DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

PhD THESIS

Contributions to Bayesian network classifiers and interneuron classification

Author

Bojan Mihaljević
MS Artificial Intelligence

PhD supervisors

Pedro Larrañaga
PhD Computer Science

Concha Bielza
PhD Computer Science

2018

Thesis Committee

President: Víctor Robles Forcada, Universidad Politécnica de Madrid

External Member: Marco Scutari, University of Oxford

Member: Robert Castelo, Universitat Pompeu Fabra

Member: Javier de Felipe, Consejo Superior de Investigaciones Científicas

Secretary: Juan Antonio Fernández del Pozo, Universidad Politécnica de Madrid

To my parents, Svjetlana and Dragan, who
with help from grandma Ruža and grandpa Savo
brought me up lovingly and selflessly,
and my godparents, Despina and Mihalis

Acknowledgements

This has been a long journey and many people have made good things happen along the way. This is an opportunity to mention at least some of them.

I would have not entered research had it not been for my advisors, Concha Bielza and Pedro Larrañaga. I am very grateful to them for their guidance and all the things I have learned during these years. Collaborating with Javier DeFelipe and Ruth Benavides-Piccione has been a privilege, as has my three-month stay at École Polytechnique Fédérale de Lausanne with Sean Hill. Pilar Flores Romero has been very helpful with operative matters related to the Cajal Blue Brain and Human Brain projects.

The research carried out in this thesis has been supported by funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 785907 (HBP SGA2), by the Spanish Ministry of Economy and Competitiveness through the Cajal Blue Brain (C080020-09; the Spanish partner of the EPFL Blue Brain initiative) and TIN2016-79684-P projects, by the Regional Government of Madrid through the S2013/ICE-2845-CASI-CAM-CM project, and by a Fundación BBVA grant to Scientific Research Teams in Big Data 2016. I thankfully acknowledge a UPM grant that funded my three-months stay at the École Polytechnique Fédérale de Lausanne.

Part of my research builds on work by Pedro López-Cruz and Luis Guerra. Luis Rodriguez-Luján has helped me by creating the NeuroSTR library. I think I have learned something from everyone in our research group and a big thank you is due to all. Lida Kanari assisted me with the set of neurons from the Markram laboratory.

This work would absolutely have not been possible without software and resources that people release for free on the internet. I want to specially thank the authors of R, the devtools and bnlearn R packages, R markdown, git, vim, and the GNU utilities for making my life easier. Special thanks to Hadley Wickham, for all the tools and the freely available R books.

I want to thank my colleagues, past and present, at the Computational Intelligence Group, for all the nice moments and the support. I especially want to mention Sergio, Iñaki, Luis, Marco and Martín, who all made it to my wedding in Serbia, as well as Gherardo and Alberto. Thanks to Elisabeta, Christian, Silvia, and Ernesto for making my stay in Geneva much nicer.

Lastly, but most importantly, the biggest thank you is for my loving wife Nina. Both of us know that this work, like anything else I do, is yours as much as it is mine. The look in your eyes once this journey is completed will be my dearest prize.

Abstract

Machine learning is concerned with the automatic extraction of patterns from data. Its most common form is supervised classification, where we learn a predictive model from examples labeled as belonging to one out of a number of possible classes. One type of model, based on encoding the probability distribution over the examples and the classes, are Bayesian network classifiers. Most search and score algorithms for learning Bayesian network classifiers traverse the space of directed acyclic graphs (DAGs), making arbitrary yet possibly suboptimal arc directionality decisions. This can be remedied by learning in the space of DAG equivalence classes. This is commonly done for general Bayesian networks but has not been widely applied for Bayesian network classifiers. First, we identify the smallest subspace of DAGs that covers all possible class-posterior distributions when data is complete. All the DAGs in this space, which we call minimal class-focused DAGs (MC-DAGs), are such that their every arc is directed towards a child of the class variable. Second, we adapt the greedy equivalence search (GES) by adding operator validity criteria which ensure GES only visits states within our space. Third, we specify how to efficiently evaluate the discriminative score of a GES operator for an MC-DAG in time independent of the number of variables and without converting the completed partially DAG, which represents an equivalence class, into a DAG. Our adapted algorithm has shown promising results on preliminary experiments.

Few of the many proposed methods for learning Bayesian network classifiers are available in popular software tools. Some promising methods are only implemented as standalone, often scarcely documented, programs, while many others lack available implementations. We have implemented the bnclassify package for the R environment for statistical computing that provides a unified interface and documentation to a number of such algorithms. Structure learning algorithms include variants of the greedy hill-climbing search that can be combined with discriminative scores, an adaptation of the Chow-Liu algorithm, and the averaged one-dependence estimator. Parameter estimation includes naive-Bayes-specific methods based on discriminative score optimization and Bayesian model averaging.

A major problem in neuroscience is to determine the types of GABAergic interneurons. These neurons are very diverse with regards to morphological, electro-physiological, molecular, and synaptic properties. Most researchers consider that interneurons can be grouped into types with much less variability within types than among them. Yet, among the many morphological types established in the literature, only the chandelier and Martinotti types have widely accepted definitions. Some authors expect that a fully data-driven approach will solve this with an objective classification by clustering neurons according to their molecular, morphological, and electrophysiological features. Currently, however, the data are too scarce for this and are almost always limited to either morphological, electro-physiological, or molecular features.

We learned supervised classifiers from neuron morphologies, pre-classified into some of the established interneuron types. We were interested in whether these types, some of which lack widely accepted definitions, could be distinguished by quantitative models. Also, interpretable and accurate models could allow us to better understand these types. Leveraging

prior knowledge on existing types, via class labels, may allow us to extract insight from the existing scarce data, which is likely insufficient for a fully unsupervised classification.

We used two sets of neuron morphology reconstructions. The first, Gardener's, set consisted of 237 cells which were classified by 42 leading neuroscientists into ten interneuron types. The degree of agreement among neuroscientists varied widely, with 29 cells such that at least 35 neuroscientists agreed on its type, yet 67 cells such that no more than 15 agreed on a type. The neuroscientists also also classified the cells according to four categorical features of axonal morphology, largely agreeing on their definitions. Learning to reproduce the classification of this representative group of neuroscientists could provide objective, consensus-based, models of interneuron type and features. We first learned separate Bayesian network classifier models for the type and four axonal features, labelling each cell by its majority vote among the 42 neuroscientists. We did this with different subsets of neurons, formed by increasing the threshold on label reliability, which we defined as the minimal number of neuroscientists agreeing on the majority type. Second, we used probabilistic class labels while predicting the five variables at once. For this end, we proposed an instance-based classifier that handles instances with probabilistic labels encoded as Bayesian networks. This classifier predicts the multi-dimensional probabilistic labels by forming a consensus among a set of Bayesian networks. Third, we considered a descriptive, rather than predictive, approach, by semi-supervised projected clustering of the data. We 1) unlabelled some of instances; 2) initialized a cluster for each type with the cells of that type; and 3) clustered the unlabeled cells, into either the clusters of known types or newly created clusters. We sought potential subtypes of the established types, while estimating localized feature relevance for the types/subtypes. For this, we adapted a mixture of Gaussians with localized feature selection by simplifying its definition of feature irrelevance.

The second, Markram's, set of reconstructions consisted of 217 of rat interneurons from the Markram laboratory, pre-classified into one of seven morphological types, with a single class label per cell. We trained classifiers for each type in a one-versus-all fashion by applying state-of-the-art learning algorithms. With seven chandelier and 15 bitufted —yet 123 basket— cells, the sample was insufficient to accurately distinguish each of the seven types. We were able to, however, learn accurate and interpretable models for the Martinotti, basket, and nest basket types, and moderately accurate models for the double boquet and small basket types.

Resumen

El aprendizaje automático trata de la extracción automática de patrones a partir de datos. Su formato más común es la clasificación supervisada, donde aprendemos un modelo predictivo a partir de ejemplos etiquetados como miembros de una de las varias clases posibles. Un tipo de modelo, basado en el modelado de la distribución de probabilidad sobre los ejemplos y las clases, son los clasificadores basados en redes Bayesianas. La mayoría de los algoritmos de búsqueda y puntuación para aprender clasificadores basados en redes Bayesianas recorren el espacio de grafos dirigidos acíclicos (DAGs), tomando arbitrariamente decisiones posiblemente suboptimas sobre la direccionalidad de arcos. Esto puede ser evitado mediante el aprendizaje dentro del espacio de clases de equivalencia de DAGs pero, sin embargo, esto no se ha aplicado ampliamente para clasificadores basados en redes Bayesianas. Primero, identificamos el subespacio de DAGs mínimo que cubre todas las posibles distribuciones a posteriori de la clase cuando los datos son completos. Los DAGs en este espacio, al que llamamos el de los DAGs enfocados en la clase mínimos (MC-DAGs), tienen todos los arcos dirigidos hacia un nodo que es hijo del nodo clase. Segundo, adaptamos el algoritmo greedy equivalence search (GES) añadiendo criterios de validez de operadores de búsqueda que aseguran que el GES adaptado solamente visite estados que están dentro de nuestro espacio de búsqueda. Tercero, especificamos cómo evaluar de manera eficiente puntuaciones discriminativas de operadores del GES para los MC-DAGs en tiempo que es independiente del número de las variables y sin convertir el DAG parcialmente dirigido completado, que representa a una clase de equivalencia, a un DAG. El GES adaptado ha mostrado buenos resultados en experimentos preliminares.

Pocos de los métodos propuestos para el aprendizaje de clasificadores basados en redes Bayesianas están disponibles en herramientas de software populares. Algunos métodos prometedores solamente están implementados como programas independientes, comúnmente poco documentados, mientras que para otros no se tienen implementaciones públicamente disponibles. Hemos implementado el paquete bnclassify, para el lenguaje y entorno para el análisis estadístico R, que provee una interfaz y documentación unificada para varios tales métodos. Entre los algoritmos para el aprendizaje de estructura es incluyen variantes de la búsqueda voráz hill climbing, que se pueden combinar con puntuaciones discriminativas, una adaptación del algoritmo Chow-Liu y el averaged one-dependence estimator. La estimación de parámetros incluye métodos específicos para el naive Bayes, como los basados en optimización de puntuaciones discriminativas y en el Bayesian model averaging.

Un problema importante dentro de la neurociencia es la determinación de los tipos de interneuronas GABAergicas. Estas neuronas son muy diversas con respecto a propiedades morfológicas, electro-fisiológicas, moleculares, y sinapticas. La mayoría de los investigadores consideran que las interneuronas se pueden agrupar en tipos, con mucha menos variedad dentro de los tipos que entre ellos. Sin embargo, dentro de los muchos tipos morfológicos establecidos en la literatura, solamente los tipos candelabro y Martinotti tienen definiciones ampliamente aceptadas entre los neurocientíficos. Algunos autores consideran que un enfoque completamente dirigido por datos resolvera este problema mediante una clasificación objetiva

a través del clustering de neuronas tomando en cuenta sus propiedades morfológicas, electrofisiológicas, y moleculares. Actualmente, sin embargo, los datos existentes no son suficientes para ello, y casi siempre están limitados a una de las tres dimensiones – morfológicas, electrofisiológicas, y moleculares – y no a una combinación de varias.

Hemos aprendido clasificadores supervisados a partir de datos morfológicos, pre-clasificados en tipos establecidos de interneuronas. Estábamos interesados en saber si estos tipos, algunos de los cuales no tienen definiciones ampliamente aceptadas, podrían ser distinguidos por un modelo cuantitativo. También, un modelo de alto porcentaje de clasificación correcta e interpretable podría permitir una mejor comprensión de esos tipos. Utilizar conocimiento previo, mediante las etiquetas de clase, nos puede permitir extraer conocimiento a datos limitados, con los cuales una clasificación completamente no-supervisada probablemente no es viable.

Hemos usado dos conjuntos de reconstrucciones de morfologías neuronales. El primero, el conjunto del Jardinaro, consistía en 237 células clasificadas por 42 neurocientíficos punteros en diez tipos de interneuronas. El nivel de acuerdo entre los neurocientíficos variaba mucho, habiendo 29 células en las cuales al menos 35 neurocientíficos habían coincidido en el tipo, pero también 67 tales que no más de 15 neurocientíficos habían coincidido en un tipo de interneurona. Los neurocientíficos también clasificaron las células de acuerdo a cuatro características categóricas de morfología axonal y en gran médida sí coincidían en los valores para éstas. Aprender a reproducir la clasificación de interneuronas hecha por este grupo representativo de neurocientíficos podría proveer modelos objetivos, basados en el consenso, del tipo de interneuronas y las características axonales. Primero, hemos aprendido modelos independientes con clasificadores Bayesianos discretos para el tipo interneuronal y las cuatro características axonales, etiquetando cada célula con su voto mayoritario entre los 42 neurocientíficos. Hemos considerado múltiples subconjuntos de neuronas, formados al incrementar el umbral de fiabilidad de la etiqueta, definida como el número mínimo de neurocientíficos que han coincidido en la etiqueta mayoritaria. Segundo, hemos considerado etiquetas de clase probabilísticas prediciendo las cinco variables clase a la vez. Para ello, hemos propuesto un clasificador instance-based que puede tratar con etiquetas probabilísticas codificadas como redes Bayesianas. El clasificador predice las etiquetas probabilísticas multi-dimensionales formado un consenso entre un conjunto de redes Bayesianas. Tercero, hemos considerado un enfoque más descriptivo que predictivo, a través del clustering proyectado semi-supervisado. 1) desetiquetamos algunas de las instancias; 2) inicializamos un cluster para cada tipo, asignándole todos las células del dicho tipo; 3) hacemos clustering de las células desetiquetadas, un clusters correspondientes a los tipos existentes o clusters nuevos. Con ello buscabamos potenciales subtipos de los tipos establecidos, a la vez que identificabamos variables localmente relevantes para los clusters. Para ello, hemos adaptado un modelo de mixtura de Gausianas con selección de variables localizada, simplificando la definición de irrelevancia de una variable.

El segundo conjunto, de Markram, consistia en 217 reconstrucciones de interneuronas de ratas del laboratorio Markram. Los autores de las reconstrucciones las habían clasificado en uno de ocho tipos morfológicos, por lo cual cada célula tenía una etiqueta de clase única. Hemos entrenado un clasificador para cada tipo, del modo uno contra todos, aplicando clasificadores supervisados punteros. Con siete neuronas candelabro y 15 neuronas bitufted —pero 123 células basket—, la muestra no era suficiente para modelos con alto porcentaje de acierto para todos los tipos. Sin embargo, hemos aprendido modelos interpretables con alto porcentaje de acierto para los tipos Martinotti, basket, y nest basket, y con un porcentaje de acierto mediano para los tipos double boquet y small basket.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is concerned with the automatic extraction of patterns from data. It is increasingly applied across both science and industry [Jordan and Mitchell, 2015]. This addoption is driven by the large amounts of data being stored and processed, the low cost of computation, and the need for extracting knowledge from this data.

The most common form of machine learning is supervised classification, where we learn a predictive model from examples labeled as belonging to one out of a number of possible classes. Many different classifiers exist, offering different tradeoffs which make them suitable for different settings.

One family of such models, based on encoding the probability distribution over the examples and the classes, are Bayesian network classifiers [Bielza and Larrañaga, 2014b, Friedman et al., 1997]. Most search and score algorithms for learning Bayesian network classifiers traverse the space of directed acyclic graphs (DAGs), making arbitrary yet possibly suboptimal arc directionality decisions. This can be remedied by learning in the space of DAG equivalence classes. This is common for general Bayesian networks but has not been widely applied for Bayesian network classifiers (en exception is Acid et al. [2005]). First, we identify the smallest subspace of DAGs that covers all possible class-posterior distributions when data is complete. All the DAGs in this space, which we call minimal class-focused DAGs (MC-DAGs), are such that their every arc is directed towards a child of the class variable. Second, we adapt the greedy equivalence search (GES) by adding operator validity criteria which ensure GES only visits states within our space. Third, we specify how to efficiently evaluate the discriminative score of a GES operator for an MC-DAG in time independent of the number of variables and without converting the completed partially DAG, which represents an equivalence class, into a DAG. Our adapted algorithm has shown promising results on preliminary experiments.

Although a lot of research has been carried out on Bayesian network classifiers [Bielza and Larrañaga, 2014b], most proposals are not available in commonly-used software tools. Some are only implemented as standalone, often scarcely documented, programs, while many are not available at all. We have implemented the bnclassify package for the R environment for statistical computing [R Core Team, 2015] that provides a unified interface and documentation to a number of such algorithms. Structure learning algorithms include variants of the greedy hill-climbing search that can be combined with discriminative scores, an adaptation of the Chow-Liu algorithm, and the averaged one-dependence estimator. Parameter estimation includes naive-Bayes-specific methods based on discriminative score optimization and Bayesian model averaging. The implementation is efficient enough to handle computation-intensive discriminative scores with medium-sized data sets. The interpretad nature and syntax of the R programming language allow the package to be used without advanced programming skills.

Cortical interneurons are very diverse with regards to morphological, electro-physiological, molecular, and synaptic properties [Fairen et al., 1984, Peters and Jones, 1984, White, 1989, DeFelipe, 1993, Kawaguchi and Kubota, 1997, Markram et al., 2004, Jiang et al., 2015, Tremblay et al., 2016]. Most researchers consider that interneurons can be grouped into types [Ascoli et al., 2008] with much less variability within types than among them. There is, however, no unique catalogue of types [Ascoli et al., 2008, DeFelipe et al., 2013]. High-throughput generation of data is expected to enable learning a systematic taxonomy within a decade [Zeng and Sanes, 2017], by clustering [Tasic et al., 2016, Cauli et al., 1997] molecular, morphological, and electrophysiological features. Currently, however, researchers use [e.g., Markram et al., 2015] and refer to established morphological types such as chandelier, Martinotti, neurogliaform, and basket [Markram et al., 2004, DeFelipe et al., 2013, Feldmeyer et al., 2018, Tremblay et al., 2016]. These types are identified on the basis of the target innervation location —e.g., the peri-somatic area for basket cells— and somatodendritic and axonal morphological features. The latter can be subjective and lead to different classifications: e.g., while Wang et al. [2002] distinguish between large, nest, and small basket cell types, based on features such as axonal arbor density and branch length, DeFelipe et al. [2013] only distinguish between large and common basket types. The different classification schemes [Markram et al., 2004, DeFelipe et al., 2013] only partially overlap. There is, however, consensus on the morphological features of the chandelier, Martinotti, and neurogliaform types [DeFelipe et al., 2013].

Having a model to automatically classify interneurons into these morphological types [Armañanzas and Ascoli, 2015] could bring insight and be useful to practitioners [DeFelipe et al., 2013]. A sufficiently simple and accurate model would provide an interpretable mapping from the quantitative characteristics to the types, such as, for example, the classification tree [Breiman et al., 1984] model by Toledo-Rodriguez et al. [2005] relating mRNA expression to anatomical type. Unlike with the neuroscientist, a classifier's assignment of an interneuron into a particular type can be understood by analyzing the model. Also, many models can quantify the confidence in their decision. Identifying cells that cannot be reliably classified into any of the *a priori* known types might lead to refining the classification taxonomy —these cells might belong to a novel type— or suggest that the boundary between a pair of types is unclear —if many interneurons are very likely to belong to both of them simultaneously. Sufficiently accurate models could then be used by all practitioners to 'objectively' classify interneurons, rather than each of them assigning their own classification. Furthermore, learning such models may help enable future unsupervised type discovery by identifying and fostering the development and definition of useful morphometrics. With thousands of neuronal morphology reconstructions [Parekh and Ascoli, 2013, Ascoli, 2006] available at online repositories such as Neuromorpho.org [Ascoli et al., 2007, 2017] and the Allen Brain Cell Types Database[1], this seems more attainable than ever, especially for the rodent brain.

There are, however, practical obstacles and aspects to consider when learning such models. First, it is important that class labels (i.e., the *a priori* classification) are assigned according to well-established criteria, to avoid learning idiosyncrasies of the annotating neuroscientist. Second, reconstructions at Neuromorpho.org are often incomplete (e.g., insufficient axonal length or interrupted axons), lack relevant metadata, such as the cell body's cortical area and layer, and there is a lot of variability if combining data across species, age, brain region [DeFelipe, 1993], as well as histological, imaging, and reconstruction protocol [Scorcioni et al., 2004, Polavaram et al., 2014, Peng et al., 2015], whereas focusing on a homogeneous data set shrinks the sample size. Third, infinitely many morphometrics [Uylings and Van Pelt, 2002] —variables that quantify morphological features— can be computed and their choice will influence the model [Kong et al., 2005]. While the Petilla convention [Ascoli et al., 2008] provided a reference point by identifying a set of features to distinguish interneuron types, only some of them are readily quantified with software such as L-Measure [Scorcioni et al., 2008] and Neurolucida Explorer (MicroBrightField), as many either rely on often-missing metadata (e.g., laminar extent), or are vaguely defined (e.g., 'dense plexus of highly branched axons'). Indeed, researchers have often recurred to quantifying interneurons with custom-computed morphometrics [Helmstaedter et al., 2009b,c, Dumitriu et al., 2007, Markram et al., 2015].

We trained models in supervised [Murphy, 2012, Hastie et al., 2009, Guerra et al., 2011] and semi-supervised fashion, with the cells pre-classified into some of the established interneuron types. This could tell whether these types, some of which lack widely accepted definitions, could be distinguished by quantitative models. Second, interpretable and accurate models could allow us to better understand these types. Leveraging prior knowledge on existing types, via class labels, may allow us to extract insight from the existing scarce data, likely insufficient for a fully unsupervised classification.

