relative merits of each technique.

Chapter 12

Learning conditional linear Gaussian classifiers from class label counts using finite mixture models

Pattern classification is one of the main problems studied in artificial intelligence [153]. In Chapter 2 we saw that a classifier can be defined as a function which uses a set of distinctive features of an object to select a label for this object from a set of class labels. Different machine learning problems were identified depending on the type of information available about the class labels of the training instances: supervised (see Section 2.3), unsupervised (see Section 2.4), semi-supervised (see Section 2.5), etc.

In this chapter, we study a different problem. The information about the class of each training instance is given as a set of (possibly repeated) class labels. This scenario is motivated by the interneuron classification problem (see Chapter 10), where the true class labels of the training instances are not known, perhaps because the class labels are not clearly defined. Therefore, selecting one class label for each instance based on only one source of information can yield wrong or biased class information. One solution is to ask a set of experts to individually provide a class label for each instance. This scenario occurs naturally, e.g., in medical practice, where several physicians could have different opinions about the diagnosis, prognosis or the most appropriate treatment for a given patient. Therefore, a set of class labels including the opinion of each expert could be generated for that instance. Partially supervised learning is a particular case of our problem, where at most one observation of each class label is allowed for each instance. However, our problem is more general because class labels can be repeated for the same instance, and such information is used as a way to measure our certainty about the class labels.

The simplest approach for solving this problem would be to transform it into a regular supervised learning problem by labeling each instance with its most voted class label. This is the standard procedure in ensembles of classifiers [306, 307]. However, our goal is to learn a single classifier which takes into account the class labels of all the experts. Learning classifiers

with uncertain class labels has attracted considerable interest [96, 131, 132, 161, 502]. These researchers induce classifiers where the uncertain class labels of each instance are encoded using belief functions [127, 455]. However, these uncertain class labels do not appear in the final classifiers, i.e., the uncertain class labels are used to fit, but are not explicitly modeled in, the classifiers. This can be a drawback in real applications where the interpretation of the resulting classifiers is important.

In this chapter, we propose modeling the number of votes received by each class label for a given instance. Therefore, the information about the class for each instance is encoded as a count vector with the number of experts who assigned that instance to each class label. We induce BNCs (see Section 3.4) by including the class count vectors as predictive variables modeled using conditional multinomial distributions. The joint distribution over the predictive variables of a BNC can be defined as an FMM. The EM algorithm [128] is used to find the ML parameters of the FMM for the classifier. The estimated parameters of the probability distributions in the FMM are plugged into the BNCs, and three classification rules are proposed for labeling a previously unseen instance. We term our proposal the counts multinomial expectation maximization (CoMEM) algorithm.

FMMs for count data have been previously studied in the literature, e.g., see [57, 58, 388, 389]. Those works focused on clustering problems where the features describing the instances were modeled as count vectors. On the other hand, our proposal focuses on a partially supervised learning problem where the predictive features describing the instances are modeled with Gaussian distributions, and the class count vectors encoding the experts' opinions for each instance are modeled with multinomial distributions. Therefore, we fit hybrid FMMs with Gaussian (continuous data) and multinomial (count data) distributions.

We compare our proposal to Côme et al.'s method [96]. They presented a framework for learning MG classifiers (see Section 3.4.2.1) where the uncertain class labels were codified as belief functions. A generalized likelihood criterion based on the belief function theory was derived, and the EM algorithm was used to find the parameters of the FMM which maximize the generalized likelihood. Here, we particularize their EM algorithm to the case where the information about the class for each instance is given as a probability distribution over the class labels. We name this algorithm the probabilistic label expectation maximization (PLEM) algorithm. Additionally, we compare the proposed approach to the classical EM algorithm for unsupervised learning (see Section 3.5.2.2). The final classifiers obtained with the PLEM and EM algorithms are regular BNCs which do not explicitly model the uncertain class labels. On the contrary, the count vectors encoding the uncertain class labels in CoMEM are included as random multinomial variables in the BNCs.

Côme et al. [96] only considered MG classifiers, which need a full covariance matrix of the predictive variables to be estimated for each class label. However, (conditional) independence relationships between the predictive variables frequently occur in real problems. Therefore, we can successfully model our problem domain using sparser models which do not need full covariance matrices to be estimated from data. It is also worthwhile to capture these independence relationships when model interpretation is important. Moreover, when a high

number of variables n and/or a small number of training instances N are available, the estimated covariance matrices in MG classifiers might not be very accurate. BNCs exploit some conditional independence relationships between the predictive variables and the class variable. Therefore, fewer parameters need to be estimated when learning BNCs. In this chapter, three kinds of CLG classifiers with different structural complexities are induced and compared (see Section 3.4.2.1): MG, NB and AODE classifiers.

Part of this research has been published in López-Cruz et al. [341].

Chapter outline

This chapter is organized as follows. Section 12.1 presents the proposed approach (CoMEM) for inducing CLG classifiers when the information about the class is encoded as a count vector for each instance and modeled with multinomial distributions. Section 12.2 introduces the PLEM algorithm as a particularization of [96] for the case where the information about the class for each instance is provided as a probability distribution over the class labels. Experimental results over 16 datasets and a thorough comparison of the proposed approach (CoMEM) to the PLEM and the classical EM algorithms are reported in Section 12.3. Finally, Section 12.4 concludes with some discussion and outlines future work.

12.1 Learning Bayesian classifiers with class count vectors provided by a group of experts

In this section, we introduce the proposed approach for learning BNCs when the true class labels of the training instances are not available, but a group of experts provide a set of class labels individually. This is a three-step process, as follows:

1. We obtain a count vector for each instance, representing the number of experts who assigned each possible class label to the instance.

2. We find the FMM defining the joint probability over the predictive variables, and use the EM algorithm to compute the ML estimates of the parameters of the probability distributions in the FMM.

3. We set the estimated ML parameters in the respective probability distributions of the BNC, and apply the classification rules to assign a class label to previously unseen instances.

The following sections detail each one of these steps. First, Section 12.1.1 shows the computation of the class count vectors. Section 12.1.2 analyzes the structure and parameters of the BNCs and their FMMs defining the JPD over the predictive variables. Section 12.1.3 details the ML estimation of the parameters of the BNCs using the EM algorithm. Section 12.1.4 explains how to use the proposed models to classify a previously unseen instance.

Table 12.1: Example of a dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ with N instances characterized by $n = 4$ predictive variables. The unknown class variable has three values $\Omega_C = \{1, 2, 3\}$. A set of $N_e = 10$ experts have provided a class label for each instance, where l_m refers to the class labels selected by the mth expert.

j	X_1	X_2	X_3	X_4	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	C
1	5.1	3.5	1.4	0.2	1	3	3	2	1	3	2	3	3	1	?
2	7	3.2	4.7	1.4	2	2	3	3	2	3	2	2	2	1	?
...		...								...					...
N	5.9	3.0	5.1	1.8	3	3	3	3	2	2	3	3	2	3	?

12.1.1 Obtaining class count vectors for each instance from a group of experts

Our problem domain is modeled with a vector of n predictive variables $\mathbf{X} = (X_1, \ldots, X_n)$ and a class variable C. The predictive variables are continuous $\Omega_{X_i} \subseteq \mathbb{R}$, $i = 1, \ldots, n$, and the class variable is discrete with values in $\Omega_C = \{1, \ldots, K\}$. We have a dataset with N instances $\mathcal{D}_{\mathbf{X},\mathbf{L}} = \{(\mathbf{x}_1, \mathbf{l}_1), \ldots, (\mathbf{x}_N, \mathbf{l}_N)\}$. The jth instance is characterized by a set of values $\mathbf{x}_j = (x_{j1}, \ldots, x_{jn})$ for the vector of n predictive variables $\mathbf{X} = (X_1, \ldots, X_n)$. The true class label of the instances in $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ is unknown. Instead, a set of N_e experts are asked to provide a single class label for each instance $\mathbf{x}_j$ in $\mathcal{D}_{\mathbf{X},\mathbf{L}}$, and a label vector is built: $\mathbf{l}_j = (l_{j1}, \ldots, l_{jN_e})$ with $l_{jm} \in \Omega_C$, $m = 1, \ldots, N_e$. Table 12.1 shows an example of a dataset with $n = 4$ variables and $N_e = 10$ experts for a problem with three class labels $\Omega_C = \{1, 2, 3\}$.

In our problem, a set of class labels $\mathbf{l}_j$ is available for each training instance. These class labels are summarized in a count vector encoding the number of experts who selected each class label for each instance. These count vectors are explicitly included in the model using conditional multinomial distributions. The training dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ is transformed into a new dataset $\mathcal{D}_{\mathbf{X},\mathbf{V}} = \{(\mathbf{x}_1, \mathbf{v}_1), \ldots, (\mathbf{x}_N, \mathbf{v}_N)\}$ with $\mathbf{v}_j = (v_{j1}, \ldots, v_{jK})$, where v_{jc} is the number of votes that class c, $c \in \Omega_C$, received in the jth instance

$$v_{jc} = \sum_{m=1}^{N_e} \delta(l_{jm}, c), \tag{12.1}$$

with $\delta(x, y) = 1$ if $x = y$ and zero otherwise, and $\sum_{c \in \Omega_C} v_{jc} = N_e$. Table 12.2 shows the transformed dataset $\mathcal{D}_{\mathbf{X},\mathbf{V}}$ for dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ shown in Table 12.1. Our goal is to learn BNCs using the information in Table 12.2.

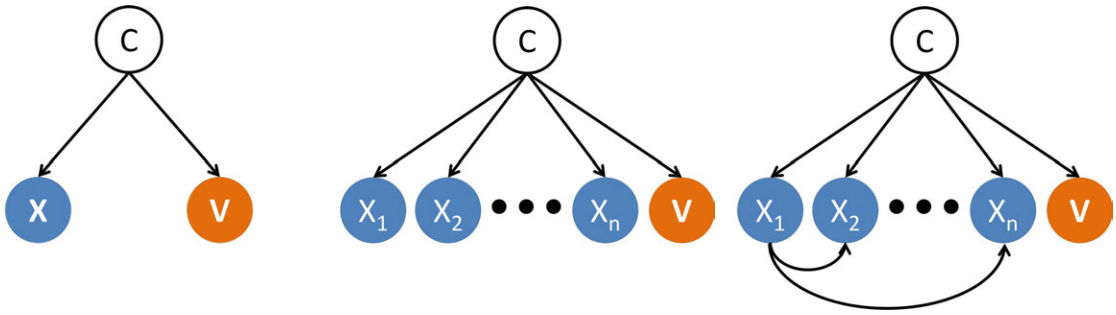
12.1.2 Conditional linear Gaussian classifiers

We studied three BNCs with different structures (see Section 3.4.2.1): MG, NB and AODE classifiers. The class is modeled with a categorical probability distribution $p_C(c; \boldsymbol{\theta}_C)$, $c \in \Omega_C$, where $\boldsymbol{\theta}_C = \{\theta_1, \ldots, \theta_K\}$. The vector of predictive variables $\mathbf{X}$ is modeled with multivariate, univariate or CLG distributions depending on the complexity of the classifiers. The number of votes received by each class is encoded in a vector of random variables $\mathbf{V} = (V_1, \ldots, V_K)$,

Table 12.2: Example of a dataset $\mathcal{D}_{\mathbf{X},\mathbf{V}}$ corresponding to dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ in Table 12.1 with the information about the class modeled as count vectors. The N instances are characterized by $n = 4$ predictive variables, and a count vector $\mathbf{v}_j$ has been derived for each instance.

j	X_1	X_2	X_3	X_4	V_1	V_2	V_3	C
1	5.1	3.5	1.4	0.2	3	2	5	?
2	7	3.2	4.7	1.4	1	6	3	?
...		...				...		...
N	5.9	3.0	5.1	1.8	0	3	7	?

where $\Omega_{\mathbf{V}}$ is the K-part composition of N_e, i.e., the simplex-lattice design containing all the vectors $\mathbf{v} = (v_1, \ldots, v_K)$ such that $v_c \in \{0, 1, \ldots, N_e\}$ and $\sum_{c \in \Omega_C} v_c = N_e$. These count variables $\mathbf{V}$ are modeled using conditional multinomial distributions $p_{\mathbf{V}|C}\left(\mathbf{v}|c\ ;\ \boldsymbol{\theta}_{\mathbf{V}|C}\right)$, where $\boldsymbol{\theta}_{\mathbf{V}|C} = \left\{\theta_{V_1|c}, \ldots, \theta_{V_K|c}\right\}_{c \in \Omega_C}$ are the parameters of the conditional multinomial distribution for each class label c and $\sum_{k \in \Omega_C} \theta_{V_k|c} = 1$. The goal is to use the information in $\mathbf{V}$ as predictive information in the model. Therefore, the number of votes $\mathbf{V}$ received for each class are considered independent of the values of the predictive variables in $\mathbf{X}$ given the class variable C. Figure 12.1 shows the structures of the three BNCs studied in this chapter, where the count variables $\mathbf{V}$ are included in the model.



(a) MG classifier

(b) NB classifier

(c) TAN classifier, as the first classifier of an AODE classifier

Figure 12.1: Network structures of the three CLG classifiers learned with the CoMEM algorithm. Class C is discrete, and the vector of predictive variables $\mathbf{X} = (X_1, \ldots, X_n)$ is continuous. The number of votes received by each instance for each class label are codified in $\mathbf{V}$.

12.1.2.1 Multivariate Gaussian classifier

MG classifiers (see also Section 3.4.2.1) model the features which describe the instances belonging to each class as a MG distribution (see Figure 12.1a) with conditional MG probability density functions $f_{\mathbf{X}|C}(\mathbf{x}|c\ ;\ \boldsymbol{\theta}_{\mathbf{X}|C})$ for $c \in \Omega_C$, where $\boldsymbol{\theta}_{\mathbf{X}|C} = \left\{\boldsymbol{\mu}_{\mathbf{X}|c}, \boldsymbol{\Sigma}_{\mathbf{X}|c}\right\}_{c \in \Omega_C}$ are the parameters of the MG distribution with mean $\boldsymbol{\mu}_{\mathbf{X}|c}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{X}|c}$ for $c \in \Omega_C$.

The joint probability density over the variables $(\mathbf{X}, \mathbf{V})$ of an MG classifier is the FMM

$$f_{\mathbf{X},\mathbf{V}}(\mathbf{x},\mathbf{v}) = \sum_{c\in\Omega_C} p_C(c;\boldsymbol{\theta}_C) p_{\mathbf{V}|C}\left(\mathbf{v}|c\ ;\ \boldsymbol{\theta}_{\mathbf{V}|C}\right) f_{\mathbf{X}|C}\left(\mathbf{x}|c\ ;\ \boldsymbol{\theta}_{\mathbf{X}|C}\right). \tag{12.2}$$

The number of parameters which need to be fitted from data in this MG classifier is $K-1+K(K-1)+2Kn+K\frac{1}{2}n(n-1)$. When the number of variables n is very high or the number of training data N available is low, estimating the full covariance matrix $\boldsymbol{\Sigma}_{\mathbf{X}|c}$ for each class label c can yield inaccurate values.

12.1.2.2 Naive Bayes classifier

The NB classifier (see also Section 3.4.2.1) assumes that all the predictive variables in $\mathbf{X}$ are conditionally independent given the class variable. In a NB classifier, we fit a univariate conditional Gaussian density function $f_{X_i|c}(x_i\ ;\ \boldsymbol{\theta}_{X_i|C})$ for each variable X_i and each class label c, where $\boldsymbol{\theta}_{X_i|C} = \left\{\mu_{X_i|c}, \sigma^2_{X_i|c}\right\}_{c\in\Omega_C}$, and $\mu_{X_i|c}$ and $\sigma^2_{X_i|c}$ are the mean and the variance of variable X_i given $C=c$, respectively. The conditional independence assumption in the NB classifier simplifies Equation (12.2) so that the joint probability density over the variables $(\mathbf{X}, \mathbf{V})$ is the FMM

$$f_{\mathbf{X},\mathbf{V}}(\mathbf{x},\mathbf{v}) = \sum_{c\in\Omega_C} p_C(c;\boldsymbol{\theta}_C) p_{\mathbf{V}|C}\left(\mathbf{v}|c\ ;\ \boldsymbol{\theta}_{\mathbf{V}|C}\right) \prod_{i=1}^{n} f_{X_i|C}\left(x_i|c\ ;\ \boldsymbol{\theta}_{X_j|C}\right). \tag{12.3}$$

The NB classifier only needs to estimate the main diagonal of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{X}|c}$ for each class label c in Equation (12.2), so the final number of parameters in this model is $K-1+K(K-1)+2Kn$.

12.1.2.3 Aggregated one-dependence estimator classifier

Here, we use CLGs as BNCs with a discrete class variable C and a vector of n continuous predictive variables $\mathbf{X} = (X_1, \ldots, X_n)$ (see Figure 12.1c). The conditional density function for a continuous variable X_i having parents $\{\mathbf{Y}_i, C\}$, where $\mathbf{Y}_i \subseteq \mathbf{X}\backslash\{X_i\}$, is defined as a CLG distribution $f_{X_i|\mathbf{Y}_i,C}(x_i|\mathbf{y}_i, c; \boldsymbol{\theta}_{X_i|\mathbf{Y}_i,C})$, where $\boldsymbol{\theta}_{X_i|\mathbf{Y}_i,C} = \left\{\beta_{0X_i|\mathbf{Y}_i,c}, \boldsymbol{\beta}^T_{X_i|\mathbf{Y}_i,c}, \sigma^2_{X_i|\mathbf{Y}_i,c}\right\}_{c\in\Omega_C}$, where $\beta_{0X_i|\mathbf{Y}_ic}$ and $\boldsymbol{\beta}_{X_i|\mathbf{Y}_ic}$ are the regression coefficients of X_i over $\mathbf{Y}_i$ and $C=c$, and $\sigma^2_{X_i|\mathbf{Y}_ic}$ is the conditional variance of X_i given $\mathbf{Y}_i$ for $C=c$. The joint probability density over the variables $(\mathbf{X}, \mathbf{V})$ (Equation (12.2)) of a CLG classifier is given by the FMM

$$f_{\mathbf{X},\mathbf{V}}(\mathbf{x},\mathbf{v}) = \sum_{c\in\Omega_C} p_C(c;\boldsymbol{\theta}_C) p_{\mathbf{V}|C}\left(\mathbf{v}|c\ ;\ \boldsymbol{\theta}_{\mathbf{V}|C}\right) \prod_{i=1}^{n} f_{X_i|\mathbf{Y}_i,C}\left(x_i|\mathbf{y}_i, c\ ;\ \boldsymbol{\theta}_{X_i|\mathbf{Y}_i,C}\right). \tag{12.4}$$

The total number of parameters to be estimated in a CLG classifier depends on the cardinality $Card(\mathbf{Y}_i)$ of the set of parents $\mathbf{Y}_i$ of each variable X_i, and is given by $K-1+K(K-1)+2Kn+K\sum_{i=1}^{n}\left(Card(\mathbf{Y}_i)+\frac{1}{2}Card(\mathbf{Y}_i)(Card(\mathbf{Y}_i)-1)\right)$.

Here, we implemented the AODE classifier (see Section 3.4.2.1). Figure 12.1(c) shows one of the TAN classifier included in the AODE classifier, where variable X_1 is a parent of all the other variables $(X_2, \ldots, X_n)$. For the ith TAN classifier in AODE, the joint density over the variables $(\mathbf{X}, \mathbf{V})$ in Equation (12.4) further simplifies to the FMM

$$f_{\mathbf{X},\mathbf{V}}(\mathbf{x}, \mathbf{v}) = \sum_{c \in \Omega_C} p_C(c; \boldsymbol{\theta}_C) p_{\mathbf{V}|C}\left(\mathbf{v}|c \ ; \ \boldsymbol{\theta}_{\mathbf{V}|C}\right) f_{X_i|C}\left(x_i|c \ ; \ \boldsymbol{\theta}_{X_i|C}\right) \times \prod_{s=1, s \neq i}^{n} f_{X_s|X_i,C}\left(x_s|x_i, c \ ; \ \boldsymbol{\theta}_{X_s|X_i,C}\right). \quad (12.5)$$

The number of parameters to be estimated in this specific TAN classifier is $K - 1 + K(K - 1) + 2Kn + K(n - 1)$. Then, the final number of parameters of an AODE classifier is $n(K - 1 + K(K - 1) + 2Kn + K(n - 1))$. When classifying a new instance, AODE computes the posterior probability of each class label as the mean of the posterior probabilities yielded by each individual TAN classifier.

12.1.3 The counts multinomial expectation maximization algorithm

The EM algorithm is suited for estimating the parameters $\boldsymbol{\Theta}$ of the FMMs defined for each classifier in Section 12.1.2. We illustrate the model for an MG classifier (Section 12.1.2.1), although it can be straightforwardly adapted for NB (Section 12.1.2.2) and AODE classifiers (Section 12.1.2.3). The log-likelihood of a dataset $\mathcal{D}_{\mathbf{X},\mathbf{V}} = \{(\mathbf{x}_1, \mathbf{v}_1), \ldots, (\mathbf{x}_N, \mathbf{v}_N)\}$ (as in Table 12.2) given the FMM in Equation (12.2) with parameters $\boldsymbol{\Theta} = \left\{\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\mathbf{V}|C}, \boldsymbol{\theta}_{\mathbf{X}|C}\right\}$ is

$$\ell(\mathcal{D}_{\mathbf{X},\mathbf{V}}|\boldsymbol{\Theta}) = \sum_{j=1}^{N} \ln \left(\sum_{c \in \Omega_C} p_C(c; \boldsymbol{\theta}_C) p_{\mathbf{V}|C}\left(\mathbf{v}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{V}|C}\right) f_{\mathbf{X}|C}\left(\mathbf{x}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{X}|C}\right) \right). \quad (12.6)$$

The expectation step at iteration q computes the posterior probability that the jth instance belongs to class c given parameters $\boldsymbol{\Theta}^{(q)}$ as

$$t_{jc}^{(q)} = \frac{p_C\left(c \ ; \ \boldsymbol{\theta}_C^{(q)}\right) p_{\mathbf{V}|C}\left(\mathbf{v}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{V}|C}^{(q)}\right) f_{\mathbf{X}|C}\left(\mathbf{x}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{X}|C}^{(q)}\right)}{\sum_{k \in \Omega_C} p_C\left(k \ ; \ \boldsymbol{\theta}_C^{(q)}\right) p_{\mathbf{V}|C}\left(\mathbf{v}_j|k \ ; \ \boldsymbol{\theta}_{\mathbf{V}|C}^{(q)}\right) f_{\mathbf{X}|C}\left(\mathbf{x}_j|k \ ; \ \boldsymbol{\theta}_{\mathbf{X}|C}^{(q)}\right)}. \quad (12.7)$$

Then, the expected log-likelihood of the complete data $\mathcal{Q}\left(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(q)}\right)$ is computed as the sum of the expected log-likelihood of each instance j, $\mathcal{Q}_j\left(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(q)}\right)$:

$$\mathcal{Q}\left(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(q)}\right) = \sum_{j=1}^{N} \mathcal{Q}_j\left(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(q)}\right) = \sum_{j=1}^{N} \sum_{c \in \Omega_C} t_{jc}^{(q)} \left[\ln p_C\left(c \ ; \ \boldsymbol{\theta}_C^{(q)}\right) + \ln p_{\mathbf{V}|C}\left(\mathbf{v}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{V}|C}^{(q)}\right) + \ln f_{\mathbf{X}|C}\left(\mathbf{x}_j|c \ ; \ \boldsymbol{\theta}_{\mathbf{X}|C}^{(q)}\right) \right]. \quad (12.8)$$

In the maximization step at iteration q, we have to find the parameters for the next iteration $\boldsymbol{\Theta}^{(q+1)}$ which maximize Equation (12.8). Therefore, we have to derive Equation (12.8) with respect to every parameter in $\boldsymbol{\Theta}^{(q)}$ and solve for zero. Since $\mathbf{X}$ and $\mathbf{V}$ are assumed to be independent given C, these parameters can be easily computed. The parameters of the class prior probability distribution $\boldsymbol{\theta}_C^{(q)} = \{\theta_1, \ldots, \theta_K\}$ and the parameters of the conditional MG distributions $\boldsymbol{\theta}_{\mathbf{X}|C} = \left\{\boldsymbol{\mu}_{\mathbf{X}|c}, \boldsymbol{\Sigma}_{\mathbf{X}|c}\right\}_{c \in \Omega_C}$ are computed as in the classical EM algorithm for fitting a finite mixture of Gaussian distributions:

$$\begin{aligned}
\theta_c^{(q+1)} &= \frac{1}{N} \sum_{j=1}^{N} t_{jc}^{(q)}, \\
\boldsymbol{\mu}_{\mathbf{X}|c}^{(q+1)} &= \frac{1}{\sum_{j=1}^{N} t_{jc}^{(q)}} \sum_{j=1}^{N} t_{jc}^{(q)} \mathbf{x}_j, \\
\boldsymbol{\Sigma}_{\mathbf{X}|c}^{(q+1)} &= \frac{1}{\sum_{j=1}^{N} t_{jc}^{(q)}} \sum_{j=1}^{N} t_{jc}^{(q)} \left(\mathbf{x}_j - \boldsymbol{\mu}_{\mathbf{X}|c}^{(q+1)}\right) \left(\mathbf{x}_j - \boldsymbol{\mu}_{\mathbf{X}|c}^{(q+1)}\right)^T.
\end{aligned} \tag{12.9}$$

The parameters of the conditional multinomial distributions of the count variables $\mathbf{V}$ given the class label c are obtained using Lagrange multipliers and including the restriction $\sum_{k \in \Omega_C} {\theta_{V_k|c}}^{(q+1)} = 1$ [388, 389]:

$$\theta_{V_k|c}^{(q+1)} = \frac{\sum_{j=1}^{N} t_{jc}^{(q)} v_{jk}}{\sum_{k'=1}^{K} \sum_{j=1}^{N} t_{jc}^{(q)} v_{jk'}}. \tag{12.10}$$

The EM algorithm iteratively computes the expectation and maximization steps until it reaches a local maximum in the log-likelihood (Equation (12.6)). Therefore, there is no guarantee that the algorithm will find the global maximum. Usually, several runs of the algorithm are performed with different initial values of parameters $\boldsymbol{\Theta}^{(1)}$ in the first expectation step (Equation (12.7)) of the algorithm, and the model with the highest log-likelihood in all the runs is returned. Instead, in our CoMEM algorithm, we initialize the values of $t_{ic}^{(1)}$ to the proportion of experts who classified the jth item as belonging to class c: v_{jc}/N_e. The first maximization step (Equations (12.9) and (12.10)) is computed using those values, and the algorithm is run until convergence. Therefore, instead of running the CoMEM algorithm several times with different initializations, we use the information about the class provided by the experts for each instance to initialize the expected posterior probabilities $t_{jc}^{(1)}$ and only one complete run of the algorithm is performed. The stopping criterion used to check the convergence of the algorithm is

$$\frac{\ell(\mathcal{D}_{\mathbf{X},\mathbf{V}}|\boldsymbol{\Theta}^{(q)}) - \ell(\mathcal{D}_{\mathbf{X},\mathbf{V}}|\boldsymbol{\Theta}^{(q-1)})}{\left|\ell(\mathcal{D}_{\mathbf{X},\mathbf{V}}|\boldsymbol{\Theta}^{(q-1)})\right|} < \epsilon. \tag{12.11}$$

We set the convergence parameter to $\epsilon = 10^{-6}$.

The CoMEM algorithm can be easily particularized for learning NB (Section 12.1.2.2)

or AODE classifiers (Section 12.1.2.3). In the FMM for the NB classifier (Equation (12.3)), only the main diagonal of the covariance matrix $\mathbf{\Sigma}_{\mathbf{X}|c}^{(q+1)}$ (Equation (12.9)) has to be estimated for each class label c. In the AODE classifier, only the variance of each variable $\Sigma_{X_i|c}^{(q+1)}$ and the covariance between each variable and its parent $\mathbf{\Sigma}_{X_iY_i|c}^{(q+1)}$ have to be estimated in Equation (12.9). These covariances are then used to compute the parameters $\beta_{0X_s|X_ic}, \beta_{X_s|X_ic}$ and $\sigma^2_{X_s|X_ic}$ of the CLG distributions in the FMM of the TAN classifiers (Equation (12.5)) that build up the AODE classifier.

12.1.4 Classification of a new instance

The CoMEM algorithm is used to find the ML estimates of the FMM for each classifier in Section 12.1.2. These parameters can be directly plugged into the (conditional) probability distributions making up the BNCs, so that they can be used for classification. In traditional BNCs, we use the maximum a posteriori decision rule so that $\mathbf{x}$ is assigned to the class c^* with maximum posterior probability

$$c^* = \arg\max_{c\in\Omega_C} p_{C|\mathbf{X}}(c|\mathbf{x}) = \arg\max_{c\in\Omega_C} p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|c).$$

The BNCs studied here also include the vector of count variables $\mathbf{V}$ as predictive information, and, applying the Bayes rule, they yield $p_{C|\mathbf{X},\mathbf{V}}(c|\mathbf{x},\mathbf{v})$. However, values $\mathbf{v}$ are not available for a new instance $\mathbf{x}$ to classify. Here, we investigate three approximations of $p_{C|\mathbf{X}}(c|\mathbf{x})$:

`The independence classification rule` (CoMEM-IND) ignores variables $\mathbf{V}$ when classifying $\mathbf{x}$ and computes the posterior probability $p_{C|\mathbf{X}}(c|\mathbf{x})$ as

$$p_{C|\mathbf{X}}(c|\mathbf{x}) = \sum_{\mathbf{v}\in\Omega_\mathbf{V}} p_{C|\mathbf{X},\mathbf{V}}(c|\mathbf{x},\mathbf{v}) p_{\mathbf{V}|\mathbf{X}}(\mathbf{v}|\mathbf{x}) = \frac{p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|c)}{\sum_{k\in\Omega_C} p_C(k) f_{\mathbf{X}|C}(\mathbf{x}|k)}. \tag{12.12}$$

`The multinomial maximum classification rule` (CoMEM-MUL) classifies a new instance $\mathbf{x}$ taking into account all the possible combinations of votes $\mathbf{v}$ for the class labels. Here, we approximate $p_{C|\mathbf{X}}(c|\mathbf{x})$ with the maximum of the addends in Equation (12.12):

$$p_{C|\mathbf{X}}(c|\mathbf{x}) \approx \max_{\mathbf{v}\in\Omega_\mathbf{V}} p_{C|\mathbf{X},\mathbf{V}}(c|\mathbf{x},\mathbf{v}) p_{\mathbf{V}|\mathbf{X}}(\mathbf{v}|\mathbf{x}) = \max_{\mathbf{v}\in\Omega_\mathbf{V}} \frac{p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|c) p_{\mathbf{V}|C}(\mathbf{v}|c)}{\sum_{k\in\Omega_C} p_C(k) f_{\mathbf{X}|C}(\mathbf{x}|k)},$$

where $\Omega_\mathbf{V}$ is the domain of all vectors $\mathbf{v} = (v_1, \ldots, v_K)$ such that $v_c \in \{0, 1, \ldots, N_e\}$ and $\sum_{c\in\Omega_C} v_c = N_e$.

`The binomial maximum classification rule` (CoMEM-BIN) assumes that the only relevant information for computing the posterior probability $p_{C|\mathbf{X}}(c|\mathbf{x})$ is the number of votes v_c that instance $\mathbf{x}$ could have received for class label c, independently of the

number of votes received by the other class labels. The posterior probability $p_{C|\mathbf{X}}(c|\mathbf{x})$ is approximated by the maximum:

$$p_{C|\mathbf{X}}(c|\mathbf{x}) \approx \max_{v_c \in \Omega_{V_c}} p_{C|\mathbf{X},V_c}(c|\mathbf{x}, v_c) p_{V_c|\mathbf{X}}(v_c|\mathbf{x}) = \max_{v_c \in \Omega_{V_c}} \frac{p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|C) p_{V_c|C}(v_c|c)}{\sum_{k \in \Omega_C} p_C(k) f_{\mathbf{X}|C}(\mathbf{x}|k)},$$

where $\Omega_{V_c} = \{0, 1, \ldots, N_e\}$. The probability distribution of $\mathbf{V}$ given the class label c was modeled as a multinomial distribution $p_{\mathbf{V}|C}(\mathbf{v}|c\,;\,\boldsymbol{\theta}_{\mathbf{V}|C})$. Therefore, the conditional probability of a single variable V_c given the class label k is a binomial distribution: $p_{V_c|C}(v_c|k\,;\,\theta_{V_c|k})$.