We used two sets of neuron morphology reconstructions. The first, Gardener's set, [DeFelipe et al., 2013] consisted of 237 cells which were classified by 42 leading neuroscientists into ten interneuron types. The cells come from different cortical areas of the rat, mouse, and monkey. The degree of agreement among neuroscientists varied widely, with 29 cells such that at least 35 neuroscientists agreed on its type, yet 67 cells such that no more than 15 agreed on a type. The neuroscientists also also classified the cells according to four categorical features of axonal morphology, largely agreeing on their definitions. Learning to reproduce the classification of this representative group of neuroscientists could provide objective, consensus-based, models of interneuron type and features. DeFelipe et al. [2013] obtained limited accuracy for the interneuron type and one of the axonal features, by using majority vote class labels and considering multiple classifiers. We first learned separate discrete Bayesian network classifier

[1]http://celltypes.brain-map.org/

models for the type and four axonal features, labelling each cell by its majority vote among the 42 neuroscientists. We did this with different subsets of neurons, formed by increasing the threshold on label reliability, which we defined as the minimal number of neuroscientists agreeing on the majority type. Second, we used probabilistic class labels while predicting the five variables at once. For this end, we proposed an instance-based classifier that handles instances with probabilistic labels encoded as Bayesian networks. This classifier predicts the multi-dimensional [Bielza et al., 2011] probabilistic labels by forming a consensus among a set of Bayesian networks. Third, we considered a descriptive, rather than predictive, approach, by semi-supervised projected clustering of the data. We 1) unlabelled some of instances; 2) initialized a cluster for each type with the cells of that type; and 3) clustered the unlabeled cells, into either the clusters of known types or newly created clusters. We sought potential subtypes of the established types, while estimating localized feature relevance for the types/subtypes. For this, we adapted a mixture of Gaussians model with localized feature selection [Guerra et al., 2013b] by simplifying its definition of feature irrelevance.

The second, Markram's, set of reconstructions consisted of 217 rat interneurons [Ramaswamy et al., 2015], which Markram et al. [2015] used for simulating the cortical column. Each cell was pre-classified into one of seven morphological types, with a single class label per cell, mainly on the basis of criteria by Markram et al. [2004], Wang et al. [2002, 2004] regarding the soma's layer and anatomical features. We trained classifiers for each type in a one-versus-all fashion by applying state-of-the-art learning algorithms. The sample contained only seven chandelier and 15 bitufted cells, yet 50 Martinotti and 123 basket ones.

1.1 Hypotheses and objectives

Out research hypotheses are:

- Completed partially DAGs can be usefully leveraged for learning Bayesian network classifiers
- Supervised and semi-supervised learning can produce accurate and interpretable models of GABAergic interneuron types

Based on these hypotheses, this thesis pursues the following specific objectives:

- Adapt the GES algorithm to learn Bayesian network classifiers in the space of equivalence classes
- Implement state-of-the-art Bayesian network classifiers in an R package
- Propose and implement morphometrics for quantifying interneuron morphologies
- Learn accurate supervised classifiers for the Gardener's scheme from interneurons classified by 42 leading neuroscientists
- Develop a method to predict multi-dimensional probabilistic labels encoded as Bayesian networks-
- Apply semi-supervised projected clustering to validate / discover Gardener's interneuron types / subtypes and identify relevant variables

- Learn accurate and interpretable per-type models of the Markram interneuron types

1.2 Document organization

In Part I we introduce notation and provide necessary background. In Part II we present our present our proposal for learning Bayesian network classifiers with equivalence classes, as well as the bnclassify R package. In Part III we present the contributions to interneuron classification. In Part IV we summarize our contributions and list publications and developed software. Appendix A lists the morphometrics used for classifying Markram interneurons, including the definitions of custom morphometrics. Appendix B provides detailed results for the classification of Markram interneurons. Appendix C provides additional results on the neuroscientists' classification of Gardener's interneurons.

Part I

BACKGROUND

Chapter 2

Notation and terminology

We denote random variables with uppercase letters (e.g., X or Y) and their values with corresponding lowercase letters (e.g., x or y). We denote vector with boldface letters (e.g., $\mathbf{X}$, $\mathbf{x}$). Ω_X is the domain of a random variable X and $|\Omega_X|$ the cardinality of Ω_X. Ω_X is finite for a discrete variable X while $\Omega_X = [a, b] \subseteq \mathbb{R}$ for a real-valued variable X. A probability distribution for a discrete X is denoted $P(x)$ and a probability density function for a continuous X is denoted $p(x)$. The indicator function $\mathbb{I}(a)$ returns 1 if its argument a is true and 0 otherwise.

In supervised classification, we have vector of n predictor variables or features $\mathbf{X} = (X_1, \ldots, X_n)$ and a discrete class variable C. When $\Omega_C = \{c_0, c_1\}$ we talk of a positive class c_1 and a negative class c_0. We have a data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, c^{(i)})\}_1^N$ consisting of N instances (data points, examples) $\mathbf{x}^{(i)}$ with their class label $c^{(i)}$. We say that the data are complete if every $\mathbf{x}$ as a full instantiation of all variables in $\mathbf{X}$ (C is observed for all instances by assumption). A classifier is a function $f\colon \Omega_{X_1} \times \ldots \times \Omega_{X_n} \mapsto \Omega_C$. A learning algorithm produces $\hat{f}$, an estimate of f, from a training set of observed values of $\mathbf{X}$ and C.

A graph $\mathcal{G}$ is a pair $(\mathbf{V}, \mathbf{E}_\mathcal{G})$, where $\mathbf{V}$ is the set of nodes and $\mathbf{E}_\mathcal{G}$ the set of edges connecting the nodes in $\mathbf{V}$. En edge may be undirected or directed. An undirected edge $E \in \mathbf{E}_\mathcal{G}$ is a pair $\{V_i, V_j\}$ with $V_i, V_j \in \mathbf{V}$. A directed edge, or arc, is an ordered pair (V_i, V_j) encoding the arc directed from V_i to V_j. We also denote an undirected edge $\{V_i, V_j\}$ as $V_i - V_j$ and an arc (V_i, V_j) as $V_i \to V_j$. An undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}_\mathcal{G})$ is such that every $E \in \mathbf{E}_\mathcal{G}$ is undirected, whereas a directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}_\mathcal{G})$ contains only directed edges. A directed path is a sequence of edges over $V_i \ldots V_k$ such that for each $i \in \{1, \ldots, k-1\}$, either $V_i \to V_{i+1}$ or $V_i - V_{i+1}$ is in $\mathbf{E}_\mathcal{G}$, with at least one edges being directed. A cycle is a directed path $V_i = V_k$ such that $V_i = V_k$. A directed acyclic graph (DAG) is a directed graph with no cycles. An acyclic partially DAG (PDAG) $\mathcal{P}$ may contain both directed and undirected edges and has no directed cycles.

$\mathbf{Ch}_\mathcal{P}(X)$ denotes the children of X in $\mathcal{P}$, $\mathbf{Ch}_\mathcal{P}(X)$ the parents of X if $\mathcal{P}$, $\mathbf{Nbr}_\mathcal{P}(X)$ the neighbour nodes connected to X in $\mathcal{P}$ by an undirected edge, and $\mathbf{Adj}_\mathcal{P}(X) = \mathbf{Ch}_\mathcal{P}(X) \cup \mathbf{Pa}_\mathcal{P}(X) \cup \mathbf{Nbr}_\mathcal{P}(X)$ the nodes adjacent to X in $\mathcal{P}$. A v-structure is an ordered triple of nodes (X, Y, Z) such that $\mathcal{P}$ contains the edges $X \to Y$ and $Z \to Y$, and X and Z are not adjacent in $\mathcal{P}$. These definition are easily specialized for DAGs. We drop the subscript to $\mathbf{Adj}(\cdot)$, $\mathbf{Nbr}(\cdot)$, $\mathbf{Ch}(\cdot)$, and $\mathbf{Pa}(\cdot)$ when clear from context.

In chapters of Part III notation differs somewhat from the convention presented here. We specify notation details in those cases.

Chapter 3

Machine learning

3.1 Introduction

Machine learning is concerned with the automatic extraction of patterns from data. Within the field of artificial intelligence, it is used for tasks such as computer vision, speech recognition, and natural language processing. The reason is that it is easier to train a system to do this by showing it examples of desired input-output pairs than to program it by anticipating the output for every possible input [Jordan and Mitchell, 2015]. Machine learning is increasingly applied across both science and industry, with use examples such as cell type detection and fraud detection. We can only expect this trend to continue, driven by the large amounts of data being stored and processed, the low cost of computation, and the need for extracting knowledge from this data.

This chapter provides a brief introduction to machine learning topics relevant to this thesis. For a thorough treatment we suggests the books by [Bishop, 2007, Murphy, 2012, Hastie et al., 2009, Duda et al., 2000].

Section 3.2 presents an overview of machine learning, and Section 3.3 introduces basic topics in supervised classification. Section 3.4 treats model selection and assessment while Section 3.5 deals with loss functions. Section 3.6, Section 3.7, and Section 3.8 treat, respectively, learning with class labels from multiple-annotators, with probabilistic class labels, and with class label noise. Section 3.9 covers class imbalance. Section 3.10 briefly treats feature selection. Section 3.11 and Section 3.12 present, respectively, the classifiers and feature selection methods used in Chapter 11.

3.2 Overview

The most most common types of machine learning are supervised and unsupervised learning. In supervised, or predictive, learning we want to learn from data $\mathcal{D} = \{(\mathbf{x}^{(i)}, c^{(i)})\}_1^N$ a function $\hat{f}$ that maps the features $\mathbf{X}$ to the values of a target variable C. This is called classification when C is discrete and regression when C is real-valued. A typical example of classification is deciding whether an email is spam or ham given frequency counts of words from a dictionary [Androutsopoulos et al., 2000]. The optimal prediction for an instance $\mathbf{x}$ is the class $c^* = \arg\max_c P(c \mid \mathbf{x})$. The goal of learning, then, is to estimate $P(c \mid \mathbf{x})$ from $\mathcal{D}$. We want a model that generalizes well, accurately classifying instances (e.g., emails) that were not included in $\mathcal{D}$.

A more general setting is multi-output or multi-dimensional classification [Bielza et al., 2011], where we have a vector of class variables $\mathbf{C}$ instead of a single variable C. Its special case is multi-label classification [Tsoumakas et al., 2009], where an instance is associated with a subset of classes Ω_C. For example, a news article can be labeled as related to both religion and politics. This can be viewed as multi-dimensional classification with $|\Omega_C|$ binary variables indicating the presence or absence of a class. In such settings, prediction can be improved by improved by taking into account dependencies among the class variables [Read et al., 2009]. We can explicitly model them with Bayesian networks (see Chapter 4).

In unsupervised, or descriptive, learning the data set contains only observations of the features $\mathbf{X}$, $\mathcal{D} = \{\mathbf{x}^{(i)}\}_1^N$. The most common task in this setting is clustering [Jain, 2010, Everitt et al., 2011]. Clustering assumes that there is an unobserved class variable C and assign the instances in $\mathcal{D}$ to the classes in Ω_C, that is, set values for the unobserved variable C. Examples include customer segmentation for targeted advertising [Berkhin, 2006] and cell type detection from flow-cytometry data [Lo, 2009]. In addition to clustering, unsupervised learning includes density estimation, or the estimation of $P(\mathbf{x})$ for discrete and $p(\mathbf{x})$ for real-valued $\mathbf{X}$, and dimensionality reduction. A useful tool for density estimation are Bayesian networks as they can encode distributions over many variables by assuming conditional independencies (see Chapter 4).

Unsupervised learning is more widely applicable than supervised learning because it does not require the instances to be labeled. Getting data to be labeled by experts can costly and time-consuming. An intermediate setting is semi-supervised learning [Chapelle et al., 2006, Zhu and Goldberg, 2009], with L ($L < N$) data instances labeled and $N - L$ instances lacking a label. We may use the unlabeled instances to train a better predictive model for the classes present in $\mathcal{D}$, or learn a descriptive model, such as the clustering the unlabeled instances by assigning values to their corresponding instantiations of C. Crowd-sourcing services, such as Amazon's Mechanical Turk[1], can provide inexpensive class labels by many annotators [e..g, Snow et al., 2008], that is, a vector $\mathbf{c}^{(i)} = (c_1^{(i)}, \ldots, c_{R_i}^{(i)})$ of candidate class labels by R_i annotators for each $\mathbf{x}^{(i)}$. These labels are possibly noisy as the annotators are often not experts in the domain.

Besides the practical value of predicting an unknown value, a predictive model can be descriptive of a domain. For example, we can read conditional independencies among the variables from the learned structure of a Bayesian network model. We may obtain a compact model of a domain with a subset of the variables $\mathbf{X}$ by feature selection [Guyon and Elisseeff, 2003, Guyon et al., 2006].

One can distinguish between parametric and non-parametric models. A simple supervised

[1]Available at https://www.mturk.com.

non-parametric model is the k-nearest neighbors (k-nn) classifier [Aha, 1997, Fix and Hodges, 1989]. It approximates $P(c \mid \mathbf{x})$ with the estimate from the k nearest neighbors of $\mathbf{x}$ in $\mathcal{D}$ according to, for example, Euclidean distance if $\mathbf{X}$ are real-valued. While this is reasonable for a low number of features n, as n grows the neighbors become more distant and a reliable estimation requires an exponentially larger N. An alternative is to assume that the variables follow some a parametric distribution. For example, the naive Bayes [Minsky, 1961] classifier may assume that, for each value c of the class variable, a feature X_j follows a Gaussian distribution with mean $\mu_{j,c}$ and variance $\sigma^2_{j,c}$. With k-nn, the effective number of parameters that we are estimating is $\frac{N}{k}$, while with the naive Bayes it is fixed and does not depend on N. The less flexible parametric approach may allow us to learn a useful model with a smaller N, as long as its assumptions are reasonable.

3.3 Supervised classifiers

Ideally, $\hat{f}$ minimizes the expected loss, $\mathbb{E}[L(c, \hat{f}(\mathbf{x}))]$, where the expectation is over the unknown true distribution of $\mathbf{X}$ and C and $L(\cdot, \cdot)$ is a loss function that penalizes misclassifications. In classification, the most common one is 1/0 loss, with $L(\hat{f}(\mathbf{x})), c) = \mathbb{I}(c = \hat{f}(\mathbf{x}))$. A k-nn with $k = 1$ will achieve 0 loss on the training data $\mathcal{D}$ by memorizing the class of each instance. However, $\mathcal{D}$ almost certainly contains noise and a model that learns it perfectly may be more complex than the true generating model. Learning thus generally combines the optimization of a loss function on $\mathcal{D}$ with the restriction, or regularization, of model complexity. Means for such restriction include penalizing objective loss functions with a term that is a function of the number of parameters, as well as making simplifying assumptions, such as that the features are conditionally independent given the class, as does the naive Bayes classifier.

One useful distinction is between discriminative and generative models. Generative ones, such as the naive Bayes, model the joint distribution over the class and the features, $P(c, \mathbf{x})$, while discriminative ones model the class-conditional distribution, $P(c \mid \mathbf{x})$. Generative models can handle missing data and allow for semi-supervised settings more easily than generative ones. Discriminative models, on the other hand, allow for easy feature pre-processing, such as replacing X_1 with X_1^2. The discriminative approach of modeling the distribution of interest, $P(c \mid \mathbf{x})$, rather than $P(c, \mathbf{x})$, gives a lower asymptotic loss (as the number of training instances becomes large). The generative model, however, may converge to its asymptotic loss faster than the discriminative one [Ng and Jordan, 2001].

There are thus many different models, with different assumptions about the true distribution of $P(c \mid \mathbf{x})$. They can be learned with different algorithms, which offer different runtime/accuracy trade-offs. No single model is best for all possible domains [Wolpert, 1996], because assumptions that work well in one domain may fail in another. Some models, however, often outperform others. Fernández-Delgado et al. [2014] found that that random forest and support vector machines with Gaussian kernels (see Section 3.11) were significantly better than other 177 classifiers on 121 data sets from the UCI repository [Dheeru and

Karra Taniskidou, 2017].

3.4 Model selection and assessment

Model selection refers to choosing one model out of several of of different complexity. For example, we might want to pick the k parameter for the k-nn. A related task is to assess how well our model generalizes to unseen data. While the training set estimate of the expected loss is optimistic, we can approximate it with a data set that we did not use for training. That is, we train models on a training set $\mathcal{D}^t$, compute their losses on a validation set $\mathcal{D}^v$, and choose the model M with the optimal loss l. Note that l is not an unbiased estimate of expected loss of M, since it comes from data, $\mathcal{D}^v$, which we used to pick the very model M. We would thus need a third, test set, $\mathcal{D}^{test}$, to estimate the loss of M. For example, we might use 50% of our data for training, 25% for validation, and 25% for testing.

We often want to use all available data for training, and thus cannot afford separate validation and test sets. We can get unbiased estimates of the expected loss with resampling techniques such as cross-validation Stone [1974] and boostrap [Efron, 1979]. Cross-validation randomly partitions $\mathcal{D}$ into k roughly equal-sized subsets called folds. We use each of the k folds to compute the loss of the model learned from the remaining k - 1 folds. The average loss over the k folds gives an estimate of the loss of a model learned on all of $\mathcal{D}$. Each point is predicted once yet used for training k-1 times, and thus the per-fold losses are not independent for k $>$ 2. If we do both model selection and assessment, we need nested cross-validation. Stratified cross-validation is a variant such that the proportion of the class is roughly the same as all folds.

3.5 Loss functions

As mentioned above, the most common loss function, or predictive performance metric, is the 1/0 loss. The 1/0 loss is also known as misclassification rate, while $1 - $ the misclassification rate is known as classification accuracy.

Alternative metrics are useful in some settings, such as when the cost of misclassification is asymmetric. For example, we prefer to classify a spam email as valid than to classify a valid email as spam. If spam is the positive class, this means that we prefer false negative (FN; spas as valid email) over false positive (FP) misclassifications. While the 1/0 loss does not distinguish among these errors, metrics such as the following do

$$\text{sensitivity} = \frac{TP}{TP + FN},$$

$$\text{specificity} = \frac{TN}{TN + FP},$$

$$\text{F-measure} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FN} + \text{FP}},$$

	‘Positive’	‘Negative’
Positive	TP	FN
Negative	FP	TN

Table 3.1: A confusion matrix for two classes. The rows represent to true class and the columns the predicted class. TP: true positives; FP: false positives; FN: false negatives; TN: true negatives.

where TP are the true positive classifications (see Table 3.1).

3.6 Multiple annotators

Rather than having a unique class label $c^{(i)}$ per instance $\mathbf{x}^{(i)}$, we may have a vector of R_i labels $\mathbf{c}^{(i)} = (c_1^{(i)}, \ldots, c_{R_i}^{(i)})$, each label provided by a different expert or annotator. An example [Raykar et al., 2010] is that of different radiologists providing subjective class labels by looking for malignant lesions at medical image, while the true label (i.e., ground truth) can only be determined by a biopsy. There may be strong disagreement among the annotators, for example, due to different degrees of expertise.

In order to train and evaluate a classifier with such multi-annotator labels we need an estimate of the ground truth. Simple estimators of the ground truth are: 1) majority voting, i.e., the one selected by most annotators, $\arg\max_{c_j} \sum_{r=1}^{R_i} \mathbb{I}(c_r^{(i)} = c_j)$; and 2) label frequencies, giving a probabilistic ground truth $P(c^{(i)} = c_j) = \frac{1}{R_i} \sum_{r=1}^{R_i} \mathbb{I}(c_r^{(i)} = c_j)$. Method 1) allows us to use any supervised classifier and many can be adapted to deal with method 2) (see Section 3.7). Both approaches, however, assume that all annotators are equally accurate. This does not hold in many settings, notably when learning from a *crowd* of annotators [Snow et al., 2008, Sorokin and Forsyth, 2008, Welinder et al., 2010, Raykar et al., 2010, Raykar and Yu, 2012]. Thus, the general approach is to model annotator reliability in order to decrease the influence of the less reliable ones on the ground truth estimate Dawid and Skene [1979], Whitehill et al. [2009], Welinder et al. [2010], Raykar et al. [2010], Raykar and Yu [2012]. While this adds parameters to be learned (the annotators' reliability), it can be better than the simple approaches. The method by Raykar et al. [2010], which learns the classifier and ground truth labels simultaneously, outperformed majority voting on three real-world data sets. It was better even with five annotators, when three of them were novices, although in one of the samples majority voting was better with less than 40 annotators.

3.7 Probabilistic labels

Probabilistic or soft labels are natural when class membership is inherently uncertain, such as in text categorization [Thiel et al., 2007, Schwenker and Trentin, 2014]. When a classifier cannot be adapted to consider such labels, an alternative is to replicate training instances according to the soft labels.

3.8 Label noise

Noise in a finite sample may occur in the features and in the class labels. We may thus have a unique class label c per each instance yet this label might, due to reasons such as lack of information or human mistakes, be wrong [Frénay and Verleysen, 2014]. Methods for dealing with label noise include data cleaning and modelling label uncertainty. One data cleaning method is that by Brodley and Friedl [1999], which considers instances misclassified by different models, that is, models from different families, as mislabeled.

3.9 Class imbalance

Class imbalance occurs when some classes are much less numerous that others. Many objective functions for learning classifiers implicitly optimize classification accuracy, which can be high on a imbalanced set even with a poor prediction of the minority class. Approaches for dealing with class imbalance [He and Garcia, 2009] include training data sampling and cost-sensitive learning. Oversampling augments the training set with instances of the minority class. The SMOTE [Chawla et al., 2002] method, in particular, creates a synthetic instance of the minority class by randomly choosing a point on the line between some minority class instance $\mathbf{x}$ and one of its k nearest neighbors from the minority class. Undersampling, on the other hand, involves removing a number of instances from the majority class. One has to determine the number of instances to add to (remove from) the training set; albeit a balanced training set is usually desirable, many synthetic minority class instances can lead to overfitting, while losing many majority class instances can mean losing valuable information [He and Garcia, 2009].

In small-sample class-imbalance settings, univariate feature selection can improve predictive performance more than over- and under-sampling, at least for the support vector machine [Wasikowski and Chen, 2010].

3.10 Feature selection

Feature selection [Liu and Motoda, 2007, Guyon et al., 2006, Saeys et al., 2007] deals with selecting a subset of the variables to learn the model with. The goals are [Guyon and Elisseeff, 2003] to improve the model, reduce the amount of data to be collected and processed, and provide better understanding of the process that generated the data.

In supervised learning, the basic distinction is among filter, wrapper or embedded methods. Filters are pre-processing steps previous and independent of model learning. Wrappers [Kohavi and John, 1997], no the other hand, use the model as a blackbox to evaluate sets of features. Embedded methods select features simultaneously with model learning. They include Bayesian network structure learning algorithms that may render a feature independent from the class node.

Another basic distinction is between univariate and multivariate feature selection. Univariate feature selection evaluates each feature independently. While fast an statistically robust, it does not, for example, detect redundant features. Multivariate feature selection considers sets of variables. Since the 2^n possible feature subsets cannot be evaluated except for a small n, one usually uses heuristic search algorithms, such as greedy hill-climbing, to traverse the space of feature subsets.

3.11 Used classifiers

3.11.1 Naive Bayes

The naive Bayes [Minsky, 1961] is a simple approximation to the joint probability distribution $P(C, \mathbf{X})$. It assumes that predictors are conditionally independent given the class and classifies an instance according to

$$c^* = \arg\max_c P(c|\mathbf{x}) \propto P(c) \prod_{j=1}^{n} p(x_j|c).$$

Albeit a simple model, the naive Bayes often performs well, generally due to its low variance. See Chapter 4 for more details and for Bayesian network classifiers other than the naive Bayes.

3.11.2 Linear discriminant analysis

Like multinomial regression, the LDA [Fisher, 1936, Rao, 1948] is a linear classifier, with piecewise hyperplanar decision boundaries. It assumes $p(\mathbf{x} \mid c_l) = \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma})$, that is, multivariate normal class-conditional distributions, with an $\boldsymbol{\mu}_l$ mean vector for each class c_l and a shared covariance matrix $\boldsymbol{\Sigma}$, equal for all classes. The quadratic discriminant analysis model has a covariance matrix $\boldsymbol{\Sigma}_l$ for each class. With diagonal $\boldsymbol{\Sigma}_l$, that corresponds to the naive Bayes.

3.11.3 Regularized logistic regression

According to the (binomial) logistic regression model [e.g., Hastie et al., 2009, Chapter 4], the log odds of a class c_0 and class c_1 are a linear function of $\mathbf{x}$:

$$\ln \frac{P(c_0 \mid \mathbf{x})}{P(c_1 \mid \mathbf{x})} = \beta_0 + \boldsymbol{\beta}^T \mathbf{x},$$

where β_0 and $\boldsymbol{\beta}$ are the model's coefficients. While the $\boldsymbol{\beta}$ can be fit by maximum likelihood estimation, regularizing the model by shrinking them can reduce variance. The lasso [Tibshirani, 1996] regularization finds the $\boldsymbol{\beta}$ by maximizing:

$$\max_{\beta_0, \boldsymbol{\beta}} \frac{1}{N} \sum_{i=1}^{N} \log P(c^{(i)} \mid \mathbf{x}^{(i)}) - \lambda \sum_{j=1}^{n} |\beta_j|,$$

where $P(c^{(i)} \mid \mathbf{x}^{(i)})$ is the probability, under the model, of $c^{(i)}$ given $\mathbf{x}^{(i)}$, while λ specifies the degree of penalty on the magnitude of the coefficients. The lasso tends to shrink some coefficients to zero, effectively selecting, for interpretation purposes, the non-zero coefficient variables (features with $\beta_j = 0$ are effectively omitted from the model). The $\boldsymbol{\beta}$ coefficients are straightforward to interpret: keeping all other predictors fixed, a unit increase in a improved predictor X_j increases the log-odds of the positive class by β_j. Thus, the higher $|\beta_j|$, the more useful is X_j. For groups of correlated predictors, lasso tends to keep a single non-zero coefficient and shrink the rest to zero. Implementations such as the `glmnet` package [Friedman et al., 2010] can efficiently optimize λ according to the cross-validated estimate of a loss function such as classification error.

3.11.4 Support vector machine

The SVM [Boser et al., 1992, Cortes and Vapnik, 1995] finds the maximal margin hyperplane that separates the two classes. It uses kernel functions to project the data onto a higher dimensional space, where they are more likely to be linearly separable. It searches for a separating hyperplane, determined by a coefficient vector $\boldsymbol{\beta}$ and an intercept β_0, by finding

$$\min_{\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}} \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} + R\sum_{i=1}^{N}\xi^{(i)}$$

$$\text{subject to } \xi^{(i)} \geq 0,\ c^{(i)}\phi(\boldsymbol{\beta}^T\mathbf{x}^{(i)} + \beta_0) \geq 1 - \xi^{(i)},\ \forall i,$$

with $c^{(i)} \in \{-1, 1\}$, $\xi^{(i)} = 0$ if $\mathbf{x}^{(i)}$ is on the correct side of the hyperplane, and $R > 0$ is the complexity parameter, with larger values narrowing the margin and yielding less training set misclassifications, while ϕ maps $\mathbf{x}$ to a higher dimensional space. ϕ is given by a kernel function K such that $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$. A common example is the radial basis function, $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma||\mathbf{x} - \mathbf{x}'||^2\right)$, whose parameter $\gamma > 0$ indicates spread from the target instance $\mathbf{x}$.

3.11.5 CART

The CART algorithm [Breiman et al., 1984] produces a classification tree by recursively partitioning the training samples according to a single predictor at a time. For each node a of the tree, CART selects the splitting predictor X_j, and its threshold value t, by minimizing 'class impurity' G,

$$\min_{j,t}\{G(\mathcal{D}^a_{jt}) + G(\mathcal{D}^a \setminus \mathcal{D}^a_{jt})\},$$

where $\mathcal{D}^a$ is a subset of $\mathcal{D}$ at node a, while $\mathcal{D}^a_{jt}$ and $\mathcal{D}^a \setminus \mathcal{D}^a_{jt}$ are the left and right splits, respectively, of $\mathcal{D}^a$ according to X_j and threshold t,

$$\mathcal{D}^a_{jt} = \{(\mathbf{x}^{(i)}, c^{(i)}) : x^{(i)}_j \leq t, \mathbf{x}^{(i)} \in \mathcal{D}^a\}.$$

One measure of impurity is the Gini criterion,

$$G(\mathcal{D}^a) = \sum_{l=1}^{k} P_{\mathcal{D}^a}(c_l)(1 - P_{\mathcal{D}^a}(c_l)),$$

where $P_{\mathcal{D}^a}$ is the empirical probability of class c_l in $\mathcal{D}^a$. Deep trees can overfit the data, and options for regulating complexity include $|\mathcal{D}^a|$, the minimum size of $\mathcal{D}^a$ required in order to attempt a split, and $|\mathcal{D}^l|$, the minimum size of some leaf node $\mathcal{D}^l$.

3.11.6 Random forest

A CART tree can overfit the training data. Besides pruning, another way to reduce variance is to use an ensemble of trees, such as the random forest classifier [Breiman, 2001]. One draws T bootstrap [Efron, 1979] samples (size N samples from $\mathcal{D}$ with replacement), and on each learns an unpruned CART tree. At each split, consider only $m \leq n$ randomly selected features. To make a prediction, choose the majority class among the T trees. Due to averaging over bootstrap samples, the random forest is generally robust to overfitting.

3.11.7 k-nearest neighbors

k-nn [Aha, 1997, Fix and Hodges, 1989] classifies an instance $\mathbf{x}$ according to its nearest neighbors in feature space, by choosing the most common class label among them. The number of neighbors k is a parameter to the model, with a lower value reducing bias but increasing variance (a lower k fits the training data better). The neighbors are usually identified using a variant of the Minkowski distance, such as Euclidean distance. A common extension is to predict c^* by giving more importance to the points that are closer to the target point. Kernel functions are a common means of expressing such weight functions, with weights decreasing smoothly with distance from the target point $\mathbf{x}$ [see, e.g., Hechenbichler and Schliep, 2004].

3.12 Used feature selection methods

3.12.1 Kruskal-Wallis test

The null hypothesis of the Kruskal-Wallis test [Kruskal and Wallis, 1952] is that the medians of k samples are the same. In our case, these samples correspond to the k different classes. It is a non-parametric procedure and as such it does not assume that the data follow a particular distribution. Its special case for $k = 2$ is the Mann-Whitney-Wilcoxon test [Wilcoxon, 1945, Mann and Whitney, 1947]. The test statistic H_j, for some feature X_j, is

$$H_j = (N-1)\frac{\sum_{l=1}^{k} N_l(\bar{r}_{l\cdot} - \bar{r})^2}{\sum_{l=1}^{k}\sum_{i=1}^{N_l}(r_{li} - \bar{r})^2},$$

where r_{li} is the rank of i-th sample in class c_l, $\bar{r}_{l\cdot}$ is the average rank of samples in class c_l, $\bar{r}$ is the average rank, and N_l is the number of instances in class c_l. Under the null hypothesis

H_j asymptotically follows the χ^2 distribution and thus we compute the test's p-value as $P(\chi^2_{k-1} \geq H_j)$. With small N_l, the χ^2 approximation is less accurate and results in reduced test power [Sheskin, 2003]. We adjusted the p-values (obtained with the χ^2 test) for multiple testing by using the false discovery rate procedure [Benjamini and Hochberg, 1995]. Some examples of its use for feature selection are Golugula et al. [2011], Christin et al. [2013].

3.12.2 RF variable importance

Variable importance (VI) is given by the out-of-bag (OOB) accuracy of the trees in the random forest. An OOB sample for a tree t consists of instances which were not in the bootstrap subsample from which t was learned. Let a_t be the percentage of correct classifications in the OOB sample for tree t, and a_{ptj} the percentage of correct classifications after randomly permuting the values of X_j in the OOB sample. Then,

$$VI(X_j) = \frac{1}{T}\sum_{t=1}^{T}(a_{tj} - a_{ptj}),$$

where $a_{tj} = a_{ptj} = 0$ if X_j is not in tree t; otherwise $a_{tj} = a_t$; T, as mentioned in Section 3.11.6, is the number of trees in the ensemble. Alternatively, one can compute per-class VIs by measuring changes in class-specific accuracies.

VI can loosely be interpreted as the feature's effect on accuracy and it provides a ranking of the features (obtained in a multivariate way). Useful features will have positive values whereas useless ones will have VIs around or below zero. A drawback is that the VI ranking tends to favor correlated predictors, especially for low values of m [i.e., the number of features considered at each split; see Strobl et al., 2008]. Because the ranking is stochastic, it is important to use enough trees for it to stabilize.

The above-described VI is less effective in imbalanced settings, as misclassifications due to imbalance can overcome those due to class label permutation. While Janitza et al. [2013] proposed a VI derived from the change in area under the ROC curve [Swets, 1988, Fawcett, 2006], rather than the change in accuracy, so as to balance both types of errors (i.e., false positives and false negatives), it is only implemented for the RF variant based on conditional inference trees [Hothorn et al., 2006].

Finally, given a VI-based ranking, it is not straightforward to determine the cut-point that separates useful features from useless ones. While Breiman [2001] suggests a statistical test for the purpose, it has some undesirable statistical properties (i.e., its power increases with the number of trees and decreases with sample size) and is thus not recommended [Strobl and Zeileis, 2008]. The scaled VI also increases with the number of trees. Alternatives include permutation tests [Wang et al., 2010, Altmann et al., 2010] and methods based on OOB accuracy of nested RF models, corresponding to different cut-points along the ranking [Svetnik et al., 2003, Díaz-Uriarte and De Andres, 2006, Genuer et al., 2010].

Chapter 4

Bayesian network classifiers

Bayesian network classifiers [Bielza and Larrañaga, 2014a, Friedman et al., 1997] are competitive performance classifiers [e.g., Zaidi et al., 2013] with the added benefit of interpretability. Their simplest member, the naive Bayes [Minsky, 1961], is well-known [Hand and Yu, 2001]. More elaborate models exist, taking advantage of the Bayesian network [Pearl, 1988, Koller and Friedman, 2009] formalism for representing complex probability distributions. The tree augmented naive Bayes [Friedman et al., 1997] and the averaged one-dependence estimator [Webb et al., 2005] are among the most prominent.

This chapter provides a brief introduction to Bayesian network classifier topics relevant to this thesis. We refer the reader to Bielza and Larrañaga [2014a] for a survey and to Koller and Friedman [2009] for a detailed treatment of Bayesian networks.

4.1 Bayesian networks

We are interested in modelling a distribution over a set of variables $\{C, \mathbf{X}\}$, with C being a discrete variable representing the class, and $\mathbf{X} = (X_1, \ldots, X_n)$ being n discrete or real-valued predictor variables. We also use X, Y, Z and Q to refer to variables in $\{C, \mathbf{X}\}$. A Bayesian network $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ models a probability distribution $P_{\mathcal{G}}(\mathbf{x}, c)$. $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ is a directed acyclic graph (DAG) with vertices (i.e., nodes) $\mathbf{V}$ corresponding to variables in $\{C, \mathbf{X}\}$, and directed edges (i.e., arcs) $\mathbf{E}_{\mathcal{G}}$ among the vertices. $P_{\mathcal{G}}(\mathbf{x}, c)$ factorizes according to $\mathcal{G}$,

$$P_{\mathcal{G}}(\mathbf{x}, c) = P_{\mathcal{G}}(c \mid \mathbf{pa}_{\mathcal{G}}(c)) \prod_{i=1}^{n} P_{\mathcal{G}}(x_i \mid \mathbf{pa}_{\mathcal{G}}(x_i)),$$

where $\mathbf{pa}_{\mathcal{G}}(x)$ are the values of parents of X in $\mathcal{G}$. $\mathcal{G}$ imposes independence constraints on $P_{\mathcal{G}}(\cdot)$, and they can all be derived from the constraint that each variable is independent of its non-descendents in $\mathcal{G}$ given its parents. $X_1 \perp\!\!\!\perp_{\mathcal{G}} X_2 \mid C$ denotes that X_1 is conditionally independent of X_2 given C in $P_{\mathcal{G}}(\cdot)$.

The parameters $\boldsymbol{\theta}$ specify the local conditional distributions of each variable given its parents' values, with each θ_{ijk} encoding $P(X_i = k \mid \mathbf{Pa}(X_i) = j)$.

4.2 Structure learning

We learn $\mathcal{B}$ from a data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \dots, (\mathbf{x}^N, c^N)\}$ of N observations of $\mathbf{X}$ and C. There are two main approaches to learning the structure $\mathcal{G}$ from $\mathcal{D}$: a) the *constraint-based*, by testing for conditional independence among triplets of variables and b) *search and score*, searching a space of possible structures in order to optimize a network quality score. Under assumptions such as a limited number of parents per variable, the constraint-based approach can produce the correct network in polynomial time [Cheng et al., 2002, Tsamardinos et al., 2003].

Common scores in structure learning are the penalized log-likelihood scores, such as the Akaike information criterion (AIC) [Akaike, 1974] and Bayesian information criterion (BIC) [Schwarz, 1978], and the Bayesian ones such as Bayesian Dirichlet equivalence (BDe) Heckerman et al. [1995]. The log-likelihood based ones measure the model's fitting of the empirical distribution $\widehat{P}(c, \mathbf{x})$ adding a penalty term that is a function of structure complexity. The penalty helps avoid overfitting the training data $\mathcal{D}$. Penalized log-likelihood scores are decomposable with respect to $\mathcal{G}$, allowing for efficient search algorithms.

Finding the optimal structure even with at most two parents per variable is NP-hard [Chickering et al., 2004]. We can, however, find the optimal tree or forest in time quadratic in n with the Chow-Liu [Chow and Liu, 1968] algorithm and a decomposable score. Thus, heuristic local search algorithms are commonly used [see e.g., Koller and Friedman, 2009]. These include greedy hill-climbing and the tabu meta-heuristic [Glover and Laguna, 2013] which allows for score-degrading operators while, for efficiency, avoiding those that undo the effect of recently applied ones.

Two different DAGs can encode an identical joint probability distribution $P(C, \mathbf{X})$. The equivalence relation partitions the set of DAG structures into equivalence classes, and searching in this space, rather than in the space of DAGs, can be more efficient and avoid arbitrary arc choices. The greedy equivalence search (GES) [Chickering, 2002a] and k-greedy equivalence search (KES) [Nielsen et al., 2003] are two algorithms that operate in this space. Both guarantee learning the optimal DAG at the large sample limit if the data is sampled from a Bayesian network.

4.3 Structure learning for classifiers

With limited N and a large n, discriminative scores based on $P(c \mid \mathbf{x})$, such as conditional log-likelihood and classification accuracy, are more suitable to the classification task [Friedman et al., 1997]. These, however, are not decomposable according to $\mathcal{G}$. While one can add a complexity penalty to discriminative scores [e.g., Grossman and Domingos, 2004], they are instead often cross-validated to induce preference towards structures that generalize better, making their computation even more time demanding.

For Bayesian network classifiers, a common [Bielza and Larrañaga, 2014a] structure space is that of augmented naive Bayes [Friedman et al., 1997] models (see Figure 4.1), factorizing

$P(\mathbf{X}, C)$ as

$$P(\mathbf{X}, C) = P(C) \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)), \tag{4.1}$$

with $C \in \mathbf{Pa}(X_i)$ for all X_i and $\mathbf{Pa}(C) = \emptyset$.

Models of different complexity arise by extending or shrinking the parent sets $\mathbf{Pa}(X_i)$, ranging from the NB [Minsky, 1961] with $\mathbf{Pa}(X_i) = \{C\}$ for all X_i, to those with a limited-size $\mathbf{Pa}(X_i)$ [Friedman et al., 1997, Sahami, 1996], to those with unbounded $\mathbf{Pa}(X_i)$ [Pernkopf and O'Leary, 2003]. While the NB can only represent linearly separable classes [Jaeger, 2003], more complex models are more expressive [Varando et al., 2015]. Simpler models, with sparser $\mathbf{Pa}(X_i)$, may perform better with less training data, due to their lower variance, yet worse with more data as the bias due to wrong independence assumptions will tend to dominate the error. In addition, simpler models allow for more efficient learning and inference algorithms.

The algorithms that produce the above structures are generally instances of greedy hill-climbing [Keogh and Pazzani, 2002, Sahami, 1996], with arc inclusion and removal as their search operators. Some add node inclusion or removal [Pazzani, 1996], thus embedding feature selection [Guyon and Elisseeff, 2003] within structure learning. Alternatives include the adaptation [Friedman et al., 1997] of the Chow-Liu [Chow and Liu, 1968] algorithm to find the optimal one-dependence estimator (ODE; see Section 4.6) with respect to decomposable penalized log-likelihood scores in time quadratic in n. Some structures, such as NB or AODE, are fixed and thus require no search.

There has been little work on learning Bayes network classifiers in the space of DAG equivalence classes and the only work we are aware of is that by Acid et al. [2005].

4.4 Parameters learning

Given $\mathcal{G}$, learning $\boldsymbol{\theta}$ in order to best approximate the underlying $P(C, \mathbf{X})$ is straightforward with complete data. For discrete variables X_i and $\mathbf{Pa}(X_i)$, Bayesian estimation can be obtained in closed form by assuming a Dirichlet prior over $\boldsymbol{\theta}$. With all Dirichlet hyper-parameters equal to α,

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + \alpha}{N_{\cdot j \cdot} + |\Omega_{X_i}|\alpha}, \tag{4.2}$$

where N_{ijk} is the number of instances in $\mathcal{D}$ such that $X_i = k$ and $\mathbf{Pa}(X_i) = j$, corresponding to the j-th possible instantiation of $\mathbf{Pa}(x_i)$, $N_{\cdot j \cdot}$ is the number of instances in which $\mathbf{Pa}(x_i) = j$, while $|\Omega_{X_i}|$ is the cardinality of X_i. $\alpha = 0$ in Equation 4.2 yields the maximum likelihood estimate of θ_{ijk}. With incomplete data, the parameters of local distributions are no longer independent and we cannot separately maximize the likelihood for each X_i as in Equation 4.2. Optimizing the likelihood requires a time-consuming algorithm like expectation maximization [Dempster et al., 1977] which does not guarantee convergence to the global optimum.

While the NB can separate any two linearly separable classes given the appropriate $\boldsymbol{\theta}$, learning by approximating $P(C, \mathbf{X})$ cannot render the optimal $\boldsymbol{\theta}$ in some cases [Jaeger, 2003]. Multiple methods [Hall, 2007, Zaidi et al., 2013, 2017] learn a weight $w_i \in [0, 1]$ for each feature and then update $\boldsymbol{\theta}$ as

$$\theta_{ijk}^{weighted} = \frac{(\theta_{ijk})^{w_i}}{\sum_{k=1}^{|\Omega_{X_i}|} (\theta_{ijk})^{w_i}}.$$

A $w_i < 1$ reduces the effect of X_i on the class posterior, with $w_i = 0$ omitting X_i from the model, making weighting more general than feature selection. Ways to compute the weights include maximizing a discriminative score [Zaidi et al., 2013] and computing the usefulness of a feature in a classification tree [Hall, 2007]. Mainly applied to naive Bayes models, a generalization for augmented naive Bayes classifiers has been recently developed [Zaidi et al., 2017].

Another parameter estimation method for the naive Bayes is by means of Bayesian model averaging over the 2^n possible naive Bayes structures with up to n features [Dash and Cooper, 2002]. It is computed in time linear in n and provides the posterior probability of an arc from C to X_i.

4.5 Inference

A Bayesian network allows us to query for the conditional probability $P(\mathbf{Q} \mid \mathbf{O} = \mathbf{o})$ for any $\mathbf{O}$ and $\mathbf{Q}$ in $\{\mathbf{X}, C\}$. In classification, however, we are only interested in the conditional probability $P(c \mid \mathbf{x})$ as our aim is to predict the class c^* for an instance $\mathbf{x}$: $c^* = \arg\max_c P(c \mid \mathbf{x}) = \arg\max_c P(\mathbf{x}, c)$.

Computing $P(c \mid \mathbf{x})$ for a fully observed $\mathbf{x}$ only requires multiplying the corresponding $\boldsymbol{\theta}$ and the time complexity is thus linear in n. With an incomplete $\mathbf{x}$, however, exact inference requires summing over parameters of the local distributions and is NP-hard in the general case [Cooper, 1990], yet can be tractable with limited-complexity structures such as polytrees.

The AODE ensemble computes $P(c \mid \mathbf{x})$ as the average of the $P(c \mid \mathbf{x})$ of the n base models. One method specific to Bayesian network classifiers is the lazy elimination [Zheng and Webb, 2006] heuristic which omits x_i from Equation 4.1 if $P(x_i \mid x_j) = 1$ for some x_j.