In real applications, the class labels are usually related, e.g., some classes are frequently confused whereas others are easily distinguishable. The CoMEM-MUL classification rule can take into account these relationships between class labels because it considers all the possible combinations of votes $\mathbf{v} \in \Omega_{\mathbf{V}}$. However, the cardinality of $\Omega_{\mathbf{V}}$ is $\binom{N_e+K-1}{K-1}$, so it rapidly increases with the number of experts N_e and the number of class labels K. Therefore, these classification rules can become extremely inefficient in real applications. The CoMEM-BIN classification rule considers partial information about the class labels because only the number of votes v_c are used for computing $p_{C|\mathbf{X}}(c|\mathbf{x})$. This rule ignores possible relationships between the class labels. However, the cardinality of Ω_{V_c} is $N_e + 1$, so this classification rule is much more efficient than CoMEM-MUL. Note also that the multinomial distribution reduces to the binomial distribution for binary classes. Therefore, CoMEM-MUL and CoMEM-BIN are equivalent when the classification problem has two class labels.

12.2 Related work: Modeling probabilistic class labels with belief functions

Côme et al. [96] proposed a framework for learning MG classifiers where the information about the class for each instance is modeled with belief functions in the context of the transferable belief model. We tailor their approach to our problem, where a set of class labels is available for each training instance. Here, the information about the class for each instance is summarized as a probability distribution over the class labels. Therefore, the training dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ is transformed into a new dataset $\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}} = \{(\mathbf{x}_1, \boldsymbol{\pi}_1), \ldots, (\mathbf{x}_N, \boldsymbol{\pi}_N)\}$, where $\boldsymbol{\pi}_j = (\pi_{j1}, \ldots, \pi_{jK})$ is a probability distribution over Ω_C so that π_{jc} is the probability of instance j belonging to class c

$$\pi_{jc} = \frac{\sum_{m=1}^{N_e} \delta(\ell_{jm}, c)}{N_e}, \tag{12.13}$$

with $0 \leq \pi_{jc} \leq 1$ and $\sum_{c \in \Omega_C} \pi_{jc} = 1$. Table 12.3 shows the transformed dataset $\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}}$ for dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ shown in Table 12.1.

In [96], the information about the class of each instance $\mathbf{x}_j$ is modeled as a basic belief assignment (BBA) [455, 464]. A BBA is a function $m_j^{\Omega_C} : 2^{\Omega_C} \to [0, 1]$ over the powerset

Table 12.3: Example of a dataset $\mathcal{D}_{\mathbf{X},\mathbf{\Pi}}$ with probabilistic class labels corresponding to dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ in Table 12.1. The N instances are characterized by $n = 4$ predictive variables and a probability $\pi_{\ c}$ for each class label $c \in \Omega_C = \{1, 2, 3\}$.

j	X_1	X_2	X_3	X_4	$\pi_{\ 1}$	$\pi_{\ 2}$	$\pi_{\ 3}$	C
1	5.1	3.5	1.4	0.2	0.3	0.2	0.5	?
2	7	3.2	4.7	1.4	0.1	0.6	0.3	?
...		...				...		...
N	5.9	3.0	5.1	1.8	0.0	0.3	0.7	?

Table 12.4: Example of a general BBA $m_j^{\Omega_C}(\omega)$ in [502] (left) and a Bayesian BBA (right), and the corresponding plausibility functions $pl_j^{\Omega_C}(\omega)$. The class variable C has three values $\Omega_C = \{1, 2, 3\}$. The Bayesian BBA (right) corresponds to the class information of the first instance in $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ (Table 12.1) and $\mathcal{D}_{\mathbf{X},\mathbf{\Pi}}$ (Table 12.3).

ω	General BBA		Bayesian BBA	
	$m_j^{\Omega_C}(\omega)$	$pl_j^{\Omega_C}(\omega)$	$m_j^{\Omega_C}(\omega)$	$pl_j^{\Omega_C}(\omega)$
$\emptyset$	0.0	0.0	0.0	0.0
$\{1\}$	0.1	0.7	0.3	0.3
$\{2\}$	0.0	0.3	0.2	0.2
$\{3\}$	0.3	0.7	0.5	0.5
$\{1, 2\}$	0.2	0.7	0.0	0.5
$\{1, 3\}$	0.3	1.0	0.0	0.8
$\{2, 3\}$	0.0	0.9	0.0	0.7
Ω_C	0.1	1.0	0.0	1.0

2^{Ω_C}, verifying $\sum_{\omega \subseteq \Omega_C} m_j^{\Omega_C}(\omega) = 1$. Table 12.4 shows an example of a general BBA (left) in [502], where different beliefs are assigned to each possible subset $\omega \subseteq \Omega_C$. In our scenario, each BBA is a probability distribution over Ω_C, so all the focal sets (subsets $\omega \subseteq 2^{\Omega_C}$ with $m_j^{\Omega_C}(\omega) > 0$) are singletons, and the BBA is called a Bayesian BBA (rightmost column in Table 12.4). Therefore, $m_j^{\Omega_C}(\{c\}) = \pi_{jc}$ for each singleton set $\{c\} \subseteq 2^{\Omega_C}$ and $m_j^{\Omega_C}(\omega) = 0$ for all the non-singleton sets $\omega \subseteq 2^{\Omega_C}, Card(\omega) \neq 1$.

For each instance, a plausibility function $pl_j^{\Omega_C}(\omega)$ can be built from the BBA $m_j^{\Omega_C}(\omega)$ as

$$pl_j^{\Omega_C}(\omega) = \sum_{\substack{\gamma \subseteq \Omega_C, \\ \gamma \cap \omega \neq \emptyset}} m_j^{\Omega_C}(\gamma), \ \forall \omega \subseteq \Omega_C.$$

Since our $m_j^{\Omega_C}(\omega)$ are Bayesian, the plausibility functions $pl_j^{\Omega_C}(\omega)$ are probability measures. Table 12.4 shows the plausibilities for a general BBA and a Bayesian BBA.

The PLEM algorithm is a generalization of the EM algorithm [128]. PLEM uses the belief function theory in the context of the transferable belief model [464] for fitting a finite mixture of MG distributions with K components, one for each class label $c \in \Omega_C$:

$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{c \in \Omega_C} p_C(c \ ; \ \boldsymbol{\theta}_C) f_{\mathbf{X}|C}\left(\mathbf{x}|c \ ; \ \boldsymbol{\theta}_{\mathbf{X}|C}\right). \tag{12.14}$$

Côme et al. [96] derived a generalized likelihood criterion which measures the plausibility $pl^{\boldsymbol{\Theta}}(\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}}|\boldsymbol{\Theta})$ of a dataset $\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}}$ given the FMM (Equation (12.14)) with parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_C, \boldsymbol{\theta}_{\mathbf{X}|C}\}$. Then, the PLEM algorithm finds the parameters $\boldsymbol{\Theta}$ which maximize the generalized log-likelihood criterion

$$\ln(pl^{\boldsymbol{\Theta}}(\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}}|\boldsymbol{\Theta})) = \sum_{j=1}^{N} \ln \left(\sum_{c \in \Omega_C} pl_{jc} p_C(c \,;\, \boldsymbol{\theta}_C) f_{\mathbf{X}|C}\left(\mathbf{x}_j | c \,;\, \boldsymbol{\theta}_{\mathbf{X}|C}\right) \right), \tag{12.15}$$

where $pl_{jc} = pl_j^{\Omega_C}(\{c\})$ are the plausibilities of the jth instance for the set $\{c\}$. For Bayesian BBAs we have that $pl_{jc} = pl_j^{\Omega_C}(\{c\}) = m_j^{\Omega_C}(\{c\}) = \pi_{jc}$, so the generalized log-likelihood criterion (Equation (12.15)) simplifies to

$$\ell(\mathcal{D}_{\mathbf{X},\boldsymbol{\Pi}}|\boldsymbol{\Theta})) = \sum_{j=1}^{N} \ln \left(\sum_{c \in \Omega_C} \pi_{jc} p_C(c \,;\, \boldsymbol{\theta}_C) f_{\mathbf{X}|C}\left(\mathbf{x}_j | c \,;\, \boldsymbol{\theta}_{\mathbf{X}|C}\right) \right). \tag{12.16}$$

The PLEM algorithm is then particularized to maximize the expected generalized log-likelihood criterion (Equation (12.16)) by alternating the two steps:

Expectation step in iteration q: compute the expected posterior probabilities

$$t_{jc}^{(q)} = \frac{\pi_{jc} p_C\left(c \,;\, \boldsymbol{\theta}_C^{(q)}\right) f_{\mathbf{X}|C}\left(\mathbf{x}_j | c \,;\, \boldsymbol{\theta}_{\mathbf{X}|C}^{(q)}\right)}{\sum_{k \in \Omega_C} \pi_{jk} p_C\left(k \,;\, \boldsymbol{\theta}_C^{(q)}\right) f_{\mathbf{X}|C}\left(\mathbf{x}_j | k \,;\, \boldsymbol{\theta}_{\mathbf{X}|C}^{(q)}\right)}. \tag{12.17}$$

Maximization step in iteration q: find the parameters which maximize the expected generalized log-likelihood of the complete data. The parameters of the class prior probability distribution $\boldsymbol{\theta}_C^{(q+1)}$ and the parameters of the conditional MG distributions $\boldsymbol{\theta}_{\mathbf{X}|C}^{(q+1)}$ for the next iteration are computed as in Equation (12.9).

The PLEM algorithm, like the EM algorithm, iteratively maximizes Equation (12.16) until a local maximum is achieved. The same convergence criterion as in the CoMEM algorithm was used (Equation (12.11)). As before, the expected posterior probabilities $t_{jc}^{(1)}$ in the first iteration (Equation (12.17)) are initialized to π_{jc} and the first maximization step is performed with those initial values. Then, the algorithm is run until convergence.

This approach does not explicitly model the class probabilities $\boldsymbol{\pi}_j$ for each instance. In fact, these class probabilities stay constant throughout the whole algorithm and do not appear in the final model (see Figures 3.2, 3.3 and 3.5). Therefore, this particularized PLEM algorithm could be seen as a weighted version of the classical EM algorithm, where probabilities $\boldsymbol{\pi}_j$ are used as weights in the log-likelihood of the model (Equation (12.16)).

Fitting the parameters of NB or AODE classifiers is straightforward by computing the necessary covariances in Equation (12.9) and using the factorization of the joint probability over $\mathbf{X}$ in a similar way to Equations (12.3) and (12.4), respectively.

12.3 Experiments

This section includes the evaluation of the classifiers over 16 datasets taken from the UCI [27], KEEL [9] and LibSVM [78] repositories. Each variable in the datasets was standardized by subtracting the mean and dividing by the standard deviation. We deleted the eighth variable in the `glass` dataset because 82.24% of the values were zero and the estimated covariance matrices were not positive definite in some runs. Similarly, we deleted the first variable in the `ion` dataset because 89.17% of its values were equal to 1. Table 12.5 shows the main features of the final datasets used in the experimentation.

Table 12.5: Datasets used in the experiments.

Name	No. instances N	No. pred. variables n	No. class labels K
`appendicitis`	106	7	2
`fourclass`	862	2	2
`glass`	214	8	2
`haberman`	306	3	2
`ion`	351	32	2
`iris`	150	4	3
`liver`	341	6	2
`newthyroid`	215	5	3
`phoneme`	5,404	5	2
`ring`	7,400	20	2
`svmguide1`	7,089	4	2
`twonorm`	7,400	20	2
`vehicle`	846	18	4
`waveform`	5,000	21	3
`wdbc`	569	30	2
`wine`	178	13	3

Three different classifiers were considered (see Section 12.1.2): MG, NB and AODE. The parameters of the probability distributions for each classifier were found by fitting FMMs using three algorithms: PLEM, CoMEM and EM. Additionally, three classification rules were provided for CoMEM (see Section 12.1.4). The probability distributions for the class labels $\boldsymbol{\pi}_j$ were used to initialize the posterior probabilities $t_{jc}^{(1)}$ in the expectation step of the first iteration of the three algorithms. Therefore, the following classification scenarios were studied:

CoMEM: The information about the class was codified as count vectors and explicitly modeled in the BNCs as conditional multinomial distributions. The CoMEM algorithm was used to find the ML estimates of the parameters of the model as explained in Section 12.1. We studied the three classification rules proposed in Section 12.1.4: CoMEM-IND, CoMEM-MUL and CoMEM-BIN.

PLEM: The information about the class is encoded as probability distributions over the class labels modeled with belief functions. The parameters of the BNCs were found using the PLEM algorithm (see Section 12.2).

EM: The BNCs were fitted as in an unsupervised learning scenario, i.e., no information

about the true class labels is available. The classical EM algorithm [128] was used for estimating the parameters of the probability distributions in the classifiers.

12.3.1 Dataset generation and stratified h-fold cross-validation with experts' class labels

We took the original datasets in Table 12.5 and generated a dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ with experts' class labels. For each instance $\mathbf{x}_j$, a set of experts' class labels $\mathbf{l}_j = \{l_{j1}, \ldots, l_{jN_e}\}$ was generated. The mistakes made by the experts were modeled using a beta distribution with mean μ_B and standard deviation σ_B. For the jth instance and the mth expert, we sampled a value b_{jm} from the beta distribution $B(\mu_B, \sigma_B)$. Then, a value u_{jm} was drawn from a uniform distribution in $[0, 1]$. If $u_{jm} \geq b_{jm}$, the expert's label l_{jm} was set to the true class label c_j of the jth instance. Otherwise, the expert's label l_{jm} was set to any other class label in $\Omega_C \setminus \{c_j\}$ with equal probability. The beta distribution models the mistakes made by an expert when classifying the instances. Low values of μ_B yielded a low number of mistakes and most of the experts assigned the correct label to the instances, whereas high values of μ_B yielded a lot of mistakes when labeling the instances.

Stratified h-fold cross-validation was used to honestly evaluate the models, taking into account the different number of instances belonging to each class (see Section 2.3.2). The true class label of the instances was not available in $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ (see Table 12.1), so the stratified cross-validation process was based on the number of votes v_{jc} received by each class label for each instance (see Equation (12.1)). A simple greedy algorithm was designed for generating the folds in the cross-validation process. The goal was to generate folds with the same proportion of instances of each class label as the dataset with experts' class labels $\mathcal{D}_{\mathbf{X},\mathbf{L}}$. An instance was randomly drawn from the dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ without replacement. The class label which yielded the maximum number of votes for that instance was found, and the instance was introduced in the fold which had the lowest number of votes for that class label (adding the votes of the instances already included in the fold). Ties were broken randomly. Table 12.6 shows the probabilities for each class label in each dataset in Table 12.5, the mean class probability in the transformed dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ with experts' labels and the mean and standard deviation of the mean probabilities of each class label averaged over the h folds obtained with the proposed stratified h-fold cross-validation procedure. The proposed algorithm yielded folds with similar proportions to the transformed dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$, even when the class labels were unbalanced (e.g., `appendicitis` or `glass` datasets). Also, the cross-validation generated folds with a low standard deviation in the class probabilities. Once the folds were generated, we proceeded as in a classical cross-validation setting. Each fold was considered once for testing the classifier learned using the other $h - 1$ training folds.

12.3.2 Evaluation measures

Two measures were considered for evaluating the quality of the classifiers. First, we computed the error of the classifiers with respect to the true class labels of the instances in the dataset

Table 12.6: Comparison between the class probability in the original datasets in Table 12.5, the class probabilities in the transformed dataset $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ with experts' class labels generated using a beta distribution $B(0.1, 0.01)$ and the mean and standard deviation of the class probabilities in one run of the stratified h-fold cross validation ($h = 10$).

Dataset	Class label	Class probability in the original dataset	Mean class probability in the transformed dataset with experts labels $\mathcal{D}_{\mathbf{X},\mathbf{L}}$	Mean class probability and standard deviation in the h folds
appendicitis	1	0.8019	0.7396	0.7400 ± 0.0201
	2	0.1981	0.2604	0.2600 ± 0.0201
fourclass	1	0.3561	0.3819	0.3819 ± 0.0022
	2	0.6439	0.6181	0.6181 ± 0.0022
glass	1	0.2383	0.2958	0.2956 ± 0.0095
	2	0.7617	0.7042	0.7044 ± 0.0095
haberman	1	0.7353	0.6817	0.6817 ± 0.0058
	2	0.2647	0.3183	0.3183 ± 0.0058
ion	1	0.6410	0.6091	0.6091 ± 0.0066
	2	0.3590	0.3909	0.3909 ± 0.0066
iris	1	0.3333	0.3367	0.3368 ± 0.0140
	2	0.3333	0.3160	0.3159 ± 0.0138
	3	0.3333	0.3473	0.3473 ± 0.0114
liver	1	0.4164	0.4358	0.4358 ± 0.0048
	2	0.5836	0.5642	0.5642 ± 0.0048
newthyroid	1	0.6977	0.6433	0.6434 ± 0.0118
	2	0.1628	0.1874	0.1873 ± 0.0109
	3	0.1395	0.1693	0.1693 ± 0.0075
phoneme	1	0.7065	0.6675	0.6675 ± 0.0002
	2	0.2935	0.3325	0.3325 ± 0.0002
ring	1	0.4951	0.4950	0.4950 ± 0.0003
	2	0.5049	0.5050	0.5050 ± 0.0003
svmguide1	1	0.5643	0.5506	0.5506 ± 0.0004
	2	0.4357	0.4494	0.4494 ± 0.0004
twonorm	1	0.5004	0.5013	0.5013 ± 0.0004
	2	0.4996	0.4987	0.4987 ± 0.0004
vehicle	1	0.2577	0.2537	0.2537 ± 0.0030
	2	0.2352	0.2384	0.2384 ± 0.0029
	3	0.2506	0.2511	0.2511 ± 0.0027
	4	0.2565	0.2569	0.2569 ± 0.0028
waveform	1	0.3294	0.3304	0.3304 ± 0.0003
	2	0.3392	0.3359	0.3359 ± 0.0005
	3	0.3314	0.3338	0.3338 ± 0.0006
wdbc	1	0.3726	0.4018	0.4018 ± 0.0033
	2	0.6274	0.5982	0.5982 ± 0.0033
wine	1	0.3315	0.3275	0.3276 ± 0.0141
	2	0.3989	0.3933	0.3933 ± 0.0136
	3	0.2697	0.2792	0.2792 ± 0.0156

(see Section 2.3.2). However, in a real problem, the true class label of the instances is not known and the error cannot be computed. Moreover, even if the true class labels are available, we could define the goal of a classifier as being to reproduce the information given by the set of experts. Thus, we could compare the proportion of experts $\boldsymbol{\pi}_j = (\pi_{j1}, \ldots, \pi_{jK})$ who classify the jth instance with a given class label (Equation (12.13)) and the posterior probability yielded by the BNC $\left(p_{C|\mathbf{X}}(1|\mathbf{x}_j), \ldots, p_{C|\mathbf{X}}(K|\mathbf{x}_j)\right)$. For each instance $\mathbf{x}_j$, the mean squared

error (MSE) was used to measure the difference between the "true probability" $\boldsymbol{\pi}_j$ and the posterior probability yielded by the BNC:

$$MSE_j = \frac{1}{K} \sum_{c \in \Omega_C} \left(\pi_{jc} - p_{C|\mathbf{X}}(c|\mathbf{x}_j)\right)^2. \tag{12.18}$$

12.3.3 Results

This section reports the results of the experimental evaluation of the models proposed in this paper. For each one of the 16 datasets in Table 12.5, three classifiers (MG, NB and AODE) were induced with the studied algorithms and classification rules: CoMEM-IND, CoMEM-MUL, CoMEM-BIN, PLEM and EM. For each original dataset in Table 12.5, we generated ten datasets with experts' class labels $\mathcal{D}_{\mathbf{X},\mathbf{L}}$ using beta distributions with $\mu_B = 0.1, 0.2, 0.3, 0.4$ and $\sigma_B = 0.01$ (Section 12.3.1). A stratified 10-fold cross-validation was performed with each dataset with experts' class labels $\mathcal{D}_{\mathbf{X},\mathbf{L}}$. Appendix C reports the mean and the standard deviation of the classification error and the MSE (Equation (12.18)) computed over the 10 folds and the 10 datasets $\mathcal{D}_{\mathbf{X},\mathbf{L}}$. The goal was to minimize the two evaluation measures, so lower values of the evaluation measures yielded better classifiers.

First, we looked for statistically significant differences between the algorithms (CoMEM-IND, CoMEM-MUL, CoMEM-BIN, PLEM and EM) using all the combinations of datasets, classifiers and values of μ_B as input information for the tests. Table 12.7 shows the average ranking obtained by each algorithm for each evaluation measure. Higher values of the ranking yielded better performances. CoMEM-MUL and CoMEM-BIN were ranked highest and second highest, respectively, in terms of both classification error and MSE. EM yielded the worst average rank for classification error and MSE. CoMEM-IND outperformed PLEM on MSE, whereas PLEM outperformed CoMEM-IND on classification error. Friedman's test [195] and Iman-Davenport's test [271] were performed to check the null hypothesis that all the algorithms performed similarly. Boldface numbers indicate statistically significant results at a significance level $\alpha = 0.05$. This null hypothesis was rejected for the two evaluation measures with both tests (all p-values < 0.0001). García and Herrera [207] extended the tests proposed by Demšar [129] for performing all pairwise comparisons between several algorithms on a collection of datasets. Therefore, we performed Bergmann-Hommel's [42] post-hoc test as computed in [207] to find statistical differences in the performance of all pairs of algorithms. Table 12.8 shows the adjusted p-values of the tests for every pair of algorithms and each evaluation measure. Looking at the ordered rankings in Table 12.7, we can conclude from the statistically significant differences detected in Table 12.8 that all the algorithms outperformed EM for the two evaluation measures. We found no significant differences between CoMEM-MUL and CoMEM-BIN. CoMEM-MUL and CoMEM-BIN outperformed PLEM on both accuracy and MSE. CoMEM-IND outperformed PLEM according to the MSE evaluation measure, whereas the two algorithms performed similarly with respect to classification error.

We also compared the three classifiers (MG, NB and AODE) by studying their perfor-

Table 12.7: Average rankings of the five algorithms for each evaluation measure, and p-values of Friedman's and Iman-Davenport's tests of equal performance between all algorithms.

Classification error		Mean squared error	
Algorithm	Ranking	Algorithm	Ranking
CoMEM-MUL	3.5182	CoMEM-MUL	3.7292
CoMEM-BIN	3.4661	CoMEM-BIN	3.6927
PLEM	3.1198	CoMEM-IND	3.5781
CoMEM-IND	3.0599	PLEM	2.6406
EM	1.8359	EM	1.3594
Friedman's test	p-value < **0.0001**		p-value < **0.0001**
Iman-Davenport's test	p-value < **0.0001**		p-value < **0.0001**

Table 12.8: Adjusted p-values of Bergmann-Hommel's post-hoc tests for all pairwise comparisons between the five algorithms (CoMEM-IND, CoMEM-MUL, CoMEM-BIN, PLEM and EM) when all the datasets (16), classifiers (3) and values of μ_B (4) are considered.

H_1	Classification error p-value	Mean squared error p-value
CoMEM-MUL $\neq$ CoMEM-BIN	1.0000	1.0000
CoMEM-MUL $\neq$ CoMEM-IND	**0.0271**	1.0000
CoMEM-MUL $\neq$ PLEM	**0.0406**	**<0.0001**
CoMEM-MUL $\neq$ EM	**<0.0001**	**<0.0001**
CoMEM-BIN $\neq$ CoMEM-IND	**0.0355**	1.0000
CoMEM-BIN $\neq$ PLEM	**0.0406**	**<0.0001**
CoMEM-BIN $\neq$ EM	**<0.0001**	**<0.0001**
CoMEM-IND $\neq$ PLEM	1.0000	**<0.0001**
CoMEM-IND $\neq$ EM	**<0.0001**	**<0.0001**
PLEM $\neq$ EM	**<0.0001**	**<0.0001**

mances in the 16 datasets, the five algorithms and the four possible values of μ_B. Table 12.9 shows the three algorithms ordered according to their average ranking for each evaluation measure. MG yielded the highest ranking for classification error, whereas AODE achieved the highest ranking for MSE. NB yielded the worst ranking for both evaluation measures. Friedman's and Iman-Davenport's tests (all p-values < 0.0001) reveal significant differences between the classifiers for the two evaluation measures. We performed all the pairwise comparisons between the three classifiers and reported the adjusted p-values of the Bergmann-Hommel's post-hoc tests in Table 12.10. MG and AODE outperformed NB for both evaluation measures. No significant differences between MG and AODE were found when considering the classification error. However, AODE outperformed MG on MSE.

We also studied the differences between the algorithms taking each individual classifier and each value of $\mu_B = 0.1, 0.2, 0.3, 0.4$ independently. This way, we were able to analyze the behavior of each algorithm and each classifier as we increased the errors in the class labels provided by the experts. Tables 12.11 to 12.14 show the pairwise comparisons between the algorithms for each value of $\mu_B = 0.1, 0.2, 0.3, 0.4$. The number of datasets out of 16 in which

Table 12.9: Average rankings of the three classifiers for each evaluation measure, and p-values of Friedman's and Iman-Davenport's tests of equal performance between all classifiers.

Classification error		Mean squared error	
Algorithm	Ranking	Algorithm	Ranking
MG	2.2062	AODE	2.4687
AODE	2.0969	MG	2.0563
NB	1.6969	NB	1.4750
Friedman's test	p-value < **0.0001**		p-value < **0.0001**
Iman-Davenport's test	p-value < **0.0001**		p-value < **0.0001**

Table 12.10: Adjusted p-values of Bergmann-Hommel's post-hoc tests for all pairwise comparisons between the three classifiers (MG, NB and AODE) when all the datasets (16), algorithms (5) and values of μ_B (4) are considered.

H_1	Classification error	Mean squared error
	p_{Berg}	p_{Berg}
MG $\neq$ NB	**<0.0001**	**<0.0001**
AODE $\neq$ MG	0.1665	**<0.0001**
AODE $\neq$ NB	**<0.0001**	**<0.0001**

the first algorithm won, tied or lost against the second algorithm is shown for each pair of algorithms (W/T/L column). Also, the tables include the p-values of a two-tailed binomial test and the Bergmann-Hommel's post-hoc test to look for significant differences between the performance of the algorithms. Boldface numbers indicate statistically significant results at a significance level $\alpha = 0.05$. For each pair of algorithms in the test, the algorithm's name is highlighted in boldface if it significantly outperforms the other classifier in any test or evaluation measure.

First, we compared the three classification rules of the proposed method (CoMEM-IND, CoMEM-MUL and CoMEM-BIN) against PLEM. PLEM never outperformed any of the proposed classification rules on either classification error or MSE. For low values of $\mu_B = 0.1$ (Table 12.11), the MG classifier learned with CoMEM-IND, CoMEM-MUL and CoMEM-BIN outperformed PLEM on classification error according to both the binomial test and the Bergmann-Hommel's test. Also, for $\mu_B = 0.2$ (Table 12.12), the MG classifiers learned with these proposed rules outperformed PLEM on classification error according to the binomial test. According to MSE, MG classifiers learned with the proposed approach outperformed PLEM for low values of $\mu_B = 0.1, 0.2$. AODE classifiers learned with CoMEM-IND, CoMEM-MUL and CoMEM-BIN outperformed PLEM on MSE for all values of μ_B, whereas there were no significant differences in accuracy. In NB classifiers, no significant differences between PLEM and the proposed classification rules could be found.

Second, we compared the classification rules for the proposed model. On the one hand, CoMEM-IND never outperformed CoMEM-MUL or CoMEM-BIN for any value of μ_B and any classifier. On the other hand, CoMEM-MUL and CoMEM-BIN outperformed CoMEM-IND in classification error for some specific scenarios, e.g., for NB and AODE classifiers

Table 12.11: Comparison of the proposed methods (CoMEM-MUL, CoMEM-BIN, CoMEM-IND, PLEM and EM) considering 3 classifiers (MG, NB, AODE), all the datasets (16) and $\mu_B = 0.1$.

H_1	Classification error			Mean squared error		
	W/T/L	Binomial	Bergmann	W/T/L	Binomial	Bergmann
			MG			
CoMEM-MUL ≠ CoMEM-BIN	2/13/1	1.0000	1.0000	2/10/4	0.6875	1.0000
CoMEM-MUL ≠ CoMEM-IND	6/5/5	1.0000	1.0000	6/0/10	0.4545	1.0000
CoMEM-MUL ≠ PLEM	14/0/2	**0.0042**	**0.0437**	15/0/1	**0.0005**	**0.0087**
CoMEM-MUL ≠ EM	14/0/2	**0.0042**	**0.0004**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	4/6/6	0.7539	1.0000	5/0/11	0.2101	1.0000
CoMEM-BIN ≠ PLEM	13/1/2	**0.0074**	**0.0437**	16/0/0	**<0.0001**	**0.0043**
CoMEM-BIN ≠ EM	14/0/2	**0.0042**	**0.0006**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-IND ≠ PLEM	13/1/2	**0.0074**	**0.0437**	15/0/1	**0.0005**	**0.0009**
CoMEM-IND ≠ EM	14/0/2	**0.0042**	**0.0004**	16/0/0	**<0.0001**	**<0.0001**
PLEM ≠ EM	14/0/2	**0.0042**	0.5844	16/0/0	**<0.0001**	0.1767
			NB			
CoMEM-MUL ≠ CoMEM-BIN	3/11/2	1.0000	1.0000	5/10/1	0.2188	1.0000
CoMEM-MUL ≠ CoMEM-IND	8/5/3	0.2266	1.0000	10/0/6	0.4545	1.0000
CoMEM-MUL ≠ PLEM	9/0/7	0.8036	1.0000	12/0/4	0.0768	0.2316
CoMEM-MUL ≠ EM	13/0/3	**0.0213**	**0.0175**	15/0/1	**0.0005**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	7/8/1	0.0703	1.0000	9/0/7	0.8036	1.0000
CoMEM-BIN ≠ PLEM	9/0/7	0.8036	1.0000	12/0/4	0.0768	0.3150
CoMEM-BIN ≠ EM	13/0/3	**0.0213**	**0.0175**	15/0/1	**0.0005**	**<0.0001**
CoMEM-IND ≠ PLEM	9/0/7	0.8036	1.0000	12/0/4	0.0768	0.3594
CoMEM-IND ≠ EM	13/0/3	**0.0213**	0.1170	15/0/1	**0.0005**	**0.0001**
PLEM ≠ EM	12/0/4	0.0768	0.1014	16/0/0	**<0.0001**	**0.0208**
			AODE			
CoMEM-MUL ≠ CoMEM-BIN	2/11/3	1.0000	1.0000	2/11/3	1.0000	1.0000
CoMEM-MUL ≠ CoMEM-IND	7/3/6	1.0000	1.0000	7/0/9	0.8036	1.0000
CoMEM-MUL ≠ PLEM	9/0/7	0.8036	1.0000	13/0/3	**0.0213**	0.1158
CoMEM-MUL ≠ EM	13/0/3	**0.0213**	**0.0211**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	6/4/6	1.0000	1.0000	6/0/10	0.4545	1.0000
CoMEM-BIN ≠ PLEM	9/0/7	0.8036	1.0000	13/0/3	**0.0213**	0.1158
CoMEM-BIN ≠ EM	13/0/3	**0.0213**	**0.0211**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-IND ≠ PLEM	8/0/8	1.0000	1.0000	13/0/3	**0.0213**	0.0608
CoMEM-IND ≠ EM	13/0/3	**0.0213**	**0.0211**	16/0/0	**<0.0001**	**<0.0001**
PLEM ≠ EM	14/0/2	**0.0042**	**0.0211**	16/0/0	**<0.0001**	**0.0208**

when $\mu_B = 0.2$ (Table 12.12) or NB classifiers when $\mu_B = 0.3$ (Table 12.13). No significant differences between CoMEM-MUL and CoMEM-BIN were found in any experiment.