4.6 Used structures and learning algorithms

We now list of the augmented naive Bayes structures and learning algorithms used throughout this thesis. The naive Bayes (Figure 4.1a) assumes that the predictors are conditionally independent given the class,

$$p(c|\mathbf{x}) \propto p(c) \prod_{i=1}^{n} p(x_i|c).$$

One way to handle violation of conditional independencies is to omit some of the features [Langley and Sage, 1994]. The forward sequential selection naive Bayes (NB-FSS) algorithm

[Langley and Sage, 1994] learns a *selective naive Bayes* with a greedy forward search guided by classification accuracy. In other words, it starts from a model consisting solely of the class variable and progressively incorporates predictors as long as they do not degrade the score.

As mentioned above, a generalization of the selective naive Bayes is the *weighted naive Bayes*, given by

$$p(c|\mathbf{x}) \propto p(c) \prod_{i=1}^{n} p(x_i|c)^{w_i}, \tag{4.3}$$

with $w_i \in [0, 1]$. Equation 4.3 yields a model equivalent to a naive Bayes with parameters learned with Equation 4.4. The *weighting to alleviate the naive Bayes independence assumption* (WANBIA; Zaidi et al. [2013]) method computes $\mathbf{w}$ by optimizing either the conditional log-likelihood or the mean root squared error of the predictions. The attribute weighted naive Bayes (AWNB; Hall [2007]) sets $w_i = \frac{1}{\frac{1}{T}\sum_{t=1}^{T}\sqrt{d_i^t}}$, where d_i^t is an estimate of X_i's dependence on other features defined as the minimum depth at which X_i is tested in an unpruned classification tree ($\frac{1}{\sqrt{d_i^t}} = 0$ if X_i is not in the tree), with T trees learned from bootstrap samples [Breiman, 1996] of $\mathcal{D}$.

The *model averaged naive Bayes* (MANB) [Dash and Cooper, 2002] parameter estimation method corresponds to exact Bayesian model averaging over the naive Bayes models obtained from all 2^n subsets of the n features, yet it is computed in time linear in n. The estimate for a parameter θ_{ijk}^{MANB} is

$$\theta_{ijk}^{MANB} = \theta_{ijk} P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D}) + \theta_{ik} P(\mathcal{G}_{C \perp\!\!\!\perp X_i}),$$

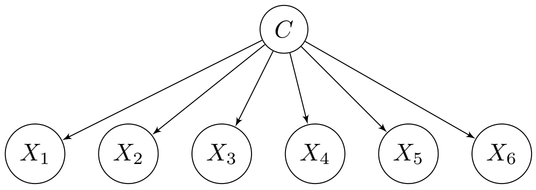
where $P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D})$ is the local posterior probability of an arc from C to X_i, whereas $P(\mathcal{G}_{C \perp\!\!\!\perp X_i}) = 1 - P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D})$ is that of the absence of such an arc (which is equivalent to omitting X_i from the model), while θ_{ijk} and θ_{ik} are the Bayesian parameter estimates obtained with Equation 4.2 given the corresponding structures (i.e., with and without the arc from C to X_i).

A one-dependence estimator (ODE) allows up to one feature parent per predictor. A special case is the *tree augmented naive Bayes* (TAN; [Friedman et al., 1997] Figure 4.1b) with exactly $n - 1$ augmenting arcs, while all other ODEs are *forest-augmented naive Bayes* models (FAN; Figure 4.1c). A special case of the TAN is the super-parent ODE (SPODE) where one feature X_i is the parent of all others (a super-parent), $X_i \in \mathbf{Pa}(X_j)\ \forall j \neq i$ [Keogh and Pazzani, 2002]. With a penalized log-likelihood score the adaptation of the Chow-Liu algorithm due to Friedman et al. [1997] can produce the optimal ODE in time quadratic in n; this ODE is necessarily a TAN if no penalty term is added to the log-likelihood. [Keogh and Pazzani, 2002] learn an ODE with a discriminative score and a greedy hill-climbing search.

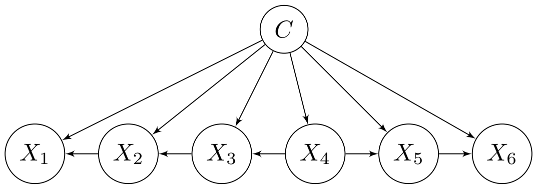
k-dependence Bayesian classifier [Sahami, 1996] allows up to k parent features for each feature X_i. Learning algorithms include those due to Sahami [1996], Pernkopf and Bilmes [2010].

The *semi-naive Bayes* (SNB; Figure 4.1d) [Pazzani, 1996] structure has either a complete or a disconnected subgraph induced by any subset of $\mathbf{X}$. The complexity of its structure is not *a priori* bounded and the learning algorithm may produce a complete DAG in the extreme case. Learning algorithms backward and forward greedy algorithms [Pazzani, 1996].

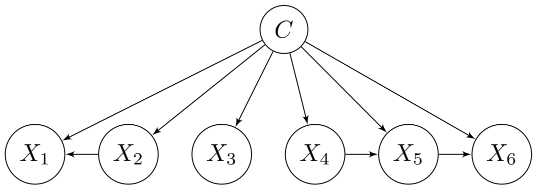
The averaged one-dependence estimator (AODE) [Webb et al., 2005] is an ensemble of n SPODE structures.



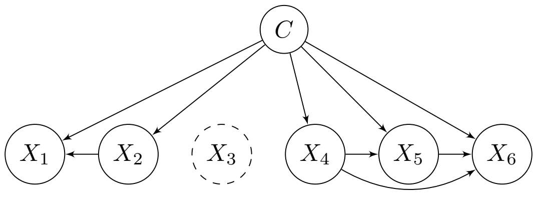
(a) $p(c, \mathbf{x}) = p(c)p(x_1|c)p(x_2|c)p(x_3|c)p(x_4|c)$
$p(x_5|c)p(x_6|c)$



(b) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c, x_3)p(x_3|c, x_4)p(x_4|c)$
$p(x_5|c, x_4)p(x_6|c, x_5)$



(c) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c)p(x_3|c)p(x_4|c)$
$p(x_5|c, x_4)p(x_6|c, x_5)$



(d) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c)p(x_4|c)$
$p(x_5|c, x_4)p(x_6|c, x_4, x_5)$

Figure 4.1: Examples of Bayesian network classifiers structures. (a) NB; (b) TAN (c) FAN (d) SNB. The NB assumes that the features are independent given the class. ODE allows each predictor to depend on at most one other predictor: the TAN is a special case with exactly $n - 1$ augmenting arcs (i.e., inter-feature arcs) while a FAN may have less than $n - 1$. The k-DB allows for up to k parent features per feature X_i, with NB and ODE as its special cases with $k = 0$ and $k = 1$, respectively. The SNB does not restrict the number of parents but requires that connected feature subgraphs be complete (connected, after removing C, subgraphs in (d): $\{X_1, X_2\}$, and $\{X_4, X_5, X_6\}$), also allowing the removal of features (X_3 omitted in (d)).

Chapter 5

GABAergic interneurons classification

5.1 Introduction

The human nervous system is the most complex biological system. In detecting and responding to changes in the environment, it is capable of learning, self-awareness, and gives rise to the intellect. While many fundamental aspects of neuronal structure and function are well understood, many questions remain open. Answering them is becoming more urgent, mainly due to enormous social and economic cost of nervous system disorders. Brain disorders, such as dementia, depression, and addiction, account for 36% of the burden of all disease in high-income countries [Silberberg et al., 2015], with eight millions deaths a year attributable to them [Walker et al., 2015]. The monetary cost of Alzheimer's disease alone in the United States in 2010 was estimated between 157 and 215 billion American dollars [Hurd et al., 2013].

Progressing towards understanding the brain is a monumental endeavor. To this end, ambitious neuroscience projects have been launched globally [Huang and Luo, 2015] over the last decade or so. These include the Human Brain Project [Markram, 2012, Amunts et al., 2016] in the European Union, the Brain Research through Advancing Innovative Neurotechnologies (BRAIN) initiative [Insel et al., 2013] and the Allen Institute for Brain Science in United States of America, and others in Canada, China, Japan, Korea, and Israel [Huang and Luo, 2015, Grillner et al., 2016]. Most of these are extremely large projects, reflecting the complexity of the task. The Human Brain Project, for example, is one of the largest European-funded research projects ever, with the total funding planned to be around one billion euros. It is an interdisciplinary effort, including experts in computer science, physics, and mathematics [Amunts et al., 2016], in addition to those in neuroscience and related life sciences.

The Human Brain Project, the Allen Institute and the BRAIN initiative have neuron type identification among their primary goals [Huang and Luo, 2015, Grillner et al., 2016]. Part of the problem is knowing how to distinguish among types. While each neuron is unique, and broad categories clearly exist, the most useful grouping ought to be somewhere in between [Zeng and Sanes, 2017]. Indeed, high-throughput generation of data, is expected to enable learning a systematic taxonomy within a decade [Zeng and Sanes, 2017], by clustering [Tasic et al., 2016, Cauli et al., 1997] molecular, morphological, and electrophysiological features.

The rest of this chapter introduces neuroscience concepts relevant to our work on supervised and semi-supervised classification of interneurons. Section 5.2 and Section 5.3 provide basic neuroscience concepts. Section 5.4 covers interneuron classification, including the Gardener's and Markram's classification schemes, according to which the neurons used in this thesis have been pre-classified. Section 5.5 treats machine learning classification of interneuron morphologies, which requires computing morphometric variables (Section 5.7) from digital morphology reconstructions (Section 5.6).

5.2 The neuron

The basic structural and functional unit of the nervous system is the nerve cell or neuron. There are around 10^{11} Azevedo et al. [2009] neurons in the human brain, with 10^{15} connections among them [Sporns, 2011]. A neuron's function it to receive and integrate information from sensory receptors or other neurons and transmit it to other neurons or organs. Each neuron has a single cell body, or *soma*, with branching processes, or neurites, called *dendrites* and *axon*, emerging from it. The dendrites receive chemical signals, or *neurotransmitters*, from axons of other neurons and transform them into electrical signals. The soma integrates incoming signals and may send a signal to other neurons, by an electrical potential that travels down the axon and away from the soma. At axon terminals, or *boutons*, this potential triggers the release of a neurotransmitter, into the *synapse*, the region between two adjacent neurons, passing the signal to the post-synaptic neuron. The post-synaptic component is often located on a dendritic *spine*, a protuberance on a dendrite.

Neurons are generally studied in terms of their morphological, molecular and electrophysiological features.

5.3 Cortical neurons

The nervous system is divided into the central and the peripheral nervous systems. The central nervous system contains the spinal cord and the brain, dominated by the cerebral hemispheres [Kandel et al., 2000]. The largest part of the brain hemispheres is a superficial sheet of gray matter called cerebral cortex [Crossman, 2010]. The term gray matter refers to an area densely populated with neuron cell bodies, while the term white matter designates parts of the nervous system mostly populated with axons. The cerebral cortex is responsible for many important functions, such as learning, memory, and goal-directed behaviour [Harris and Mrsic-Flogel, 2013, Li et al., 2015]. In terms of evolutionary development, it can be divided into the older allocortex and the larger and newer neocortex. The cerebral hemispheres are divided into four lobes on the basis of surface topography. Beneath the cortex, the hemispheres contain white matter and a mass of cell bodies called the basal ganglia, while they are covered from above by the pia membrane layer.

Neocortical neurons are arranged in six layers, numbered I to VI, that connect to different cortical and subcortical regions [White, 1989], with layer I nearest to the pia and layer VI closest to subcortical regions. The layers are connected by short-range connections [Schubert et al., 2007], forming a vertical cortical column with a diameter that varies between 200 μm and 600 μm, depending on cortical area and species [Mountcastle, 1997, DeFelipe et al., 2012]. The cortical column is widely considered the elementary cortical unit of operation [DeFelipe et al., 2012, Mountcastle, 1997].

Between 70% and 80% of neocortical neurons are excitatory pyramidal neurons [DeFelipe and Fariñas, 1992, White, 1989, Peters and Jones, 1984]. These cells are relatively uniform in terms of morphological, physiological and molecular properties [DeFelipe and Fariñas, 1992]. The remaining 20–30% neurons are interneurons. They are the main component of inhibitory cortical circuits, which are associated with disorders such as epilepsy [DeFelipe, 1999, Hunt et al., 2013], autism [Rubenstein and Merzenich, 2003], and schizophrenia [Curley and Lewis, 2012, Lewis, 2011, Inan et al., 2013, Joshi et al., 2014]. They are mostly inhibitory, that is, use the gamma-amino butyric acid (GABA) as their neurotransmitter, and have short axons that do not leave the cortex and dendrites with few or no spines.

5.4 Interneuron types

Cortical interneurons are very diverse with regards to morphological, electro-physiological, molecular, and synaptic properties [Fairen et al., 1984, Peters and Jones, 1984, White, 1989, DeFelipe, 1993, Kawaguchi and Kubota, 1997, Markram et al., 2004, Jiang et al., 2015, Tremblay et al., 2016]. Most researchers consider that interneurons can be grouped into types [Ascoli et al., 2008] with much less variability within types than among them. There is, however, no unique catalogue of types [Ascoli et al., 2008, DeFelipe et al., 2013]. High-throughput generation of data is expected to enable learning a systematic taxonomy within a decade [Zeng and Sanes, 2017], by clustering [Tasic et al., 2016, Cauli et al., 1997] molecular, morphological, and electrophysiological features. Currently, however, researchers use [e.g., Markram et al., 2015] and refer to established morphological types such as chandelier, Martinotti, neurogliaform, and basket [Markram et al., 2004, DeFelipe et al., 2013, Feldmeyer et al., 2018, Tremblay et al., 2016]. These types are identified on the basis of the target innervation location —e.g., the peri-somatic area for basket cells— and somatodendritic and axonal morphological features. The latter can be subjective and lead to different classifications: e.g., while Wang et al. [2002] distinguish between large, nest, and small basket cell types, based on features such as axonal arbor density and branch length, DeFelipe et al. [2013] only distinguish between large and common basket types. The different classification schemes [Markram et al., 2004, DeFelipe et al., 2013] only partially overlap. There is, however, consensus on the morphological features of the chandelier, Martinotti, and neurogliaform types

[DeFelipe et al., 2013].

5.4.1 Markram's scheme

Markram et al. [2004] provided a widely cited morphological classification scheme for inhibitory interneurons in layers L2/3 to L6. It specifies nine distinct types (see Figure 5.1) on the basis of axonal and dendritic features, including fine-grained ones such as bouton distribution. This scheme is often refined [e.g., Markram et al., 2015, Jiang et al., 2015] by adding a layer prefix to each type (e.g., layer 2/3 Martinotti, layer 4 Martinotti, etc.) for a total of $4 \times 9 = 36$ types. This scheme includes the following types (see Figure 5.1): bitufted cell; chandelier cell; double bouquet cell; large basket cell; martinotti cell; nest basket cell; small basket cell; neurogliaform cell; and bipolar (BP) cell. We refer to this scheme as the Markram scheme. These types are defined mainly in terms of axonal arbor shape but also of fine-grained properties such as bouton density and distribution (e.g., that of the chandelier cell is characteristic; see Figure 5.1) or axonal thickness, which are rarely captured in morphology reconstructions (see Section 5.6).

5.4.2 Gardener's scheme

DeFelipe et al. [2013] proposed a classification scheme based mainly on patterns of axonal arborization.

The scheme contemplates ten interneuron types (see Figure 5.2): arcade, Cajal-Retzius, chandelier, common basket, common type, horse-tail, large basket, Martinotti, neurogliaform, and other, while DeFelipe et al. [2013] introduced the common type for cells without a strikingly recognizable shape, while other is meant to be chosen when a type missing from the scheme is considered most adequate. The gardener's scheme partially overlaps with the Markram scheme, sharing the neurogliaform, chandelier, and Martinotti types[1].

In addition to interneuron type, the gardener's scheme contemplates to five high-level axonal features, such as whether the axon is restricted or not to the layer of its soma. These features, termed features F1, F2, F3, F4, and F6 (in the scheme, F5 is the interneuron type) have the following categories (see Figure 5.3):

- F1: intralaminar and translaminar
- F2: intracolumnar and transcolumnar
- F3: centered and displaced
- F4: ascending, descending, and both
- F6: characterized and uncharacterized

[1]We used Table 1 in Markram et al. [2015] to map between the two schemes. While the large basket type was also common to the two schemes, Table 1 in Markram et al. [2015] maps it to the common basket type in DeFelipe et al. [2013].

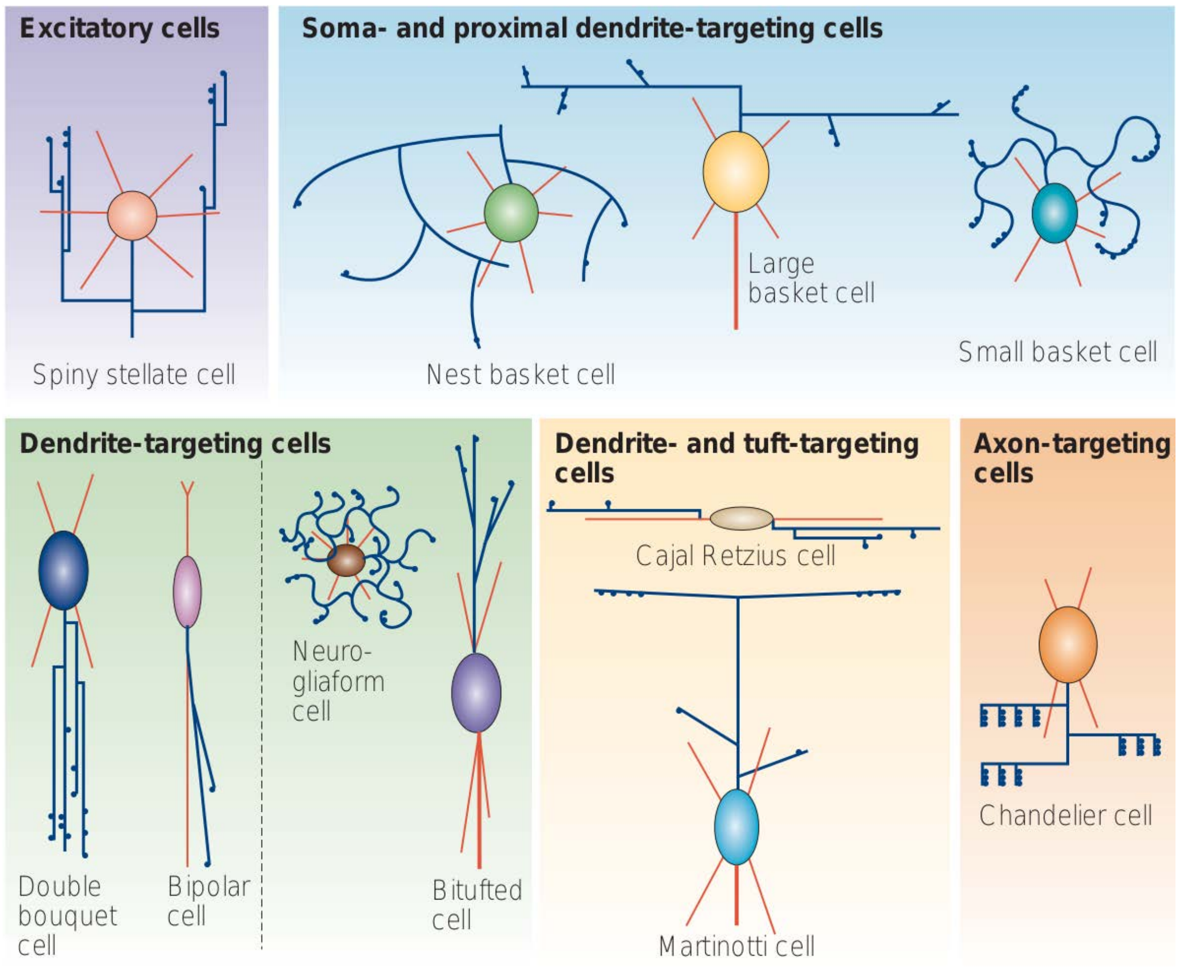


Figure 5.1: The Markram scheme of cortical interneuron types. Axons are shown with blue lines and axonal boutons as blue dots; dendrites with red lines. Figure from [Markram et al., 2004]. Reprinted by permission from Nature Reviews Neuroscience.

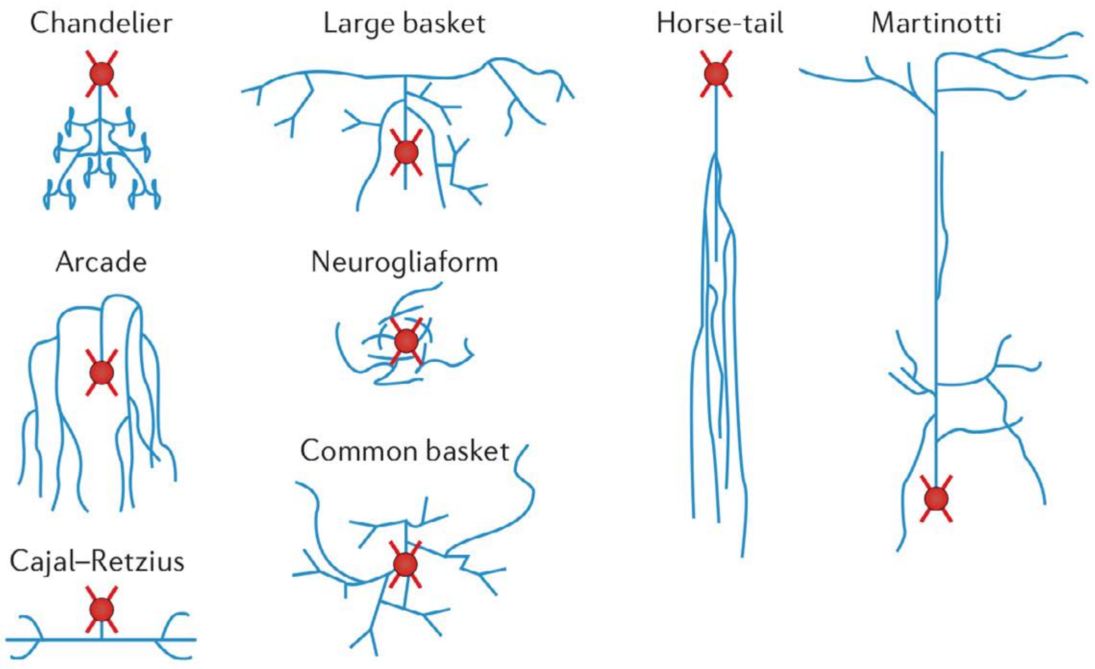


Figure 5.2: Interneuron types in the gardener's scheme. Figure from [DeFelipe et al., 2013]. Reprinted by permission from Nature Reviews Neuroscience.

Features F1 and F2 refer to the distribution of the axonal arbor relative to the cortical layer and column of the soma, respectively. Cells with the axon predominantly in the soma's cortical layer are intralaminar, whereas the rest are translaminar. Likewise, regarding F2, cells with the axon mainly confined to the soma's cortical column are intracolumnar; the rest are transcolumnar. Feature F3 refers to the relative location of axonal and dendritic arbors. Cells with the dendritic arbor mainly located in the center of the axonal arborization are centered whereas the rest are displaced. Feature F4 allows for further distinguishing between translaminar and displaced cells: cells with an axon mainly ascending towards the cortical surface are ascending, cells with an axon mainly descending towards the white matter are descending, whereas the rest are termed both. Regarding F6, a cell is uncharacterized if its reconstruction does not allow for the characterization according to the remaining features, due to, e.g., insufficient axonal reconstruction; otherwise, a cell is characterized.

5.4.3 Consensus among neuroscientists

DeFelipe et al. [2013] asked 42 leading neuroscientists to classify 320 (cat, human, monkey, mouse, rabbit and rat) interneurons according to the gardener's scheme. Neuroscientists categorized interneurons by observing images such as those in Figure 5.4. They were told the neuron's cortical layer and its approximate thickness, cortical area, when these data were available, and species of the animal. DeFelipe et al. [2013] found highest inter-expert agreeement for the chandelier type, followed by the Martinotti, horse tail, and neurogliaform

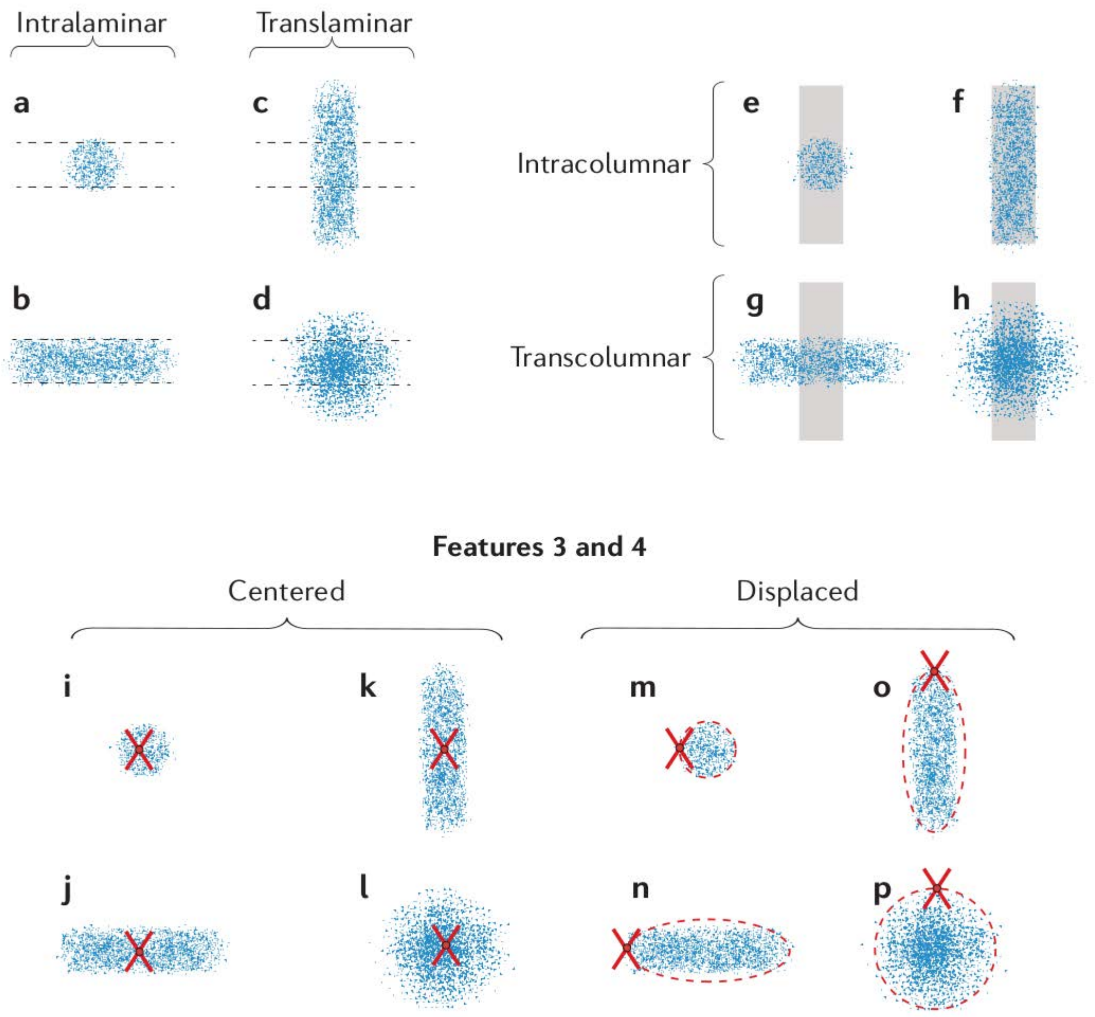


Figure 5.3: Axonal features F1 to F4 in the gardener's scheme. Figure from [DeFelipe et al., 2013]. Reprinted by permission from Nature Reviews Neuroscience.

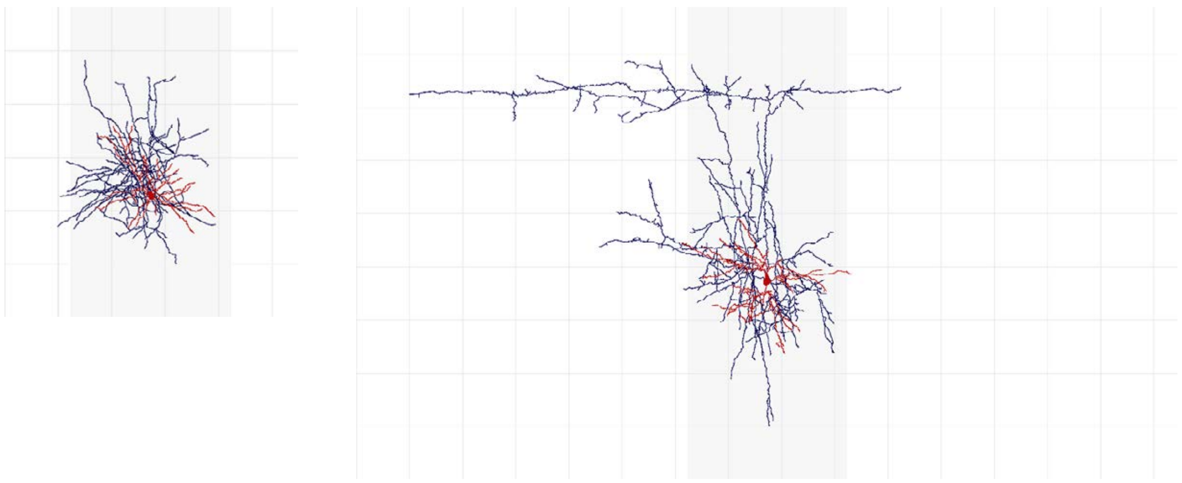


Figure 5.4: Examples of interneurons of different types and with different high-level axonal features. (a) is an `intralaminar`, `intracolumnar`, and `centered` cell, according to 38 (out of 42) experts. Most of its axon (shown in blue) is located within 200 μm horizontally from the soma (shown in red; the grid lines are established every $100 \mu m$), thus appearing to be in the same layer as the soma; it is within the soma's cortical column (the gray vertical shadows depict a 300 μm-wide cortical column); and it seems to be centered around the dendritic arbor (also shown in red). Because this cell is not `translaminar` and `displaced`, but rather `intralaminar` and `centered`, it is not characterizable according to axonal feature F4. According to 24 experts, this is a `common basket` cell. (b) is a `translaminar`, `transcolumnar`, `displaced`, and `ascending` cell according to 38 (out of 42) experts. Unlike (a), this cell's axon reaches over 300 μm horizontally above soma (i.e., it seems to extend to another layer); a large portion of its axon is outside of the soma's cortical column; its dendrites are not in the center of the axonal arborization; and its axon is predominantly above the soma. According to 29 experts, this is a `Martinotti` cell.

types. Inter-expert agreement was, in general, high regarding the high-level axonal features F1-F4.

5.5 Data-driven classification

Both supervised classification and clustering have been used with neurons, usually with either morphological, electrophysiological, or molecular features and rarely with a combination thereof [Armañanzas and Ascoli, 2015]. Authors have used clustering to look for neuron types, using methods such as hierarchical clustering [Cauli et al., 2000, Wang et al., 2002, Tsiola et al., 2003, Benavides-Piccione et al., 2006, Dumitriu et al., 2007, Helmstaedter et al., 2009a,b] k-means (e.g., Karagiannis et al. [2009], affinity propagation [Santana et al., 2013]. Supervised classification [Guerra et al., 2011] has been used to validate proposed groupings [Marin et al., 2002, Druckmann et al., 2013]. Often, the code used classify neurons is not publicly available [Armañanzas and Ascoli, 2015].

DeFelipe et al. [2013] trained supervised classifiers using their 241 digitally reconstructed morphologies, training a model for each of the six axonal features. They computed over 2000 morphometrics and labeled each cell with its majority vote among the neuroscientists. Their models were accurate for axonal features F1, F2, F3, and F6, and only moderately accurate for F4 and F5.

5.6 Morphology reconstructions

A typical neuronal morphology reconstruction [Parekh and Ascoli, 2013] is a sequence of connected cylinders [Cannon et al., 1998], called segments (or compartments), each characterized by six values: the Euclidean coordinates (X, Y and Z) and radius of its terminating point, all given in μm, the identity of its parent segment, and its process type (whether soma, dendrite or axon), with soma's centroid usually at coordinates $(0, 0, 0)$. A branch is the sequence of segments between two bifurcation points (i.e., terminal point of a segment having multiple child segments), while linked branches form an arbor.

The reconstructions are most commonly traced by hand [Parekh and Ascoli, 2013] and there is substantial inter-operator variability [Scorcioni et al., 2004], especially regarding fine-grained properties, such as dendritic and axonal thickness and local branching angles, while bouton locations are seldom included. In addition, histological processing of brain slices makes the tissue shrink, increasing arbor tortuosity (decreasing reach while maintaining total length) [Jaeger, 2010]. Current efforts to improve and standardize automatic reconstruction, such as BigNeuron [Peng et al., 2015] may remove reconstruction-specific differences, increasing the usability of produced morphologies.

Thousands of morphology reconstructions are freely available online in repositories such as Neuromorpho.org [Ascoli et al., 2007, 2017] and the Allen Brain Cell Types Database[2]. NeuroMorpho.Org is a curated repository that gathers reconstructions from laboratories worldwide. It stores over 50,000 reconstructions from 36 distinct species, more than 200 brain regions, and over 300 cell types, contributed by more than 250 laboratories [Ascoli et al., 2017].

5.7 Morphometrics

The Petilla convention [Ascoli et al., 2008] established a set of morphological features that distinguish cortical interneuron types. They include characteristics such as branching angles, axon terminal branch shape (curved / straight), bouton density and clustering patterns, dendritic polarity, whether the axon is ascending or descending, or intra- or trans-laminar, or the presence of distinctive patterns of arborization, such as `bundles of long, vertical branches or tufts' or`dense plexus of highly branched axons'. Many of these correspond to standard neuronal morphometrics (e.g., branching angles) or can be quantified rather directly (e.g., one can compute the tortuosity of terminal branches). Others are either a) often impossible to quantify, as relevant data (e.g., bouton density) may be missing from the digital morphology reconstruction; b) can only be approximated (e.g., translaminar extent) as the data is often incomplete (we often only know the soma's layer, not its position within it); or c) are vaguely defined (e.g., 'dense plexus of highly branched axons').

[2]http://celltypes.brain-map.org/

Standard neuronal morphometrics [Uylings and Van Pelt, 2002] are either metric (e.g., branch length) or topological [partition asymmetry; Van Pelt et al., 1992], and are computed either at the whole arbor(s) level (e.g., height) or for a part of the tree, such as a branch or a bifurcation (e.g., branch length); the latter are then quantified with summarizing statistics across the arbor(s) (e.g., mean and maximal branch length). These morphometrics can be computed with software such as the free, yet closed-source, L-Measure [Scorcioni et al., 2008], the commercial Neurolucida Explorer (MicroBrightField), and open-source alternatives under active development such as NeuroSTR and NeuroM[3]. L-measure provides 42 analyses of morphology, with five summary statistics per analysis; 19 out of the 42 analyses depend on arbor diameter or local bifurcation angles, which often differ across laboratories [Scorcioni et al., 2004, Polavaram et al., 2014], and it seems to assume bifurcating branches, although multi-furcations can occur [Verwer and Van Pelt, 1990].

Researchers have often quantified interneurons with custom-implemented morphometrics, ranging from simple properties such as the mean X coordinate of the axon [e.g., Markram et al., 2015], 2D (X and Y) axonal 'tile surface' and density [Dumitriu et al., 2007], the extent of axonal arborization in L1 [Helmstaedter et al., 2009c], features derived from 2D axonal and dendritic density maps [Jiang et al., 2015], dendritic polarity [Helmstaedter et al., 2009b], estimates of translaminar extent and of the radial (ascending or descending) direction of arborization [Mihaljević et al., 2014], or the position of the convex hull's centroid as a proxy for arbor orientation and extent [Dumitriu et al., 2007, Mihaljević et al., 2014].

[3]The online repository: `https://github.com/BlueBrain/NeuroM`.

Part II

CONTRIBUTIONS TO BAYESIAN NETWORK CLASSIFIERS

Chapter 6

Learning with completed partially directed acyclic graphs

6.1 Introduction

In this chapter, we present a method for learning Bayesian network classifiers in the space of DAG equivalence classes, using completed partially directed acyclic graphs.

Bayesian network classifiers [Bielza and Larrañaga, 2014a, Friedman et al., 1997] are interpretable models that offer competitive predictive performance [e.g., Zaidi et al., 2013]. They include the popular naive Bayes [Minsky, 1961] and its augmented naive Bayes [Friedman et al., 1997] variants with arcs among the features. A common way to learn them [e.g., Keogh and Pazzani, 2002, Pazzani, 1996] is by optimizing a score in the space of directed acyclic graphs (DAGs). This may needlessly prune the search space by arbitrarily directing an arc when its reversal would yield an equivalent model. Two DAGs are equivalent if they impose identical independence constraints on the joint probability distribution $P(C, \mathbf{X})$, with C being the class and $\mathbf{X}$ the predictor variables. The equivalence relation partitions the set of DAGs into equivalence classes, and by searching in this space we only set arc direction when a reversal produces a non-equivalent model. The greedy equivalence search (GES) [Chickering, 2002a] is one algorithm that operates in this space. It represents an equivalence class with a *completed partially* DAG (CPDAG), which has both directed and undirected edges.

Acid et al. [2005] learned Bayesian network classifiers by traversing the space of equivalence classes. They pruned the space of DAGs to be considered by noticing that non-equivalent DAGs could, nonetheless, be *classification equivalent*, that is, encode an identical class-posterior distribution, $P(C \mid \mathbf{x})$, for every instance $\mathbf{x}$. They showed that, with complete training data (i.e., without missing values), a minimal classification-equivalent subgraph of any DAG, which they call a C-DAG, is such that each arc is either directed towards the class variable or a child of the class variable. They traversed the equivalence classes of C-DAGs with a heuristic greedy search and a representation based on their previous work [Acid and de Campos, 2003], rather than the more standard CPDAG representation and its corresponding operators [Chickering, 2002a,b]. Unlike the CPDAG, their representation is not a canonical representative of an equivalence class.

In this chapter, we extend the work by Acid et al. [2005] in a number of ways. First, we show that the search can be reduced, without loss of generality, to a subset of the space of C-DAGs. Namely, we need not consider C-DAGs with parents for the class variable, $\mathbf{Pa}(C)$, because $P(C \mid \mathbf{x})$ is unaffected by marginal independences among $\mathbf{Pa}(C)$ while we can model the full dependencies among them conditional to C with them being children of C. Besides being smaller, we argue that this space of *minimal class-focused* DAGs, or MC-DAGs, is also more adequate for greedy forward learning of dependencies among $\mathbf{Pa}(C)$.

Second, we adapt the GES algorithm to traverse the space of MC-DAGs. We do this by providing GES operator validity conditions, that can be checked efficiently on a CPDAGs, that discard CPDAGs corresponding to equivalence classes that do not have an MC-DAG as a member.

Third, we specify how to, for complete data, efficiently evaluate the discriminative score of an operator locally, time independent of the number of variables, and on a CPDAG. This is based on a technique, described by Keogh and Pazzani [2002], for updating $P(C, \mathbf{x})$, for all $\mathbf{x}$ in our data set, for an arc addition. We adapt the technique for the GES operators over CPDAGs in the MC-DAG space.

We applied our method on thirteen real-world data sets using cross-validated accuracy as the learning score, and compared it to other greedy and non-greedy augmented naive Bayes classifiers. Our method outperformed them on three data sets while it was outperformed on one.

The research covered in this chapter has been submitted in Mihaljević et al. [2018b].

The rest of this chapter is organized as follows. Section 6.2 introduces notation and terminology. Section 6.3 describes the MC-DAG space and discusses its advantages over the C-DAG space. Section 6.4 describes our adaptation of GES for learning MC-DAGs from data. Section 6.5 presents the local updating of $P(C, \mathbf{X})$, used to compute discriminative scores. Section 6.6 shows the evaluation of our algorithm on real-world data sets. We conclude in Section 6.7.

6.2 Preliminaries

The reader can see Chapter 4 for the definition of a Bayesian network and related concepts as well as the basics of learning them from data. Here we recall that we are interested in modeling a distribution over a set of variables $\{C, \mathbf{X}\}$, with C being a discrete variable representing the class, and $\mathbf{X} = (X_1, \ldots, X_n)$ being n discrete or real-valued predictor random variables. We also use X, Y, Z and Q to refer to variables in $\{C, \mathbf{X}\}$. We use a Bayesian network $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ to model a probability distribution $P_{\mathcal{G}}(\mathbf{x}, c)$. We are interested in learning $\mathcal{B}$ from a data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(N)}, c^{(N)})\}$ of N observations of $\mathbf{X}$ and C.

Two DAGs $\mathcal{G}$ and $\mathcal{H}$ are equivalent if the independence constraints that they impose on $P_{\mathcal{G}}(\cdot)$ and $P_{\mathcal{H}}(\cdot)$, respectively, are identical. Searching in this space requires a score-equivalent function and a way to represent an equivalence class of DAGs. All discriminative and many generative scores are score-equivalent, that is, they score all equivalent DAGs equally. A partially DAG (PDAG) $\mathcal{P}$, containing both directed and undirected edges, represents the class of DAGs equivalent to a DAG $\mathcal{G}$ obtained by orienting undirected edges in $\mathcal{P}$. We say that such a $\mathcal{G}$ is a consistent extension of $\mathcal{P}$, $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$, while any DAG $\mathcal{H}$ equivalent to $\mathcal{G}$ is a member of the equivalence class of DAGs corresponding to $\mathcal{P}$, $\mathcal{H} \in \mathcal{E}(\mathcal{P})$. A completed PDAG (CPDAG) $\mathcal{P}$ for en equivalence class $\mathcal{E}(\mathcal{P})$ is the PDAG with an oriented edge for every edge that is identically oriented in every $\mathcal{G} \in \mathcal{E}(\mathcal{P})$, and an undirected edge for all other edges in $\mathcal{E}(\mathcal{P})$. We refer to the directed arcs in a CPDAG as compelled for the equivalence class, and to the undirected ones as reversible. A CPDAG $\mathcal{P}$ is unique for an equivalence class and has every $\mathcal{G} \in \mathcal{E}(\mathcal{P})$ as a consistent extension, $\mathbf{cext}(\mathcal{P}) = \mathcal{E}(\mathcal{P})$.

The GES algorithm starts from a CPDAG $\mathcal{P} = (\mathbf{V}, \mathbf{E}_{\mathcal{P}} = \emptyset)$ and proceeds with the Insert$(X, Y, \mathbf{T})$ operator (to be defined below) considering all arc additions to every DAG in the current equivalence class $\mathcal{E}(\mathcal{P})$. It adds the best among the considered arcs and sets the equivalence class of the obtained DAG as the new state $\mathcal{P}'$, and applies the Insert$(X, Y, \mathbf{T})$ operator to $\mathcal{P}'$. Once it reaches a local optimum, it starts its backward phase, with the Delete$(X, Y, \mathbf{H})$ operator (to be defined below) considering the removal of every arc in every DAG in the current equivalence class. The operators are scored locally on the CPDAG $\mathcal{P}$, without generating the DAGs to which the visited states correspond. A set of conditions verifiable on $\mathcal{P}$ ensure that the operators correspond to valid DAGs in the desired neighbourhood of $\mathcal{P}$.

6.3 Minimal C-DAGs

We begin with the definitions of classification equivalence and class-focused DAGs, or C-DAGs.