Third, the classical EM algorithm never outperformed any of the other methods studied in this paper. EM was frequently outperformed by PLEM, CoMEM-IND, CoMEM-MUL and CoMEM-BIN in all the classifiers, evaluation measures and values of μ_B, and specially for MG classifiers. NB classifiers learned with EM were outperformed on classification error for low values of μ_B, whereas the algorithms performed more similarly for higher values of μ_B. Also, on MSE, NB classifiers learned with EM were outperformed for values of $\mu_B = 0.1, 0.2, 0.3$. Similarly, AODE classifiers learned with PLEM or the proposed classification rules usually outperformed those learned with EM on both classification error and MSE for lower values of μ_B. For higher values of μ_B, significant differences could only be found on MSE.

Table 12.12: Comparison of the proposed methods (CoMEM-MUL, CoMEM-BIN, CoMEM-IND, PLEM and EM) considering 3 classifiers (MG, NB, AODE), all the datasets (16) and $\mu_B = 0.2$.

H_1	Classification error			Mean squared error		
	W/T/L	Binomial	Bergmann	W/T/L	Binomial	Bergmann
			MG			
CoMEM-MUL ≠ CoMEM-BIN	1/14/1	1.0000	1.0000	2/10/4	0.6875	1.0000
CoMEM-MUL ≠ CoMEM-IND	9/3/4	0.2668	1.0000	8/0/8	1.0000	1.0000
CoMEM-MUL ≠ PLEM	12/1/3	**0.0352**	0.1521	15/0/1	**0.0005**	**0.0043**
CoMEM-MUL ≠ EM	14/0/2	**0.0042**	**0.0002**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	9/4/3	0.1460	1.0000	9/0/7	0.8036	1.0000
CoMEM-BIN ≠ PLEM	12/1/3	**0.0352**	0.1521	15/0/1	**0.0005**	**0.0039**
CoMEM-BIN ≠ EM	14/0/2	**0.0042**	**0.0002**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-IND ≠ PLEM	12/1/3	**0.0352**	0.3971	15/0/1	**0.0005**	**0.0043**
CoMEM-IND ≠ EM	14/0/2	**0.0042**	**0.0032**	16/0/0	**<0.0001**	**<0.0001**
PLEM ≠ EM	15/0/1	**0.0005**	0.1544	15/0/1	**0.0005**	0.2294
			NB			
CoMEM-MUL ≠ CoMEM-BIN	1/13/2	1.0000	1.0000	4/9/3	1.0000	0.9110
CoMEM-MUL ≠ CoMEM-IND	10/4/2	**0.0386**	0.4986	10/0/6	0.4545	0.8645
CoMEM-MUL ≠ PLEM	9/0/7	0.8036	1.0000	12/0/4	0.0768	0.1314
CoMEM-MUL ≠ EM	14/0/2	**0.0042**	**0.0007**	15/0/1	**0.0005**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	10/6/0	**0.0020**	0.4986	12/0/4	0.0768	0.8645
CoMEM-BIN ≠ PLEM	9/0/7	0.8036	1.0000	12/0/4	0.0768	0.1314
CoMEM-BIN ≠ EM	14/0/2	**0.0042**	**0.0004**	15/0/1	**0.0005**	**<0.0001**
CoMEM-IND ≠ PLEM	8/0/8	1.0000	1.0000	12/0/4	0.0768	0.4375
CoMEM-IND ≠ EM	14/0/2	**0.0042**	0.0755	15/0/1	**0.0005**	**0.0014**
PLEM ≠ EM	14/0/2	**0.0042**	**0.0070**	14/0/2	**0.0042**	0.0755
			AODE			
CoMEM-MUL ≠ CoMEM-BIN	3/12/1	0.6250	1.0000	2/11/3	1.0000	1.0000
CoMEM-MUL ≠ CoMEM-IND	10/5/1	**0.0117**	0.4986	9/0/7	0.8036	1.0000
CoMEM-MUL ≠ PLEM	9/0/7	0.8036	1.0000	15/0/1	**0.0005**	**0.0029**
CoMEM-MUL ≠ EM	12/0/4	0.0768	**0.0144**	15/0/1	**0.0005**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	9/7/0	**0.0039**	0.5844	10/0/6	0.4545	1.0000
CoMEM-BIN ≠ PLEM	8/0/8	1.0000	1.0000	15/0/1	**0.0005**	**0.0026**
CoMEM-BIN ≠ EM	12/0/4	0.0768	**0.0262**	15/0/1	**0.0005**	**<0.0001**
CoMEM-IND ≠ PLEM	8/0/8	1.0000	0.5844	15/0/1	**0.0005**	**0.0073**
CoMEM-IND ≠ EM	12/0/4	0.0768	0.5844	15/0/1	**0.0005**	**<0.0001**
PLEM ≠ EM	14/0/2	**0.0042**	**0.0405**	14/0/2	**0.0042**	0.7188

Finally, we observed that the number of statistically significant differences between algorithms decreased with higher values of μ_B. Low values of μ_B yielded more information about the true class labels of the training instances, because a higher number of votes $\mathbf{v}_j$ in CoMEM and a higher probability $\boldsymbol{\pi}_j$ in PLEM were simulated for the true class label of each instance $\mathbf{x}_j$. In these scenarios, the different ways of managing the uncertain class data of CoMEM and PLEM yield significantly different performances. On the other hand, little information about the true class labels of the training instances is available with high values of μ_B. In these settings, the number of votes $\mathbf{v}_j$ and the class probabilities $\boldsymbol{\pi}_j$ are more uniform for each class label, so CoMEM and PLEM performances are more alike.

Table 12.13: Comparison of the proposed methods (CoMEM-MUL, CoMEM-BIN, CoMEM-IND, PLEM and EM) considering 3 classifiers (MG, NB, AODE), all the datasets (16) and $\mu_B = 0.3$.

H_1	Classification error			Mean squared error		
	W/T/L	Binomial	Bergmann	W/T/L	Binomial	Bergmann
			MG			
CoMEM-MUL ≠ CoMEM-BIN	1/12/3	0.6250	1.0000	4/9/3	1.0000	1.0000
CoMEM-MUL ≠ CoMEM-IND	4/10/2	0.6875	1.0000	9/0/7	0.8036	1.0000
CoMEM-MUL ≠ PLEM	11/0/5	0.2101	0.7907	11/0/5	0.2101	0.4986
CoMEM-MUL ≠ EM	13/0/3	**0.0213**	**0.0048**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	5/11/0	0.0625	1.0000	9/0/7	0.8036	1.0000
CoMEM-BIN ≠ PLEM	11/0/5	0.2101	0.7873	11/0/5	0.2101	0.4986
CoMEM-BIN ≠ EM	13/0/3	**0.0213**	**0.0018**	16/0/0	**<0.0001**	**<0.0001**
CoMEM-IND ≠ PLEM	11/0/5	0.2101	1.0000	11/0/5	0.2101	0.4986
CoMEM-IND ≠ EM	13/0/3	**0.0213**	**0.0122**	16/0/0	**<0.0001**	**<0.0001**
PLEM ≠ EM	15/0/1	**0.0005**	0.1014	14/0/2	**0.0042**	**0.0102**
			NB			
CoMEM-MUL ≠ CoMEM-BIN	1/13/2	1.0000	1.0000	5/11/0	0.0625	1.0000
CoMEM-MUL ≠ CoMEM-IND	8/7/1	**0.0391**	0.4868	12/0/4	0.0768	0.6299
CoMEM-MUL ≠ PLEM	5/0/11	0.2101	0.8750	9/0/7	0.8036	0.8645
CoMEM-MUL ≠ EM	10/2/4	0.1796	0.2650	13/0/3	**0.0213**	**0.0018**
CoMEM-BIN ≠ CoMEM-IND	8/8/0	**0.0078**	0.4868	12/0/4	0.0768	0.8645
CoMEM-BIN ≠ PLEM	5/0/11	0.2101	0.8750	9/0/7	0.8036	1.0000
CoMEM-BIN ≠ EM	10/2/4	0.1796	0.2650	13/0/3	**0.0213**	**0.0086**
CoMEM-IND ≠ PLEM	5/0/11	0.2101	0.0834	9/0/7	0.8036	1.0000
CoMEM-IND ≠ EM	10/1/5	0.3018	1.0000	13/0/3	**0.0213**	0.1346
PLEM ≠ EM	13/0/3	**0.0213**	**0.0211**	14/0/2	**0.0042**	**0.0292**
			AODE			
CoMEM-MUL ≠ CoMEM-BIN	3/13/0	0.2500	1.0000	4/11/1	0.3750	1.0000
CoMEM-MUL ≠ CoMEM-IND	6/7/3	0.5078	1.0000	10/0/6	0.4545	1.0000
CoMEM-MUL ≠ PLEM	7/0/9	0.8036	1.0000	14/0/2	**0.0042**	**0.0026**
CoMEM-MUL ≠ EM	9/0/7	0.8036	0.8766	16/0/0	**<0.0001**	**<0.0001**
CoMEM-BIN ≠ CoMEM-IND	5/7/4	1.0000	1.0000	9/0/7	0.8036	1.0000
CoMEM-BIN ≠ PLEM	7/0/9	0.8036	1.0000	14/0/2	**0.0042**	**0.0063**
CoMEM-BIN ≠ EM	9/0/7	0.8036	1.0000	16/0/0	**<0.0001**	**<0.0001**
CoMEM-IND ≠ PLEM	8/0/8	1.0000	1.0000	14/0/2	**0.0042**	**0.0104**
CoMEM-IND ≠ EM	10/0/6	0.4545	1.0000	16/0/0	**<0.0001**	**<0.0001**
PLEM ≠ EM	14/0/2	**0.0042**	0.4417	12/0/4	0.0768	0.4701

12.4 Conclusion

In this chapter, we studied the problem of learning BNCs when the true class label of the instances is not known. Instead, a set of class labels provided by a group of experts individually are available for each instance. This is a naturally occurring scenario in a number of fields, e.g., it is common in medical practice that several physicians are asked to give their opinion on a diagnosis, prognosis or treatment. Therefore, a set of experts' class labels is obtained for each instance.

We proposed a novel approach for learning BNCs using that class information. We modeled the information about the class for each instance as a count vector containing the number of experts who assigned each class label for the given instance. This information was explic-

Table 12.14: Comparison of the proposed methods (CoMEM-MUL, CoMEM-BIN, CoMEM-IND, PLEM and EM) considering 3 classifiers (MG, NB, AODE), all the datasets (16) and $\mu_B = 0.4$.

H_1	Classification error			Mean squared error		
	W/T/L	Binomial	Bergmann	W/T/L	Binomial	Bergmann
			MG			
CoMEM-MUL ≠ CoMEM-BIN	2/13/1	1.0000	1.0000	3/11/2	1.0000	1.0000
CoMEM-MUL ≠ CoMEM-IND	6/10/0	**0.0313**	1.0000	6/0/10	0.4545	1.0000
CoMEM-MUL ≠ PLEM	11/0/5	0.2101	0.5612	7/0/9	0.8036	1.0000
CoMEM-MUL ≠ EM	13/1/2	**0.0074**	**0.0011**	13/0/3	**0.0213**	**0.0246**
CoMEM-BIN ≠ CoMEM-IND	4/12/0	0.1250	1.0000	7/0/9	0.8036	1.0000
CoMEM-BIN ≠ PLEM	11/0/5	0.2101	0.5612	7/0/9	0.8036	1.0000
CoMEM-BIN ≠ EM	13/1/2	**0.0074**	**0.0017**	13/0/3	**0.0213**	**0.0246**
CoMEM-IND ≠ PLEM	11/0/5	0.2101	1.0000	7/0/9	0.8036	1.0000
CoMEM-IND ≠ EM	13/1/2	**0.0074**	**0.0146**	14/0/2	**0.0042**	**0.0053**
PLEM ≠ EM	14/0/2	**0.0042**	0.1170	14/0/2	**0.0042**	**0.0053**
			NB			
CoMEM-MUL ≠ CoMEM-BIN	2/13/1	1.0000	1.0000	4/11/1	0.3750	1.0000
CoMEM-MUL ≠ CoMEM-IND	4/10/2	0.6875	1.0000	11/0/5	0.2101	0.7213
CoMEM-MUL ≠ PLEM	6/0/10	0.4545	1.0000	6/0/10	0.4545	1.0000
CoMEM-MUL ≠ EM	9/2/5	0.4240	1.0000	12/0/4	0.0768	0.0713
CoMEM-BIN ≠ CoMEM-IND	3/12/1	0.6250	1.0000	11/0/5	0.2101	0.8035
CoMEM-BIN ≠ PLEM	6/0/10	0.4545	1.0000	6/0/10	0.4545	1.0000
CoMEM-BIN ≠ EM	9/2/5	0.4240	1.0000	12/0/4	0.0768	0.1170
CoMEM-IND ≠ PLEM	6/0/10	0.4545	0.9735	6/0/10	0.4545	0.7051
CoMEM-IND ≠ EM	9/2/5	0.4240	1.0000	12/0/4	0.0768	0.7188
PLEM ≠ EM	12/1/3	**0.0352**	0.1888	12/0/4	0.0768	**0.0365**
			AODE			
CoMEM-MUL ≠ CoMEM-BIN	3/13/0	0.2500	1.0000	2/11/3	1.0000	1.0000
CoMEM-MUL ≠ CoMEM-IND	7/9/0	**0.0156**	1.0000	6/0/10	0.4545	1.0000
CoMEM-MUL ≠ PLEM	10/0/6	0.4545	1.0000	13/0/3	**0.0213**	**0.0238**
CoMEM-MUL ≠ EM	10/2/4	0.1796	0.0729	14/0/2	**0.0042**	**0.0071**
CoMEM-BIN ≠ CoMEM-IND	6/8/2	0.2891	1.0000	7/0/9	0.8036	1.0000
CoMEM-BIN ≠ PLEM	10/0/6	0.4545	1.0000	13/0/3	**0.0213**	**0.0185**
CoMEM-BIN ≠ EM	10/2/4	0.1796	0.1755	14/0/2	**0.0042**	**0.0039**
CoMEM-IND ≠ PLEM	10/0/6	0.4545	1.0000	13/0/3	**0.0213**	**0.0071**
CoMEM-IND ≠ EM	10/2/4	0.1796	0.5248	15/0/1	**0.0005**	**0.0009**
PLEM ≠ EM	13/0/3	**0.0213**	0.5844	9/0/7	0.8036	1.0000

itly added to the model using conditional multinomial distributions. Therefore, the induced models more closely fit the original experiment setting. Also, the multinomial distributions take into account the number of experts who provide the class labels for each training instance. The CoMEM algorithm, an adaptation of the EM algorithm [128], was proposed to find the parameters of the probability distributions in the BNCs by fitting an FMM. Three CLG classifiers with different complexities were studied: MG, NB and AODE classifiers. Three classification rules were proposed for assigning a class label to a previously unseen (and unlabeled) instance.

We compared the proposed models against the approach in [96] and the classical EM algorithm for unsupervised learning. The PLEM algorithm, a particularization of the algorithm method in [96], was considered for the scenario where the class information for each

instance is given as a probability distribution over the class labels by encoding the proportion of experts who selected each class label for the given instance. An extensive evaluation was performed on 16 datasets. A greedy algorithm was proposed to perform stratified h-fold cross-validation when the true class labels of the training instances were not known, but a set of experts' class labels was available for each instance. This algorithm generated folds with similar proportions of instances of each class label to the original datasets. Two evaluation measures were considered for studying the performance of the classifiers and the algorithms. The accuracy (classification error) depends on the true class label of the instances being known in order to evaluate the classifier. In a real setting, however, these class labels are not available and the goal might be to accurately reproduce the experts' opinions. Therefore, we also evaluated the classifiers using the mean squared error between the "true" probability of each class label (proportion of experts who classified the instances with each given class label) and the posterior probability predicted by the classifiers.

We found statistically significant differences between the methods studied in this paper (CoMEM, PLEM and EM). In general, CoMEM-MUL and CoMEM-BIN were the best methods for both evaluation measures. CoMEM-IND never outperformed CoMEM-MUL or CoMEM-BIN. Additionally, we studied the differences between the three classifiers: MG, NB and AODE. MG and AODE outperformed NB for both evaluation measures. We could not find statistically significant differences between MG and AODE regarding classification error. However, AODE outperformed MG in terms of MSE. The combination of TAN classifiers in AODE reduces the classifier variance [513] and that might be why it is able to reproduce the true probability distributions over the class labels.

Therefore, we can conclude that explicitly adding the uncertain information about the class labels to the model yielded better results than using PLEM or the classical EM algorithm. Moreover, modeling this uncertain information can be an advantage when model interpretation is important. When applied to a real problem, it would be possible to study the relationships between the class labels by analyzing the conditional probability distributions for the class count vectors and thus understand the causes of the disagreements between experts.

Future work includes applying these approaches to learn other BNCs with different structures. The classifiers considered in this work (MG, NB and AODE) had fixed structures. We could use the structural EM [196] and extend the PLEM and CoMEM algorithms to find the conditional independence relationships between the predictive variables $\mathbf{X}$ given the class variable. Also, we jointly modeled the number of votes $\mathbf{V}$ received by the class labels. However, we could also exploit the conditional independences between the variables $V_c \in \mathbf{V}$. This could be interesting when the number of class labels is high and the cardinality of $\mathbf{V}$ increases. Additionally, we could extend this work to other BNCs such as the TAN [199], the k-DB [438] or the semi-naive Bayes [394] classifiers. Both the TAN and k-DB algorithms for inducing BNCs are based on the computation of the conditional MI between each pair of variables given the class variable. However, estimating this conditional MI when the true class labels are not available is a matter of open research.

BNCs are known to suffer from redundant variables [309]. Therefore, feature subset selection methods are expected to improve the performance of these classifiers. On the one hand, wrapper approaches could be easily applied to the proposed models. However, these methods are computationally expensive because a classifier has to be induced and evaluated to estimate the quality of each subset of predictive variables. On the other hand, filter methods compute the quality of a subset of predictive variables by measuring the information that they provide about the class, e.g., using MI or correlation measures. Adapting these measures to the scenario where the information about the class is provided as a group of experts' class labels is another challenge.

Different distributions could be considered when modeling the class count vectors in the proposed approach. For instance, the multinomial Dirichlet distribution, also known as the Polya distribution, was proposed for modeling word counts in the context of document classification [160, 348], and has been successfully applied in other domains [60]. This distribution is useful for modeling the phenomenon known as "word burstiness", which refers to the fact that, when a rare word appears in a document, the probability of it appearing again in that same document increases. A similar phenomenon is likely to occur when a group of experts assign class labels to the training instances, i.e., the probability of a class label is higher if other experts have already classified the instance with that class label. Therefore, modeling the class count vectors with multinomial Dirichlet distributions could yield better results. One drawback of both the multinomial distribution and the multinomial Dirichlet distribution is that their covariance matrix is strictly negative [57], i.e., no positive correlations between class labels can be modeled. Other distributions for count data which allow for positive correlations between the variables have been recently proposed, such as the multinomial generalized Dirichlet distribution [59] or the multinomial beta-Liouville distribution [58]. Investigating the ability of these distributions for modeling expert class labels is also interesting.

Finally, in this chapter, we have assumed that all the experts are equally reliable when providing a class label for the instances. However, significant differences between experts can be identified in real problems [426] (see also Chapter 10). Extending the proposed models to take into account the distinctive behavior of each expert might increase classifier accuracy. Additionally, explicitly modeling the individual experts might provide some interesting insights into the problem domain by studying the differences between experts.

Part V

CONCLUSIONS

Chapter 13

Conclusions and future work

13.1 Summary of contributions

The contributions in this dissertation have been divided into two parts:

Part III includes the research on BN learning. BNs are used in Chapter 7 for modeling and simulating basal dendritic trees from pyramidal cells from the mouse neocortex. A large number of variables encoding information about the dendritic trees, subtrees, branches, neighboring segments, etc. are measured from 3D reconstructions of real dendritic trees, and BNs are learned from these data. The relationships between the variables are encoded in the BN structures, and the models are interpreted to obtain interesting insights into the dendritic structure. A simulation algorithm is proposed for sampling virtual dendrites from the BNs. A multidimensional test based on the KL divergence is used to compare the virtual and real dendritic trees.

Chapter 8 proposes methods for learning MoP approximations of one-dimensional, multidimensional and conditional probability densities from data using B-spline interpolation. The methods are evaluated by sampling datasets from known probability distributions (including complex mixtures of distributions) and by comparing the MoP approximations with the true generating densities. Also, we empirically compare the proposed methods for learning MoPs using B-splines with the previous proposals based on LIPs. The proposed methods are used as a non-parametric density estimation technique in BNCs.

Chapter 9 studies the supervised learning problem when the predictive variables are angular or directional. We introduce directional NB classifiers using von Mises or von Mises–Fisher densities. We also study hybrid classifiers with linear, directional and discrete variables. The classifiers are studied from a theoretical point of view by analyzing the decision functions they induce. Also, the directional classifiers are empirically compared with classical NB classifiers using Gaussian densities or discretization. Appendices A and B include the detailed derivations of the decision surfaces studied in Chapter 9.

Part IV includes our work on statistical methods for analyzing and modeling the consensus between experts. Chapter 10 explains the problem of the classification of GABAergic interneurons from the neocortex. A web application is designed to retrieve the classification of a set of interneurons by a group of experts from laboratories all around the world. Then, the class labels provided by the group of experts are analyzed using statistical tools. The consensus between experts is measured using agreement indices. Then, we cluster the neurons according to the classifications provided by the group of experts. Additionally, we induce one BN representing the behavior of each expert, and we analyze and compare the BNs of the experts. Finally, automatic classifiers are learned using the most frequently voted class label for each neuron. All these analyses helped identifying agreed neuronal types and confusing terms commonly used in the literature.

Based on the data collected in Chapter 10, we propose in Chapter 11 a method for building a BN that models the opinions of the group of experts regarding the classification of the GABAergic interneurons. One BN is learned for each expert, representing his/her behavior in the classification experiment. Then, we propose a method for clustering the BNs into groups corresponding to experts with similar behaviors. We induce a representative BN for each cluster, modeling the common behavior of the experts in the same cluster. Finally, a consensus BN is built as a finite mixture of BNs and represented as a Bayesian multinet. The proposed method is able to confirm and extend our findings regarding the classification of GABAergic interneurons.

Inspired by the interneuron classification problem, in Chapter 12 we propose a method for learning BNCs when the true class labels of the training instances are not known. Instead, a vector of class labels provided by a group of experts is available for each instance. We summarize the experts class labels as a count vector modeling the number of votes received by each class label for each instance. We consider different BNC structures and use the EM algorithm for finding the parameters of the probability distributions in the BNCs. The proposed BNCs are empirically studied using artificial datasets. Appendix C includes the detailed results of the classifiers.

13.2 List of publications

The research reported in this dissertation has produced the following list of publications and submissions.

1. Peer-reviewed journals

   P. L. López-Cruz, C. Bielza, P. Larrañaga, R. Benavides-Piccione, and J. DeFelipe. Models and simulation of 3D neuronal dendritic trees using Bayesian networks. *Neuroinformatics*, 9(4):347–369, 2011. Impact factor (JCR 2011): 2.973. Ranking: 13/99. Category: Computer science, interdisciplinary applications.

J. DeFelipe, P. L. López-Cruz, R. Benavides-Piccione, C. Bielza, P. Larrañaga, S. Anderson, A. Burkhalter, B. Cauli, A. Fairén, D. Feldmeyer, G. Fishell, D. Fitzpatrick, T. F. Freund, G. González-Burgos, S. Hestrin, S. Hill, P. R. Hof, J. Huang, E. G. Jones, Y. Kawaguchi, Z. Kisvárday, Y. Kubota, D. A. Lewis, O. Marín, H. Markram, C. J. McBain, H. S. Meyer, H. Monyer, S. B. Nelson, K. Rockland, J. Rossier, J. L. R. Rubenstein, B. Rudy, M. Scanziani, G. M. Shepherd, C. C. Sherwood, J. F. Staiger, G. Tamás, A. Thomson, Y. Weng, R. Yuste, and G. A. Ascoli. New insights into the classification and nomenclature of cortical GABAergic interneurons. *Nature Reviews Neuroscience*, 14(3):202–216, 2013. Impact factor (JCR 2012): 31.673. Ranking: 1/251. Category: Neurosciences.

P. L. López-Cruz, P. Larrañaga, J. DeFelipe, and C. Bielza. Bayesian network modeling of the consensus between experts: An application to neuron classification. *International Journal of Approximate Reasoning*, in press, 2013. Impact Factor (JCR 2012): 1.729. Ranking: 35/114. Category: Computer science, artificial intelligence.

P. L. López-Cruz, C. Bielza, and P. Larrañaga. Directional naive Bayes classifiers. *Pattern Analysis and Applications*, in press, 2013. Impact factor (JCR 2012): 0.814. Ranking: 78/114. Category: Computer science, artificial intelligence.

P. L. López-Cruz, P. Larrañaga, and C. Bielza. Learning mixtures of polynomials of multidimensional densities from data. *International Journal of Approximate Reasoning*, submitted, 2013. Impact Factor (JCR 2012): 1.729. Ranking: 35/114. Category: Computer science, artificial intelligence.

2. Congress contributions and communications

P. L. López-Cruz, C. Bielza, P. Larrañaga, R. Benavides-Piccione, and J. DeFelipe. 3D simulation of dendritic morphology using Bayesian networks. *16th Annual Meeting of the Organization for Human Brain Mapping (HBM 2010)*. Barcelona, Spain, 6–10 June 2010.

P. L. López-Cruz, C. Bielza, P. Larrañaga, R. Benavides-Piccione, and J. DeFelipe. 3D simulation of dendritic morphology using Bayesian networks. *Single Neuron Morphology and Function Workshop*. Amsterdam, Netherlands, 8–9 July 2010.

P. L. López-Cruz, C. Bielza, and P. Larrañaga. The von Mises naive Bayes classifier for angular data. In J. A. Lozano, J. A. Gámez, and J. A. Moreno, editors, *Advances in Artificial Intelligence, Proceedings of the 14th Conference of the Spanish Association for Artificial Intelligence*, volume 7023 of *Lecture Notes in Computer Science*, pages 145–154. Springer, 2011.

P. L. López-Cruz, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials from data using B-spline interpolation. In A. Cano, M. Gómez-Olmedo, and T. D. Nielsen, editors, *Proceedings of the 6th European Workshop on Probabilistic Graphical Models* (PGM 2012), pages 211–218, 2012.

P. L. López-Cruz, C. Bielza, and P. Larrañaga. Learning conditional linear Gaussian classifiers with probabilistic class labels. In C. Bielza et al., editors, *Advances in Artificial Intelligence, Proceedings of the 15th MultiConference of the Spanish Association for Artificial Intelligence*, volume 8109 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2013.

P. L. López-Cruz, T. D. Nielsen, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials of conditional densities from data. In C. Bielza et al., editors, *Advances in Artificial Intelligence, Proceedings of the 15th MultiConference of the Spanish Association for Artificial Intelligence*, volume 8109 of *Lecture Notes in Computer Science*, pages 363–372. Springer, 2013.

13.3 Future work

This section summarizes the most relevant research topics and open issues that will be studied in the future. The reader can find a more thorough discussion of each one of these topics in the specific conclusions sections of each chapter.

We studied how to learn MoP approximations of one-dimensional, multidimensional and conditional probability densities from data (Chapter 8). We intend to study how to use the proposed methods for working with hybrid BNs. This includes challenging tasks such as: proposing new BN learning algorithms, designing computationally efficient methods for inference in hybrid BNs, etc. We intend to use the proposed methods for solving problems different from supervised learning, e.g., clustering problems, regression problems, etc.

We also intend to extend our work in PGMs using directional random variables (Chapter 9). The family of von Mises densities is not closed under conditioning and marginalization. Therefore, BNs using directional or angular variables are difficult to work with, and specific learning and inference methods have to be studied. Hybrid BNs including linear, directional and discrete variables are even more challenging. We foresee that approximation methods will be required for working with these complex hybrid BNs.

A number of interesting research topics have been addressed in this dissertation regarding the analysis and modeling of the consensus between experts (Chapter 11) and the generation of BNCs for partially supervised learning problems with uncertain class labels provided by a group of experts (Chapter 12). We intend to extend these works in different ways. Regarding Chapter 11, we plan to further study the problem of clustering BNs. One option includes applying probabilistic clustering using FMMs for clustering the JPDs encoded by the BNs. Also, different methods for combining and aggregating the BN structures and parameters into a consensus model will be studied. Regarding Chapter 12, we intend to further study the problem of learning BNCs with uncertain class labels. Different probability distributions will be used for modeling the count vectors and/or the probability distributions encoding the uncertain class information.

Finally, we will design and apply more complex models to some of the neuroscience problems addressed in this dissertation. Regarding Chapter 7, we want to to design complex

BNs for modeling and simulating complete neuronal morphologies of pyramidal cells including basal dendrites, apical dendrites and axons. Hybrid BNs including linear, directional and discrete variables with flexible probability distributions will be crucial for solving this research line. Regarding Chapter 10, we intend to keep studying the neuron classification problem for GABAergic interneurons. This will require methods for managing uncertain information provided by different sources and experts such as the ones proposed in this dissertation, as well as the new methods mentioned in the previous paragraph.

Part VI

APPENDICES

Appendix A

Von Mises NB classifier decision function

A.1 vMNB with one predictive variable

We start by equaling the posterior probability of each class value using the probability density function of the von Mises distribution (5.4):

$$p(C=1)\frac{1}{2\pi I_0(\kappa_{\Phi|1})}\exp(\kappa_{\Phi|1}\cos(\phi-\mu_{\Phi|1}))=p(C=2)\frac{1}{2\pi I_0(\kappa_{\Phi|2})}\exp(\kappa_{\Phi|2}\cos(\phi-\mu_{\Phi|2})).$$

Simplify the constant 2π, take logarithms and arrange all terms on the same side of the equation:

$$\kappa_{\Phi|1}\cos(\phi-\mu_{\Phi|1})-\kappa_{\Phi|2}\cos(\phi-\mu_{\Phi|2})+\ln\frac{p(C=1)}{I_0(\kappa_{\Phi|1})}-\ln\frac{p(C=2)}{I_0(\kappa_{\Phi|2})}=0.$$

Substitute $\cos(\beta-\gamma)=\cos(\beta)\cos(\gamma)+\sin(\beta)\sin(\gamma)$ and operate the logarithms:

$$\kappa_{\Phi|1}\left[\cos\phi\cos\mu_{\Phi|1}+\sin\phi\sin\mu_{\Phi|1}\right]-\kappa_{\Phi|2}\left[\cos\phi\cos\mu_{\Phi|2}+\sin\phi\sin\mu_{\Phi|2}\right]+\ln\frac{p(C=1)I_0(\kappa_{\Phi|2})}{p(C=2)I_0(\kappa_{\Phi|1})}=0.$$

Arrange using $\cos\phi$ and $\sin\phi$ as common terms:

$$(\kappa_{\Phi|1}\cos\mu_{\Phi|1}-\kappa_{\Phi|2}\cos\mu_{\Phi|2})\cos\phi+(\kappa_{\Phi|1}\sin\mu_{\Phi|1}-\kappa_{\Phi|2}\sin\mu_{\Phi|2})\sin\phi+\ln\frac{p(C=1)I_0(\kappa_{\Phi|2})}{p(C=2)I_0(\kappa_{\Phi|1})}=0.$$

Substitute

$$a = \kappa_{\Phi|1} \cos \mu_{\Phi|1} - \kappa_{\Phi|2} \cos \mu_{\Phi|2},$$
$$b = \kappa_{\Phi|1} \sin \mu_{\Phi|1} - \kappa_{\Phi|2} \sin \mu_{\Phi|2},$$
$$D = -\ln \frac{p(C=1) I_0(\kappa_{\Phi|2})}{p(C=2) I_0(\kappa_{\Phi|1})},$$

and get:

$$a \cos \phi + b \sin \phi = D.$$

Trigonometrically, this is equivalent to:

$$T \cos(\phi - \alpha) = D,$$

where $T = \sqrt{a^2 + b^2}, \cos \alpha = a/T, \sin \alpha = b/T, \tan \alpha = b/a$.