Definition 1 (Acid et al. [2005]). Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ and $\mathcal{G}' = (\mathbf{V}, \mathbf{E}_{\mathcal{G}'})$ be two DAGs. Let P be any joint probability distribution on $\mathbf{V}$, and $P_{\mathcal{G}}$ and $P_{\mathcal{G}'}$ be the probability distributions that factorize according to $\mathcal{G}$ and $\mathcal{G}'$, respectively, defined as $P_{\mathcal{G}}(V \mid \mathbf{pa}_{\mathcal{G}}(V)) = P(V \mid \mathbf{pa}_{\mathcal{G}}(V))$ and $P_{\mathcal{G}'}(V \mid \mathbf{pa}_{\mathcal{G}'}(V)) = P(V \mid \mathbf{pa}_{\mathcal{G}'}(V))$, $\forall V \in \mathbf{V}$. If $P_{\mathcal{G}}(C \mid \mathbf{x}) = P_{\mathcal{G}'}(C \mid \mathbf{x})$ $\forall \mathbf{x}$, we say that $\mathcal{G}$ and $\mathcal{G}'$ are classification-equivalent.

That is, $\mathcal{G}$ and $\mathcal{G}'$ are classification-equivalent if their class-posterior distributions are identical for any value of $\mathbf{x}$.

Definition 2 (Acid et al. [2005]). A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ is a class-focused DAG (C-DAG) with respect to the variable C if and only if it satisfies the following condition: $\forall X, Y \in \mathbf{V}$, if $X \to Y \in \mathbf{E}_{\mathcal{G}}$ then either $Y = C$ or $X = C$ or $C \to Y \in \mathbf{E}_{\mathcal{G}}$.

In words, in a C-DAG only C and children of C can have parents (see Figure 6.1). Acid et al. [2005] showed that, for any DAG $\mathcal{H}$, its C-DAG subgraph $\mathcal{H}_C$, induced by including only arcs that match Definition 2, is its minimal classification-equivalent subgraph. Acid et al. [2005] searched the space of C-DAGs to learn Bayesian network classifiers because it covers all possible class-posterior distributions.

We now show that a smaller space is sufficient. Namely, we need not consider parents for the class variable C. That is, for each DAG $\mathcal{H}$ and its C-DAG subgraph $\mathcal{H}_C$, there is a classification-equivalent C-DAG $\mathcal{G}$ with no parents for C. We first formalize such a minimal class-focused DAG (MC-DAG) structure, and then show how to convert a C-DAG into a classification-equivalent MC-DAG.

Definition 3. A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E}_{\mathcal{G}})$ is a minimal class-focused DAG (MC-DAG) with respect to the variable C if and only if it satisfies the following condition: $\forall X, Y \in \mathbf{V}$, if $X \to Y \in \mathbf{E}_{\mathcal{G}}$ then $C \to Y \in \mathbf{E}_{\mathcal{G}}$.

In words, an MC-DAG is a C-DAG that only allows children of C to have parents (see Figure 6.1). Dispensing with the parents of C, $\mathbf{Pa}_{\mathcal{H}}(C)$, while maintaining classification equivalence, is possible by observing that, unlike $P_{\mathcal{H}}(c, \mathbf{x})$, $P_{\mathcal{H}}(c \mid \mathbf{x})$ is unaffected by conditional independence constaints that hold only when C is not observed. This is because we compute $P(c \mid \mathbf{x})$ as $\propto P(c)P(\mathbf{x} \mid c)$, setting each value of C as evidence. Thus, for the C-DAG $\mathcal{H}$, where $X \perp\!\!\!\perp_{\mathcal{H}} Y$ necessarily holds for all $X, Y \in \mathbf{Pa}_{\mathcal{H}}(C)$, there exists a DAG $\mathcal{G}$ such that $X \not\perp\!\!\!\perp_{\mathcal{G}} Y$ for all $X, Y \in \mathbf{Pa}_{\mathcal{H}}(C)$ while nonetheless $P_{\mathcal{H}}(c \mid \mathbf{x}) = P_{\mathcal{G}}(c \mid \mathbf{x})$. To account for $X \not\perp\!\!\!\perp_{\mathcal{H}} Y \mid C$ for all $X, Y \in \mathbf{Pa}_{\mathcal{H}}(C)$ it suffices for $\mathcal{G}$ to have 1) every $X \in \mathbf{Pa}_{\mathcal{H}}(C)$ as a child of C and 2) an arc for every pair $\{X, Y\}$, for all $X, Y \in \mathbf{Pa}_{\mathcal{H}}(C)$. Thus, any C-DAG $\mathcal{H}$ can be represented by an MC-DAG $\mathcal{G}$ with $\mathbf{Pa}_{\mathcal{G}}(C) = \emptyset$. We prove this by showing that a C-DAG $\mathcal{H}$ can be converted into a classification-equivalent C-DAG $\mathcal{H}'$ with one class parent less (Proposition 4). The conversion to an MC-DAG follows by repeating such parent removals until there are no more parents of C (Proposition 5). Figure 6.1 (above) shows a C-DAG $\mathcal{H}$ and its classification-equivalent MC-DAG $\mathcal{G}$.

Proposition 4. For a C-DAG $\mathcal{H}$ with $X \to C$ there is a classification-equivalent C-DAG $\mathcal{H}'$ such that $X \to C$ is reversed to $C \to X$.

Proof. Consider a C-DAG $\mathcal{H}$ with $X \in \mathbf{Pa}_{\mathcal{H}}(C)$. Let $\mathcal{H}' = \mathcal{H}$. By Definition 2, $\mathbf{Pa}_{\mathcal{H}'}(X) = \emptyset$ and X may have children other than C, themselves also, by Definition 2, children of C. Now reverse the arc $X \to C$ in $\mathcal{H}'$ into $C \to X$. $\mathcal{H}'$ is a valid DAG because we did not introduce a cycle: a $Y \in \mathbf{Ch}_{\mathcal{H}'}(X)$ cannot be an ancestor of C since, by Definition 2, $Y \in \mathbf{Ch}_{\mathcal{H}'}(C)$ and a path from Y to C would imply that there was a cycle in $\mathcal{H}'$ before the reversal. Since $\mathbf{Pa}_{\mathcal{H}'}(X) = \emptyset$, the reversal did not introduce a v-structure centered around X. The reversal did drop the v-structures $X \to C \leftarrow Y$ centered around C, $\forall Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$, replacing them with serial connections $Y \to C \to X$. After the reversal, $\mathcal{H}'$ and $\mathcal{H}$ differ, for all $Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$, as follows: 1) $X \perp\!\!\!\perp_{\mathcal{H}} Y$ while $X \not\perp\!\!\!\perp_{\mathcal{H}'} Y$; and 2) $X \not\perp\!\!\!\perp_{\mathcal{H}} Y \mid C$ while $X \perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$. 1) does not affect classification equivalence of $\mathcal{H}$ and $\mathcal{H}'$ since we compute $P(c \mid \mathbf{x})$ by always conditioning on C, via $\propto P(c)P(\mathbf{x} \mid c)$. 2) can be remedied by adding to $\mathcal{H}'$ an arc $Y \to X$ for every $Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$. Now add one such arc $Y \to X$ to $\mathcal{H}'$. This introduced no cycles in $\mathcal{H}'$ because $\mathbf{Pa}_{\mathcal{H}'}(Y) = \emptyset$ and therefore there is no directed path from X to Y in $\mathcal{H}'$. Before adding $Y \to X$, the only parent of X in $\mathcal{H}'$ was C, $\mathbf{Pa}_{\mathcal{H}'}(X) = C$, and thus adding $Y \to X$ only introduced the v-structure $Y \to X \leftarrow C$. Since $Y \in \mathbf{Pa}_{\mathcal{H}'}(C)$, $Y \not\perp\!\!\!\perp_{\mathcal{H}'} C \mid X$ was already true, and the only effect of adding $Y \to X$ was to render

$X \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$, which was our purpose (because $X \not\perp\!\!\!\perp_{\mathcal{H}} Y \mid C$). After the addition, $\mathbf{Pa}_{\mathcal{H}'}(X) = \{Y, C\}$. Now add to $\mathcal{H}'$ the arc $Q \to X$, for $Q \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus \{X, Y\}$. The only independence constraint in $\mathcal{H}'$ conditional to C that this addition modified is that now $Q \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$, because the two introduced v-structures, $Q \to X \leftarrow C$ and $Q \to X \leftarrow Y$, modified no such constraints. The reasoning regarding $Q \to X \to C$ in analogous to that for $Y \to X \to C$. Regarding $Q \to X \leftarrow Y$, $Q \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$ was already true due to the v-structure $Q \to C \leftarrow Y$ in $\mathcal{H}'$. For each next $Z \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus \{X, Y, Q\}$ added to $\mathcal{H}'$ it follows by induction that the only independence constraint conditional to C modified is to render $Z \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$. After adding to $\mathcal{H}'$ arcs $Y \to X$ for every $Y \in \mathbf{Pa}_{\mathcal{H}}(C) \setminus X$, $Y \not\perp\!\!\!\perp_{\mathcal{H}'} X \mid C$ holds and there are no independence constraints conditional to C holding in $\mathcal{H}'$ that do not also hold in $\mathcal{H}$. Thus, $\mathcal{H}'$ is classification-equivalent to $\mathcal{H}$. It is easy to see that it is also a C-DAG. □

Proposition 5. For each C-DAG $\mathcal{H}$ there is a classification-equivalent MC-DAG.

Proof. The proof is constructive. Start with $\mathcal{G} = \mathcal{H}$. At each step, produce a classification-equivalent C-DAG $\mathcal{G}'$ with one less parent of C than $\mathcal{G}$. Set $\mathcal{G} = \mathcal{G}'$. Repeated application will produce a classification-equivalent MC-DAG. □

Any DAG $\mathcal{H}$ and its unique C-DAG $\mathcal{H}_C$ subgraph can be mapped to multiple equivalent MC-DAGs, obtained by choosing the arcs $X \to C$ which to reverse in a different order. An MC-DAG $\mathcal{G}$ is not a subgraph of $\mathcal{H}_C$ (unless $\mathbf{Pa}_{\mathcal{H}_C}(C) = \emptyset$ and $\mathcal{H}_C$ is an MC-DAG itself), as it contains the reversed arcs to $\mathbf{Pa}_{\mathcal{H}_C}(C)$ and arcs among $\mathbf{Pa}_{\mathcal{H}_C}(C)$ (shown in red in Figure 6.1). Note that $P_{\mathcal{H}_C}(c, \mathbf{x}) = P_{\mathcal{G}}(c, \mathbf{x})$ need not hold and thus the generative scores of $\mathcal{H}_C$ and $\mathcal{G}$ might differ.

Since every C-DAG is an MC-DAG, but not vice-versa, the MC-DAG space is smaller. It is nonetheless sufficient, as for each C-DAG there is at least one classification-equivalent MC-DAG. We argue that it is also more suitable for greedy learning algorithms. Consider a forward search starting from an empty graph $\mathcal{H}' = (\mathbf{V} = \{X, Y, Z, C\}, \mathbf{E}_{\mathcal{G}'} = \emptyset)$, using penalized log-likelihood to learn from $\mathcal{D}$, a limited-N sample taken from $\mathcal{B} = (\mathcal{G}', \boldsymbol{\theta})$, with $\mathcal{G}' = (\{X, Y, Z, C\}, \mathbf{E}_{\mathcal{G}'})$ shown in Figure 6.1. After two iterations, the search may have reached the state with $\mathbf{E}_{\mathcal{H}'} = \{X \to C, C \leftarrow Y\}$. Then, its only option to account for $X \not\perp\!\!\!\perp_{\mathcal{G}'} Z \mid C$ is to add Z as a parent of C in $\mathcal{H}'$. This, however, renders $Z \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$, although $Z \perp\!\!\!\perp_{\mathcal{G}'} Y \mid C$, producing a model more complex that $\mathcal{G}'$. Otherwise, the complexity added by having $Z \not\perp\!\!\!\perp_{\mathcal{H}'} Y \mid C$ and might lead the algorithm to halt, adding no arc between C and Y or adding Y as a child of C, either way introducing independence constraint missing from $\mathcal{G}'$. Proceeding in a MC-DAG space, however, the same algorithm could have X and Y as children of C after two steps, and could recover the true structure in the following iterations.

6.4 Adapted GES algorithm for learning MC-DAGs

In this section we describe the adapted GES algorithm which visits only equivalence classes which contain MC-DAGs. We achieve this by means of additional operator validity conditions,

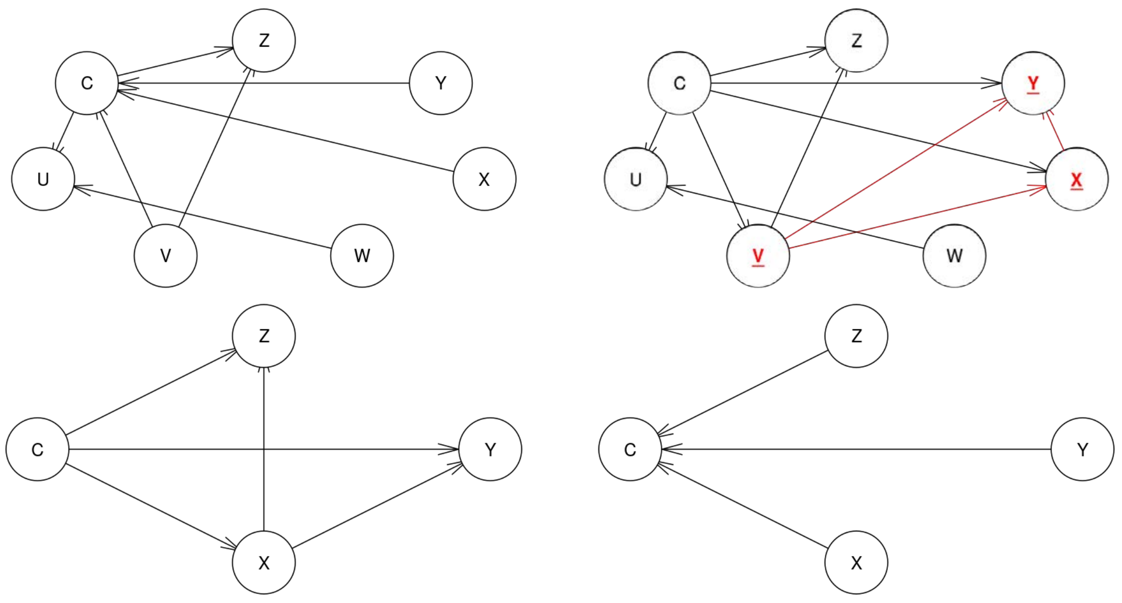


Figure 6.1: Above: A C-DAG $\mathcal{H}$ (left) and a classification-equivalent MC-DAG $\mathcal{G}$ (right). Only C and $\mathbf{Ch}_{\mathcal{H}}(C) = \{U, Z\}$ have parents in $\mathcal{H}$. In $\mathcal{G}$, all arcs incoming to C in $\mathcal{H}$ are reversed and there is an arc between every pair in $\mathbf{Pa}_{\mathcal{H}}(C)$ (shown in red). Only $\mathbf{Ch}_{\mathcal{G}}(C) = \{U, Z, X, Y, Z\}$ have parents in $\mathcal{G}$. Below: An MC-DAG $\mathcal{G}'$ (left) and a classification-equivalent C-DAG $\mathcal{H}'$ (right) which lacks the independence constraint $Y \perp\!\!\!\perp Z \mid C$ present in $\mathcal{G}'$.

discarding operators that produce CPDAGs outside this space. We begin by describing the properties of a CPDAG that has an MC-DAG consistent extension. We then provide the operator validity criteria.

6.4.1 CPDAGs for representing equivalence classes of MC-DAGs

The following conditions are needed for a CPDAG to have an MC-DAG as a consistent extension:

Proposition 6. A CPDAG $\mathcal{P} = (\mathbf{V}, \mathbf{E}_{\mathcal{P}})$ has an MC-DAG as consistent extension if:

1. $\forall X \to Y \in \mathbf{E}_{\mathcal{P}}$: $(Y \in \mathbf{Ch}(C))$ or $(X \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}, Y \in \{\mathbf{Nbr}_{\mathcal{P}}(C)\})$

2. $\forall X - Y \in \mathbf{E}_{\mathcal{P}}$: $X, Y \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$

Proof. Assume that the conditions in Proposition 6 hold for a CPDAG $\mathcal{P}$. We need to find a $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$ such that every arc $X \to Y$ in $\mathcal{G}$ is such that $Y \in \mathbf{Ch}_{\mathcal{G}}(C)$. All arcs in $\mathcal{G}$ are either directed or undirected in $\mathcal{P}$. Consider first an arc $X \to Y$ in $\mathcal{G}$ corresponding to an edge $X - Y$ in $\mathcal{P}$. By condition 2) in Proposition 6, $Y \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$. If $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$, the desired condition already holds in $\mathcal{G}$ for $X \to Y$. Otherwise, the arc $C - Y$ in $\mathcal{P}$ might be directed as $Y \to C$ in $\mathcal{G}$. $|\mathbf{Pa}_{\mathcal{G}}(C)| > 1$ may hold if every pair $Z, Q \in \mathbf{Pa}_{\mathcal{G}}(C)$ is adjacent in $\mathcal{G}$ and thus the $Z \to C \leftarrow Q$ are not v-structures. However, we will only prove the result for the case when $|\mathbf{Pa}_{\mathcal{G}}(C)| = 1$. With $|\mathbf{Pa}_{\mathcal{G}}(C)| = 1$, there is an equivalent DAG $\mathcal{G}'$ with $Y \in \mathbf{Ch}_{\mathcal{G}}(C)$, obtained by reversing $Y \to C$ in $\mathcal{G}$. $\mathcal{G}'$ has no cycles, because $\mathbf{Pa}_{\mathcal{G}}(C) = Y$ and thus there is no other path from Y to C in $\mathcal{G}'$. The reversal does not introduces any new v-structures into $\mathcal{G}'$ and thus does not modify its independence constraints. This is because reverting $C - Y$ as $C \to Y$ only adds v-structures $C \to Y \leftarrow Z$ for $Z \in \mathbf{Pa}_{\mathcal{G}'}(Y) \setminus \{C, \mathbf{Adj}_{\mathcal{G}}(C)\}$. Necessarily, $\mathbf{Pa}_{\mathcal{G}'}(Y) \subseteq \mathbf{Nbr}_{\mathcal{P}}(Y) \cup \mathbf{Pa}_{\mathcal{P}}(Y)$. For all $Z \in \mathbf{Nbr}_{\mathcal{P}}(Y)$ it follows from condition 2) in Proposition 6 that $Z \in \{$ C, $\mathbf{Ch}(C)$, $\mathbf{Nbr}(C)\}$. For all $Z \in \mathbf{Pa}_{\mathcal{P}}(Y)$, it follows from condition 1) that if $Z \notin \mathbf{Adj}_{\mathcal{P}}(C)$ then $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ and $C \to Y$ is already in the desired direction in $\mathcal{P}$. Otherwise, if $Z \in \mathbf{Adj}_{\mathcal{P}}(C)$ no v-structures are introduced by orienting $C - Y$ as $C \to Y$. Thus, for a $\mathcal{P}$ that satisfies conditions 1) and 2), there is a $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$ such that all undirected edges in $\mathcal{P}$ are directed into $\mathbf{Ch}_{\mathcal{G}}(C)$. Consider now the remaining case a directed arc $X \to Y$ in $\mathcal{P}$. By condition 1), if $X \notin \{C, \mathbf{Adj}_{\mathcal{P}}(C)\}$, then $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ and the condition is met. Otherwise, $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$. As dicussed for the case of an undirected arc $X - Y$, there is a $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$ such that $Y - C$ is directed as $C \to Y$. Thus, for a $\mathcal{P}$ complying with the conditions in Proposition 6 there is an MC-DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$. □

6.4.2 Insert operators

For a current state $\mathcal{P}$, the Insert$(X, Y, \mathbf{T})$ operator is:

Definition 7 (Chickering [2002b]). Insert$(X, Y, \mathbf{T})$
For non-adjacent nodes X and Y in $\mathcal{P}$, and for any subset $\mathbf{T}$ of the neighbors of Y that are not adjacent to X, the Insert$(X, Y, \mathbf{T})$ operator modifies $\mathcal{P}$ by (1) inserting the directed edge $X \to Y$, and (2) for each $T \in \mathbf{T}$, directing the previously undirected edge between T and Y as $T \to Y$.

Chickering [2002b] defined validity conditions for Insert$(X, Y, \mathbf{T})$ that can be checked on the CPDAG. The following additional conditions ensure that the operator can produce an MC-CPDAG.

Proposition 8. Let $\mathcal{P}'$ be a PDAG obtained by applying a valid Insert$(X, Y, \mathbf{T})$ to a CPDAG $\mathcal{P}$ that has a consistent extension $\mathcal{G}$ which is an MC-DAG. $\mathcal{P}'$ has a consistent extension $\mathcal{G}'$ which is an MC-DAG if $Y \in \{\mathbf{Ch}_{\mathcal{P}'}(C), \mathbf{Nbr}_{\mathcal{P}'}(C)\}$.

Proof. We need to find a DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$ that satisfies the conditions in Definition 3, that is, with each arc directed towards a child of the class variable. After applying Insert$(X, Y, \mathbf{T})$, $\mathcal{P}'$ differs form $\mathcal{P}$ in that it contains the arc $X \to Y$ and that undirected edges $T - Y$ for $T \in \mathbf{T}$ have been oriented as $T \to Y$. These arcs will be identically oriented in every $\mathcal{G}'$ $\in \mathbf{cext}(\mathcal{P}')$. The remaining arcs in $\mathcal{G}'$ are either oriented as in $\mathcal{G}$, because they are compelled in $\mathcal{P}$, or can be be oriented as in $\mathcal{G}$ because they are reversible in $\mathcal{P}$. Thus it suffices to show that, by proving $Y \in \mathbf{Ch}_{\mathcal{G}'}(C)$, the arcs $X \to Y$ and $T \to Y$, for every $T \in \mathbf{T}$, satisfy Definition 3. If $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$ our condition is already satisfied because $Y \in \mathbf{Ch}_{\mathcal{G}'}(C)$ for any $\mathcal{G}' \in \mathbf{cext}(\mathcal{P})$. If $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$, we need to prove that $C - Y$ can be oriented as $C \to Y$ in $\mathcal{G}'$ without modifying the independence constaints in $\mathcal{G}$. Orienting $C - Y$ as

$C \to Y$ forms v-structures $C \to Y \leftarrow Z$ in $\mathcal{P}'$ for all $Z \in \{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C)$ as well as $C \to Y \leftarrow X$ if $X \notin \mathbf{Adj}_{\mathcal{P}'}(C)$. We first show that $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C) = \emptyset$. Namely, because $Y \in \mathbf{Nbr}_{\mathcal{P}}(C)$ and all $T \in \mathbf{T}$, by Definition 7, have incident undirected arcs, it follows, due to condition 2 in Proposition 6, that $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \subseteq \mathbf{Adj}_{\mathcal{P}'}(C)$ and thus $\{\mathbf{T}, \mathbf{Pa}_{\mathcal{P}}(Y)\} \setminus \mathbf{Adj}_{\mathcal{P}'}(C) = \emptyset$. If $X \notin \mathbf{Adj}_{\mathcal{P}'}(C)$, by condition 1) in Proposition 6, $Y \in \mathbf{Ch}_{\mathcal{P}}(C)$, so $C \to Y$ is already properly oriented in $\mathcal{P}$. Therefore, enforcing in $\mathcal{G}$ the direction $C \to Y$ for $C - Y$ in $\mathcal{P}$ does not modify the independence constraints in $\mathcal{G}'$. Thus, for a $\mathcal{P}$ that matches the condition, the Insert$(X, Y, \mathbf{T})$ operator produces a CPDAG such that MC-DAG $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$. □

6.4.3 Delete operators

For a current state $\mathcal{P}$, we apply the Delete$(X, Y, \mathbf{H})$ operator, followed by the Post-Delete(X, Y) if $X = C$. The operators are:

Definition 9 (Chickering [2002b]). Delete$(X, Y, \mathbf{H})$
For adjacent nodes X and Y in $\mathcal{P}$ connected either as $X - Y$ or $X \to Y$, and for any subset $\mathbf{H}$ of the neighbors of Y that are adjacent to X, the Delete$(X, Y, \mathbf{H})$ operator modifies $\mathcal{P}$ by deleting the edge between X and Y, and for each $H \in \mathbf{H}$, (1) directing the previously undirected edge between Y and H as $Y \to H$ and (2) directing any previously undirected edge between X and H as $X \to H$.

Definition 10. Post-Delete(X, Y)
Let $\mathcal{P}'$ be a PDAG obtained after applying a valid Delete$(X, Y, \mathbf{H})$ to $\mathcal{P}$. Then, if $X = C$, the Post-Delete(X, Y) operator deletes in $\mathcal{P}'$ all directed arcs $Z \to Y$, and orients all undirected edges $Z - Y$ as $Y \to Z$, for any $Z \in \mathbf{V}$.

The conditions given in Chickering [2002b] specify whether the Delete$(X, Y, \mathbf{H})$ operator is valid for a given X, Y, and $\mathbf{H}$. If $X = C$, the delete renders $Y \notin \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C)\}$. Thus, if it contains incoming or undirected edges into Y, $\mathcal{P}'$ does not satisfy the conditions in Proposition 6 that ensure there is an MC-DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$. Post-Delete$(X, Y)$ removes the violating arcs ensuring that $\mathcal{P}$ satisfies the conditions in Proposition 6.

Proposition 11. Let PDAG $\mathcal{P}'$ be the result of applying a valid Delete$(X, Y, \mathbf{H})$, for $X = C$, followed by a Post-Delete(X, Y), to a PDAG $\mathcal{P}$. There exists a DAG $\mathcal{G}'$ that is a consistent extension of $\mathcal{P}'$ and is classification-equivalent to any MC-DAG $\mathcal{G}$ that is a consistent extension of $\mathcal{P}$.

We only sketch the proof for brevity. Because Post-Delete(X, Y) ensured that $\mathcal{P}'$ matches the conditions in Proposition 6, it follows that is there is a DAG $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$. Once Delete$(X, Y, \mathbf{H})$ rendered Y not adjacent to C, none of its incoming arcs affect classification-equivalence because they are not part of the minimal C-DAG subgraph of $\mathcal{P}$ (see Definition 2). Thus, we can delete them to obtain a PDAG with no parents for nodes that are no children of C. The undirected edges into Y would necesarily be oriented away from Y in a valid MC-DAG.

Proposition 12. Let PDAG $\mathcal{P}'$ be the result of applying a valid Delete$(X, Y, \mathbf{H})$, followed by a valid Post-Delete(X, Y), to a CPDAG $\mathcal{P}$ that has a consistent extension $\mathcal{G}$ which is an MC-DAG. $\mathcal{P}'$ has a consistent extension $\mathcal{G}$ which is an MC-DAG if $C \notin \mathbf{H}$.

Proof. $\mathcal{G}'$ is an MC-DAG if all its arcs are directed towards a child of C. We need to ensure that: 1) If $Y \notin \{\mathbf{Ch}_{\mathcal{G}'}(C), \mathbf{Nbr}_{\mathcal{G}'}(C)\}$, then $\mathbf{Pa}_{\mathcal{G}'}(Y) = \emptyset$; 2) $H \in \mathbf{Ch}_{\mathcal{G}'}(C)$, for all $H \in \mathbf{H}$. $Y \notin \{\mathbf{Ch}_{\mathcal{G}'}(C), \mathbf{Nbr}_{\mathcal{G}'}(C)\}$ only happens if $X = C$. In that case, Post-Delete(X, Y) ensures that $\mathbf{Pa}_{\mathcal{G}'}(Y) = \emptyset$, satisfying the first condition. Regarding the second condition, $H \in \{\mathbf{Ch}_{\mathcal{P}}(C), \mathbf{Nbr}_{\mathcal{P}}(C), C\}$, for all $H \in \mathbf{H}$, by condition 2) in Proposition 6. If $C \neq H$, the second condition can be satisfied. Otherwise, $\mathcal{P}'$ will contain a v-structure $X \rightarrow C \leftarrow Y$. Thus for every $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$, $\mathbf{Pa}_{\mathcal{G}'}(C) \neq \emptyset$ and therefore such $\mathcal{G}'$ is not an MC-DAG. Thus, for $C \notin \mathbf{H}$, Delete$(X, Y, \mathbf{H})$ followed by Post-Delete(X, Y) produce a CPDAG that has an MC-DAG as a consistent extension. □

6.5 Local discriminative scoring for CPDAGs

Chickering [2002b] specified how to efficiently score Insert$(X, Y, \mathbf{T})$ and Delete$(X, Y, \mathbf{H})$ operators for decomposable scores, without converting the current state's CPDAG into DAGs. We cannot do this for discriminative scores such as conditional log-likelihood, as 1) they do not decompose over the network; and 2) we need a fully parameterized DAG to compute the underlying class-conditional probability $P(c \mid \mathbf{x})$.

With complete data we can, however, efficiently update the $P(C, \mathbf{x})$ of the current state $\mathcal{P}$, without recomputing it from scratch and then recompute the $P(c \mid \mathbf{x})$ and the discriminative score from the updated $P(C, \mathbf{x})$. This technique was described by Keogh and Pazzani [2002] for a single arc insertion into a one-dependence Bayesian classifier. We adapt it for CPDAG Insert$(X, Y, \mathbf{T})$ and Delete$(X, Y, \mathbf{H})$ operators for MC-DAGs.

Let A be the $N \times r_c$ matrix holding the $P_{\mathcal{G}}(c, \mathbf{x})$ for our current CPDAG $\mathcal{P}$, where N is the number of training instances in $\mathcal{D}$ and r_c the number of distinct class values, and $\mathcal{G} \in \mathbf{cext}(\mathcal{P})$. Then, for a valid Insert$(X, Y, \mathbf{T})$ that yields a CPDAG $\mathcal{P}'$,

$$a'_{ij} = \begin{cases} a_{ij} \times \frac{P_{\mathcal{G}}(y^{(i)} | x^{(i)}, \mathbf{t}^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)}{P_{\mathcal{G}'}(y^{(i)} |_{\mathbf{pa}^*_{\mathcal{G}'}}(y)^{(i)}, c_j)} & \text{if } C \in \mathbf{Pa}(X) \\ a_{ij} & \text{otherwise} \end{cases}$$

where the i superscript denotes the values in the i-th instance of $\mathcal{D}$, $\mathcal{G}' \in \mathbf{cext}(\mathcal{P}')$, and $\mathbf{pa}^*_{\mathcal{G}}(X)$ denotes denotes $\mathbf{Pa}_{\mathcal{G}}(C) \setminus C$. After the update, a_{ij} is proportional to $P(C, \mathbf{x})$ for the new state $\mathcal{P}'$. If $C \notin \mathbf{Pa}(X)$, the factor for the observed variable X may be omitted because its value is equal for every c. If $C \in \mathbf{Pa}(X)$, the result assumes that there exist such DAGs $\mathcal{G}$ and $\mathcal{G}'$, such that $\mathbf{Pa}_{\mathcal{G}'}(Y) = \{\mathbf{Pa}_{\mathcal{G}}(Y), X, \mathbf{T}\}$ also holds. They exist if any undirected edges to Y in $\mathcal{P}$, other than those to $T \in \mathbf{T}, C$, are outgoing from Y in $\mathcal{G}$ and $\mathcal{G}'$. Since these edges are not compelled in $\mathcal{P}$, there is a $\mathcal{G}$ with such an orientation. Such edges in $\mathcal{P}'$ can only be comelled as outgoing from Y, since directing them towards Y could introduce v-structures with the newly oriented edges, and thus a $\mathcal{G}'$ also exists.

To compute the update, we do not need $\mathcal{G}$ and $\mathcal{G}'$, but only the probabilities $P_{\mathcal{G}}(y^{(i)} \mid x^{(i)}, \mathbf{t}^{(i)}, \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)$ and $P_{\mathcal{G}'}(y^{(i)} \mid \mathbf{pa}^*_{\mathcal{G}'}(y)^{(i)}, c_j)$ which we can re-estimate from $\mathcal{D}$ each time.

For a Delete$(X, Y, \mathbf{H})$, followed by a Post-Delete(X, Y),

$$a'_{ij} = \begin{cases} a_{ij} \times \frac{P'_{\mathcal{G}}(y^{(i)} \mid x^{(i)}, \mathbf{pa}^*_{\mathcal{G}'}(y)^{(i)}, c_j)}{P_{\mathcal{G}}(y^{(i)} \mid \mathbf{pa}^*_{\mathcal{G}}(y)^{(i)}, c_j)} & \text{if } C \in \mathbf{Pa}(X) \\ a_{ij} & \text{otherwise} \end{cases}$$

The argument for the existence of adequate $\mathcal{G}$ and $\mathcal{G}'$ is analogous to the one above. Thus, for each operator, we update $P(C, \mathbf{x})$ in $O(Nr_c)$ time, independently of the number of features n.

6.6 Experimental evaluation

We compared our method (MG, short for MC-DAG GES) to a number of *k-dependence Bayesian classifiers* [k-DB; Sahami, 1996], that is, augmented naive Bayes models with up to k predictor parents per predictor. Namely, we used the naive Bayes (DB^0, the 0 standing for 0-dependence), a 1-dependence classifier learned with a hill-climbing search and a discriminative score [DB$^1_{HC}$; Keogh and Pazzani, 2002] and its 2-dependence generalization (DB$^2_{HC}$) implemented in the bnclassify R package [Mihaljević et al., 2018], and the optimal log-likelihood 1-dependence classifier [DB$^1_{CL}$; Friedman et al., 1997]. For comparison with the corresponding k-DB, we also limited the number of parents per predictor for MG to 1 (MG^1), 2 (MG^2), or left it unbounded (MG). We considered both an empty graph and a naive Bayes as the initial state of the search (the latter indicated with the NB subscript, e.g., MG$_{NB}$).

We used data sets from the UCI repository [Dheeru and Karra Taniskidou, 2017], together with the `mofn-3-7-10` dataset by Kohavi and John [1997] (see Table 6.1). We discretized numeric variables with the Fayyad and Irani [1993] method and removed instances with missing values, except in `voting`, where we treated them as a separate value. For all MG and DB$_{HC}$ variants we used 10-fold stratified cross-validation accuracy estimate as the scoring function, with the greedy search proceeding until the score was not degraded. For the DB$^1_{CL}$, we used log-likelihood as the score.

MG$_{NB}$ outperformed all methods on `car` and `mofn-3-7-10` (see Table 6.1). Both unbounded variants of MG (i.e., MG$_{NB}$ and MG) were outperformed only on the mushroom data set, and by DB$^1_{CL}$, the only non-greedy algorithm. Differences between the best unbounded MG and the remaining methods were not significant on the remaining data sets. The difference on mushroom is probably due to the greedy nature of MG: DB$^1_{CL}$ is able to find the optimal 1-DB with respect to log-likelihood, with 21 augmenting arcs, while MG$_{NB}$ and MG added one and zero arcs, respectively, to their initial structures.

Traversing the space of equivalence classes, rather than that of DAGs, did not provide an advantage when bounding the number of predictor parents. Namely, MG$^1_{NB}$ only outperformed DB^1_{HC} on `tictac`, with no additional significant differences between them, nor between MG^2_{NB} and DB^2_{HC}. MG_{NB} only visited slightly more distinct structures than DB_{HC}. For example, on a single run on `voting` MG^1_{NB} visited 968 states after ten iterations, versus 961 states by DB^1_{HC}. While the difference grows with the number of augmenting arcs included, it would still be negligible on `voting` with the maximal 16 of iterations.

MG_{NB} strongly outperformed MG on `car`, `tictac` and `mofn-3-7-10`, while MG was better on `voting`, and the differences were insignificant on other data sets. The poor performance of MG was due to it returning extremely sparse graphs: an empty graph for `car` and `mofn-3-7-10`, and one with three arcs on `car`. On the other hand, MG was accurate on `voting` by using only four predictor variables, while $\mathrm{MB\_}NB$ added 15 augmenting arcs and used all predictors.

6.7 Conclusion

We have specified the smallest DAG subspace that covers all possible class-conditional distributions. We presented an algorithm to traverse the equivalence classes in this space by adapting the greedy equivalence search algorithm. Finally, we specified how to compute the discriminative score of a CPDAG search operator in a time that is independent of the number of variables and that does not require converting the CPDAG into a DAG.

Future work includes evaluating our algorithm on additional synthetic and real-world data sets, to better assess its merits. It is possible that a different search algorithm could take better advantage of equivalence classes when the nodes' in-degree is bounded, as for the k-dependence models. Adapting our algorithm to traverse the space of augmented naive Bayes models, rather than that of MC-DAGs, would amount to simple additional restrictions to operator validity conditions.

Table 6.1: Predictive accuracy estimated with 10-fold stratified cross-validation. N denotes the number of instances, n the number of predictor variables, and r_c the number of distinct classes.

	MG^1_{NB}	MG^2_{NB}	MG_{NB}	MG^1	MG^2	MG	DB^0	DB^1_{CL}	DB^1_{HC}	DB^2_{HC}	N	n	r_c
car	0.95	0.94	**0.98**	0.86	0.86	0.86	0.86	0.94	0.94	0.95	1728	6	4
tictac	0.74	0.89	**0.90**	0.71	0.78	0.75	0.69	0.76	0.71	**0.90**	958	9	2
breast	**0.98**	**0.98**	0.97	**0.98**	**0.98**	**0.98**	**0.98**	0.96	**0.98**	**0.98**	683	9	2
mofn-3-7-10	0.94	0.94	**1.00**	0.86	0.86	0.86	0.86	0.93	0.93	0.93	1324	10	2
wine	**0.99**	**0.99**	0.98	**0.99**	**0.99**	**0.99**	**0.99**	0.98	0.98	**0.99**	178	13	3
voting	0.91	0.9	0.91	**0.95**	0.94	**0.95**	0.9	0.94	0.91	0.9	435	16	2
tumor	**0.48**	0.46	0.47	**0.48**	**0.48**	**0.48**	**0.48**	0.38	0.47	0.45	132	17	18
mushroom	0.97	0.96	0.97	0.95	0.95	0.95	0.95	**1.00**	0.97	0.96	8123	22	2
iono	0.92	0.92	0.91	0.92	0.92	0.92	0.92	**0.93**	0.92	0.92	351	34	2
dermatology	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	**0.99**	0.98	358	34	6
soybean	0.91	0.92	0.92	0.91	0.91	0.91	0.91	**0.93**	0.91	0.92	562	35	19
molecular	0.88	0.9	**0.92**	0.91	0.91	0.91	0.91	0.81	0.9	0.9	106	57	2
kr-vs-kp	0.96	0.97	**0.98**	0.88	0.88	0.88	0.88	0.92	0.96	0.97	3196	36	2

Chapter 7

bnclassify: an R package for learning Bayesian network classifiers

7.1 Introduction

In this chapter, we present our implementation of Bayesian network classifiers in the `bnclassify` package for the R environment for statistical computing [R Core Team, 2015]. A Bayesian network classifier is simply a Bayesian network applied to classification, that is, the prediction of the probability $P(c \mid \mathbf{x})$ of some discrete (class) variable C given some features $\mathbf{X}$. The `bnlearn` [Scutari and Ness, 2018, Scutari, 2010] R package already provides state-of-the art algorithms for learning Bayesian networks from data. Yet, learning classifiers is specific, as the implicit goal is to estimate $P(c \mid \mathbf{x})$ rather than the joint probability $P(\mathbf{x}, c)$. Thus, specific search algorithms, network scores, parameter estimation and inference methods have been devised for this setting. In particular, many search algorithms consider a restricted space of structures such as that of augmented naive Bayes [Friedman et al., 1997] models. Unlike with general Bayesian networks, it makes sense to omit a feature X_i from the model as long as the estimate of $P(c \mid \mathbf{x})$ is no better than that of $P(c \mid \mathbf{x} \setminus x_i)$. Discriminative scores, related to the estimation of $P(c \mid \mathbf{x})$ rather than $P(c, \mathbf{x})$, are used to learn both structure [Keogh and Pazzani, 2002, Grossman and Domingos, 2004, Pernkopf and Bilmes, 2010, Carvalho et al., 2011] and parameters [Zaidi et al., 2013, 2017]. Some of the prominent classifiers [Webb et al., 2005] are ensembles of networks, and there are even heuristics applied at inference time, such as the lazy elimination technique [Zheng and Webb, 2006]. Many of these methods [e.g., Dash and Cooper, 2002, Zaidi et al., 2013, Keogh and Pazzani, 2002, Pazzani, 1996] are at best available in standalone implementations published alongside the original papers.

The `bnclassify` package that we have implemented provides state-of-the-art algorithms for learning structure and parameters. The implementation is efficient enough to allow for time-consuming discriminative scores on relatively medium-sized data sets. It provides utility functions for prediction and inference, model evaluation with network scores and cross-validated estimation of predictive performance, and model analysis, such as querying structure type or graph plotting via the `Rgraphviz` package [Hansen et al., 2017]. It integrates with the `caret` [Kuhn et al., 2017, Kuhn, 2008] and `mlr` [Bischl et al., 2017, 2015] packages for straightforward use in machine learning pipelines. Currently it supports only discrete variables. The functionalities are illustrated in an introductory vignette, while an additional vignette provides details on the implemented methods.[1] It includes over 200 unit and integration tests that give a code coverage of 94 percent[2].

The research covered in this chapter has been submitted in Mihaljević et al. [2018a].

The rest of this chapter is structured as follows. Section 7.2 describes the implemented functionalities. Section 7.3 illustrates usage with a synthetic data set while Section 7.4 compares the methods' runtime, predictive performance and complexity over multiple data sets. Section 7.5 discusses implementation while Section 7.6 briefly surveys related software. Finally, Section 7.7 concludes and outlines future work.

7.2 Functionalities

The package has four groups of functionalities:

1. Learning network structure and parameters
2. Analyzing the model
3. Evaluating the model
4. Predicting with the model

Learning is split into two separate steps, the first being structure learning and the second, optional, parameter learning. The obtained models can be evaluated, used for prediction or analyzed. The following provides a brief overview. For details on some of the underlying methods please see Chapter 4.

7.2.1 Structures

The learning algorithms produce the following network structures:

- Naive Bayes (NB) (Figure 4.1a) [Minsky, 1961]
- k-dependence Bayesian classifier (k-DB) [Sahami, 1996, Pernkopf and Bilmes, 2010]
    - One-dependence estimators (ODE)
        - Tree-augmented naive Bayes (TAN) (Figure 4.1b) [Friedman et al., 1997]
        - Forest-augmented naive Bayes (FAN) (Figure 4.1c)
- Semi-naive Bayes (SNB)(Figure 4.1d) [Pazzani, 1996]

[1]The vignettes are available within the R environment when `bnclassify` is installed.
[2]See https://codecov.io/github/bmihaljevic/bnclassify?branch=master

- Averaged one-dependence estimators (AODE) [Webb et al., 2005]

Figure 4.1 shows some of these structures and their factorizations of $P(c, \mathbf{x})$. We use k-DB in the sense meant by Pernkopf and Bilmes [2010] rather than that by Sahami [1996], as we impose no minimum on the number of augmenting arcs.

7.2.2 Algorithms

Each structure learning algorithm is implemented by a single R function. Table 7.1 lists these algorithms along with the corresponding structures that they produce, the scores they can be combined with, and their R functions. Below we provide their abbreviations, references, brief comments and illustrate function calls.

7.2.2.1 Fixed structure

We implement two algorithms:

- NB
- AODE

The NB and AODE structures are fixed given the number of variables, and thus no search is required to estimate them from data. For example, we can get a NB structure with

```
n <- nb('class', dataset = car)
```

where `class` is the name of the class variable C and `car` the dataset containing observations of C and $\mathbf{X}$.