Isolating ϕ from the equation, we get:

$$\phi' = \alpha + \arccos(D/T),$$
$$\phi'' = \alpha - \arccos(D/T).$$

The NB classifier finds two angles that bound the class regions.

A.1.1 Particular cases

We have also derived these angles when the conditional probability distributions share one of the parameters. We consider that the classes are equiprobable. If they are not equiprobable, the prior probabilities of the class values influence the value of D, modifying the class subregions so that more likely classes have larger subregions.

Case 1: $\kappa_{\Phi|1} = \kappa_{\Phi|2} = \kappa_\Phi$ and $\mu_{\Phi|1} \neq \mu_{\Phi|2}$. When the concentration parameter is the same in the two distributions, we have the following values for the constants:

$$a = \kappa_\Phi (\cos \mu_{\Phi|1} - \cos \mu_{\Phi|2}),$$
$$b = \kappa_\Phi (\sin \mu_{\Phi|1} - \sin \mu_{\Phi|1}),$$
$$D = -\ln \frac{p(C=1) I_0(\kappa_{\Phi|2})}{p(C=2) I_0(\kappa_{\Phi|1})} = -\ln 1 = 0.$$

Substituting in the expression of the arccosine, we get:

$$\arccos(D/T) = \arccos 0 = \pi/2.$$

To compute α, we take the trigonometric identities:

$$\cos\beta - \cos\gamma = -2\sin\left(\frac{1}{2}(\beta+\gamma)\right)\sin\left(\frac{1}{2}(\beta-\gamma)\right),$$
$$\sin\beta - \sin\gamma = 2\sin\left(\frac{1}{2}(\beta-\gamma)\right)\cos\left(\frac{1}{2}(\beta+\gamma)\right),$$

which we substitute in the following expression:

$$\begin{aligned}
\tan\alpha = \frac{b}{a} &= \frac{\kappa_\Phi(\sin\mu_{\Phi|1} - \sin\mu_{\Phi|2})}{\kappa_\Phi(\cos\mu_{\Phi|1} - \cos\mu_{\Phi|2})} \\
&= \frac{2\sin\left(\frac{1}{2}(\mu_{\Phi|1} - \mu_{\Phi|2})\right)\cos\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2})\right)}{-2\sin\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2})\right)\sin\left(\frac{1}{2}(\mu_{\Phi|1} - \mu_{\Phi|2})\right)} \\
&= -\frac{\cos\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2})\right)}{\sin\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2})\right)} \\
&= -\cot\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2})\right) \\
&= \tan\left(\frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}) + \frac{\pi}{2}\right).
\end{aligned}$$

Thus,

$$\alpha = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}) + \frac{\pi}{2}.$$

Now we can compute the decision angles found by the classifier:

$$\phi = \alpha \pm \arccos(D/T) = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}) + \frac{\pi}{2} \pm \frac{\pi}{2}.$$

The two decision angles are:

$$\phi' = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}),$$
$$\phi'' = \frac{1}{2}(\mu_{\Phi|1} + \mu_{\Phi|2}) + \pi.$$

These two angles correspond to the bisector angle of the two mean directions.

Case 2: $\kappa_{\Phi|1} \neq \kappa_{\Phi|2}$ and $\mu_{\Phi|1} = \mu_{\Phi|2} = \mu_\Phi$. In this scenario the mean directions are

equal, so the constants reduce to:

$$\begin{aligned}
a &= (\kappa_{\Phi|1} - \kappa_{\Phi|2}) \cos \mu_\Phi, \\
b &= (\kappa_{\Phi|1} - \kappa_{\Phi|2}) \sin \mu_\Phi, \\
D &= -\ln \frac{p(C=1) I_0(\kappa_{\Phi|2})}{p(C=2) I_0(\kappa_{\Phi|1})} = -\ln \frac{I_0(\kappa_{\Phi|2})}{I_0(\kappa_{\Phi|1})}, \\
T &= \sqrt{a^2 + b^2} \\
&= \sqrt{(\kappa_{\Phi|1} - \kappa_{\Phi|2})^2 \cos^2 \mu_\Phi + (\kappa_{\Phi|1} - \kappa_{\Phi|2})^2 \sin^2 \mu_\Phi} \\
&= \sqrt{(\kappa_{\Phi|1} - \kappa_{\Phi|2})^2 (\cos^2 \mu_\Phi + \sin^2 \mu_\Phi)} \\
&= \kappa_{\Phi|1} - \kappa_{\Phi|2}.
\end{aligned}$$

We compute α by substituting in the expression:

$$\begin{aligned}
\tan \alpha = \frac{b}{a} &= \frac{(\kappa_{\Phi|1} - \kappa_{\Phi|2}) \sin \mu_\Phi}{(\kappa_{\Phi|1} - \kappa_{\Phi|2}) \cos \mu_\Phi} = \tan \mu_\Phi, \\
\alpha &= \mu_\Phi.
\end{aligned}$$

Therefore, the resulting decision angles are given by:

$$\begin{aligned}
\phi &= \alpha \pm \arccos(D/T), \\
\phi' &= \mu_\Phi + \arccos \frac{D}{\kappa_{\Phi|1} - \kappa_{\Phi|2}}, \\
\phi'' &= \mu_\Phi - \arccos \frac{D}{\kappa_{\Phi|1} - \kappa_{\Phi|2}}.
\end{aligned}$$

Clearly, the two angles are defined with respect to the common mean direction, and their distance to that mean direction depends on the concentration parameter values.

A.2 vMNB with two predictive variables

In this scenario, we have two circular predictive variables Φ and Ψ. The domain defined by these variables is a torus $(-\pi, \pi] \times (-\pi, \pi]$. As in the simpler case above, we compute the decision surfaces induced by the classifier by equaling the posterior probability of the two class values

$$p(C=1|\Phi=\phi, \Psi=\psi) = p(C=2|\Phi=\phi, \Psi=\psi).$$

Using Bayes' rule and the conditional independence assumption, we get

$$\begin{aligned}
p(C=1) f_{\Phi|C=1}(\phi; \mu_{\Phi|1}, \kappa_{\Phi|1}) f_{\Psi|C=1}(\psi; \mu_{\Psi|1}, \kappa_{\Psi|1}) \\
= p(C=2) f_{\Phi|C=2}(\phi; \mu_{\Phi|2}, \kappa_{\Phi|2}) f_{\Psi|C=2}(\psi; \mu_{\Psi|2}, \kappa_{\Psi|2}).
\end{aligned}$$

We substitute the von Mises density (5.4) and get:

$$p(C=1)\frac{\exp(\kappa_{\Phi|1}\cos(\phi-\mu_{\Phi|1}))}{2\pi I_0(\kappa_{\Phi|1})}\frac{\exp(\kappa_{\Psi|1}\cos(\psi-\mu_{\Psi|1}))}{2\pi I_0(\kappa_{\Psi|1})} = p(C=2)\frac{\exp(\kappa_{\Phi|2}\cos(\phi-\mu_{\Phi|2}))}{2\pi I_0(\kappa_{\Phi|2})}\frac{\exp(\kappa_{\Psi|2}\cos(\psi-\mu_{\Psi|2}))}{2\pi I_0(\kappa_{\Psi|2})}.$$

We simplify the constant 2π, take logarithms and arrange all the terms on the same side of the equation:

$$\kappa_{\Phi|1}\cos(\phi-\mu_{\Phi|1})+\kappa_{\Psi|1}\cos(\psi-\mu_{\Psi|1})-\kappa_{\Phi|2}\cos(\phi-\mu_{\Phi|2})-\kappa_{\Psi|2}\cos(\psi-\mu_{\Psi|2}) + \ln\frac{p(C=1)I_0(\kappa_{\Phi|2})I_0(\kappa_{\Psi|2})}{p(C=2)I_0(\kappa_{\Phi|1})I_0(\kappa_{\Psi|1})}=0.$$

We substitute the trigonometric identity $\cos(\beta-\gamma)=\cos(\beta)\cos(\gamma)+\sin(\beta)\sin(\gamma)$ and arrange the terms:

$$(\kappa_{\Phi|1}\cos\mu_{\Phi|1}-\kappa_{\Phi|2}\cos\mu_{\Phi|2})\cos\phi+(\kappa_{\Phi|1}\sin\mu_{\Phi|1}-\kappa_{\Phi|2}\sin\mu_{\Phi|2})\sin\phi+ (\kappa_{\Psi|1}\cos\mu_{\Psi|1}-\kappa_{\Psi|2}\cos\mu_{\Psi|2})\cos\psi+(\kappa_{\Psi|1}\sin\mu_{\Psi|1}-\kappa_{\Psi|2}\sin\mu_{\Psi|2})\sin\psi+ \ln\frac{p(C=1)I_0(\kappa_{\Phi|2})I_0(\kappa_{\Psi|2})}{p(C=2)I_0(\kappa_{\Phi|1})I_0(\kappa_{\Psi|1})}=0.$$

We define the following constants:

$$\begin{aligned}
a &= \kappa_{\Phi|1}\cos\mu_{\Phi|1}-\kappa_{\Phi|2}\cos\mu_{\Phi|2},\\
b &= \kappa_{\Phi|1}\sin\mu_{\Phi|1}-\kappa_{\Phi|2}\sin\mu_{\Phi|2},\\
c &= \kappa_{\Psi|1}\cos\mu_{\Psi|1}-\kappa_{\Psi|2}\cos\mu_{\Psi|2},\\
d &= \kappa_{\Psi|1}\sin\mu_{\Psi|1}-\kappa_{\Psi|2}\sin\mu_{\Psi|2},\\
D &= -\ln\frac{p(C=1)I_0(\kappa_{\Phi|2})I_0(\kappa_{\Psi|2})}{p(C=2)I_0(\kappa_{\Phi|1})I_0(\kappa_{\Psi|1})},
\end{aligned}$$

and substitute them to get

$$a\cos\phi+b\sin\phi+c\cos\psi+d\sin\psi=D.$$

The Cartesian coordinates of the points defined by the angles ϕ and ψ on the surface of a torus are

$$\begin{aligned}
x &= (L+l\cos\phi)\cos\psi,\\
y &= (L+l\cos\phi)\sin\psi,\\
z &= l\sin\phi,
\end{aligned}$$

where L is the distance from the center of the torus to the center of the revolving circumference

that generates the torus, and l is the radius of the revolving circumference. We isolate the trigonometric functions and get

$$\begin{aligned}
&\sin\phi = z/l,\\
&\cos\phi = \pm\sqrt{1-\sin^2\phi} = \pm\sqrt{1-\left(\frac{z}{l}\right)^2} = \pm\frac{1}{l}\sqrt{l^2-z^2},\\
&\sin\psi = \frac{y}{L+l\cos\phi},\\
&\cos\psi = \frac{x}{L+l\cos\phi}.
\end{aligned}$$

Substituting these expressions, we get the two following equations corresponding to the two signs of $\cos\phi$:

$$\begin{aligned}
&\frac{a}{l}\sqrt{l^2-z^2} + \frac{b}{l}z + \frac{c}{L+\sqrt{l^2-z^2}}x + \frac{d}{L+\sqrt{l^2-z^2}}y + D = 0,\\
&-\frac{a}{l}\sqrt{l^2-z^2} + \frac{b}{l}z + \frac{c}{L-\sqrt{l^2-z^2}}x + \frac{d}{L-\sqrt{l^2-z^2}}y + D = 0.
\end{aligned}$$

Operating and arranging the terms, we get

$$\begin{aligned}
&clx + dly - az^2 + bz\sqrt{l^2-z^2} + bLz + (aL+Dl)\sqrt{l^2-z^2} + al^2 + DLl = 0,\\
&clx + dly - az^2 - bz\sqrt{l^2-z^2} + bLz - (aL+Dl)\sqrt{l^2-z^2} + al^2 + DLl = 0.
\end{aligned}$$

These expressions are quadratic in z. Therefore, we conclude that von Mises NB with two predictive variables is a much more complex and flexible classifier than von Mises NB with one predictive variable.

Appendix B

Von Mises-Fisher NB classifier decision function

To study the decision function for the von Mises-Fisher NB classifier we proceed as in Appendix A. We equal the posterior probabilities of the class values using the probability density function in Equation (5.8):

$$\begin{aligned} r(\mathbf{x}) = 0 \Leftrightarrow & p(C=1)\frac{(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1}}{\sqrt{(2\pi)^n} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} \exp(\kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}|1}^T\mathbf{x}) \\ = & p(C=2)\frac{(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1}}{\sqrt{(2\pi)^n} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})} \exp(\kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}|2}^T\mathbf{x}). \end{aligned}$$

Simplify the constants and take logarithms:

$$\ln \frac{p(C=1)(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1}}{I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} + \kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}|1}^T\mathbf{x} = \ln \frac{p(C=2)(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1}}{I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})} + \kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}|2}^T\mathbf{x}.$$

Arrange all the terms on the same side of the equation and operate the logarithms to get the following hyperplane equation:

$$(\kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}|2})^T\mathbf{x} + \ln \frac{p(C=1)(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{p(C=2)(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} = 0.$$

B.1 Particular cases

Considering that both class values have the same prior probability and that one of the parameters has the same value in both distributions, Case 1 and Case 2 can be simplified as follows. When the prior probabilities are different, the hyperplanes move away from the mean direction of the most likely class value, making their subregions larger.

Case 1: $\kappa_{\mathbf{X}|1} = \kappa_{\mathbf{X}|2} = \kappa_{\mathbf{X}}$ and $\boldsymbol{\mu}_{\mathbf{X}|1} \neq \boldsymbol{\mu}_{\mathbf{X}|2}$. When the distributions share the concentration parameter, we get the expression:

$$(\kappa_{\mathbf{X}}\boldsymbol{\mu}_{\mathbf{X}|1} - \kappa_{\mathbf{X}}\boldsymbol{\mu}_{\mathbf{X}|2})^T\mathbf{x} + \ln \frac{p(C=1)\kappa_{\mathbf{X}}^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}})}{p(C=2)\kappa_{\mathbf{X}}^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}})} = 0.$$

The logarithm reduces to 0 and we can take $\kappa_{\mathbf{X}}$ as common term:

$$\kappa_{\mathbf{X}}(\boldsymbol{\mu}_{\mathbf{X}|1} - \boldsymbol{\mu}_{\mathbf{X}|2})^T\mathbf{x} = 0.$$

Therefore, given that $\kappa > 0$ (otherwise the distributions are uniform), the hyperplane equation reduces to:

$$(\boldsymbol{\mu}_{\mathbf{X}|1} - \boldsymbol{\mu}_{\mathbf{X}|2})^T\mathbf{x} = 0.$$

That equation specifies a hyperplane that contains the origin point ($\mathbf{0}$) and goes through the middle point of the sector that connects the points of the hypersphere defined by the mean directions $\boldsymbol{\mu}_{\mathbf{X}|1}$ and $\boldsymbol{\mu}_{\mathbf{X}|2}$.

Case 2: $\kappa_{\mathbf{X}|1} \neq \kappa_{\mathbf{X}|2}$ and $\boldsymbol{\mu}_{\mathbf{X}|1} = \boldsymbol{\mu}_{\mathbf{X}|2} = \boldsymbol{\mu}_{\mathbf{X}}$. In the case where the mean directions have the same value, we can derive the following equation:

$$(\kappa_{\mathbf{X}|1}\boldsymbol{\mu}_{\mathbf{X}}^T - \kappa_{\mathbf{X}|2}\boldsymbol{\mu}_{\mathbf{X}}^T)\mathbf{x} + \ln \frac{p(C=1)(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{p(C=2)(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} = 0.$$

We can take $\boldsymbol{\mu}_{\mathbf{X}}^T$ as a common term:

$$(\kappa_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2})\boldsymbol{\mu}_{\mathbf{X}}^T\mathbf{x} + \ln \frac{(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} = 0.$$

Dividing by $(\kappa_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2})$, we get:

$$\boldsymbol{\mu}_{\mathbf{X}}^T\mathbf{x} + \frac{1}{\kappa_{\mathbf{X}|1} - \kappa_{\mathbf{X}|2}} \ln \frac{(\kappa_{\mathbf{X}|1})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|2})}{(\kappa_{\mathbf{X}|2})^{\frac{n}{2}-1} I_{\frac{n}{2}-1}(\kappa_{\mathbf{X}|1})} = 0.$$

The hyperplane defined by this equation is perpendicular to the shared mean direction vector $\boldsymbol{\mu}_{\mathbf{X}}$, and its position is given by the relationships between the concentration parameters.

Appendix C

Results of the Bayesian classifiers with class label counts

This appendix includes the detailed results of the experiments reported in Chapter 12. The BNCs with uncertain class labels were evaluated over 16 datasets taken from the UCI [27], KEEL [9] and LibSVM [78] repositories (see Table 12.5). We generated ten datasets with experts' class labels and applied the stratified 10-fold cross-validation procedure described in Section 12.3.1. The parameters of the beta distributions used for generating the datasets were: $\mu_B \in \{0.1, 0.2, 0.3, 0.4\}$ and $\sigma_B = 0.01$. We considered three classifiers (MG, NB and AODE) and induced them with the studied algorithms and classification rules (see Section 12.1): CoMEM-IND, CoMEM-MUL, CoMEM-BIN, PLEM and EM. The tables in this appendix report the mean and standard deviation of the accuracy and the mean squared error computed over the 10 repetitions of the 10-fold stratified cross-validation procedure.

Table C.1: Mean classification error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.1, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	**0.1770** ± 0.1038	**0.1770** ± 0.1038	0.1790 ± 0.1026	0.1791 ± 0.1034	0.2515 ± 0.1878
fourclass	**0.2063** ± 0.0407	**0.2063** ± 0.0407	0.2064 ± 0.0407	0.2074 ± 0.0418	0.2669 ± 0.0401
glass2	0.0936 ± 0.0766	0.0936 ± 0.0766	0.0940 ± 0.0762	**0.0911** ± 0.0714	0.1034 ± 0.0700
haberman	0.2479 ± 0.0528	0.2479 ± 0.0528	**0.2475** ± 0.0534	0.2492 ± 0.0607	0.3334 ± 0.0840
ion	0.1245 ± 0.0614	0.1245 ± 0.0614	**0.1240** ± 0.0617	0.1302 ± 0.0655	0.1402 ± 0.0658
iris	0.0274 ± 0.0392	0.0274 ± 0.0392	**0.0266** ± 0.0391	0.0286 ± 0.0394	0.0355 ± 0.0443
liver	**0.3937** ± 0.0810	**0.3937** ± 0.0810	**0.3937** ± 0.0813	0.4565 ± 0.0835	0.5045 ± 0.0686
newthyroid	0.0381 ± 0.0382	0.0381 ± 0.0382	0.0376 ± 0.0377	**0.0371** ± 0.0384	0.0409 ± 0.0392
phoneme	**0.2134** ± 0.0165	**0.2134** ± 0.0165	0.2134 ± 0.0165	0.2212 ± 0.0171	0.3008 ± 0.0211
ring	0.0206 ± 0.0045	0.0206 ± 0.0045	0.0206 ± 0.0045	0.0207 ± 0.0044	**0.0206** ± 0.0046
svmguide1	**0.0578** ± 0.0089	**0.0578** ± 0.0089	**0.0578** ± 0.0089	0.0621 ± 0.0090	0.2205 ± 0.0129
twonorm	**0.0221** ± 0.0046	**0.0221** ± 0.0046	**0.0221** ± 0.0046	0.0225 ± 0.0050	0.0235 ± 0.0051
vehicle	0.1466 ± 0.0358	0.1455 ± 0.0368	**0.1454** ± 0.0368	0.1550 ± 0.0383	0.4366 ± 0.0728
waveform	**0.1488** ± 0.0151	0.1488 ± 0.0150	0.1488 ± 0.0149	0.1498 ± 0.0150	0.1620 ± 0.0148
wdbc	**0.0441** ± 0.0290	**0.0441** ± 0.0290	**0.0441** ± 0.0290	0.0481 ± 0.0306	0.0550 ± 0.0294
wine	0.0097 ± 0.0245	0.0102 ± 0.0249	0.0102 ± 0.0249	0.0102 ± 0.0249	**0.0096** ± 0.0245
			NB		
appendicitis	0.1513 ± 0.0906	0.1513 ± 0.0906	0.1550 ± 0.0946	**0.1392** ± 0.0900	0.1422 ± 0.0971
fourclass	0.2446 ± 0.0452	0.2446 ± 0.0452	0.2447 ± 0.0454	**0.2443** ± 0.0442	0.2689 ± 0.0331
glass2	**0.0944** ± 0.0725	**0.0944** ± 0.0725	0.0958 ± 0.0726	0.0997 ± 0.0759	0.2231 ± 0.0900
haberman	0.2485 ± 0.0519	0.2485 ± 0.0519	0.2485 ± 0.0517	**0.2465** ± 0.0589	0.3334 ± 0.0840
ion	0.2579 ± 0.0869	0.2579 ± 0.0869	0.2591 ± 0.0858	**0.2445** ± 0.0675	0.3201 ± 0.0703
iris	**0.0432** ± 0.0498	0.0438 ± 0.0523	0.0438 ± 0.0523	0.0465 ± 0.0517	0.0736 ± 0.0666
liver	0.4412 ± 0.0935	0.4412 ± 0.0935	**0.4409** ± 0.0930	0.5095 ± 0.0729	0.5071 ± 0.0684
newthyroid	0.0358 ± 0.0373	0.0349 ± 0.0347	0.0349 ± 0.0347	0.0330 ± 0.0332	**0.0311** ± 0.0344
phoneme	**0.2400** ± 0.0186	**0.2400** ± 0.0186	0.2400 ± 0.0185	0.2637 ± 0.0209	0.3491 ± 0.0220
ring	0.0204 ± 0.0046	0.0204 ± 0.0046	0.0204 ± 0.0046	0.0203 ± 0.0046	**0.0201** ± 0.0045
svmguide1	**0.0631** ± 0.0088	**0.0631** ± 0.0088	**0.0631** ± 0.0088	0.0674 ± 0.0092	0.1186 ± 0.0107
twonorm	**0.0213** ± 0.0049	**0.0213** ± 0.0049	**0.0213** ± 0.0049	0.0214 ± 0.0050	0.0215 ± 0.0050
vehicle	0.5148 ± 0.0570	**0.5141** ± 0.0570	0.5145 ± 0.0569	0.5963 ± 0.0455	0.5678 ± 0.0629
waveform	**0.1908** ± 0.0153	0.1909 ± 0.0153	0.1909 ± 0.0153	0.2020 ± 0.0149	0.2550 ± 0.0167
wdbc	0.0859 ± 0.0365	0.0859 ± 0.0365	0.0859 ± 0.0363	**0.0808** ± 0.0348	0.0885 ± 0.0362
wine	**0.0214** ± 0.0297	0.0219 ± 0.0309	0.0219 ± 0.0309	0.0247 ± 0.0331	0.0355 ± 0.0374
			AODE		
appendicitis	0.1435 ± 0.0921	0.1435 ± 0.0921	**0.1435** ± 0.0929	0.2614 ± 0.1266	0.4543 ± 0.1582
fourclass	**0.2063** ± 0.0407	**0.2063** ± 0.0407	0.2064 ± 0.0407	0.2074 ± 0.0418	0.2669 ± 0.0401
glass2	0.0856 ± 0.0584	0.0856 ± 0.0584	0.0875 ± 0.0597	**0.0855** ± 0.0658	0.0974 ± 0.0660
haberman	**0.2476** ± 0.0529	**0.2476** ± 0.0529	0.2479 ± 0.0527	0.2478 ± 0.0597	0.3334 ± 0.0840
ion	0.2317 ± 0.0653	0.2317 ± 0.0653	0.2320 ± 0.0665	**0.1088** ± 0.0567	0.1365 ± 0.0553
iris	0.0241 ± 0.0377	0.0220 ± 0.0343	0.0220 ± 0.0343	**0.0200** ± 0.0322	0.0342 ± 0.0434
liver	**0.4410** ± 0.0953	**0.4410** ± 0.0953	0.4416 ± 0.0955	0.4981 ± 0.0697	0.5042 ± 0.0675
newthyroid	0.0381 ± 0.0363	0.0377 ± 0.0366	0.0372 ± 0.0368	0.0367 ± 0.0369	**0.0358** ± 0.0366
phoneme	0.2216 ± 0.0173	0.2216 ± 0.0173	**0.2215** ± 0.0173	0.2550 ± 0.0217	0.3137 ± 0.0219
ring	0.0203 ± 0.0046	0.0203 ± 0.0046	0.0203 ± 0.0046	0.0203 ± 0.0046	**0.0202** ± 0.0046
svmguide1	0.0588 ± 0.0087	0.0588 ± 0.0087	**0.0588** ± 0.0087	0.0638 ± 0.0090	0.1950 ± 0.0127
twonorm	**0.0213** ± 0.0049	**0.0213** ± 0.0049	**0.0213** ± 0.0049	0.0215 ± 0.0049	0.0217 ± 0.0050
vehicle	**0.3009** ± 0.0464	0.3092 ± 0.0482	0.3102 ± 0.0479	0.3508 ± 0.0441	0.5535 ± 0.0453
waveform	0.1558 ± 0.0135	0.1560 ± 0.0136	0.1560 ± 0.0136	**0.1489** ± 0.0161	0.2139 ± 0.0169
wdbc	**0.0669** ± 0.0337	**0.0669** ± 0.0337	**0.0669** ± 0.0337	0.0956 ± 0.0378	0.1603 ± 0.0506
wine	0.0158 ± 0.0268	0.0152 ± 0.0265	0.0146 ± 0.0262	**0.0107** ± 0.0238	0.0224 ± 0.0288

Table C.2: Mean squared error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.1, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	**0.1382** ± 0.0749	**0.1382** ± 0.0749	0.1394 ± 0.0754	0.1430 ± 0.0729	0.2026 ± 0.1419
fourclass	0.1034 ± 0.0155	0.1034 ± 0.0155	**0.1034** ± 0.0155	0.1089 ± 0.0171	0.2013 ± 0.0318
glass2	**0.0872** ± 0.0571	**0.0872** ± 0.0571	0.0874 ± 0.0573	0.0873 ± 0.0571	0.0984 ± 0.0566
haberman	0.1477 ± 0.0361	0.1477 ± 0.0361	**0.1474** ± 0.0360	0.1706 ± 0.0456	0.2708 ± 0.0657
ion	0.1152 ± 0.0520	0.1152 ± 0.0520	**0.1150** ± 0.0520	0.1198 ± 0.0540	0.1273 ± 0.0538
iris	0.0206 ± 0.0161	0.0205 ± 0.0160	**0.0204** ± 0.0157	0.0206 ± 0.0158	0.0268 ± 0.0216
liver	0.1702 ± 0.0346	0.1702 ± 0.0346	**0.1702** ± 0.0346	0.2495 ± 0.0630	0.3956 ± 0.0565
newthyroid	**0.0284** ± 0.0167	0.0284 ± 0.0168	0.0284 ± 0.0168	0.0289 ± 0.0169	0.0308 ± 0.0185
phoneme	**0.1360** ± 0.0103	**0.1360** ± 0.0103	0.1361 ± 0.0103	0.1646 ± 0.0123	0.2449 ± 0.0165
ring	0.0291 ± 0.0029	0.0291 ± 0.0029	**0.0291** ± 0.0029	0.0292 ± 0.0029	0.0292 ± 0.0029
svmguide1	0.0471 ± 0.0050	0.0471 ± 0.0050	**0.0471** ± 0.0050	0.0502 ± 0.0051	0.1574 ± 0.0091
twonorm	**0.0291** ± 0.0027	**0.0291** ± 0.0027	0.0291 ± 0.0027	0.0292 ± 0.0028	0.0295 ± 0.0027
vehicle	0.0510 ± 0.0108	0.0510 ± 0.0109	**0.0510** ± 0.0109	0.0552 ± 0.0127	0.1943 ± 0.0315
waveform	0.0581 ± 0.0042	0.0581 ± 0.0042	**0.0581** ± 0.0042	0.0585 ± 0.0043	0.0635 ± 0.0046
wdbc	**0.0499** ± 0.0212	**0.0499** ± 0.0212	0.0499 ± 0.0212	0.0532 ± 0.0229	0.0572 ± 0.0224
wine	0.0142 ± 0.0087	0.0142 ± 0.0085	**0.0141** ± 0.0085	0.0142 ± 0.0086	0.0145 ± 0.0089
			NB		
appendicitis	0.1240 ± 0.0673	0.1240 ± 0.0673	0.1253 ± 0.0676	**0.1209** ± 0.0665	0.1233 ± 0.0741
fourclass	**0.1143** ± 0.0161	**0.1143** ± 0.0161	0.1143 ± 0.0161	0.1209 ± 0.0185	0.2173 ± 0.0234
glass2	0.0894 ± 0.0557	0.0894 ± 0.0557	0.0902 ± 0.0563	**0.0892** ± 0.0582	0.1911 ± 0.0713
haberman	0.1503 ± 0.0366	0.1503 ± 0.0366	**0.1499** ± 0.0365	0.1716 ± 0.0448	0.2721 ± 0.0658
ion	0.2188 ± 0.0742	0.2188 ± 0.0742	0.2194 ± 0.0742	**0.2114** ± 0.0559	0.2727 ± 0.0582
iris	0.0302 ± 0.0238	0.0302 ± 0.0238	**0.0301** ± 0.0237	0.0307 ± 0.0239	0.0387 ± 0.0271
liver	0.1883 ± 0.0374	0.1883 ± 0.0374	**0.1882** ± 0.0374	0.3443 ± 0.0577	0.4001 ± 0.0534
newthyroid	0.0271 ± 0.0162	**0.0271** ± 0.0162	0.0271 ± 0.0162	0.0272 ± 0.0162	0.0273 ± 0.0168
phoneme	**0.1476** ± 0.0107	**0.1476** ± 0.0107	0.1476 ± 0.0107	0.1729 ± 0.0127	0.2696 ± 0.0171
ring	0.0288 ± 0.0028	0.0288 ± 0.0028	**0.0288** ± 0.0028	0.0289 ± 0.0029	0.0290 ± 0.0029
svmguide1	0.0516 ± 0.0052	0.0516 ± 0.0052	**0.0516** ± 0.0052	0.0560 ± 0.0055	0.0878 ± 0.0067
twonorm	**0.0289** ± 0.0027	**0.0289** ± 0.0027	0.0289 ± 0.0027	0.0289 ± 0.0027	0.0289 ± 0.0027
vehicle	**0.1699** ± 0.0197	0.1713 ± 0.0193	0.1716 ± 0.0192	0.2314 ± 0.0205	0.2485 ± 0.0269
waveform	**0.1008** ± 0.0072	0.1009 ± 0.0072	0.1009 ± 0.0072	0.1100 ± 0.0073	0.1454 ± 0.0087
wdbc	0.0841 ± 0.0284	0.0841 ± 0.0284	0.0841 ± 0.0283	**0.0801** ± 0.0270	0.0867 ± 0.0282
wine	**0.0209** ± 0.0143	0.0209 ± 0.0144	0.0209 ± 0.0144	0.0215 ± 0.0145	0.0260 ± 0.0168
			AODE		
appendicitis	**0.1073** ± 0.0612	**0.1073** ± 0.0612	0.1078 ± 0.0612	0.1726 ± 0.0799	0.2821 ± 0.0952
fourclass	0.1034 ± 0.0155	0.1034 ± 0.0155	**0.1034** ± 0.0155	0.1089 ± 0.0171	0.2013 ± 0.0318
glass2	**0.0687** ± 0.0410	**0.0687** ± 0.0410	0.0690 ± 0.0410	0.0766 ± 0.0492	0.0908 ± 0.0504
haberman	0.1481 ± 0.0361	0.1481 ± 0.0361	**0.1478** ± 0.0360	0.1709 ± 0.0454	0.2712 ± 0.0658
ion	0.1220 ± 0.0378	0.1220 ± 0.0378	0.1226 ± 0.0378	**0.1000** ± 0.0427	0.1249 ± 0.0467
iris	0.0190 ± 0.0128	0.0190 ± 0.0127	0.0189 ± 0.0127	**0.0188** ± 0.0140	0.0227 ± 0.0175
liver	0.1796 ± 0.0350	0.1796 ± 0.0350	**0.1796** ± 0.0350	0.2956 ± 0.0612	0.3950 ± 0.0545
newthyroid	0.0276 ± 0.0163	0.0276 ± 0.0163	**0.0276** ± 0.0163	0.0283 ± 0.0170	0.0283 ± 0.0176
phoneme	**0.1412** ± 0.0105	**0.1412** ± 0.0105	0.1412 ± 0.0105	0.1705 ± 0.0127	0.2344 ± 0.0157
ring	0.0288 ± 0.0028	0.0288 ± 0.0028	**0.0288** ± 0.0028	0.0289 ± 0.0029	0.0290 ± 0.0029
svmguide1	0.0480 ± 0.0050	0.0480 ± 0.0050	**0.0480** ± 0.0050	0.0517 ± 0.0053	0.1343 ± 0.0085
twonorm	**0.0289** ± 0.0027	**0.0289** ± 0.0027	0.0289 ± 0.0027	0.0289 ± 0.0027	0.0289 ± 0.0027
vehicle	**0.0838** ± 0.0098	0.0847 ± 0.0099	0.0849 ± 0.0099	0.1153 ± 0.0161	0.1999 ± 0.0189
waveform	**0.0594** ± 0.0041	0.0594 ± 0.0040	0.0594 ± 0.0040	0.0599 ± 0.0045	0.0777 ± 0.0046
wdbc	0.0472 ± 0.0151	0.0472 ± 0.0151	**0.0471** ± 0.0151	0.0871 ± 0.0277	0.1352 ± 0.0390
wine	0.0164 ± 0.0095	0.0164 ± 0.0096	0.0164 ± 0.0096	**0.0160** ± 0.0105	0.0195 ± 0.0123

Table C.3: Mean classification error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.2, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	0.1930 ± 0.1125	0.1930 ± 0.1125	0.1950 ± 0.1148	**0.1928** ± 0.1095	0.2655 ± 0.1504
fourclass	**0.2070** ± 0.0433	**0.2070** ± 0.0433	0.2072 ± 0.0436	0.2298 ± 0.0473	0.2749 ± 0.0546
glass2	0.1059 ± 0.0646	0.1059 ± 0.0646	0.1069 ± 0.0647	**0.0971** ± 0.0642	0.1018 ± 0.0622
haberman	0.2693 ± 0.0616	0.2693 ± 0.0616	0.2709 ± 0.0619	**0.2555** ± 0.0639	0.3334 ± 0.0840
ion	**0.1266** ± 0.0518	**0.1266** ± 0.0518	**0.1266** ± 0.0518	0.1314 ± 0.0518	0.1379 ± 0.0513
iris	0.0286 ± 0.0446	0.0286 ± 0.0446	**0.0280** ± 0.0445	0.0326 ± 0.0476	0.0473 ± 0.0648
liver	**0.4088** ± 0.0719	**0.4088** ± 0.0719	0.4091 ± 0.0719	0.4964 ± 0.0731	0.5055 ± 0.0638
newthyroid	0.0382 ± 0.0370	**0.0377** ± 0.0372	**0.0377** ± 0.0372	0.0391 ± 0.0378	0.0414 ± 0.0397
phoneme	**0.2190** ± 0.0193	**0.2190** ± 0.0193	0.2199 ± 0.0186	0.2268 ± 0.0196	0.3009 ± 0.0207
ring	0.0209 ± 0.0044	0.0209 ± 0.0044	0.0209 ± 0.0044	0.0211 ± 0.0043	**0.0208** ± 0.0044
svmguide1	0.0576 ± 0.0072	0.0576 ± 0.0072	**0.0576** ± 0.0073	0.0675 ± 0.0083	0.2265 ± 0.0107
twonorm	0.0223 ± 0.0053	0.0223 ± 0.0053	**0.0223** ± 0.0053	0.0226 ± 0.0055	0.0232 ± 0.0056
vehicle	**0.1496** ± 0.0342	0.1497 ± 0.0334	0.1497 ± 0.0332	0.1680 ± 0.0360	0.4365 ± 0.0788
waveform	**0.1472** ± 0.0155	**0.1472** ± 0.0155	0.1472 ± 0.0156	0.1502 ± 0.0150	0.1602 ± 0.0146
wdbc	**0.0494** ± 0.0288	**0.0494** ± 0.0288	0.0496 ± 0.0286	0.0499 ± 0.0293	0.0543 ± 0.0294
wine	**0.0089** ± 0.0206	**0.0089** ± 0.0206	**0.0089** ± 0.0206	**0.0089** ± 0.0206	0.0095 ± 0.0211
			NB		
appendicitis	0.1417 ± 0.0957	0.1417 ± 0.0957	0.1437 ± 0.0985	**0.1350** ± 0.1010	0.1809 ± 0.1032
fourclass	**0.2453** ± 0.0470	**0.2453** ± 0.0470	0.2457 ± 0.0467	0.2557 ± 0.0442	0.2701 ± 0.0379
glass2	0.1683 ± 0.0889	0.1683 ± 0.0889	0.1692 ± 0.0894	**0.1378** ± 0.0769	0.2246 ± 0.0739
haberman	0.2725 ± 0.0653	0.2725 ± 0.0653	0.2735 ± 0.0647	**0.2558** ± 0.0631	0.3334 ± 0.0840
ion	0.2976 ± 0.0682	0.2976 ± 0.0682	0.2996 ± 0.0673	**0.2592** ± 0.0706	0.3171 ± 0.0736
iris	0.0476 ± 0.0530	**0.0470** ± 0.0531	**0.0470** ± 0.0531	0.0491 ± 0.0544	0.0766 ± 0.0572
liver	**0.4788** ± 0.0769	**0.4788** ± 0.0769	0.4788 ± 0.0769	0.5029 ± 0.0743	0.5041 ± 0.0674
newthyroid	0.0372 ± 0.0362	0.0372 ± 0.0362	0.0376 ± 0.0360	0.0372 ± 0.0369	**0.0339** ± 0.0341
phoneme	**0.2390** ± 0.0183	**0.2390** ± 0.0183	0.2422 ± 0.0186	0.2790 ± 0.0197	0.3490 ± 0.0245
ring	0.0203 ± 0.0044	0.0203 ± 0.0044	0.0203 ± 0.0044	**0.0200** ± 0.0042	0.0201 ± 0.0043
svmguide1	**0.0622** ± 0.0075	**0.0622** ± 0.0075	**0.0622** ± 0.0075	0.0709 ± 0.0080	0.1185 ± 0.0088
twonorm	0.0215 ± 0.0052	0.0215 ± 0.0052	0.0215 ± 0.0052	**0.0215** ± 0.0053	0.0216 ± 0.0052
vehicle	**0.4983** ± 0.0419	0.5109 ± 0.0392	0.5146 ± 0.0395	0.5657 ± 0.0544	0.5597 ± 0.0572
waveform	**0.1921** ± 0.0156	**0.1921** ± 0.0156	0.1921 ± 0.0156	0.2109 ± 0.0152	0.2567 ± 0.0177
wdbc	0.0880 ± 0.0348	0.0880 ± 0.0348	0.0880 ± 0.0348	**0.0854** ± 0.0332	0.0882 ± 0.0343
wine	0.0253 ± 0.0312	**0.0247** ± 0.0311	**0.0247** ± 0.0311	0.0303 ± 0.0349	0.0359 ± 0.0377
			AODE		
appendicitis	**0.1607** ± 0.1062	**0.1607** ± 0.1062	0.1636 ± 0.1065	0.3108 ± 0.1472	0.4338 ± 0.1406
fourclass	**0.2070** ± 0.0433	**0.2070** ± 0.0433	0.2072 ± 0.0436	0.2298 ± 0.0473	0.2749 ± 0.0546
glass2	0.0958 ± 0.0587	0.0958 ± 0.0587	0.0958 ± 0.0587	**0.0874** ± 0.0559	0.0949 ± 0.0606
haberman	0.2699 ± 0.0606	0.2699 ± 0.0606	0.2709 ± 0.0602	**0.2552** ± 0.0637	0.3334 ± 0.0840
ion	0.2543 ± 0.0650	0.2543 ± 0.0650	0.2552 ± 0.0653	**0.1194** ± 0.0522	0.1459 ± 0.0635
iris	**0.0226** ± 0.0391	**0.0226** ± 0.0391	**0.0226** ± 0.0391	0.0245 ± 0.0405	0.0531 ± 0.0593
liver	**0.4636** ± 0.0771	**0.4636** ± 0.0771	0.4642 ± 0.0769	0.5017 ± 0.0710	0.5015 ± 0.0676
newthyroid	0.0386 ± 0.0381	0.0396 ± 0.0394	0.0396 ± 0.0394	0.0391 ± 0.0389	**0.0358** ± 0.0366
phoneme	**0.2238** ± 0.0186	**0.2238** ± 0.0186	0.2277 ± 0.0190	0.2844 ± 0.0199	0.3179 ± 0.0234
ring	0.0204 ± 0.0044	0.0204 ± 0.0044	0.0204 ± 0.0044	**0.0201** ± 0.0043	0.0203 ± 0.0044
svmguide1	**0.0586** ± 0.0073	**0.0586** ± 0.0073	0.0587 ± 0.0073	0.0678 ± 0.0084	0.1973 ± 0.0108
twonorm	0.0215 ± 0.0051	0.0215 ± 0.0051	0.0215 ± 0.0051	**0.0215** ± 0.0051	0.0217 ± 0.0051
vehicle	**0.2885** ± 0.0429	0.2949 ± 0.0438	0.2959 ± 0.0449	0.3913 ± 0.0486	0.5595 ± 0.0579
waveform	0.1564 ± 0.0140	0.1565 ± 0.0140	0.1565 ± 0.0139	**0.1560** ± 0.0162	0.2149 ± 0.0178
wdbc	**0.0801** ± 0.0360	**0.0801** ± 0.0360	**0.0801** ± 0.0364	0.1172 ± 0.0415	0.1738 ± 0.0405
wine	0.0168 ± 0.0269	0.0157 ± 0.0264	0.0157 ± 0.0264	**0.0151** ± 0.0261	0.0258 ± 0.0302

Table C.4: Mean squared error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.2, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	**0.1581** ± 0.0675	**0.1581** ± 0.0675	0.1591 ± 0.0680	0.1604 ± 0.0675	0.1953 ± 0.0882
fourclass	0.0856 ± 0.0130	0.0856 ± 0.0130	**0.0856** ± 0.0130	0.1068 ± 0.0175	0.1863 ± 0.0320
glass2	0.1136 ± 0.0423	0.1136 ± 0.0423	0.1137 ± 0.0424	**0.1091** ± 0.0410	0.1138 ± 0.0434
haberman	0.1724 ± 0.0409	0.1724 ± 0.0409	**0.1697** ± 0.0413	0.1781 ± 0.0370	0.2385 ± 0.0559
ion	0.1302 ± 0.0367	0.1302 ± 0.0367	**0.1302** ± 0.0366	0.1337 ± 0.0376	0.1370 ± 0.0372
iris	0.0387 ± 0.0149	0.0386 ± 0.0148	**0.0386** ± 0.0148	0.0401 ± 0.0168	0.0460 ± 0.0246
liver	**0.1430** ± 0.0315	**0.1430** ± 0.0315	0.1433 ± 0.0316	0.3043 ± 0.0455	0.3369 ± 0.0433
newthyroid	0.0446 ± 0.0166	0.0445 ± 0.0165	**0.0445** ± 0.0165	0.0451 ± 0.0168	0.0467 ± 0.0178
phoneme	**0.1387** ± 0.0100	**0.1387** ± 0.0100	0.1396 ± 0.0100	0.1684 ± 0.0111	0.2219 ± 0.0134
ring	**0.0607** ± 0.0028	**0.0607** ± 0.0028	0.0607 ± 0.0028	0.0607 ± 0.0028	0.0607 ± 0.0029
svmguide1	0.0688 ± 0.0036	0.0688 ± 0.0036	**0.0688** ± 0.0036	0.0737 ± 0.0039	0.1534 ± 0.0065
twonorm	0.0592 ± 0.0030	0.0592 ± 0.0030	**0.0592** ± 0.0030	0.0595 ± 0.0030	0.0594 ± 0.0030
vehicle	0.0559 ± 0.0089	**0.0558** ± 0.0089	0.0558 ± 0.0089	0.0647 ± 0.0112	0.1775 ± 0.0291
waveform	**0.0598** ± 0.0042	0.0598 ± 0.0042	0.0598 ± 0.0042	0.0610 ± 0.0043	0.0648 ± 0.0044
wdbc	**0.0819** ± 0.0175	**0.0819** ± 0.0175	0.0819 ± 0.0175	0.0824 ± 0.0178	0.0843 ± 0.0174
wine	**0.0329** ± 0.0085	0.0329 ± 0.0085	0.0329 ± 0.0085	0.0329 ± 0.0085	0.0331 ± 0.0088
			NB		
appendicitis	0.1360 ± 0.0628	0.1360 ± 0.0628	0.1365 ± 0.0629	**0.1310** ± 0.0636	0.1531 ± 0.0665
fourclass	**0.0918** ± 0.0139	**0.0918** ± 0.0139	0.0918 ± 0.0139	0.1225 ± 0.0206	0.1987 ± 0.0199
glass2	0.1513 ± 0.0565	0.1513 ± 0.0565	0.1518 ± 0.0568	**0.1293** ± 0.0515	0.1880 ± 0.0490
haberman	0.1756 ± 0.0414	0.1756 ± 0.0414	**0.1729** ± 0.0417	0.1792 ± 0.0368	0.2395 ± 0.0558
ion	0.2322 ± 0.0429	0.2322 ± 0.0429	0.2326 ± 0.0428	**0.2076** ± 0.0463	0.2436 ± 0.0441
iris	0.0454 ± 0.0179	**0.0453** ± 0.0178	0.0453 ± 0.0179	0.0459 ± 0.0176	0.0533 ± 0.0210
liver	0.2144 ± 0.0685	0.2144 ± 0.0685	**0.2142** ± 0.0683	0.3282 ± 0.0472	0.3388 ± 0.0450
newthyroid	0.0435 ± 0.0157	**0.0435** ± 0.0157	0.0435 ± 0.0157	0.0436 ± 0.0158	0.0435 ± 0.0159
phoneme	**0.1380** ± 0.0097	**0.1380** ± 0.0097	0.1386 ± 0.0098	0.1843 ± 0.0122	0.2368 ± 0.0153
ring	**0.0604** ± 0.0028	**0.0604** ± 0.0028	0.0604 ± 0.0028	0.0605 ± 0.0029	0.0606 ± 0.0028
svmguide1	0.0737 ± 0.0038	0.0737 ± 0.0038	**0.0737** ± 0.0038	0.0790 ± 0.0042	0.1005 ± 0.0047
twonorm	0.0590 ± 0.0030	0.0590 ± 0.0030	**0.0590** ± 0.0030	0.0590 ± 0.0030	0.0590 ± 0.0030
vehicle	**0.1717** ± 0.0148	0.1796 ± 0.0141	0.1813 ± 0.0140	0.2057 ± 0.0208	0.2229 ± 0.0217
waveform	**0.1027** ± 0.0062	0.1027 ± 0.0062	0.1027 ± 0.0062	0.1155 ± 0.0065	0.1405 ± 0.0080
wdbc	0.1081 ± 0.0236	0.1081 ± 0.0236	0.1082 ± 0.0237	**0.1051** ± 0.0223	0.1078 ± 0.0233
wine	**0.0389** ± 0.0137	0.0389 ± 0.0137	0.0389 ± 0.0137	0.0400 ± 0.0138	0.0417 ± 0.0145
			AODE		
appendicitis	**0.1075** ± 0.0508	**0.1075** ± 0.0508	0.1079 ± 0.0509	0.1803 ± 0.0785	0.2316 ± 0.0754
fourclass	0.0856 ± 0.0130	0.0856 ± 0.0130	**0.0856** ± 0.0130	0.1068 ± 0.0175	0.1863 ± 0.0320
glass2	**0.0941** ± 0.0352	**0.0941** ± 0.0352	0.0942 ± 0.0353	0.1006 ± 0.0376	0.1064 ± 0.0390
haberman	0.1726 ± 0.0400	0.1726 ± 0.0400	**0.1699** ± 0.0404	0.1784 ± 0.0369	0.2388 ± 0.0559
ion	0.1446 ± 0.0321	0.1446 ± 0.0321	0.1449 ± 0.0322	**0.1258** ± 0.0353	0.1403 ± 0.0412
iris	0.0367 ± 0.0130	**0.0367** ± 0.0129	0.0367 ± 0.0129	0.0379 ± 0.0148	0.0417 ± 0.0167
liver	**0.1620** ± 0.0358	**0.1620** ± 0.0358	0.1621 ± 0.0359	0.3147 ± 0.0469	0.3346 ± 0.0443
newthyroid	0.0438 ± 0.0160	0.0438 ± 0.0159	**0.0438** ± 0.0159	0.0445 ± 0.0163	0.0443 ± 0.0165
phoneme	**0.1347** ± 0.0097	**0.1347** ± 0.0097	0.1357 ± 0.0098	0.1910 ± 0.0123	0.2084 ± 0.0130
ring	**0.0604** ± 0.0028	**0.0604** ± 0.0028	0.0604 ± 0.0028	0.0605 ± 0.0028	0.0606 ± 0.0028
svmguide1	0.0698 ± 0.0036	0.0698 ± 0.0036	**0.0698** ± 0.0036	0.0749 ± 0.0039	0.1341 ± 0.0061
twonorm	0.0590 ± 0.0030	0.0590 ± 0.0030	**0.0590** ± 0.0030	0.0591 ± 0.0030	0.0590 ± 0.0030
vehicle	**0.0706** ± 0.0089	0.0716 ± 0.0090	0.0718 ± 0.0090	0.1222 ± 0.0151	0.1803 ± 0.0207
waveform	**0.0586** ± 0.0038	0.0586 ± 0.0038	0.0586 ± 0.0038	0.0611 ± 0.0043	0.0673 ± 0.0037
wdbc	0.0716 ± 0.0138	0.0716 ± 0.0138	**0.0716** ± 0.0138	0.1194 ± 0.0267	0.1499 ± 0.0270
wine	0.0345 ± 0.0103	**0.0345** ± 0.0103	0.0345 ± 0.0103	0.0351 ± 0.0110	0.0381 ± 0.0130

Table C.5: Mean classification error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.3, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	0.2534 ± 0.1449	0.2534 ± 0.1449	0.2534 ± 0.1449	**0.2121** ± 0.1217	0.2935 ± 0.1781
fourclass	**0.2135** ± 0.0396	**0.2135** ± 0.0396	**0.2135** ± 0.0398	0.2581 ± 0.0433	0.3482 ± 0.0822
glass2	0.1082 ± 0.0570	0.1082 ± 0.0570	0.1082 ± 0.0570	**0.1029** ± 0.0577	0.1076 ± 0.0595
haberman	0.3335 ± 0.0811	0.3335 ± 0.0811	0.3335 ± 0.0806	**0.2761** ± 0.0836	0.3332 ± 0.0813
ion	**0.1310** ± 0.0468	**0.1310** ± 0.0468	**0.1310** ± 0.0468	0.1329 ± 0.0466	0.1401 ± 0.0478
iris	0.0278 ± 0.0401	**0.0271** ± 0.0400	**0.0271** ± 0.0400	0.0278 ± 0.0380	0.1126 ± 0.0960
liver	0.4988 ± 0.0759	0.4988 ± 0.0759	0.4991 ± 0.0758	**0.4885** ± 0.0797	0.5032 ± 0.0713
newthyroid	0.0367 ± 0.0381	**0.0362** ± 0.0383	0.0367 ± 0.0381	0.0399 ± 0.0391	0.0413 ± 0.0427
phoneme	**0.2877** ± 0.0334	**0.2877** ± 0.0334	0.2890 ± 0.0328	0.2902 ± 0.0180	0.3009 ± 0.0181
ring	0.0208 ± 0.0047	0.0208 ± 0.0047	0.0208 ± 0.0047	0.0207 ± 0.0048	**0.0207** ± 0.0046
svmguide1	**0.0620** ± 0.0085	**0.0620** ± 0.0085	**0.0620** ± 0.0085	0.0813 ± 0.0092	0.2262 ± 0.0127
twonorm	**0.0226** ± 0.0058	**0.0226** ± 0.0058	**0.0226** ± 0.0058	0.0229 ± 0.0062	0.0234 ± 0.0061
vehicle	**0.1508** ± 0.0351	0.1508 ± 0.0349	0.1508 ± 0.0349	0.1938 ± 0.0480	0.4341 ± 0.0836
waveform	0.1485 ± 0.0161	**0.1485** ± 0.0162	0.1485 ± 0.0162	0.1524 ± 0.0169	0.1621 ± 0.0158
wdbc	**0.0522** ± 0.0275	**0.0522** ± 0.0275	0.0523 ± 0.0274	0.0548 ± 0.0290	0.0587 ± 0.0306
wine	**0.0119** ± 0.0245	**0.0119** ± 0.0245	**0.0119** ± 0.0245	0.0125 ± 0.0236	0.0141 ± 0.0258
			NB		
appendicitis	0.1730 ± 0.1278	0.1730 ± 0.1278	0.1730 ± 0.1278	**0.1367** ± 0.0980	0.2162 ± 0.1220
fourclass	**0.2470** ± 0.0408	**0.2470** ± 0.0408	0.2473 ± 0.0409	0.2576 ± 0.0403	0.2711 ± 0.0469
glass2	0.2235 ± 0.0783	0.2235 ± 0.0783	0.2240 ± 0.0782	**0.2044** ± 0.0786	0.2235 ± 0.0763
haberman	0.3332 ± 0.0811	0.3332 ± 0.0811	0.3332 ± 0.0811	**0.2761** ± 0.0847	0.3332 ± 0.0813
ion	0.3080 ± 0.0634	0.3080 ± 0.0634	0.3085 ± 0.0631	**0.2789** ± 0.0640	0.3179 ± 0.0682
iris	**0.0460** ± 0.0442	**0.0460** ± 0.0442	**0.0460** ± 0.0442	0.0467 ± 0.0450	0.0818 ± 0.0642
liver	0.5117 ± 0.0757	0.5117 ± 0.0757	0.5117 ± 0.0757	**0.4947** ± 0.0776	0.5064 ± 0.0760
newthyroid	0.0353 ± 0.0350	0.0348 ± 0.0358	0.0348 ± 0.0358	0.0343 ± 0.0352	**0.0324** ± 0.0344
phoneme	0.3333 ± 0.0172	0.3333 ± 0.0172	0.3352 ± 0.0172	**0.2935** ± 0.0187	0.3492 ± 0.0188
ring	0.0203 ± 0.0046	0.0203 ± 0.0046	0.0203 ± 0.0046	0.0201 ± 0.0046	**0.0201** ± 0.0045
svmguide1	**0.0651** ± 0.0088	**0.0651** ± 0.0088	0.0651 ± 0.0088	0.0761 ± 0.0094	0.1187 ± 0.0092
twonorm	0.0217 ± 0.0053	0.0217 ± 0.0053	0.0217 ± 0.0053	**0.0216** ± 0.0054	0.0216 ± 0.0053
vehicle	**0.5369** ± 0.0505	0.5388 ± 0.0498	0.5389 ± 0.0500	0.5741 ± 0.0548	0.5561 ± 0.0587
waveform	0.2247 ± 0.0170	0.2246 ± 0.0163	0.2247 ± 0.0162	**0.2192** ± 0.0153	0.2567 ± 0.0204
wdbc	0.0884 ± 0.0337	0.0884 ± 0.0337	0.0884 ± 0.0337	**0.0882** ± 0.0336	0.0884 ± 0.0337
wine	**0.0252** ± 0.0405	**0.0252** ± 0.0412	0.0258 ± 0.0412	0.0303 ± 0.0442	0.0336 ± 0.0447
			AODE		
appendicitis	**0.2954** ± 0.1484	**0.2954** ± 0.1484	**0.2954** ± 0.1484	0.3539 ± 0.1619	0.4142 ± 0.1675
fourclass	**0.2135** ± 0.0396	**0.2135** ± 0.0396	**0.2135** ± 0.0398	0.2581 ± 0.0433	0.3482 ± 0.0822
glass2	0.1089 ± 0.0588	0.1089 ± 0.0588	0.1089 ± 0.0588	**0.0913** ± 0.0561	0.0950 ± 0.0592
haberman	0.3335 ± 0.0811	0.3335 ± 0.0811	0.3332 ± 0.0816	**0.2757** ± 0.0848	0.3332 ± 0.0813
ion	0.2621 ± 0.0674	0.2621 ± 0.0674	0.2623 ± 0.0676	**0.1251** ± 0.0545	0.1382 ± 0.0587
iris	0.0218 ± 0.0366	0.0218 ± 0.0366	0.0218 ± 0.0366	**0.0199** ± 0.0360	0.0836 ± 0.0680
liver	0.5014 ± 0.0765	0.5014 ± 0.0765	0.5014 ± 0.0762	**0.4950** ± 0.0838	0.5008 ± 0.0776
newthyroid	0.0363 ± 0.0378	0.0377 ± 0.0379	0.0372 ± 0.0368	0.0386 ± 0.0386	**0.0353** ± 0.0375
phoneme	0.3250 ± 0.0160	0.3250 ± 0.0160	0.3259 ± 0.0159	**0.2948** ± 0.0175	0.3228 ± 0.0183
ring	0.0203 ± 0.0045	0.0203 ± 0.0045	0.0203 ± 0.0045	0.0202 ± 0.0046	**0.0201** ± 0.0045
svmguide1	0.0624 ± 0.0086	0.0624 ± 0.0086	**0.0624** ± 0.0086	0.0772 ± 0.0088	0.1972 ± 0.0120
twonorm	0.0217 ± 0.0054	0.0217 ± 0.0054	**0.0217** ± 0.0054	0.0217 ± 0.0054	0.0219 ± 0.0055
vehicle	**0.3472** ± 0.0559	0.3495 ± 0.0569	0.3499 ± 0.0570	0.4335 ± 0.0547	0.5683 ± 0.0529
waveform	**0.1559** ± 0.0159	0.1572 ± 0.0158	0.1573 ± 0.0159	0.1629 ± 0.0154	0.2163 ± 0.0179
wdbc	**0.0879** ± 0.0320	**0.0879** ± 0.0320	0.0880 ± 0.0321	0.1501 ± 0.0418	0.1759 ± 0.0413
wine	0.0152 ± 0.0287	0.0152 ± 0.0287	0.0152 ± 0.0287	**0.0140** ± 0.0292	0.0354 ± 0.0391

Table C.6: Mean squared error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.3, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	0.1957 ± 0.0631	0.1957 ± 0.0631	0.1961 ± 0.0632	**0.1793** ± 0.0523	0.2089 ± 0.0750
fourclass	0.0865 ± 0.0126	0.0865 ± 0.0126	**0.0864** ± 0.0126	0.1567 ± 0.0179	0.2076 ± 0.0319
glass2	0.1513 ± 0.0294	0.1513 ± 0.0294	0.1513 ± 0.0294	**0.1481** ± 0.0293	0.1514 ± 0.0304
haberman	0.2239 ± 0.0423	0.2239 ± 0.0423	0.2229 ± 0.0421	**0.1919** ± 0.0406	0.2253 ± 0.0432
ion	0.1581 ± 0.0265	0.1581 ± 0.0265	**0.1581** ± 0.0265	0.1595 ± 0.0260	0.1621 ± 0.0262
iris	**0.0636** ± 0.0163	0.0637 ± 0.0164	0.0637 ± 0.0164	0.0646 ± 0.0168	0.0876 ± 0.0285
liver	0.2773 ± 0.0511	0.2773 ± 0.0511	0.2776 ± 0.0511	**0.2722** ± 0.0376	0.2872 ± 0.0351
newthyroid	0.0696 ± 0.0153	0.0696 ± 0.0153	**0.0696** ± 0.0153	0.0705 ± 0.0154	0.0714 ± 0.0159
phoneme	**0.2039** ± 0.0197	**0.2039** ± 0.0197	0.2042 ± 0.0196	0.2106 ± 0.0082	0.2171 ± 0.0078
ring	0.1098 ± 0.0036	0.1098 ± 0.0036	**0.1098** ± 0.0036	0.1099 ± 0.0036	0.1099 ± 0.0036
svmguide1	**0.1108** ± 0.0042	**0.1108** ± 0.0042	0.1108 ± 0.0042	0.1162 ± 0.0044	0.1638 ± 0.0057
twonorm	0.1074 ± 0.0033	0.1074 ± 0.0033	**0.1074** ± 0.0033	0.1077 ± 0.0033	0.1075 ± 0.0033
vehicle	0.0658 ± 0.0087	**0.0658** ± 0.0087	0.0658 ± 0.0087	0.0822 ± 0.0134	0.1672 ± 0.0254
waveform	**0.0706** ± 0.0032	0.0707 ± 0.0032	0.0707 ± 0.0032	0.0725 ± 0.0034	0.0751 ± 0.0035
wdbc	**0.1277** ± 0.0162	**0.1277** ± 0.0162	0.1277 ± 0.0162	0.1282 ± 0.0166	0.1295 ± 0.0169
wine	0.0616 ± 0.0129	0.0616 ± 0.0129	0.0616 ± 0.0129	**0.0615** ± 0.0128	0.0619 ± 0.0131
			NB		
appendicitis	0.1704 ± 0.0519	0.1704 ± 0.0519	0.1708 ± 0.0519	**0.1598** ± 0.0460	0.1864 ± 0.0622
fourclass	0.0888 ± 0.0132	0.0888 ± 0.0132	**0.0888** ± 0.0132	0.1501 ± 0.0180	0.1972 ± 0.0228
glass2	0.1921 ± 0.0365	0.1921 ± 0.0365	0.1922 ± 0.0366	**0.1845** ± 0.0375	0.1920 ± 0.0365
haberman	0.2252 ± 0.0434	0.2252 ± 0.0434	0.2242 ± 0.0433	**0.1920** ± 0.0400	0.2259 ± 0.0433
ion	0.2282 ± 0.0295	0.2282 ± 0.0295	0.2282 ± 0.0296	**0.2161** ± 0.0310	0.2322 ± 0.0330
iris	**0.0667** ± 0.0176	0.0667 ± 0.0176	0.0667 ± 0.0176	0.0672 ± 0.0178	0.0749 ± 0.0226
liver	0.2926 ± 0.0366	0.2926 ± 0.0366	0.2927 ± 0.0366	**0.2837** ± 0.0385	0.2914 ± 0.0369
newthyroid	**0.0684** ± 0.0140	0.0684 ± 0.0139	0.0684 ± 0.0139	0.0687 ± 0.0140	0.0685 ± 0.0140
phoneme	0.2134 ± 0.0079	0.2134 ± 0.0079	0.2140 ± 0.0079	**0.1969** ± 0.0080	0.2211 ± 0.0092
ring	0.1098 ± 0.0037	0.1098 ± 0.0037	**0.1098** ± 0.0037	0.1099 ± 0.0037	0.1099 ± 0.0037
svmguide1	**0.1165** ± 0.0045	**0.1165** ± 0.0045	0.1165 ± 0.0045	0.1203 ± 0.0047	0.1317 ± 0.0051
twonorm	0.1072 ± 0.0032	0.1072 ± 0.0032	**0.1072** ± 0.0032	0.1072 ± 0.0032	0.1072 ± 0.0032
vehicle	**0.1853** ± 0.0205	0.1866 ± 0.0196	0.1867 ± 0.0195	0.2000 ± 0.0178	0.2040 ± 0.0181
waveform	**0.1227** ± 0.0055	0.1227 ± 0.0054	0.1228 ± 0.0054	0.1271 ± 0.0049	0.1438 ± 0.0072
wdbc	0.1437 ± 0.0184	0.1437 ± 0.0184	0.1437 ± 0.0184	**0.1429** ± 0.0188	0.1435 ± 0.0185
wine	**0.0648** ± 0.0150	0.0648 ± 0.0150	0.0649 ± 0.0150	0.0660 ± 0.0170	0.0672 ± 0.0174
			AODE		
appendicitis	**0.1083** ± 0.0435	**0.1083** ± 0.0435	0.1084 ± 0.0436	0.1857 ± 0.0564	0.1995 ± 0.0577
fourclass	0.0865 ± 0.0126	0.0865 ± 0.0126	**0.0864** ± 0.0126	0.1567 ± 0.0179	0.2076 ± 0.0319
glass2	**0.1268** ± 0.0269	**0.1268** ± 0.0269	0.1268 ± 0.0268	0.1404 ± 0.0275	0.1429 ± 0.0296
haberman	0.2236 ± 0.0427	0.2236 ± 0.0427	0.2227 ± 0.0426	**0.1919** ± 0.0403	0.2255 ± 0.0432
ion	**0.1524** ± 0.0280	**0.1524** ± 0.0280	0.1525 ± 0.0280	0.1539 ± 0.0255	0.1584 ± 0.0268
iris	**0.0614** ± 0.0153	0.0614 ± 0.0153	0.0614 ± 0.0153	0.0627 ± 0.0164	0.0691 ± 0.0180
liver	0.2660 ± 0.0562	0.2660 ± 0.0562	0.2661 ± 0.0562	**0.2618** ± 0.0607	0.2744 ± 0.0617
newthyroid	0.0686 ± 0.0143	0.0686 ± 0.0143	**0.0686** ± 0.0143	0.0693 ± 0.0145	0.0690 ± 0.0145
phoneme	**0.1930** ± 0.0070	**0.1930** ± 0.0070	0.1934 ± 0.0071	0.2021 ± 0.0076	0.1995 ± 0.0070
ring	0.1098 ± 0.0036	0.1098 ± 0.0036	**0.1098** ± 0.0036	0.1099 ± 0.0036	0.1099 ± 0.0036
svmguide1	**0.1119** ± 0.0042	**0.1119** ± 0.0042	0.1119 ± 0.0042	0.1167 ± 0.0044	0.1504 ± 0.0055
twonorm	0.1072 ± 0.0032	0.1072 ± 0.0032	**0.1072** ± 0.0032	0.1073 ± 0.0032	0.1072 ± 0.0032
vehicle	**0.0708** ± 0.0101	0.0710 ± 0.0102	0.0711 ± 0.0102	0.1240 ± 0.0175	0.1627 ± 0.0179
waveform	**0.0640** ± 0.0028	0.0643 ± 0.0028	0.0643 ± 0.0028	0.0703 ± 0.0031	0.0668 ± 0.0036
wdbc	0.1071 ± 0.0121	0.1071 ± 0.0121	**0.1070** ± 0.0121	0.1609 ± 0.0207	0.1698 ± 0.0202
wine	**0.0610** ± 0.0127	0.0610 ± 0.0127	0.0610 ± 0.0127	0.0617 ± 0.0133	0.0631 ± 0.0144

Table C.7: Mean classification error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.4, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	0.3528 ± 0.1967	0.3528 ± 0.1967	0.3528 ± 0.1967	**0.3253** ± 0.1815	0.3872 ± 0.2117
fourclass	**0.2635** ± 0.0535	**0.2635** ± 0.0535	0.2644 ± 0.0544	0.2736 ± 0.0576	0.4512 ± 0.0975
glass2	**0.1002** ± 0.0685	**0.1002** ± 0.0685	**0.1002** ± 0.0685	0.1026 ± 0.0684	0.1049 ± 0.0706
haberman	0.3333 ± 0.0831	0.3333 ± 0.0831	0.3333 ± 0.0831	**0.2990** ± 0.0802	0.3333 ± 0.0831
ion	**0.1413** ± 0.0589	**0.1413** ± 0.0589	**0.1413** ± 0.0589	0.1425 ± 0.0595	0.1450 ± 0.0590
iris	**0.0273** ± 0.0445	0.0280 ± 0.0446	0.0280 ± 0.0446	0.0307 ± 0.0428	0.1430 ± 0.1100
liver	0.5088 ± 0.0718	0.5088 ± 0.0718	0.5088 ± 0.0718	**0.5044** ± 0.0732	0.5088 ± 0.0689
newthyroid	**0.0372** ± 0.0374	**0.0372** ± 0.0374	**0.0372** ± 0.0374	0.0386 ± 0.0374	0.0410 ± 0.0388
phoneme	0.3013 ± 0.0188	0.3013 ± 0.0188	0.3013 ± 0.0188	**0.2972** ± 0.0172	0.3017 ± 0.0193
ring	0.0208 ± 0.0049	0.0208 ± 0.0049	0.0208 ± 0.0049	0.0208 ± 0.0049	**0.0207** ± 0.0048
svmguide1	**0.1195** ± 0.0137	**0.1195** ± 0.0137	0.1197 ± 0.0138	0.1365 ± 0.0138	0.2264 ± 0.0136
twonorm	**0.0231** ± 0.0057	**0.0231** ± 0.0057	**0.0231** ± 0.0057	0.0233 ± 0.0057	0.0717 ± 0.1235
vehicle	**0.1601** ± 0.0369	0.1603 ± 0.0372	0.1603 ± 0.0372	0.2415 ± 0.0549	0.4173 ± 0.1094
waveform	0.1498 ± 0.0135	**0.1497** ± 0.0135	0.1498 ± 0.0135	0.1531 ± 0.0141	0.1606 ± 0.0145
wdbc	0.0643 ± 0.0364	0.0643 ± 0.0364	0.0643 ± 0.0364	0.0647 ± 0.0364	**0.0643** ± 0.0369
wine	**0.0129** ± 0.0287	**0.0129** ± 0.0287	**0.0129** ± 0.0287	0.0141 ± 0.0282	0.0247 ± 0.0378
			NB		
appendicitis	0.2105 ± 0.1458	0.2105 ± 0.1458	0.2105 ± 0.1458	**0.1879** ± 0.1356	0.2342 ± 0.1573
fourclass	0.2861 ± 0.0610	0.2861 ± 0.0610	0.2868 ± 0.0617	**0.2587** ± 0.0375	0.2767 ± 0.0538
glass2	0.2236 ± 0.0861	0.2236 ± 0.0861	0.2236 ± 0.0861	**0.2213** ± 0.0862	0.2245 ± 0.0849
haberman	0.3333 ± 0.0831	0.3333 ± 0.0831	0.3333 ± 0.0831	**0.2983** ± 0.0811	0.3333 ± 0.0831
ion	0.3177 ± 0.0692	0.3177 ± 0.0692	0.3177 ± 0.0692	**0.2995** ± 0.0646	0.3206 ± 0.0672
iris	**0.0487** ± 0.0553	**0.0487** ± 0.0553	**0.0487** ± 0.0553	0.0494 ± 0.0559	0.0785 ± 0.0622
liver	0.5058 ± 0.0695	0.5058 ± 0.0695	0.5058 ± 0.0695	**0.4982** ± 0.0686	0.5073 ± 0.0694
newthyroid	0.0366 ± 0.0342	0.0362 ± 0.0338	0.0362 ± 0.0338	**0.0339** ± 0.0346	**0.0339** ± 0.0346
phoneme	0.3498 ± 0.0184	0.3498 ± 0.0184	0.3504 ± 0.0184	**0.3061** ± 0.0193	0.3514 ± 0.0188
ring	**0.0202** ± 0.0052	**0.0202** ± 0.0052	**0.0202** ± 0.0052	0.0202 ± 0.0052	**0.0202** ± 0.0051
svmguide1	**0.0844** ± 0.0099	**0.0844** ± 0.0099	0.0846 ± 0.0099	0.0899 ± 0.0109	0.1185 ± 0.0107
twonorm	**0.0217** ± 0.0054	**0.0217** ± 0.0054	**0.0217** ± 0.0054	0.0218 ± 0.0054	0.0218 ± 0.0055
vehicle	0.5637 ± 0.0468	0.5641 ± 0.0468	0.5641 ± 0.0466	0.5727 ± 0.0517	**0.5496** ± 0.0559
waveform	0.4745 ± 0.0810	0.4745 ± 0.0808	0.4745 ± 0.0808	**0.2246** ± 0.0165	0.2567 ± 0.0183
wdbc	0.0886 ± 0.0334	0.0886 ± 0.0334	0.0886 ± 0.0334	**0.0882** ± 0.0339	0.0884 ± 0.0334
wine	**0.0327** ± 0.0401	**0.0327** ± 0.0401	**0.0327** ± 0.0401	0.0338 ± 0.0407	0.0355 ± 0.0423
			AODE		
appendicitis	**0.3444** ± 0.1615	**0.3444** ± 0.1615	0.3463 ± 0.1606	0.4116 ± 0.1525	0.4385 ± 0.1626
fourclass	**0.2635** ± 0.0535	**0.2635** ± 0.0535	0.2644 ± 0.0544	0.2736 ± 0.0576	0.4512 ± 0.0975
glass2	0.1143 ± 0.0722	0.1143 ± 0.0722	0.1143 ± 0.0722	0.0966 ± 0.0618	**0.0957** ± 0.0621
haberman	0.3333 ± 0.0831	0.3333 ± 0.0831	0.3333 ± 0.0831	**0.2977** ± 0.0802	0.3333 ± 0.0831
ion	0.2782 ± 0.0688	0.2782 ± 0.0688	0.2782 ± 0.0688	**0.1347** ± 0.0589	0.1428 ± 0.0530
iris	**0.0240** ± 0.0397	0.0247 ± 0.0398	**0.0240** ± 0.0397	0.0287 ± 0.0394	0.0921 ± 0.0770
liver	**0.4976** ± 0.0698	**0.4976** ± 0.0698	**0.4976** ± 0.0698	0.5006 ± 0.0678	0.5053 ± 0.0645
newthyroid	0.0371 ± 0.0373	0.0371 ± 0.0373	0.0371 ± 0.0373	0.0367 ± 0.0368	**0.0353** ± 0.0384
phoneme	0.3379 ± 0.0228	0.3379 ± 0.0228	0.3382 ± 0.0228	**0.3025** ± 0.0177	0.3312 ± 0.0199
ring	**0.0204** ± 0.0052	**0.0204** ± 0.0052	**0.0204** ± 0.0052	0.0204 ± 0.0053	**0.0204** ± 0.0052
svmguide1	**0.0954** ± 0.0104	**0.0954** ± 0.0104	0.0955 ± 0.0104	0.1110 ± 0.0123	0.1979 ± 0.0134
twonorm	**0.0218** ± 0.0054	**0.0218** ± 0.0054	**0.0218** ± 0.0054	0.0219 ± 0.0055	0.0220 ± 0.0055
vehicle	**0.4502** ± 0.0569	0.4530 ± 0.0565	0.4526 ± 0.0564	0.4781 ± 0.0548	0.5710 ± 0.0536
waveform	0.1737 ± 0.0229	0.1742 ± 0.0232	0.1745 ± 0.0232	**0.1712** ± 0.0173	0.2241 ± 0.0198
wdbc	**0.0926** ± 0.0327	**0.0926** ± 0.0327	**0.0926** ± 0.0327	0.1541 ± 0.0456	0.1726 ± 0.0419
wine	**0.0198** ± 0.0323	**0.0198** ± 0.0323	0.0203 ± 0.0334	0.0209 ± 0.0326	0.0418 ± 0.0417

Table C.8: Mean squared error in ten repetitions of a 10-fold cross-validation. The set of experts' class labels were generated using a Beta distribution $B(0.4, 0.01)$. The best results for each dataset and algorithm are highlighted in boldface. The best result for each dataset is shaded gray.

Datasets	CoMEM-MUL	CoMEM-BIN	CoMEM-IND	PLEM	EM
			MG		
appendicitis	0.2226 ± 0.0601	0.2226 ± 0.0601	0.2226 ± 0.0602	**0.2166** ± 0.0598	0.2350 ± 0.0611
fourclass	0.1850 ± 0.0275	0.1850 ± 0.0275	0.1849 ± 0.0274	**0.1844** ± 0.0180	0.2249 ± 0.0263
glass2	0.2000 ± 0.0312	0.2000 ± 0.0312	0.2000 ± 0.0312	**0.1999** ± 0.0311	0.2007 ± 0.0310
haberman	0.2284 ± 0.0330	0.2284 ± 0.0330	0.2282 ± 0.0330	**0.2126** ± 0.0309	0.2282 ± 0.0330
ion	**0.2031** ± 0.0251	**0.2031** ± 0.0251	0.2031 ± 0.0251	0.2034 ± 0.0252	0.2039 ± 0.0254
iris	0.0986 ± 0.0203	**0.0986** ± 0.0203	0.0986 ± 0.0203	0.0999 ± 0.0207	0.1127 ± 0.0294
liver	0.2592 ± 0.0303	0.2592 ± 0.0303	0.2591 ± 0.0303	**0.2529** ± 0.0296	0.2597 ± 0.0296
newthyroid	0.1073 ± 0.0167	0.1073 ± 0.0167	0.1073 ± 0.0167	**0.1072** ± 0.0167	0.1077 ± 0.0169
phoneme	0.2297 ± 0.0080	0.2297 ± 0.0080	0.2297 ± 0.0080	**0.2287** ± 0.0078	0.2301 ± 0.0080
ring	**0.1770** ± 0.0049	**0.1770** ± 0.0049	0.1770 ± 0.0049	0.1771 ± 0.0049	0.1771 ± 0.0049
svmguide1	**0.1728** ± 0.0059	**0.1728** ± 0.0059	0.1730 ± 0.0059	0.1754 ± 0.0061	0.1911 ± 0.0057
twonorm	0.1727 ± 0.0047	0.1727 ± 0.0047	0.1727 ± 0.0047	0.1729 ± 0.0047	**0.1601** ± 0.0317
vehicle	**0.0824** ± 0.0088	0.0824 ± 0.0088	0.0824 ± 0.0088	0.1098 ± 0.0153	0.1613 ± 0.0270
waveform	0.0902 ± 0.0030	0.0902 ± 0.0030	**0.0902** ± 0.0030	0.0924 ± 0.0031	0.0938 ± 0.0030
wdbc	0.1929 ± 0.0193	0.1929 ± 0.0193	0.1929 ± 0.0193	0.1927 ± 0.0196	**0.1924** ± 0.0195
wine	0.1008 ± 0.0176	0.1008 ± 0.0176	0.1007 ± 0.0176	**0.1007** ± 0.0171	0.1013 ± 0.0181
			NB		
appendicitis	0.2117 ± 0.0500	0.2117 ± 0.0500	0.2118 ± 0.0500	**0.2047** ± 0.0478	0.2138 ± 0.0524
fourclass	**0.1789** ± 0.0211	**0.1789** ± 0.0211	0.1789 ± 0.0210	0.1871 ± 0.0245	0.2068 ± 0.0242
glass2	0.2239 ± 0.0320	0.2239 ± 0.0320	**0.2239** ± 0.0320	0.2241 ± 0.0320	0.2240 ± 0.0320
haberman	0.2290 ± 0.0329	0.2290 ± 0.0329	0.2288 ± 0.0329	**0.2129** ± 0.0308	0.2287 ± 0.0330
ion	0.2453 ± 0.0305	0.2453 ± 0.0305	0.2453 ± 0.0305	**0.2402** ± 0.0297	0.2460 ± 0.0295
iris	0.0993 ± 0.0209	0.0993 ± 0.0209	0.0993 ± 0.0209	**0.0990** ± 0.0211	0.1028 ± 0.0203
liver	0.2608 ± 0.0291	0.2608 ± 0.0291	0.2607 ± 0.0291	**0.2580** ± 0.0307	0.2618 ± 0.0293
newthyroid	0.1065 ± 0.0159	0.1065 ± 0.0159	0.1065 ± 0.0159	0.1063 ± 0.0160	**0.1062** ± 0.0159
phoneme	0.2240 ± 0.0081	0.2240 ± 0.0081	0.2240 ± 0.0081	**0.2132** ± 0.0074	0.2242 ± 0.0083
ring	**0.1771** ± 0.0049	**0.1771** ± 0.0049	0.1771 ± 0.0049	0.1771 ± 0.0049	0.1771 ± 0.0049
svmguide1	**0.1764** ± 0.0057	**0.1764** ± 0.0057	0.1764 ± 0.0057	0.1769 ± 0.0057	0.1798 ± 0.0060
twonorm	0.1726 ± 0.0046	0.1726 ± 0.0046	**0.1726** ± 0.0046	0.1726 ± 0.0046	0.1726 ± 0.0046
vehicle	0.1937 ± 0.0116	0.1937 ± 0.0116	0.1937 ± 0.0116	0.1925 ± 0.0127	**0.1925** ± 0.0151
waveform	0.2159 ± 0.0232	0.2159 ± 0.0231	0.2159 ± 0.0231	**0.1450** ± 0.0046	0.1561 ± 0.0053
wdbc	0.2007 ± 0.0187	0.2007 ± 0.0187	0.2007 ± 0.0187	**0.2007** ± 0.0186	0.2007 ± 0.0187
wine	0.1032 ± 0.0175	**0.1032** ± 0.0175	0.1032 ± 0.0175	0.1036 ± 0.0176	0.1042 ± 0.0179
			AODE		
appendicitis	0.1154 ± 0.0463	0.1154 ± 0.0463	**0.1154** ± 0.0463	0.1898 ± 0.0495	0.1916 ± 0.0514
fourclass	0.1850 ± 0.0275	0.1850 ± 0.0275	0.1849 ± 0.0274	**0.1844** ± 0.0180	0.2249 ± 0.0263
glass2	0.1743 ± 0.0337	0.1743 ± 0.0337	**0.1743** ± 0.0337	0.1961 ± 0.0321	0.1953 ± 0.0318
haberman	0.2285 ± 0.0330	0.2285 ± 0.0330	0.2284 ± 0.0330	**0.2127** ± 0.0309	0.2284 ± 0.0330
ion	**0.1750** ± 0.0278	**0.1750** ± 0.0278	0.1750 ± 0.0278	0.2051 ± 0.0263	0.2073 ± 0.0258
iris	0.0956 ± 0.0196	0.0956 ± 0.0196	0.0956 ± 0.0196	0.0984 ± 0.0202	**0.0926** ± 0.0229
liver	0.2267 ± 0.0642	0.2267 ± 0.0642	0.2266 ± 0.0642	**0.2206** ± 0.0663	0.2365 ± 0.0617
newthyroid	0.1060 ± 0.0161	0.1060 ± 0.0161	**0.1060** ± 0.0161	0.1065 ± 0.0162	0.1062 ± 0.0162
phoneme	**0.1911** ± 0.0080	**0.1911** ± 0.0080	0.1912 ± 0.0080	0.2146 ± 0.0076	0.2081 ± 0.0082
ring	**0.1771** ± 0.0049	**0.1771** ± 0.0049	0.1771 ± 0.0049	0.1771 ± 0.0049	0.1771 ± 0.0049
svmguide1	**0.1711** ± 0.0058	**0.1711** ± 0.0058	0.1711 ± 0.0058	0.1736 ± 0.0059	0.1835 ± 0.0057
twonorm	0.1725 ± 0.0046	0.1725 ± 0.0046	**0.1725** ± 0.0046	0.1726 ± 0.0046	0.1726 ± 0.0046
vehicle	**0.0792** ± 0.0120	0.0795 ± 0.0121	0.0794 ± 0.0121	0.1255 ± 0.0137	0.1504 ± 0.0156
waveform	**0.0608** ± 0.0039	0.0608 ± 0.0039	0.0608 ± 0.0039	0.0860 ± 0.0030	0.0724 ± 0.0033
wdbc	0.1554 ± 0.0187	0.1554 ± 0.0187	**0.1554** ± 0.0187	0.2093 ± 0.0201	0.2110 ± 0.0195
wine	0.0988 ± 0.0160	**0.0988** ± 0.0159	0.0988 ± 0.0160	0.1006 ± 0.0166	0.0995 ± 0.0174

Bibliography

[1] S. Abe. *Support Vector Machines for Pattern Classification*. Springer, 2010.

[2] N. Adams. Semi-supervised learning. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 172(2):530, 2009.

[3] C. Agostinelli and U. Lund. *R Package* `circular`*: Circular Statistics*, 2011. URL https://r-forge.r-project.org/projects/circular.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB 1994)*, pages 487–499. Morgan Kaufmann, 1994.

[5] A. Agresti. *An Introduction to Categorical Data Analysis*. John Wiley & Sons, second edition, 2007.

[6] D. W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2):267–287, 1992.

[7] D. W. Aha, editor. *Lazy Learning*. Springer, 1997.

[8] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[9] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287, 2011.

[10] A. P. Alivisatos, M. Chun, G. M. Church, R. J. Greenspan, M. L. Roukes, and R. Yuste. The Brain Activity Map project and the challenge of functional connectomics. *Neuron*, 74(6):970–974, 2012.

[11] A. P. Alivisatos, M. Chun, G. M. Church, K. Deisseroth, J. P. Donoghue, R. J. Greenspan, P. L. McEuen, M. L. Roukes, T. J. Sejnowski, P. S. Weiss, and R. Yuste. The Brain Activity Map. *Science*, 339:1284–1285, 2013.

[12] L. Alonso-Nanclares, A. Kastanauskaite, J.-R. Rodríguez, J. González-Soriano, and J. DeFelipe. A stereological study of synapse number in the epileptic human hippocampus. *Frontiers in Neuroanatomy*, 5(8):1–13, 2011.

[13] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[14] D. G. Altman. *Practical Statistics for Medical Research*, volume 12. Chapman & Hall/CRC, 1991.

[15] D. Amaral and P. Lavenex. Hippocampal neuroanatomy. In P. Andersen, R. Morris, D. Amaral, T. Bliss, and J. O'Keefe, editors, *The Hippocampus Book*, pages 37–114. Oxford University Press, 2006.

[16] O. Amayri and N. Bouguila. Beyond hybrid generative discriminative learning: Spherical data classification. *Pattern Analysis and Applications*, in press, 2013.

[17] H. Anwar, I. Riachi, S. Hill, F. Schürmann, and H. Markram. An approach to capturing neuron morphological diversity. In E. De Schutter, editor, *Computational Modeling Methods for Neuroscientists*, pages 211–232. The MIT Press, 2009.

[18] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

[19] R. Artstein and M. Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008.

[20] G. A. Ascoli. Successes and rewards in sharing digital reconstructions of neuronal morphology. *Neuroinformatics*, 5:154–160, 2007.

[21] G. A. Ascoli and J. L. Krichmar. L-Neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, 32-33:1003–1011, 2000.

[22] G. A. Ascoli, J. L. Krichmar, S. J. Nasuto, and S. L. Senft. Generation, description and storage of dendritic morphology data. *Philosophical Transactions of the Royal Society of London. Series B (Biological Sciences)*, 356:1131–1145, 2001.

[23] G. A. Ascoli, D. E. Donohue, and M. Halavi. Neuromorpho.org: A central resource for neuronal morphologies. *Journal of Neuroscience*, 27(35):9247–9251, 2007.

[24] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. *Journal of Intelligent Information Systems*, 20(3):255–283, 2003.

[25] F. A. C. Azevedo, L. R. B. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. L. Ferretti, R. E. P. Leite, W. J. Filho, R. Lent, and S. Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[26] F. R. Bach and M. I. Jordan. Learning graphical models with Mercer kernels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, pages 1009–1016. The MIT Press, 2002.

[27] K. Bache and M. Lichman. UCI Machine Learning Repository, URL http://archive.ics.uci.edu/ml, 2013.

[28] G. Bagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[29] I. Ballesteros-Yáñez, R. Benavides-Piccione, J. Bourgeois, J. Changeux, and J. DeFelipe. Alterations of cortical pyramidal neurons in mice lacking high-affinity nicotinic receptors. *Proceedings of the National Academy of Sciences of the United States of America*, 107(25):11567–11572, 2010.

[30] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

[31] M. Banerjee, M. Capozzoli, L. McSweeney, and D. Sinha. Beyond kappa: A review of interrater agreement measures. *Canadian Journal of Statistics*, 27(1):3–23, 1999.

[32] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[33] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(3):291–312, 2012.

[34] D. M. Bashtannyk and R. J. Hyndman. Bandwidth selection for kernel conditional density estimation. *Computational Statistics & Data Analysis*, 36(3):279–298, 2001.

[35] E. Batschelet. *Circular Statistics in Biology*. Academic Press, 1981.

[36] T. Bdiri and N. Bouguila. Positive vectors clustering using inverted Dirichlet finite mixture models. *Expert Systems with Applications*, 39(2):1869–1882, 2012.

[37] R. Benavides-Piccione. *Microestructura e Inervación Catecolaminérgica de las Células Piramidales de la Corteza Cerebral*. PhD thesis, Universidad Complutense de Madrid, 2004.

[38] R. Benavides-Piccione, I. Ballesteros-Yáñez, M. Martínez de Legrán, G. Elston, X. Estivill, C. Fillat, J. DeFelipe, and M. Dierssen. On dendrites in Down syndrome and DS murine models: A spiny way to learn. *Progress in Neurobiology*, 74:111–126, 2004.

[39] R. Benavides-Piccione, F. Hamzei-Sichani, I. Ballesteros-Yáñez, J. DeFelipe, and R. Yuste. Dendritic size of pyramidal neurons differs among mouse cortical regions. *Cerebral Cortex*, 16:990–1001, 2006.

[40] R. Benavides-Piccione, I. Fernaud-Espinosa, V. Robles, R. Yuste, and J. DeFelipe. Age-based comparison of human dendritic spine structure using complete three-dimensional reconstructions. *Cerebral Cortex*, 23(8):1798–1810, 2013.

[41] P. Berens. CircStat: A MATLAB toolbox for circular statistics. *Journal of Statistical Software*, 31(10):1–21, 2009.

[42] G. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In P. Bauer, G. Hommel, and E. Sonnemann, editors, *Multiple Hypotheses Testing*, pages 100–115. Springer, 1988.

[43] D. S. Berkholz, P. B. Krenesky, J. R. Davidson, and P. A. Karplus. Protein geometry database: A flexible engine to explore backbone conformations and their relationships to covalent geometry. *Nucleic Acids Res*, 38(suppl 1):D320–D325, 2010.

[44] M. Berry, T. Hollingworth, E. M. Anderson, and R. M. Flinn. Application of network analysis to the study of the branching patterns of dendritic fields. *Advances in Neurology*, 12:217–245, 1975.

[45] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[46] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[47] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999.

[48] R. K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, and W. Wittek. Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2–3):511–528, 2004.

[49] R. Bordley. A multiplicative formula for aggregating probability assessments. *Management Science*, 28:1137–1148, 1982.

[50] L. Bos, S. De Marchi, M. Vianello, and Y. Xu. Bivariate Lagrange interpolation at the Padua points: The ideal theory approach. *Numerische Mathematik*, 108(1):47–57, 2007.

[51] M. Bota and L. W. Swanson. The neuron classification problem. *Brain Research Reviews*, 56(1):79–88, 2007.

[52] S. G. Bøttcher. *Learning Bayesian Networks with Mixed Variables*. PhD thesis, Aalborg University, 2004.

[53] S. G. Bøttcher and C. Dethlefsen. Learning Bayesian networks with R. In K. Hornik, F. Leisch, and A. Zeileis, editors, *Proceedings of the Third International Workshop on Distributed Statistical Computing (DSC 2003)*, 2003.

[54] R. R. Bouckaert. Optimizing causal orderings for generating DAGs from data. In D. Dubois and M. P. Wellman, editors, *Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1992)*, pages 9–16. Morgan Kaufmann, 1992.

[55] R. R. Bouckaert. Estimating replicability of classiffier learning experiments. In C. E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning*. ACM, 2004.

[56] R. R. Bouckaert, E. F. Castillo, and J. M. Gutiérrez. A modified simulation scheme for inference in Bayesian networks. *International Journal of Approximate Reasoning*, 14(1):55–80, 1996.

[57] N. Bouguila. Clustering of count data using generalized Dirichlet multinomial distributions. *IEEE Transactions on Knowledge and Data Engineering*, 20(4):462–474, 2008.

[58] N. Bouguila. Count data modeling and classification using finite mixtures of distributions. *IEEE Transactions on Neural Networks*, 22(2):186–198, 2011.

[59] N. Bouguila and W. ElGuebaly. Discrete data clustering using finite mixture models. *Pattern Recognition*, 42(1):33–42, 2009.

[60] N. Bouguila and D. Ziou. High-dimensional unsupervised selection and estimation of a finite generalized Dirichlet mixture model based on minimum message length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1716–1731, 2007.

[61] N. Bouguila and D. Ziou. A countably infinite mixture model for clustering and feature selection. *Knowledge and Information Systems*, 33(2):351–370, 2012.

[62] N. Bouguila, D. Ziou, and J. Vaillancourt. Unsupervised learning of a finite mixture model based on the Dirichlet distribution and its application. *IEEE Transactions on Image Processing*, 13(11):1533–1543, 2004.

[63] S. Boutemedjet, D. Ziou, and N. Bouguila. Model-based subspace clustering of non-Gaussian data. *Neurocomputing*, 73(10-12):1730–1739, 2010.

[64] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[65] P. Brennan and A. Silman. Statistical methods for assessing observer variability in clinical measures. *British Medical Journal*, 304(6840):1491–1494, 1992.

[66] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. Peckham, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997)*, pages 265–276. ACM Press, 1997.

[67] K. M. Brown, T. A. Gillette, and G. A. Ascoli. Quantifying neuronal size: Summing up trees and splitting the branch difference. *Seminars in Cell & Developmental Biology*, 19:485–493, 2008.

[68] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

[69] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[70] T. Byrt, J. Bishop, and J. Carlin. Bias, prevalence and kappa. *Journal of Clinical Epidemiology*, 46(5):423–429, 1993.

[71] S. Calderara, A. Prati, and R. Cucchiara. Mixtures of von Mises distributions for people trajectory shape analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):457–471, 2011.

[72] M. Caliari, S. De Marchi, and M. Vianello. Hyperinterpolation in the cube. *Computers & Mathematics with Applications*, 55(11):2490–2497, 2008.

[73] M. Caliari, S. De Marchi, A. Sommariva, and M. Vianello. Padua2DM: Fast interpolation and cubature at the Padua points in Matlab/Octave. *Numerical Algorithms*, 56: 45–60, 2011.

[74] R. C. Cannon, D. A. Turner, G. K. Pyapali, and H. V. Wheal. An on-line archive of reconstructed hippocampal neurons. *Journal of Neuroscience Methods*, 84:49–54, 1998.

[75] A. Cano, M. Gómez-Olmedo, and S. Moral. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52(1):49–62, 2011.

[76] J. Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[77] A. M. Carvalho, T. Roos, A. L. Oliveira, and P. Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 12:2181–2210, 2011.

[78] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines, 2011.

[79] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

[80] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 138–180. The AAAI Press, 1996.

[81] B. Chen, Q. Liao, and Z. Tang. A clustering based Bayesian network classifier. In *Proceedings of the IEEE Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 444–448. IEEE Computer Society, 2007.

[82] J.-Y. Chen. A simulation study investigating the impact of dendritic morphology and synaptic topology on neuronal firing patterns. *Neural Computation*, 22(4):1086–1111, 2009.

[83] R. Chen, K. Sivakumar, and H. Kargupta. Collective mining of Bayesian networks from distributed heterogeneous data. *Knowledge and Information Systems*, 6:164–187, 2004.

[84] J. Cheng and R. Greiner. Learning Bayesian belief network classifiers: Algorithms and system. In E. Stroulia and S. Matwin, editors, *Proceedings of the Fourteenth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, volume 2056 of *Lecture Notes in Computer Science*, pages 141–151. Springer, 2001.

[85] D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data*, volume 112 of *Lecture Notes in Statistics*, pages 121–130. Springer, 1996.

[86] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[87] S.-T. Chiu. A comparative review of bandwidth selection for kernel density estimation. *Statistica Sinica*, 6:129–146, 1996.

[88] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[89] T. G. Chowdhury, J. C. Jimenez, J. M. Bomar, A. Cruz-Martin, J. P. Cantle, and C. Portera-Cailliau. Fate of Cajal–Retzius neurons in the postnatal mouse neocortex. *Frontiers in Neuroanatomy*, 4(10):1–8, 2010.

[90] K. W. Church and W. A. Gale. Poisson mixtures. *Natural Language Engineering*, 1(2): 163–190, 1995.

[91] D. V. Cicchetti and A. R. Feinstein. High agreement but low kappa: II. Resolving the paradoxes. *Journal of Clinical Epidemiology*, 43(6):551–558, 1990.

[92] H. T. Cline. Dendritic arbor development and synaptogenesis. *Current Opinion in Neurobiology*, 11(1):118–126, 2001.

[93] B. Cobb, P. P. Shenoy, and R. Rumí. Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, 16:193–308, 2006.

[94] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[95] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 115–123. Morgan Kaufmann, 1995.

[96] E. Côme, L. Oukhellou, T. Denœux, and P. Aknin. Learning from partially supervised data using mixture models and belief functions. *Pattern Recognition*, 42:334–348, 2009.

[97] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[98] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[99] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1225–1261, 2011.

[100] A. Crampton and A. B. Forbes. Spline approximation using knot density functions. In A. Iske and J. Levesley, editors, *Algorithms for Approximation*, pages 249–258. Springer, 2007.

[101] G. Csárdi and T. Nepusz. The `igraph` software package for complex network research. *InterJournal, Vol. Complex Systems*, 1695:1–9, 2006.

[102] H. Cuntz. The dendritic density field of a cortical pyramidal cell. *Frontiers in Neuroanatomy*, 6(2):1–6, 2012.

[103] H. Cuntz, A. Borst, and I. Segev. Optimization principles of dendritic structure. *Theoretical Biology and Medical Modelling*, 4(21):1–8, 2007.

[104] H. Cuntz, F. Forstner, A. Borst, and M. Häusser. One rule to grow them all: A general theory of neuronal branching and its practical application. *PLoS Computational Biology*, 6(8):e1000877, 2010.

[105] H. Cuntz, A. Mathy, and M. Häusser. A scaling law derived from optimal dendritic wiring. *Proceedings of the National Academy of Sciences of the United States of America*, 109(27):11014–11018, 2012.

[106] J. E. da Silva, J. Marques de Sá, and J. Jossinet. Classification of breast tissue by electrical impedance spectroscopy. *Medical and Biological Engineering and Computing*, 38(1):26–30, 2000.

[107] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[108] R. Daly, Q. Shen, and S. Aitken. Learning Bayesian networks: Approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157, 2011.

[109] P. Damien and S. Walker. A full Bayesian analysis of circular data using the von Mises distribution. *Canadian Journal of Statistics*, 27(2):291–298, 1999.

[110] A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[111] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[112] D. Dash and G. Cooper. Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203, 2004.

[113] P. Dayan and L. F. Abbott. *Theoretical Neuroscience*. The MIT Press, 2001.

[114] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[115] L. M. de Campos and J. F. Huete. Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing. In *Proceedings of the Eight Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 333–340, 2000.

[116] J. G. De Gooijer and D. Zerom. On conditional density estimation. *Statistica Neerlandica*, 57(2):159–176, 2003.

[117] J. DeFelipe. El nacimiento de la neurociencia moderna. *Trébede*, 63, 2002.

[118] J. DeFelipe. Cortical interneurons: From Cajal to 2001. *Progress in Brain Research*, 136:215–238, 2002.

[119] J. DeFelipe. The neuroanatomist's dream, the problems and solutions, and the ultimate aim. *Frontiers in Neuroscience*, 2:10–12, 2008.

[120] J. DeFelipe. From the connectome to the synaptome: An epic love story. *Science*, 330 (6008):1198–1201, 2010.

[121] J. DeFelipe and I. Fariñas. The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs. *Progress in Neurobiology*, 39:563–607, 1992.

[122] J. DeFelipe, H. Markram, and K. Rockland. The neocortical column. *Frontiers in Neuroanatomy*, 6(22):1–2, 2012.

[123] J. DeFelipe, P. L. López-Cruz, R. Benavides-Piccione, C. Bielza, P. Larrañaga, S. Anderson, A. Burkhalter, B. Cauli, A. Fairén, D. Feldmeyer, G. Fishell, D. Fitzpatrick, T. F. Freund, G. González-Burgos, S. Hestrin, S. Hill, P. R. Hof, J. Huang, E. G.

Jones, Y. Kawaguchi, Z. Kisvárday, Y. Kubota, D. A. Lewis, O. Marín, H. Markram, C. J. McBain, H. S. Meyer, H. Monyer, S. B. Nelson, K. Rockland, J. Rossier, J. L. R. Rubenstein, B. Rudy, M. Scanziani, G. M. Shepherd, C. C. Sherwood, J. F. Staiger, G. Tamás, A. Thomson, Y. Weng, R. Yuste, and G. A. Ascoli. New insights into the classification and nomenclature of cortical GABAergic interneurons. *Nature Reviews Neuroscience*, 14(3):202–216, 2013.

[124] G. L. deHaas–Lorentz. *Die Brownsche Bewegung und einige verwandte Erscheinungen*. Friedr. Vieweg und Sohn, 1913.

[125] M. J. del Jesus, J. A. Gámez, P. González, and J. M. Puerta. On the discovery of association rules by means of evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):397–415, 2011.

[126] J. del Valle, S. Bayod, A. Camins, C. Beas-Zárate, D. A. Velázquez-Zamora, I. González-Burgos, and M. Pallàs. Dendritic spine abnormalities in hippocampal CA1 pyramidal neurons underlying memory deficits in the SAMP8 mouse model of Alzheimer's disease. *Journal of Alzheimer's Disease*, 32(1):233–240, 2012.

[127] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[128] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[129] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[130] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. Automatic Bayesian curve fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60 (2):333–350, 1998.

[131] T. Denœux. A k-nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5):804–813, 1995.

[132] T. Denœux and L. M. Zouhal. Handling possibilistic labels in pattern classification using evidential reasoning. *Fuzzy Sets and Systems*, 122:409–424, 2001.

[133] C. Dethlefsen and S. Højsgaard. A common platform for graphical models in R: The gRbase package. *Journal of Statistical Software*, 14(17):1–12, 2005.

[134] J. M. Devaud, B. Quenet, J. Gascuel, and C. Masson. Statistical analysis and parsimonious modelling of dendrograms of in vitro neurones. *Bulletin of Mathematical Biology*, 62:657–674, 2000.

[135] D. Devlaminck, W. Waegeman, B. Bauwens, B. Wyns, P. Santens, and G. Otte. From circular ordinal regression to multilabel classification. In *Proceedings of the 2010 Workshop on Preference Learning, European Conference on Machine Learning*, 2010.

[136] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[137] F. J. Díez. Local conditioning in Bayesian networks. *Artificial Intelligence*, 87(1-2): 1–20, 1996.

[138] I. DiMatteo, C. R. Genovese, and R. E. Kass. Bayesian curve-fitting with free-knot splines. *Biometrika*, 88(4):1055–1071, 2001.

[139] B. Ding, R. Gentleman, and V. Carey. *bioDist: Different distance measures*, 2010. R package version 1.18.0.

[140] M. Ding and D. Glanzman. *The Dynamic Brain: An Exploration of Neuronal Variability and its Functional Significance*. Oxford University Press, 2011.

[141] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Communications of the ACM*, 54(4):86–96, 2011.

[142] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero–one loss. *Machine Learning*, 29:103–130, 1997.

[143] D. E. Donohue and G. A. Ascoli. Models of neuronal outgrowth. In S. Koslow and S. Subramaniam, editors, *Databasing the Brain: From Data to Knowledge*, pages 303–326. Wiley, New York, 2005.

[144] D. E. Donohue and G. A. Ascoli. Local diameter fully constrains dendritic size in basal but not apical trees of CA1 pyramidal neurons. *Journal of Computational Neuroscience*, 19(2):223–238, 2005.

[145] D. E. Donohue and G. A. Ascoli. A comparative computer simulation of dendritic morphology. *PLoS Computational Biology*, 4(6):e1000089, 2008.

[146] D. E. Donohue and G. A. Ascoli. Automated reconstruction of neuronal morphology: An overview. *Brain Research Reviews*, 67:94–102, 2011.

[147] G. Dougherty. *Pattern Recognition and Classification: An Introduction*. Springer, 2012.

[148] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 194–202. Morgan Kaufmann, 1995.

[149] T. D. Downs. Spherical regression. *Biometrika*, 90(3):655–668, 2003.

[150] T. D. Downs and K. V. Mardia. Circular regression. *Biometrika*, 89(3):683–697, 2002.

[151] E. Driver and D. Morrell. Implementation of continuous Bayesian networks using sums of weighted Gaussians. In P. Besnard and S. Hanks, editors, *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pages 134–140. Morgan Kaufmann, 1995.

[152] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[153] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, second edition, 2001.

[154] T. Duong. *ks: Kernel smoothing*, 2012. URL http://CRAN.R-project.org/package=ks. R package version 1.8.11.

[155] K. Eben. Classification into two von Mises distributions with unknown mean directions. *Aplikace Matematiky*, 28(3):230–237, 1983.

[156] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[157] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–75, 1986.

[158] K. El Emam. Benchmarking kappa: Interrater agreement in software process assessments. *Empirical Software Engineering*, 4(2):113–133, 1999.

[159] S. El Khattabi and F. Streit. Identification analysis in directional statistics. *Computational Statistics & Data Analysis*, 23:45–63, 1996.

[160] C. Elkan. Clustering documents with an exponential-family approximation of the Dirichlet compound multinomial distribution. In W. W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 289–296. ACM, 2006.

[161] Z. Elouedi, K. Mellouli, and P. Smets. Belief decision trees: Theoretical foundations. *International Journal of Approximate Reasoning*, 28:91–124, 2001.

[162] G. Elston and M. Rosa. The occipito-parietal pathway of the macaque monkey: Comparison of pyramidal cell morphology in layer III of functionally related cortical visual areas. *Cerebral Cortex*, 7(5):432–452, 1997.

[163] G. N. Elston, R. Benavides-Piccione, A. Elston, J. DeFelipe, and P. R. Manger. Specialization in pyramidal cell structure in the sensory-motor cortex of the vervet monkey (Cercopethicus pygerythrus). *Neuroscience*, 134(3):1057–1068, 2005.

[164] K. Etminani, M. Naghibzadeh, and J. M. Peña. DemocraticOP: A democratic way of aggregating Bayesian network parameters. *International Journal of Approximate Reasoning*, 54(5):602–614, 2013.

[165] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley, 5th edition, 2011.

[166] I. Faux and M. Pratt. *Computational Geometry for Design and Manufacture*. Wiley, 1979.

[167] M. P. Fay and M. A. Proschan. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39, 2010.

[168] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, 1996.

[169] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, 1996.

[170] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann, 1993.

[171] A. R. Feinstein and D. V. Cicchetti. High agreement but low kappa: I. The problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549, 1990.

[172] M. Feldman. Morphology of the neocortical pyramidal neuron. In A. Peters and E. Jones, editors, *Cerebral Cortex. Vol. 1. Cellular Components of the Cerebral Cortex*, pages 201–253. Plenum Press, 1984.

[173] D. Feldmeyer, V. Egger, J. Lubke, and B. Sakmann. Reliable synaptic connections between pairs of excitatory layer 4 neurones within a single barrel of developing rat somatosensory cortex. *Journal of Physiology*, 521:169–190, 1999.

[174] J. Feng. *Computational Neuroscience: A Comprehensive Approach*. Chapman & Hall, 2004.

[175] A. Fernández, M. Morales, and A. Salmerón. Tree augmented naive Bayes for regression using mixtures of truncated exponentials: Application to higher education management. In *Advances in Intelligent Data Analysis VII, Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA'07), LNCS 4723*, pages 59–69, 2007.

[176] A. Fernández, J. D. Nielsen, and A. Salmerón. Learning Bayesian networks for regression from incomplete databases. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 18(1):69–86, 2010.

[177] A. Fernández, J. A. Gámez, R. Rumí, and A. Salmerón. Data clustering using hidden variables in hybrid Bayesian networks. In *Proceedings of the 4th International Conference of the European Research Consortium for Informatics and Mathematics - Working Group on Computing & Statistics (ERCIM2011), 2011*, 2011.

[178] A. Figueiredo. Discriminant analysis for the von Mises-Fisher distribution. *Communications in Statistics–Simulation and Computation*, 38(9):1991–2003, 2009.

[179] A. Figueiredo and P. Gomes. Discriminant analysis based on the Watson distribution defined on the hypersphere. *Statistics: A Journal of Theoretical and Applied Statistics*, 40(5):435–445, 2006.

[180] N. I. Fisher. *Statistical Analysis of Spherical Data*. Cambridge University Press, 1987.

[181] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993.

[182] N. I. Fisher and A. J. Lee. Regression models for an angular response. *Biometrics*, 48: 665–677, 1992.

[183] R. A. Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A (Mathematical and Physical Sciences)*, 217(1130):295–305, 1953.

[184] P. A. Flach and N. Lachiche. Confirmation-guided discovery of first-order rules with tertius. *Machine Learning*, 42(1-2):61–95, 2001.

[185] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[186] J. L. Fleiss, J. Cohen, and B. S. Everitt. Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, 72(5):323–327, 1969.

[187] J. L. Fleiss, B. Levin, and M. C. Paik. *Statistical Methods for Rates and Proportions*. Wiley, 3rd edition, 2003.

[188] I. Flesch, A. Fernández, and A. Salmerón. Incremental supervised classification for the MTE distribution: A preliminary study. In I. Rojas and H. Pomares, editors, *Proceedings of the Second Simposio de Inteligencia Computacional (SICO 2007)*, pages 217–224. Thomson, 2007.

[189] M. J. Flores, J. A. Gámez, A. M. Martínez, and J. M. Puerta. GAODE and HAODE: Two proposals based on AODE to deal with continuous variables. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the Twenty-Sixth Annual International Conference on Machine Learning (ICML 2009)*, volume 382 of *ACM International Conference Proceeding Series*, pages 313–320. ACM, 2009.

[190] M. J. Flores, J. A. Gámez, A. M. Martínez, and J. M. Puerta. Handling numeric attributes when comparing Bayesian network classifiers: Does the discretization method matter? *Applied Intelligence*, 34(3):372–385, 2011.

[191] M. J. Flores, J. A. Gámez, A. M. Martínez, and A. Salmerón. Mixture of truncated exponentials in supervised classification: Case study for the naive Bayes and averaged one-dependence estimators classifiers. In S. Ventura, A. Abraham, K. J. Cios, C. Romero, F. Marcelloni, J. M. Benítez, and E. L. G. Galindo, editors, *Proceedings of the Eleventh International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, pages 593–598. IEEE Computer Society, 2011.

[192] M. J. Flores, A. E. Nicholson, A. Brunskill, K. B. Korb, and S. Mascaro. Incorporating expert knowledge when learning Bayesian network structure: A medical case study. *Artificial Intelligence in Medicine*, 53:181–204, 2011.

[193] G. M. Foody. On the compensation for chance agreement in image classification accuracy assessment. *Photogrammetric Engineering and Remote Sensing*, 58(10):1459–1460, 1992.

[194] E. Frank, L. Trigg, G. Holmes, and I. H. Witten. Technical note: Naive Bayes for regression. *Machine Learning*, 41(1):5–25, 2000.

[195] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[196] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)*, pages 125–133. Morgan Kaufmann, 1997.

[197] N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, pages 157–165. Morgan Kaufmann, 1996.

[198] N. Friedman and D. Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–125, 2003.

[199] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[200] N. Friedman, M. Goldszmith, and A. Wyner. Data analysis with Bayesian networks: A bootstrap approach. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pages 196–205. Morgan Kaufmann, 1999.

[201] M. J. Fryer. A review of some non-parametric methods of density estimation. *IMA Journal of Applied Mathematics*, 20(3):335–354, 1977.

[202] G. Fu, F. Y. Shih, and H. Wang. A kernel-based parametric method for conditional density estimation. *Pattern Recognition*, 44:284–294, 2011.

[203] L. D. Fu. A Comparison of State-of-the-Art Algorithms for Learning Bayesian Network Structure From Continuous Data. Master's thesis, Vanderbilt University, Nashville, Tennessee, 2005.

[204] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1): 3–54, 1999.

[205] J. A. Gámez, R. Rumí, and A. Salmerón. Unsupervised naive Bayes for data clustering with mixtures of truncated exponentials. In M. Studený and J. Vomlel, editors, *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM2006)*, pages 123–130, 2006.

[206] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129, 2010.

[207] S. García and F. Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[208] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013.

[209] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan. Statistical methods for eliciting probability distributions. *Journal of the American Statistical Association*, 100(470): 680–701, 2005.

[210] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Advances in Computational Mathematics*, 12(4):377–410, 2000.

[211] R. Gatto and S. R. Jammalamadaka. The generalized von Mises distribution. *Statistical Methodology*, 4(3):341–353, 2007.

[212] D. Geiger and D. Heckerman. Learning Gaussian networks. In R. López de Mántaras and D. Poole, editors, *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1994)*, pages 235–243. Morgan Kaufmann, 1994.

[213] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[214] C. Genest and J. V. Zideck. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–135, 1986.

[215] R. Gentleman, E. Whalen, W. Huber, and S. Falcon. *`graph`: A package to handle graph data structures*, 2012. R package version 1.36.1.

[216] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004.

[217] A. Gerasoulis and T. Yang. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. *Journal of Parallel and Distributed Computing*, 16 (4):276–291, 1992.

[218] J. D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Chapman & Hall, 5th edition, 2010.

[219] J. R. Glaser and E. M. Glaser. Neuron imaging with Neurolucida – A PC-based system for image combining microscopy. *Computerized Medical Imaging and Graphics*, 14(5): 307–317, 1990.

[220] A. Goh and R. Vidal. Unsupervised Riemannian clustering of probability density functions. In W. Daelemans, B. Goethals, and K. Morik, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 377–392. Springer, 2008.

[221] J. H. Goldberg, G. Tamas, D. Aronov, and R. Yuste. Calcium microdomains in aspiny dendrites. *Neuron*, 40(4):807–821, 2003.

[222] I. J. Good. A causal calculus. *Philosophy of Science*, 11:305–318, 1961.

[223] K. A. Greene, J. M. Kniss, G. F. Luger, and C. R. Stern. Satisficing the masses: Applying game theory to large-scale, democratic decision problems. In *International Conference on Computational Science and Engineering*, pages 1156–1162. IEEE Computer Society, 2009.

[224] K. A. Greene, J. M. Kniss, and G. F. Luger. Representing diversity in communities of Bayesian decision-makers. In *IEEE Second International Conference on Social Computing*, pages 315–322. IEEE Computer Society, 2010.

[225] R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3): 297–322, 2005.

[226] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In C. E. Brodley, editor, *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[227] L. Guerra, V. Robles, C. Bielza, and P. Larrañaga. A comparison of clustering quality indices using outliers and noise. *Intelligent Data Analysis*, 16(4):703–715, 2012.

[228] H. Guo and W. Hsu. A survey of algorithms for real-time Bayesian network inference. In H. Guo, E. Horvitz, W. H. Hsu, and E. J. Santos, editors, *Proceedings of the AAAI Workshop on Real-Time Decision Support and Diagnosis Systems*, pages 1–12. AAAI Press, 2002.

[229] H. Guo, B. B. Perry, J. A. Stilson, and W. H. Hsu. A genetic algorithm for tuning variable orderings in Bayesian network structure learning. In R. Dechter and R. S. Sutton, editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 951–952. AAAI Press / The MIT Press, 2002.

[230] Y. Gurwicz and B. Lerner. Bayesian network classification using spline-approximated kernel density estimation. *Pattern Recognition Letters*, 26(11):1761–1771, 2005.

[231] P. Guttorp and R. A. Lockhart. Finding the location of a signal: A Bayesian analysis. *Journal of the American Statistical Association*, 83:322–330, 1988.

[232] H. A. Güvenir, B. Acar, G. Demiröz, and A. Çekin. A supervised machine learning algorithm for arrhythmia analysis. In A. Murray and S. Swiryn, editors, *Computers in Cardiology 1997*, pages 433–436, 1997.

[233] K. L. Gwet. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48, 2008.

[234] K. L. Gwet. *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters*. Advanced Analytics, 3rd edition, 2012.

[235] M. Halavi, K. Hamilton, R. Parekh, and G. A. Ascoli. Digital reconstructions of neuronal morphology: Three decades of research trends. *Frontiers in Neuroscience*, 6(49): 1–11, 2012.

[236] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[237] M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato, 1999.

[238] D. J. Hamilton, G. M. Shepherd, M. E. Martone, and G. A. Ascoli. An ontological approach to describing neurons and their relationships. *Frontiers in Neuroinformatics*, 6(15):1–11, 2012.

[239] P. Hamilton. A language to describe the growth of neurites. *Biological Cybernetics*, 68 (6):559–565, 1993.

[240] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 2000)*, pages 1–12. ACM, 2000.

[241] D. J. Hand and K. Yu. Idiot's Bayes: Not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

[242] L. A. Harris. Bivariate Lagrange interpolation at the Chebyshev nodes. *Proceedings of the American Mathematical Society*, 138(12):4447–4453, 2010.

[243] D. P. Hartmann. Considerations in the choice of interobserver reliability estimates. *Journal of Applied Behavior Analysis*, 10(1):103–116, 1977.

[244] M. Häusser and B. Mel. Dendrites: Bug or features? *Current Opinion in Neurobiology*, 13(3):372–383, 2003.

[245] A. F. Hayes and K. Krippendorff. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77–89, 2007.

[246] S. S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2009.

[247] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Corporation, 1996.

[248] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[249] M. Helmstaeder, B. Sakmann, and D. Feldmeyer. The relation between dendritic geometry, electrical excitability, and axonal projections of L2/3 interneurons in rat barrel cortex. *Cerebral Cortex*, 19(4):938–950, 2009.

[250] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence (UAI 1986)*, pages 149–164. Elsevier, 1986.

[251] H. G. Hentschel and A. van Ooyen. Models of axon guidance and bundling during development. *Proceedings of the Royal Society of London. Series B (Biological Sciences)*, 266:2231–2238, 1999.

[252] L. D. Hernández. Algoritmos de propagación I: Métodos exactos. In J. A. Gámez and J. M. Puerta, editors, *Sistemas Expertos Probabilísticos*, pages 500–506. Universidad de Castilla-La Mancha, 1998.

[253] L. D. Hernández, S. Moral, and A. Salmerón. A Monte Carlo algorithm for probabilistic propagation in belief networks based on importance sampling and stratified simulation techniques. *International Journal of Approximate Reasoning*, 18(1-2):53–91, 1998.

[254] T. Heskes. Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems*, pages 266–272. The MIT Press, 1998.

[255] H. Heumann and G. Wittum. The tree-edit-distance, a measure for quantifying neuronal morphology. *Neuroinformatics*, 7(3):179–190, 2009.

[256] R. F. Hevner, T. Neogi, C. Englund, R. A. M. Daza, and A. Fink. Cajal-Retzius cells in the mouse: Transcription factors, neurotransmitters, and birthdays suggest a pallial origin. *Developmental Brain Research*, 141(1):39–54, 2003.

[257] D. Hillman. Neuronal shape parameters and substructures as a basis of neuronal form. In F. Schmitt, editor, *The Neurosciences: Fourth Study Program*, pages 477–498. The MIT Press, 1979.

[258] P. R. Hof, I. I. Glezer, F. Condé, R. A. Flagg, M. B. Rubin, E. A. Nimchinsky, and D. M. Vogt Weisenhorn. Cellular distribution of the calcium-binding proteins parvalbumin, calbindin, and calretinin in the neocortex of mammals: Phylogenetic and developmental patterns. *Journal of Chemical Neuroanatomy*, 16(2):77–116, 1999.

[259] R. Hofmann and V. Tresp. Discovering structure in continuous variables using Bayesian networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, pages 500–506. The MIT Press, 1995.

[260] S. Højsgaard. Graphical independence networks with the gRain package for R. *Journal of Statistical Software*, 46(10):1–26, 2012.

[261] S. Højsgaard. *gRim: Graphical interaction models*, 2012. URL http://CRAN.R-project.org/package=gRim. R package version 0.1-15.

[262] S. Højsgaard, D. Edwards, and S. Lauritzen. *Graphical Models with R*. Springer, 2012.

[263] K. Hornik and B. Grün. On conjugate families and Jeffreys priors for von Mises-Fisher distributions. *Journal of Statistical Planning and Inference*, 143(5):992–999, 2013.

[264] G. Hripcsak and D. F. Heitjan. Measuring agreement in medical informatics reliability studies. *Journal of Biomedical Informatics*, 35(2):99–110, 2002.

[265] C.-N. Hsu, H.-J. Huang, and T.-T. Wong. Why discretization works for naive Bayesian classifiers. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 399–406. Morgan Kaufmann, 2000.

[266] C.-N. Hsu, H.-J. Huang, and T.-T. Wong. Implications of the Dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers. *Machine Learning*, 53(3):235–263, 2003.

[267] W. H. Hsu, H. Guo, B. B. Perry, and J. A. Stilson. A permutation genetic algorithm for variable ordering in learning Bayesian networks from data. In W. B. Langdon, E. Cantú-Paz, K. E. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. K. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 383–390. Morgan Kaufmann, 2002.

[268] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[269] J. Hughlings. On classification and on methods of investigation. In *Selected Writings of John Hughlings Jackson*. Hodder and Stoughton, 1931.

[270] A. Hussein and E. Santos. Exploring case-based Bayesian networks and Bayesian multinets for classification. In A. Y. Tawfik and S. D. Goodwin, editors, *Proceedings of the Seventeenth Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI-2004)*, volume 3060 of *Lecture Notes in Computer Science*, pages 485–492. Springer, 2004.

[271] R. L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

[272] S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. In *Proceedings of the Seventh Pacific Symposium on Biocomputing*, pages 175–186. World Scientific Press, 2002.

[273] S. Imoto, S. Kim, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, and S. Miyano. Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. In *Proceedings of the First IEEE Bioinformatics Conference of the Computer Society (CBS 2002)*, pages 219–227. IEEE Computer Society, 2002.

[274] T. S. Jaakola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, 1997.

[275] B. Jacobs, M. Schall, M. Prather, E. Kapler, L. Driscoll, S. Baca, J. Jacobs, K. Ford, M. Wainwright, and M. Treml. Regional dendritic and spine variation in human cerebral cortex: A quantitative Golgi study. *Cerebral Cortex*, 11(6):558–571, 2001.

[276] S. R. Jammalamadaka and A. SenGupta. *Topics in Circular Statistics*. World Scientific, 2001.

[277] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[278] F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, 1990.

[279] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In P. Besnard and S. Hanks, editors, *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pages 338–345. Morgan Kaufmann, 1995.

[280] R. A. Johnson and T. E. Wehrly. Some angular-linear distributions and related regression models. *Journal of the American Statistical Association*, 73(363):602–606, 1978.

[281] S. G. Johnson and B. Narasimhan. *Cubature: Adaptive multivariate integration over hypercubes*, 2011. URL http://CRAN.R-project.org/package=cubature. R package version 1.1-1.

[282] M. C. Jones and A. Pewsey. A family of symmetric distributions on the circle. *Journal of the American Statistical Association*, 100(472):1422–1428, 2005.

[283] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, 1996.

[284] J. Jossinet. Variability of impedivity in normal and pathological breast tissue. *Medical and Biological Engineering and Computing*, 34(5):346–350, 1996.

[285] A. Juan and E. Vidal. On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710, 2002.

[286] S. Jung, K. H. Lee, and D. Lee. Enabling large-scale Bayesian network learning by preserving intercluster directionality. *IEICE Transactions on Information and Systems*, E90-D:1018–1027, 2007.

[287] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, School of Computer Science and Engineering, University of New South Wales, 2002.

[288] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, 4th edition, 2000.

[289] S. Kato and M. C. Jones. A family of distributions on the circle with links to, and applications arising from, Möbius transformation. *Journal of the American Statistical Association*, 105(489):249–262, 2010.

[290] S. Kato and M. C. Jones. An extended family of circular distributions related to wrapped Cauchy distributions via Brownian motion. *Bernoulli*, 19(1):154–171, 2013.

[291] S. Kato, K. Shimizu, and G. Shieh. A circular-circular regression model. *Statistica Sinica*, 18(2):633–645, 2008.

[292] W. E. Kaufmann and H. W. Moser. Dendritic anomalies in disorders associated with mental retardation. *Cerebral Cortex*, 10(10):981–991, 2000.

[293] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649, 2001.

[294] J. T. Kent. The Fisher-Bingham distribution on the sphere. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(1):71–80, 1982.

[295] J. H. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference systems. In A. Bundy, editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI 1983)*, pages 190–193. William Kaufmann, 1983.

[296] C. Koch and I. Segev. The role of single neurons in information processing. *Nature Neuroscience*, 3:1171–1177, 2000.

[297] C. Koch, T. Poggio, and V. Torres. Retinal ganglion cells: A functional interpretation of dendritic morphology. *Philosophical Transactions of the Royal Society of London. Series B (Biological Sciences)*, 298(1090):227–263, 1982.

[298] R. A. Koene, B. Tijms, P. van Hees, F. Postma, A. de Ridder, G. J. A. Ramakers, J. van Pelt, and A. van Ooyen. NETMORPH: A framework for the stochastic generation of large scale neuronal networks with realistic neuron morphologies. *Neuroinformatics*, 7 (3):195–210, 2009.

[299] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD 1996)*, pages 202–207. AAAI Press, 1996.

[300] D. Koller and N. Friedman. *Probabilistic Graphical Models. Principles and Techniques*. The MIT Press, 2009.

[301] A. O. Komendantov and G. A. Ascoli. Dendritic excitability and neuronal morphology as determinants of synaptic efficacy. *Journal of Neurophysiology*, 101(4):1847–1866, 2009.

[302] I. Kononenko. Semi-naive Bayesian classifier. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning*, volume 482 of *Lecture Notes in Computer Science*, pages 206–219. Springer, 1991.

[303] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[304] W. L. Kovach. Quantitative methods for the study of lycopod megaspore ultrastructure. *Rev Palaeobot Palynology*, 57(3–4):233–246, 1989.

[305] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[306] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. W. Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, 6:22–31, 2003.

[307] L. Lam and C. Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 27(5):553–568, 1997.

[308] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[309] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In R. López de Mántaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI 1994)*, pages 399–406. Morgan Kaufmann, 1994.

[310] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In W. R. Swartout, editor, *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pages 223–228. AAAI Press / The MIT Press, 1992.

[311] H. Langseth and T. D. Nielsen. Classification using hierarchical naïve Bayes models. *Machine Learning*, 63(2):135–159, 2006.

[312] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Inference in hybrid Bayesian networks. *Reliability Engineering and System Safety*, 94:1499–1509, 2009.

[313] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Maximum likelihood learning of conditional MTE distributions. In C. Sossai and G. Chemello, editors, *Proceedings of the Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2012)*, volume 5590 of *Lecture Notes on Computer Science*, pages 240–251. Springer, 2009.

[314] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51:485–498, 2010.

[315] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53:212–227, 2012.

[316] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Learning mixtures of truncated basis functions from data. In A. Cano, M. Gómez-Olmedo, and T. D. Nielsen, editors, *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM 2012)*, pages 163–170, 2012.

[317] C. A. Lantz and E. Nebenzahl. Behavior and interpretation of the κ statistic: Resolution of the two paradoxes. *Journal of Clinical Epidemiology*, 49(4):431–434, 1996.

[318] A. Larkman. Dendritic morphology of pyramidal neurones of the visual cortex of the rat: I. Branching patterns. *Journal of Comparative Neurology*, 306(2):307–319, 1991.

[319] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans*, 26(4): 487–493, 1996.

[320] S. D. Larson and M. E. Martone. Ontologies for neuroscience: What are they and what are they good for? *Frontiers in Neuroscience*, 3(1):60–67, 2009.

[321] S. L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.

[322] S. L. Lauritzen and F. V. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11(2):191–203, 2001.

[323] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[324] S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1): 31–57, 1989.

[325] A. Lee. Circular data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4): 477–486, 2010.

[326] F. Leitenstorfer and G. Tutz. Knot selection by boosting techniques. *Computational Statistics & Data Analysis*, 51(9):4605–4621, 2007.

[327] P. Leray and O. Francois. BNT structure learning package: Documentation and Experiments. Technical Report FRE CNRS 2645, Laboratoire PSI - INSA Rouen, 2006.

[328] M. P. Lévy. L'addition des variables aléatoires définies sur une circonférence. *Bulletin de la Société Mathématique de France*, 67:1–41, 1939.

[329] R. A. Lew and J. S. Lew. A general permutation test for pairwise agreement. Technical report, IBM T. J. Watson Research Center, 1995.

[330] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nedellec and C. Rouveirol, editors, *Proceedings of the Tenth European Conference on Machine Learning (ECML 1998)*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15. Springer, 1998.

[331] G. H. Li and C. D. Qin. A model for neurite growth and neuronal morphogenesis. *Mathematical Biosciences*, 132(1):97–110, 1996.

[332] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pages 369–376. IEEE Computer Society, 2001.

[333] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[334] K. A. Lindsay, D. J. Maxwell, J. R. Rosenberg, and G. Tucker. A new approach to reconstruction models of dendritic branching patterns. *Mathematical Biosciences*, 205 (2):271–296, 2007.

[335] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998)*, pages 80–86. AAAI Press, 1998.

[336] B. Liu, Y. Ma, and C. K. Wong. Improving an association rule based classifier. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 504–509. Springer, 2000.

[337] P. L. López-Cruz, C. Bielza, and P. Larrañaga. The von Mises naive Bayes classifier for angular data. In J. A. Lozano, J. A. Gámez, and J. A. Moreno, editors, *Advances in Artificial Intelligence, Proceedings of the 14th Conference of the Spanish Association for Artificial Intelligence*, volume 7023 of *Lecture Notes in Computer Science*, pages 145–154. Springer, 2011.

[338] P. L. López-Cruz, C. Bielza, P. Larrañaga, R. Benavides-Piccione, and J. DeFelipe. Models and simulation of 3D neuronal dendritic trees using Bayesian networks. *Neuroinformatics*, 9(4):347–369, 2011.

[339] P. L. López-Cruz, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials from data using B-spline interpolation. In A. Cano, M. Gómez-Olmedo, and T. D. Nielsen, editors, *Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM 2012)*, pages 211–218, 2012.

[340] P. L. López-Cruz, C. Bielza, and P. Larrañaga. Directional naive Bayes classifiers. *Pattern Analysis and Applications*, in press, 2013.

[341] P. L. López-Cruz, C. Bielza, and P. Larrañaga. Learning conditional linear Gaussian classifiers with probabilistic class labels. In C. Bielza, A. Salmerón, and A. Alonso-Betanzos, editors, *Advances in Artificial Intelligence, Proceedings of the 15th Multi-*

Conference of the Spanish Association for Artificial Intelligence, volume 8109 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2013.

[342] P. L. López-Cruz, P. Larrañaga, J. DeFelipe, and C. Bielza. Bayesian network modeling of the consensus between experts: An application to neuron classification. *International Journal of Approximate Reasoning*, in press, 2013.

[343] P. L. López-Cruz, T. D. Nielsen, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials of conditional densities from data. In C. Bielza, A. Salmerón, and A. Alonso-Betanzos, editors, *Advances in Artificial Intelligence, Proceedings of the 15th MultiConference of the Spanish Association for Artificial Intelligence*, volume 8109 of *Lecture Notes in Computer Science*, pages 363–372. Springer, 2013.

[344] R. López de Mántaras and E. Armengol. Machine learning from examples: Inductive and lazy methods. *Data & Knowledge Engineering*, 25(1-2):99–123, 1998.

[345] F. López-Muñoz, J. Boya, and C. Alamo. Neuron theory, the cornerstone of neuroscience, on the centenary of the Nobel Prize award to Santiago Ramón y Cajal. *Brain Research Bulletin*, 70(4-6):391–405, 2006.

[346] A. Luczak. Spatial embedding of neuronal trees modeled by diffusive growth. *Journal of Neuroscience Methods*, 157(1):132–141, 2006.

[347] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. California, USA, 1967.

[348] R. E. Madsen, D. Kauchak, and C. Elkan. Modeling word burstiness using the Dirichlet distribution. In L. De Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 545–552. ACM, 2005.

[349] Z. F. Mainen and T. J. Sejnowski. Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature*, 382:363–366, 1996.

[350] R. Malach. Cortical columns as devices for maximizing neuronal diversity. *Trends in Neurosciences*, 17(3):101–104, 1994.

[351] B. B. Mandelbrot. *The Fractal Geometry of Nature*. Freeman, 1982.

[352] K. V. Mardia. Statistics of directional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 37(3):349–393, 1975.

[353] K. V. Mardia. On some recent advancements in applied shape analysis and directional statistics. In S. Barber, P. D. Baxter, and K. V. Mardia, editors, *Systems Biology & Statistical Bioinformatics*, pages 9–17. Leeds University Press, 2006.

[354] K. V. Mardia. Bayesian analysis for bivariate von Mises distributions. *Journal of Applied Statistics*, 37(3):515–528, 2010.

[355] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 2000.

[356] K. V. Mardia, C. C. Taylor, and G. K. Subramaniam. Protein bioinformatics and mixtures of bivariate von Mises distributions for angular data. *Biometrics*, 63(2):505–512, 2007.

[357] M. Marín-Padilla. Cajal–Retzius cells and the development of the neocortex. *Trends in Neurosciences*, 21(2):64–71, 1998.

[358] H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006.

[359] A. M. Martínez, G. I. Webb, M. J. Flores, and J. A. Gámez. Non-disjoint discretization for aggregating one-dependence estimator classifiers. In E. Corchado, V. Snásel, A. Abraham, M. Wozniak, M. Graña, and S.-B. Cho, editors, *Proceedings of the Seventh International Conference on Hybrid Artificial Intelligent Systems*, volume 7209 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2012.

[360] M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, and J. C. Riquelme. Mining quantitative association rules based on evolutionary computation and its application to atmospheric pollution. *Integrated Computer-Aided Engineering*, 17(3):227–242, 2010.

[361] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, and J. C. Riquelme. An evolutionary algorithm to discover quantitative association rules in multidimensional time series. *Soft Computing*, 15(10):2065–2084, 2011.

[362] A. R. Masegosa and S. Moral. An interactive approach for Bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54 (8):1168–1181, 2013.

[363] I. Matzkevich and B. Abramson. The topological fusion of Bayes nets. In D. Dubois and M. P. Wellman, editors, *Proceedings of the Eight Conference on Uncertainty in Artificial Intelligence*, pages 191–198. Morgan Kaufmann, 1992.

[364] I. Matzkevich and B. Abramson. Deriving a minimal I-map of a belief network relative to a target ordering of its nodes. In D. Heckerman and E. H. Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 159–165. Morgan Kaufmann, 1993.

[365] P. Maynard-Reich II and U. Chajewska. Aggregating learned probabilistic beliefs. In J. S. Breese and D. Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 354–361. Morgan Kaufmann, 2001.

[366] A. K. McAllister. Cellular and molecular mechanisms of dendrite growth. *Cerebral Cortex*, 10(10):963–973, 2000.

[367] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

[368] M. Meilă and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

[369] G. Meyer, A. M. Goffinet, and A. Fairén. What is a Cajal-Retzius cell? A reassessment of a classical cell type based on recent observations in the developing neocortex. *Cerebral Cortex*, 9(8):765–775, 1999.

[370] J. Miina and T. Pukkala. Application of ecological field theory in distance-dependent growth modelling. *Forest Ecology and Management*, 161:101–107, 2002.

[371] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[372] M. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30, 1961.

[373] S. Monti and G. F. Cooper. Learning Bayesian belief networks with neural network estimators. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, pages 578–584. The MIT Press, 1996.

[374] S. Monti and G. F. Cooper. A multivariate discretization method for learning Bayesian networks from mixed data. In G. F. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI 1998)*, pages 404–413. Morgan Kaufmann, 1998.

[375] J. A. Mooney, P. J. Helms, and I. T. Jolliffe. Fitting mixtures of von Mises distributions: A case study involving sudden infant death syndrome. *Computational Statistics & Data Analysis*, 41(3-4):505–513, 2003.

[376] S. Moral, R. Rumí, and A. Salmerón. Mixtures of truncated exponentials in hybrid Bayesian networks. In S. Benferhat and P. Besnard, editors, *Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2001)*, volume 2143 of *Lecture Notes in Artificial Intelligence*, pages 145–167. Springer, 2001.

[377] M. Morales, C. Rodríguez, and A. Salmerón. Selective naive Bayes for regression based on mixtures of truncated exponentials. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 15(6):697–716, 2007.

[378] J. E. Morris and P. J. Laycock. Discriminant analysis of directional data. *Biometrika*, 61(2):335–341, 1974.

[379] F. Mosteller and J. W. Tukey. Data analysis, including statistics. In G. Lindzey and E. Aronson, editors, *Handbook of Social Psychology*, volume 2, pages 80–203. Addison-Wesley, 2nd edition, 1968.

[380] V. B. Mountcastle. *Perceptual Neuroscience: The Cerebral Cortex*. Harvard University Press, 1998.

[381] K. P. Murphy. Inference and learning in hybrid Bayesian networks. Technical Report UCB/CSD-98-990, Computer Science Division, University of California, 1998.

[382] K. P. Murphy. A variational approximation for Bayesian networks with discrete and continuous latent variables. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pages 457–466. Morgan Kaufmann, 1999.

[383] K. P. Murphy. The Bayes net toolbox for Matlab. In E. J. Wegman, A. Braverman, A. Goodman, and P. Smyth, editors, *Proceedings of the Thirty-Third Symposium on the Interface*, pages 331–350. Interface Foundation of North America, 2001.

[384] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[385] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285, 2004.

[386] R. Nagarajan, M. Scutari, and S. Lèbre. *Bayesian Networks in R: with Applications in Systems Biology*. Springer, 2013.

[387] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Publications, 2002.

[388] K. Nigam, A. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2):103–134, 2000.

[389] J. Novovičová and A. Malík. Text document classification based on mixture models. *Kybernetika*, 40(3):293–304, 2004.

[390] K. G. Olesen. Causal probabilistic networks with both discrete and continuous variables. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):275–279, 1993.

[391] M. R. Osborne, B. Presnell, and B. A. Turlach. Knot selection for regression splines via the lasso. In S. Weisberg, editor, *Dimension Reduction, Computational Complexity, and Information*, volume 30 of *Computing Science and Statistics*, pages 44–49. Interface Foundation of North America, 1998.

[392] W. Pan and X. Shen. Penalized model-based clustering with application to variable selection. *Journal of Machine Learning Research*, 8:1145–1164, 2007.

[393] J. D. Park and A. Darwiche. A differential semantics for jointree algorithms. *Artificial Intelligence*, 156(2):197–216, 2004.

[394] M. J. Pazzani. Searching for dependencies in Bayesian classifiers. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V. Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 239–248. Springer, 1995.

[395] J. M. Peña. *On Unsupervised Learning of Bayesian Networks and Conditional Gaussian Networks*. PhD thesis, University of the Basque Country, Spain, 2001.

[396] J. M. Peña. Finding consensus Bayesian network structures. *Journal of Artificial Intelligence Research*, 42:661–687, 2011.

[397] J. M. Peña, J. A. Lozano, and P. Larrañaga. An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21(8):779–786, 2000.

[398] J. M. Peña, J. A. Lozano, P. Larrañaga, and I. Inza. Dimensionality reduction in unsupervised learning of conditional Gaussian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):590–603, 2001.

[399] J. M. Peña, J. A. Lozano, and P. Larrañaga. Learning recursive Bayesian multinets for data clustering by means of constructive induction. *Machine Learning*, 47(1):63–89, 2002.

[400] J. M. Peña, J. A. Lozano, and P. Larrañaga. Unsupervised learning of Bayesian networks via estimation of distribution algorithms: An application to gene expression data clustering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(supp01):63–82, 2004.

[401] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[402] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[403] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[404] D. Peel and G. McLachlan. Robust mixture modeling using the t distribution. *Statistics and Computing*, 10:339–348, 2000.

[405] D. Peel, W. J. Whiten, , and G. J. McLachlan. Fitting mixtures of Kent distributions to aid in joint set identification. *Journal of the American Statistical Association*, 96 (453):56–63, 2001.

[406] D. M. Pennock and M. P. Wellman. Graphical representations of consensus belief. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 531–540. Morgan Kaufmann, 1999.

[407] M. A. Peot. Geometric implications of the naive Bayes assumption. In E. Horvitz and F. V. Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI 1996)*, pages 414–419. Morgan Kaufmann, 1996.

[408] A. Pérez, P. Larrañaga, and I. Inza. Supervised classification with conditional Gaussian networks: Increasing the structure complexity from naive Bayes. *International Journal of Approximate Reasoning*, 43:1–25, 2006.

[409] A. Pérez, P. Larrañaga, and I. Inza. Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, 50(2): 341–362, 2009.

[410] F. Pernkopf and J. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In L. De Raedt and S. Wrobel, editors, *Proceedings of the Twenty-Second International Conference on Machine learning (ICML 2005)*, pages 657–664. ACM, 2005.

[411] F. Perrin. Étude mathématique du mouvement Brownien de rotation. *Annales Scientifiques de l'École Normale Supérieure*, 45:1–51, 1928.

[412] A. Peters and E. G. Jones. *Cerebral Cortex. Cellular Components of the Cerebral Cortex*, volume 1. Plenum Press, 1984.

[413] Petilla Interneuron Nomenclature Group. Petilla terminology: Nomenclature of features of GABAergic interneurons of the cerebral cortex. *Nature Reviews Neuroscience*, 9(7): 557–568, 2008.

[414] A. Pewsey. The wrapped stable family of distributions as a flexible model for circular data. *Computational Statistics & Data Analysis*, 52(3):1516–1523, 2008.

[415] D. T. Pham and G. A. Ruz. Unsupervised training of Bayesian networks for data clustering. *Proceedings of the Royal Society. Series A: Mathematical, Physical and Engineering Science*, 465(2109):2927–2948, 2009.

[416] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–205. The MIT Press, 1998.

[417] R. Popping. On agreement indices for nominal data. In W. E. Saris and I. N. Gallhofer, editors, *Sociometric Research: Volume 1, Data Collection and Scaling*, pages 90–105. St. Martin's Press, 1988.

[418] O. Pourret, P. Naïm, and B. Marcot. *Bayesian Networks: A Practical Guide to Applications*. Wiley, 2008.

[419] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer-Verlag, 2002.

[420] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[421] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[422] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/.

[423] P. Rakic. Confusing cortical columns. *Proceedings of the National Academy of Sciences of the United States of America*, 105(34):12099–12100, 2008.

[424] S. Ramón y Cajal. Estructura de los centros nerviosos de las aves. *Revista Trimestral de Histología Normal y Patológica*, 1:1–10, 1988.

[425] C. E. Rasmussen. The infinite Gaussian mixture model. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 554–560. The MIT Press, 1999.

[426] V. C. Raykar, S. Yu, L. H. Zhao, G. Hermosillo-Valadez, C. Florin, L. Bogoni, L. Moy, and D. Blei. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

[427] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 2008.

[428] L.-P. Rivest and T. Chang. Regression and correlation for 3×3 rotation matrices. *Canadian Journal of Statistics*, 34(2):187–202, 2006.

[429] M. E. Robert and J. D. Sweeney. Computer model: Investigating the role of filopodia-based steering in experimental neurite galvanotropism. *Journal of Theoretical Biology*, 188(3):277–288, 1997.

[430] A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):365–378, 2005.

[431] K. S. Rockland. Five points on columns. *Frontiers in Neuroanatomy*, 4(22):1–10, 2010.

[432] V. Romero, R. Rumí, and A. Salmerón. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42:54–68, 2006.

[433] S. T. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[434] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L-systems*. Academic Press, 1980.

[435] R. Rumí, A. Salmerón, and S. Moral. Estimating mixtures of truncated exponentials in hybrid Bayesian networks. *Test*, 15(2):397–421, 2006.

[436] S. G. Sadeghi, M. J. Chacron, M. C. Taylor, and K. E. Cullen. Neural variability, detection thresholds, and information transmission in the vestibular system. *Journal of Neuroscience*, 27(4):771–781, 2007.

[437] J. Sagrado and S. Moral. Qualitative combination of Bayesian networks. *International Journal of Intelligent Systems*, 18:237–249, 2003.

[438] M. Sahami. Learning limited dependence Bayesian classifiers. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD 1996)*, pages 335–338. AAAI Press, 1996.

[439] A. V. Samsonovich and G. A. Ascoli. Statistical morphological analysis of hippocampal principal neurons indicates cell-specific repulsion of dendrites from their own cell. *Journal of Neuroscience Research*, 71(2):173–187, 2003.

[440] E. Santos and A. Hussein. Case-based Bayesian network classifiers. In V. Barr and Z. Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2004.

[441] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. Part A: On the problem of smoothing of graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4:45–99, 1946.

[442] E. L. Schwartz, editor. *Computational Neuroscience*. The MIT Press, 1993.

[443] G. E. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

[444] E. K. Scott and L. Luo. How do dendrites take their shape? *Nature Neuroscience*, 4 (4):359–365, 2001.

[445] W. A. Scott. Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly*, 19(3):321–325, 1955.

[446] M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[447] R. Segal and O. Etzioni. Learning decision lists using homogeneous rules. In B. Hayes-Roth and R. E. Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 619–625. AAAI Press / The MIT Press, 1994.

[448] T. J. Sejnowski, C. Koch, and P. S. Churchland. Computational neuroscience. *Science*, 241(4871):1299–1306, 1988.

[449] A. SenGupta and S. Roy. A simple classification rule for directional data. In N. Balakrishnan, H. N. Nagaraja, and N. Kannan, editors, *Advances in Ranking and Selection, Multiple Comparisons, and Reliability*, Statistics for Industry and Technology, pages 81–90. Birkhäuser Boston, 2005.

[450] A. SenGupta and F. I. Ugwuowo. A classification method for directional data with application to the human skull. *Communications in Statistics – Theory and Methods*, 40:457–466, 2011.

[451] R. D. Shachter. Intelligent probabilistic inference. In L. N. Kanal and J. F. Lemmer, editors, *Proceedings of the First Annual Conference on Uncertainty in Artificial Intelligence (UAI 1985)*, pages 371–382. Elsevier, 1985.

[452] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[453] R. D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604, 1988.

[454] R. D. Shachter and C. R. Kenley. Gaussian influence diagrams. *Management Science*, 35(5):527–550, 1989.

[455] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[456] P. P. Shenoy. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning*, 53(5):847–866, 2012.

[457] P. P. Shenoy and G. Shafer. Axioms for probability and belief functions propagation. In R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer, editors, *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1988)*, pages 169–198. North-Holland, 1990.

[458] P. P. Shenoy and J. C. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657, 2011.

[459] G. M. Shepherd, editor. *The Synaptic Organization of the Brain*. Oxford University Press, 5th edition, 2004.

[460] C. C. Sherwood, C. D. Stimpson, C. Butti, C. J. Bonar, A. L. Newton, J. M. Allman, and P. R. Hof. Neocortical neuron types in xenarthra and afrotheria: Implications for brain evolution in mammals. *Brain Structure and Function*, 213(3):301–328, 2009.

[461] P. E. Shrout. Measurement reliability and agreement in psychiatry. *Statistical Methods in Medical Research*, 7(3):301–317, 1998.

[462] S. Siegel and N. J. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 2nd edition, 1988.

[463] J. Sim and C. C. Wright. The kappa statistic in reliability studies: Use, interpretation, and sample size requirements. *Physical Therapy*, 85(3):257–268, 2005.

[464] P. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66: 191–243, 1994.

[465] D. J. Spiegelhalter. Probabilistic reasoning in predictive expert systems. In L. N. Kanal and J. F. Lemmer, editors, *Proceedings of the First Annual Conference on Uncertainty in Artificial Intelligence (UAI 1985)*, pages 47–68. Elsevier, 1985.

[466] P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, Prediction & Search*. The MIT Press, 2nd edition, 2000.

[467] R. L. Spitzer and J. L. Fleiss. A reanalysis of the reliability of psychiatric diagnosis. *British Journal of Psychiatry*, 125(10):341–347, 1974.

[468] N. Spruston. Pyramidal neurons: Dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221, 2008.

[469] S. Sra. A short note on parameter approximation for von Mises-Fisher distributions: And a fast implementation of $I_s(x)$. *Computationl Statistics*, 27(1):177–190, 2012.

[470] J. F. Staiger, I. Flagmeyer, D. Schubert, K. Zilles, R. Kötter, and H. J. Luhmann. Functional diversity of layer IV spiny neurons in rat somatosensory cortex: Quantitative morphology of electrophysiologically characterized and biocytin labeled cells. *Cerebral Cortex*, 14(6):690–701, 2004.

[471] H. Steck and T. S. Jaakkola. Predictive discretization during model selection. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *Proceedings of the Twenty-Sixth Symposium of the German Association for Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 1–8. Springer, 2004.

[472] R. B. Stein, E. R. Gossen, and K. E. Jones. Neuronal variability: Noise or part of the signal? *Nature Reviews Neuroscience*, 6:389–397, 2005.

[473] V. Steuber, E. De Schutter, and D. Jaeger. Passive models of neurons in the deep cerebellar nuclei: The effect of reconstruction errors. *Neurocomputing*, 58-60:563–568, 2004.

[474] C. J. Stone, M. H. Hansen, C. Kooperberg, and Y. K. Truong. Polynomial splines and their tensor products in extended linear modeling. *The Annals of Statistics*, 25(4): 1371–1470, 1997.

[475] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.

[476] M. H. Stone. The generalized Weierstrass approximation theorem. *Mathematics Magazine*, 21(5):237–254, 1948.

[477] J. Su and H. Zhang. Full Bayesian network classifiers. In W. W. Cohen and A. Moore, editors, *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML 2006)*, volume 148 of *ACM International Conference Proceeding Series*, pages 897–904. ACM, 2006.

[478] J. Su, H. Zhang, C. X. Ling, and S. Matwin. Discriminative parameter learning for Bayesian networks. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML 2008)*, volume 307 of *ACM International Conference Proceeding Series*, pages 1016–1023. ACM, 2008.

[479] H. J. Suermondt and G. F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, 4(4): 283–306, 1990.

[480] H. J. Suermondt and G. F. Cooper. Initialization for the method of conditioning in Bayesian belief networks. *Artificial Intelligence*, 50(1):83–94, 1991.

[481] A. Sumida, I. Terazawa, A. Togashi, and A. Komiyama. Spatial arrangement of branches in relation to slope and neighbourhood competition. *Annals of Botany*, 89(3): 301–310, 2002.

[482] M. Svensén and C. M. Bishop. Robust Bayesian mixture modelling. *Neurocomputing*, 64:235–252, 2005.

[483] K. Svoboda. The past, present, and future of single neuron reconstruction. *Neuroinformatics*, 9(2-3):97–98, 2011.

[484] A. Tanabe, K. Fukumizu, S. Oba, T. Takenouchi, and S. Ishii. Parameter estimation for von Mises-Fisher distributions. *Computational Statistics*, 22:145–157, 2007.

[485] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[486] F. Thabtah. A review of associative classification mining. *The Knowledge Engineering Review*, 22(01):37–65, 2007.

[487] B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of DAG models. In G. F. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 504–513. Morgan Kaufmann, 1998.

[488] B. Torben-Nielsen, K. Tuyls, and E. O. Postma. Shaping realistic neuronal morphologies: An evolutionary computation method. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, pages 573–580. IEEE, 2006.

[489] B. Torben-Nielsen, K. Tuyls, and E. O. Postma. On the neuronal morphology-function relationship: A synthetic approach. In K. Tuyls, R. Westra, Y. Saeys, and A. Nowé, editors, *Proceedings of the First International Workshop Knowledge Discovery and Emergent Complexity in Bioinformatics (KDECB 2006)*, volume 4366 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2007.

[490] B. Torben-Nielsen, K. Tuyls, and E. O. Postma. EvOL-Neuron: Neuronal morphology generation. *Neurocomputing*, 71:963–972, 2008.

[491] B. Torben-Nielsen, S. Vanderlooy, and E. O. Postma. Non-parametric algorithmic generation of neuronal morphologies. *Neuroinformatics*, 6:257–277, 2008.

[492] T. P. Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, 2nd edition, 2010.

[493] E. G. Tsionas. Bayesian analysis of finite mixtures of Weibull distributions. *Communications in Statistics - Theory and Methods*, 31(1):37–48, 2002.

[494] H. B. M. Uylings and J. van Pelt. Measures for quantifying dendritic arborizations. *Network: Computation in Neural Systems*, 13:397–414, 2002.

[495] H. B. M. Uylings, A. Ruiz-Marcos, and J. Van Pelt. The metric analysis of three-dimensional dendritic tree patterns: A methodological review. *Journal of Neuroscience Methods*, 18:127–151, 1986.

[496] T. V. Van and T. Pham-Gia. Clustering probability distributions. *Journal of Applied Statistics*, 37(11):1891–1910, 2010.

[497] J. Van Pelt and H. B. M. Uylings. Modeling the natural variability in the shape of dendritic trees: Application to basal dendrites of small rat cortical layer 5 pyramidal neurons. *Neurocomputing*, 26-27:305–311, 1999.

[498] J. Van Pelt and H. B. M. Uylings. Natural variability in the geometry of dendritic branching patterns. In G. N. Reeke, R. R. Poznanski, K. A. Lindsay, J. R. Rosenberg, and O. Sporns, editors, *Modeling in the Neurosciences: From Biological Systems to Neuromimetic Robotics*, pages 89–116. CRC Press, 2005.

[499] J. van Pelt and H. B. M. Uylings. The flatness of bifurcations in 3D dendritic trees: An optimal design. *Frontiers in Computational Neuroscience*, 5(54):1–27, 2012.

[500] J. Van Pelt, A. van Ooyen, and H. B. M. Uylings. Modeling dendritic geometry and the development of nerve connections. In E. De Schutter, editor, *Computational Neuroscience: Realistic Modeling for Experimentalists*, pages 179–208. CRC Press, 2001.

[501] M. P. Van Veen and J. Van Pelt. Terminal and intermediate segment lengths in neuronal trees with finite length. *Bulletin of Mathematical Biological*, 55:277–294, 1993.

[502] P. Vannoorenberghe and P. Smets. Partially supervised learning by a credal EM approach. In L. Godo, editor, *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 956–967. Springer, 2005.

[503] J. J. Verbeek, N. A. Vlassis, and B. J. A. Kröse. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.

[504] R. W. H. Verwer and J. van Pelt. Analysis of binary trees when occasional multifurcations can be considered as aggregates of bifurcations. *Bulletin of Mathematical Biology*, 52(2):629–641, 1990.

[505] R. W. H. Verwer, J. van Pelt, and H. B. M. Uylings. An introduction to topological analysis of neurones. In M. G. Stewart, editor, *Quantitative Methods in Neuroanatomy*, pages 292–323. Wiley, 1992.

[506] P. Vetter, A. Roth, and M. Häusser. Propagation of action potentials in dendrites depends on dendritic morphology. *Journal of Neurophysiology*, 85(2):926–937, 2001.

[507] R. Vilalta, G. Blix, and L. Rendell. Global data analysis and the fragmentation problem in decision tree induction. In *Proceedings of the Ninth European Conference on Machine Learning Machine Learning (ECML 1997)*, volume 1224 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 1997.

[508] R. von Mises. Uber die "Ganzzahligkeit" der Atomgewichte und verwandte Fragen. *Physikal Z*, 19:490–500, 1918.

[509] M. Wand and B. Ripley. *KernSmooth: Functions for kernel smoothing for Wand & Jones (1995) "Kernel Smoothing"*, 2012. URL http://CRAN.R-project.org/package=KernSmooth. R package version 2.23-8.

[510] Q. Wang, S. R. Kulkarni, and S. Verdú. A nearest-neighbor approach to estimating divergence between continuous random vectors. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2006)*, pages 242–246. IEEE, 2006.

[511] Y. Wang, A. Gupta, M. Toledo-Rodríguez, C. Zhi Wu, and H. Markram. Anatomical, physiological, molecular and circuit properties of nest basket cells in the developing somatosensory cortex. *Cerebral Cortex*, 12(4):395–410, 2002.

[512] G. I. Webb. Discovering associations with numeric variables. In D. Lee, M. Schkolnick, F. J. Provost, and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge discovery and Data Mining (ACM SIGKDD 2001)*, pages 383–388. ACM, 2001.

[513] G. I. Webb, J. R. Boughton, and Z. Wang. Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58:5–24, 2005.

[514] E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[515] Q. Wen, A. Stepanyants, G. N. Elston, A. Y. Grosberg, and D. B. Chklovskii. Maximization of the connectivity repertoire as a statistical principle governing the shapes of dendritic arbors. *Proceedings of the National Academy of Sciences of the United States of America*, 106(30):12536–12541, 2009.

[516] N. Wermuth. Linear recursive equations, covariance selection, and path analysis. *Journal of the American Statistical Association*, 75(372):963–972, 1980.

[517] E. White. *Cortical Circuits: Synaptic Organization of the Cerebral Cortex. Structure, Function and Theory*. Birkhauser, Boston, 1989.

[518] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

[519] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[520] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[521] A. T. Wood. Simulation of the von Mises-Fisher distribution. *Communications in Statistics - Simulation and Computation*, 23(1):157–164, 1994.

[522] C. Wu, E. Ivanova, J. Cui, Q. Lu, and Z.-H. Pan. Action potential generation at an AIS-like process in the axonless retinal AII amacrine cell. *Journal of Neuroscience*, 31 (41):14654–14659, 2012.

[523] B. Xie, W. Pan, and X. Shen. Variable selection in penalized model-based clustering via regularization on grouped parameters. *Biometrics*, 64:921–930, 2008.

[524] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[525] R. Xu and D. Wunsch. *Clustering*. Wiley-IEEE Press, 2008.

[526] Y. Yang and G. I. Webb. On why discretization works for naive-Bayes classifiers. In T. D. Gedeon and L. C. C. Fung, editors, *Proceedings of the Sixteenth Australian Conference on Artificial Intelligence*, volume 2903 of *Lecture Notes in Computer Science*, pages 440–452. Springer, 2003.

[527] Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 101–116. Springer, 2010.

[528] E. A. Yfantis and L. E. Borgman. An extension of the von Mises distribution. *Communications in Statistics - Theory and Methods*, 11(15):1695–1706, 1982.

[529] W. Yu and B. Lu. Synapses and dendritic spines as pathogenic targets in Alzheimer's disease. *Neural Plasticity*, 2012(247150):1–8, 2012.

[530] G. U. Yule. Notes on the theory of association of attributes in statistics. *Biometrika*, 2:121–134, 1903.

[531] R. Yuste. *Dendritic spines*. The MIT Press, 2010.

[532] R. Yuste. Dendritic spines and distributed circuits. *Neuron*, 71(5):772–781, 2011.

[533] R. Yuste and T. Bonhoeffer. Genesis of dendritic spines: Insights from ultrastructural and imaging studies. *Nature Reviews Neuroscience*, 5:24–34, 2004.

[534] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[535] R. S. Zemel, C. K. I. Williams, and M. C. Mozer. Lending direction to neural networks. *Neural Networks*, 8(4):503–512, 1995.

[536] M.-L. Zhang and Z. Zhi-Hua. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineerings*, in press, 2013.

[537] Y. Zhang, K. Yue, M. Yue, and W. Liu. An approach for fusing Bayesian networks. *Journal of Information & Computational Science*, 8(2):194–201, 2011.

[538] Z. Zheng and G. I. Webb. Lazy learning of Bayesian rules. *Machine Learning*, 41(1): 53–84, 2000.

[539] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.

[540] Z. Zong. *Information-Theoretic Methods for Estimating Complicated Probability Distributions*. Elsevier, 2006.

[541] Z. Zong and K. Lam. Estimation of complicated distributions using B-spline functions. *Structural Safety*, 20(4):341–355, 1998.