7.2.2.2 Optimal ODEs with decomposable scores

We implement one algorithm:

- Chow-Liu for ODEs (CL-ODE; Friedman et al. [1997])

Maximizing log-likelihood will always render a TAN while maximizing penalized log-likelihood may render a FAN since including some arcs can degrade such a score. With incomplete data our implementation does not guarantee the optimal ODE as that would require computing maximum likelihood parameters. The parameters of the `tan_cl` function are the network score to use and, optionally, the root for features' subgraph:

```
n <- tan_cl('class', car, score = 'AIC', root = 'buying')
```

7.2.2.3 Greedy hill-climbing with global scores

`bnclassify` implements five algorithms:

- Hill-climbing tree augmented naive Bayes (HC-TAN) [Keogh and Pazzani, 2002]
- Hill-climbing super-parent tree augmented naive Bayes (HC-SP-TAN) [Keogh and Pazzani, 2002]
- Backward sequential elimination and joining (BSEJ) [Pazzani, 1996]
- Forward sequential selection and joining (FSSJ) [Pazzani, 1996]
- Hill-climbing k-dependence Bayesian classifier (k-DB)

These algorithms use the cross-validated estimate of predictive accuracy as score. Only the FSSJ and BSEJ provide feature selection. The parameters of the corresponding functions include the number of cross-validation folds `k` and the minimal absolute score improvement `epsilon` required for continuing the search:

```
fssj <- fssj('class', car, k = 5, epsilon = 0)
```

Table 7.1: Implemented structure learning algorithms.

Structure	Search algorithm	Score	Feature selection	Function
NB	-	-	-	`nb`
TAN/FAN	CL-ODE	log-lik, AIC, BIC	-	`tan_cl`
TAN	TAN-HC	accuracy	-	`tan_hc`
TAN	TAN-HCSP	accuracy	-	`tan_hcsp`
SNB	FSSJ	accuracy	forward	`fssj`
SNB	BSEJ	accuracy	backward	`bsej`
AODE	-	-	-	`aode`
kDB	kDB	accuracy	-	`kdb`

7.2.3 Parameters

`bnclassify` only handles discrete features. With fully observed data, it estimates the parameters with maximum likelihood or Bayesian estimation, according to Equation 4.2, with a single α for all local distributions. With incomplete data it uses *available case analysis* and substitutes $N_{\cdot j \cdot}$ in Equation 4.2 with $N_{ij\cdot} = \sum_{k=1}^{|\Omega_{X_i}|} N_{ijk}$, i.e., the count of instances in which $\mathbf{Pa}(X_i) = j$ and X_i is observed.

We implement two methods for weighted naive Bayes parameter estimation:

- Weighting attributes to alleviate naive Bayes' independence assumption (WANBIA) [Zaidi et al., 2013]
- Attribute-weighted naive Bayes (AWNB) [Hall, 2007]

And one method for estimation by means of Bayesian model averaging over all NB structures with up to n features:

- Model averaged naive Bayes (MANB) [Dash and Cooper, 2002]

It makes little sense to apply WANBIA, MANB and AWNB to structures other than NB. WANBIA, for example, learns the weights by optimizing the conditional log-likelihood of the NB. Parameter learning is done with the `lp` function. For example,

```
a <- lp(n, smooth = 1, manb_prior = 0.5)
```

computes Bayesian parameter estimates with $\alpha = 1$ (the `smooth` argument) for all local distributions, and updates them with the MANB estimation obtained with a 0.5 prior probability for each class-to-feature arc.

7.2.4 Utilities

Single-structure-learning functions, as opposed to those that learn an ensemble of structures, return an S3 object of class `bnc_dag`. The following functions can be invoked on such objects:

- Plot the network: `plot`
- Query model type: `is_tan`, `is_ode`, `is_nb`, `is_aode`, ...
- Query model properties: `narcs`, `families`, `features`, ...
- Convert to `gRain` and `bnlearn` objects: `as_grain`

Ensembles are of type `bnc_aode` and only `print` and model type queries can be applied on such objects. Fitting the parameters (by calling `lp`) of a `bnc_dag` produces a `bnc_bn` object. In addition to all `bnc_dag` functions, the following are meaningful:

- Predict class labels and class posterior probability: `predict`
- Predict joint distribution: `compute_joint`
- Network scores: `AIC,BIC,logLik,clogLik`
- Cross-validated accuracy: `cv`
- Query model properties: `nparams`
- Parameter weights: `manb_arc_posterior`, `weights`

The above functions for `bnc_bn` can also be applied to an ensemble with fitted parameters.

7.2.5 Documentation

This vignette provides an overview of the package and background on the implemented methods. Calling `?bnclassify` gives a more concise overview of the functionalities, with pointers to relevant functions and their documentation. The "usage" vignette presents more detailed usage examples and shows how to combine the functions. The "methods" vignette provides details on the underlying methods and documents implementation specifics, especially where they differ from or are undocumented in the original paper.

7.3 Usage example

We illustrate the four groups of functionalities (see Section 7.2) with the synthetic `car` data set with six features. We begin with a simple example for each functionality group and then elaborate on the options in the following sections. We first load the package and the dataset,

```
library(bnclassify)
data(car)
```

then learn a naive Bayes structure and its parameters,

```
nb <- nb('class', car)
nb <- lp(nb, car, smooth = 0.01)
```

get the number of arcs in the network,

```
narcs(nb)
```

```
[1] 6
```

get the 10-fold cross-validation estimate of accuracy,

```
cv(nb, car, k = 10)
```

```
[1] 0.8623058
```

and finally classify the entire data set

```
p <- predict(nb, car)
head(p)
```

```
[1] unacc unacc unacc unacc unacc unacc
Levels: unacc acc good vgood
```

7.3.1 Learning

The functions for structure learning, shown in Table 7.1, correspond to the different algorithms. They all receive the name of the class variable and the data set as their first two arguments, which are then followed by optional arguments. The following runs the CL-ODE algorithm with the AIC score, followed by the FSSJ algorithm to learn another model:

```
ode_cl_aic <- tan_cl('class', car, score = 'aic')
set.seed(3)
fssj <- fssj('class', car, k = 5, epsilon = 0)
```

The `bnc` function is a shorthand for learning structure and parameters in a single step,

```
ode_cl_aic <- bnc('tan_cl', 'class', car, smooth = 1, dag_args = list(score = 'aic'))
```

where the first argument is the name of the structure learning function while and optional arguments go in `dag_args`.

7.3.2 Analyzing

Printing the model, such as the above `ode_cl_aic` object, provides basic information about it.

```
ode_cl_aic
```

```

  Bayesian network classifier

  class variable:        class
  num. features:   6
  num. arcs:   9
  free parameters:   131
  learning algorithm:    tan_cl
```

While plotting the network is especially useful for small networks, printing the structure in the `deal` [Bottcher and Dethlefsen, 2013] and `bnlearn` format may be more useful for larger ones:

```
ms <- modelstring(ode_cl_aic)
strwrap(ms, width = 60)
```

```
[1] "[class] [buying|class] [doors|class] [persons|class]"
[2] "[maint|buying:class] [safety|persons:class]"
[3] "[lug_boot|safety:class]"
```

We can query the type of structure; `params` lets us access the conditional probability tables (CPTs); while `features` lists the features:

```
is_ode(ode_cl_aic)
```

```
[1] TRUE
```

```
params(nb)$buying
```

```
       class
buying         unacc          acc         good        vgood
  low   0.2132243562 0.2317727320 0.6664252607 0.5997847478
  med   0.2214885458 0.2994740131 0.3332850521 0.3999077491
  high  0.2677680077 0.2812467451 0.0001448436 0.0001537515
  vhigh 0.2975190903 0.1875065097 0.0001448436 0.0001537515
```

```
length(features(fssj))
```

```
[1] 5
```

For example, `fssj` has selected five out of six features.

`manb_arc_posterior` provides the MANB posterior probabilities for arcs from the class to each of the features:

```
manb <- lp(nb, car, smooth = 0.01, manb_prior = 0.5)
manb_arc_posterior(manb)
```

```
      buying        maint        doors      persons     lug_boot 
1.000000e+00 1.000000e+00 3.937961e-20 1.000000e+00 9.980275e-01 
      safety 
1.000000e+00 
```

With the posterior probability of 0 for the arc from ‘class‘ to ‘doors‘, and 100% for all others, MANB renders `doors` independent from the class while leaving the other features' parameters unaltered. We can see this by printing out the CPTs:

```
params(manb)$doors
```

```
       class
doors   unacc  acc good vgood
  2      0.25 0.25 0.25  0.25
  3      0.25 0.25 0.25  0.25
  4      0.25 0.25 0.25  0.25
  5more  0.25 0.25 0.25  0.25
```

```
all.equal(params(manb)$buying, params(nb)$buying)
```

```
[1] TRUE
```

For more functions for querying a structure with parameters (`bnc_bn`) see `?inspect_bnc_bn`. For a structure without parameters (`bnc_dag`), see `?inspect_bnc_dag`.

7.3.3 Evaluating

Multiple scores can be computed:

```
logLik(ode_cl_aic, car)
```

```
'log Lik.' -13307.59 (df=131)
```

```
AIC(ode_cl_aic, car)
```

```
[1] -13438.59
```

The `cv` function estimates the predictive accuracy of one or more models with stratified cross-validation. In the following we assess the above models produced by NB and CL-ODE algorithms:

```
set.seed(0)
cv(list(nb = nb, ode_cl_aic = ode_cl_aic), car, k = 5, dag = TRUE)

        nb ode_cl_aic
 0.8582303  0.9345913
```

Above, `k` is the desired number of folds, and `dag = TRUE` evaluates structure and parameter learning, while `dag = FALSE` keeps the structure fixed and evaluates just the parameter learning. The output gives 86% and 93% accuracy estimates for NB and CL-ODE, respectively. `mlr` and `caret` packages provide additional options for evaluating predictive performance, such as different metrics, and `bnclassify` is integrated with both.

7.3.4 Predicting

As shown above, we can predict class labels with `predict`. We can also get the class posterior probabilities:

```
pp <- predict(nb, car, prob = TRUE)
head(pp)

         unacc          acc         good        vgood
[1,] 1.0000000 2.171346e-10 8.267214e-16 3.536615e-19
[2,] 0.9999937 6.306269e-06 5.203338e-12 5.706038e-19
[3,] 0.9999908 9.211090e-06 5.158884e-12 4.780777e-15
[4,] 1.0000000 3.204714e-10 1.084552e-15 1.015375e-15
[5,] 0.9999907 9.307467e-06 6.826088e-12 1.638219e-15
[6,] 0.9999864 1.359469e-05 6.767760e-12 1.372573e-11
```

7.4 Runtimes

We illustrate the algorithms' runtimes, resulting structure complexity and predictive performance on the datasets listed in Table 7.2. We only used complete data sets as time-consuming inference with incomplete data makes cross-validated scores costly for medium-sized or large data sets. The structure and parameter learning methods are listed in the legends of Figure 7.1, Figure 7.2, and Figure 7.3.

Figure 7.1 shows that the algorithms with cross-validated scores, followed by WANBIA, are the most time-consuming. Runtime is still not prohibitive: TAN-HC ran for 90 seconds

Table 7.2: Used data sets, from the UCI repository [Lichman, 2013]. Incomplete rows have been removed. r_c is the number of classes (i.e., distinct class labels).

N	n	r_c	Dataset
1728	7	4	car
958	10	2	tic-tac-toe
435	17	2	voting
351	35	2	ionosphere
562	36	19	soybean
3196	37	2	kr-vs-kr
3190	61	3	splice

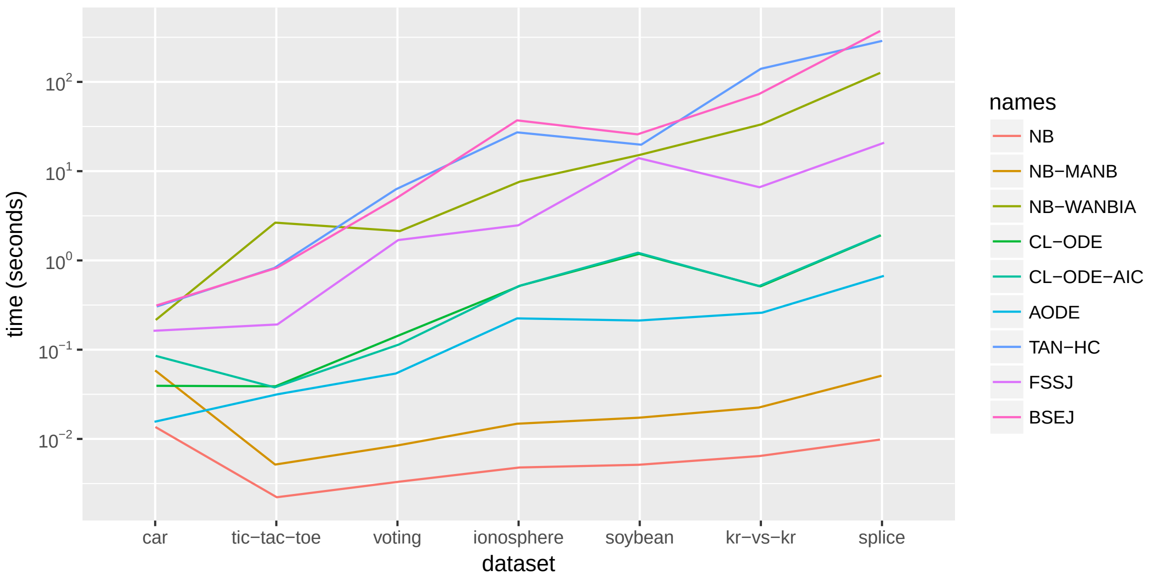


Figure 7.1: Runtimes of the algorithms on a Ubuntu 16.04 machine with 8 GB of RAM and a 2.5 GHz processor, on a $\log_{10}$ scale. We used the default options for all algorithms and `k = 5` and `epsilon = 0` for the wrappers. CL-ODE-AIC is CL-ODE with the AIC rather than the log-likelihood score. The lines have been horizontally and vertically jittered to avoid overlap where identical.

on kr-vs-kp and 123 seconds on splice, adding 27 augmenting arcs on the former and 7 on the latter (a added arcs mean a iterations of the search algorithm). Note that their runtime is linear in the number of cross-validation folds `k`; using `k` $= 10$ instead of `k` $= 5$ would have roughly doubled the time.

CL-ODE tended to produce the most complex structures (see Figure 7.2), with FSSJ learning complex models on car, soybean and splice, yet simple ones, due to feature selection, on voting and tic-tac-toe. The NB models with alternative parameters, WANBIA and MANB, have as much parameters as the NB because we are not counting the length-n weights vector, rather just the parameters $\boldsymbol{\theta}$ of the resulting NB (the weights simply produce an alternative parameterization of the NB).

In terms of accuracy, NB and MANB performed comparatively poorly on car, voting, tic-tac-toe, and kr-vs-kp, possibly because of many wrong independence assumptions (see

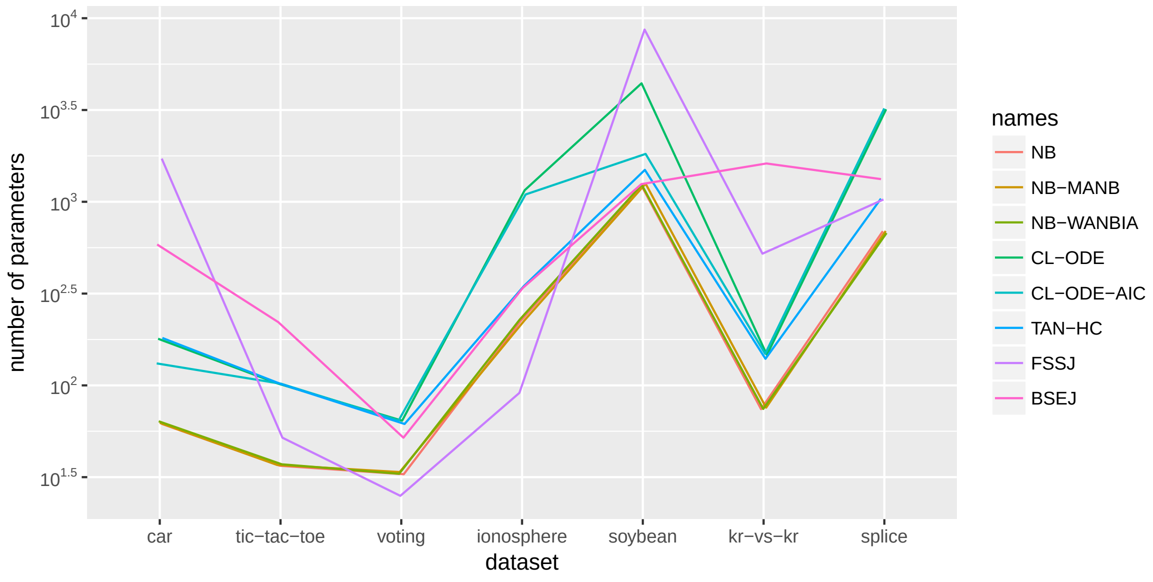


Figure 7.2: The number of Bayesian network parameters $\boldsymbol{\theta}$ of the resulting structures, on a $\log_{10}$ scale. The lines have been horizontally and vertically jittered to avoid overlap where identical.

Figure 7.3). WANBIA may accounted for some of these violations on voting and kr-vs-kp, as it outperformed NB and MANB on these datasets, showing that a simple model can perform well on them when adequately parameterized. More complex models, such as CL-ODE and AODE, performed better on car.

7.5 Implementation

With complete data, `bnclassify` implements prediction for augmented naive Bayes models as well as for ensembles of such models. It multiplies the corresponding $\boldsymbol{\theta}$ in logarithmic space, applying the *log-sum-exp* trick before normalizing, to reduce the chance of underflow. On instances with missing entries, it uses the `gRain` package [Højsgaard, 2016, Højsgaard, 2012] to perform exact inference, which is notably slower. Network plotting is implemented by the `Rgraphviz` package. Some functions are implemented in C++ with `Rcpp` for efficiency. The package is extensively tested, with over 200 unit and integrated tests that give a 94% code coverage.

7.6 Related software

NB, TAN, and AODE are available in general-purpose tools such as `bnlearn` and Weka, while WANBIA[3] and MANB[4] are only available in stand-alone software, published along with the original publications. We are not aware of available implementations of the remaining methods implemented in `bnclassify`.

[3] `https://sourceforge.net/projects/rawnaivebayes`
[4] `http://www.dbmi.pitt.edu/content/manb`

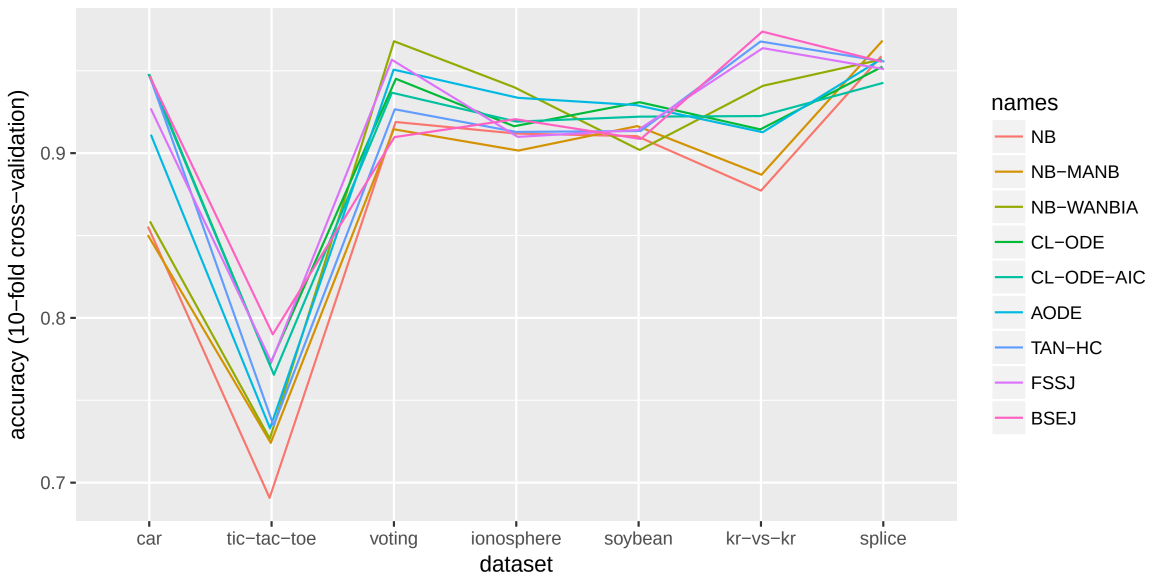


Figure 7.3: Accuracy of the algorithms estimated with stratified 10-fold cross-validation. The lines have been horizontally and vertically jittered to avoid overlap where identical.

`bnlearn` implements algorithms for learning general purpose Bayesian networks. Among them, algorithms for Markov blanket learning by testing for independences, such as IAMB [Tsamardinos and Aliferis, 2003] and GS [Margaritis and Thrun, 2000], can be very useful for classification as they can look for the Markov blanket of the class variable. `bnlearn` combines the search algorithms, such as greedy hill-climbing and tabu search [Glover and Laguna, 2013], only with generative scores such as penalized log-likelihood. Among classification models, it implements the discrete NB and CL-ODE. It does not handle incomplete data and provides cross-validation and prediction only for the NB and TAN models, but not for the unrestricted Bayesian networks.

Version 3.8 of Weka [Hall et al., 2009, Bouckaert, 2004] provides variants of the AODE [Webb et al., 2005] as well as the CL-ODE and NB. It implements five additional search algorithms, such as K2 [Cooper and Herskovits, 1992], tabu search and simulated annealing [Kirkpatrick et al., 1983], combining them only with generative scores. Except for the NB, Weka only handles discrete data and uses simple imputation (replacing with the mode or mean) to handle incomplete data. It provides two constraint-based algorithms, but performs conditional independence tests in an ad-hoc way [Bouckaert, 2004]. Weka provides Bayesian model averaging for parameter estimation [Bouckaert, 1995].

jBNC[5] (version 1.2.2) is a Java library which learns ODE classifiers from Sacha et al. [2002] by optimizing penalized log-likelihood or the cross-validated estimate of accuracy. The CGBayes (version 7.14.14) package [McGeachie et al., 2014] for MATLAB implements conditional Gaussian networks [Lauritzen and Wermuth, 1989]. It provides four structure learning algorithms, including a variant of K2 and a greedy hill-climber, all optimizing the marginal

[5]`http://jbnc.sourceforge.net/`

likelihood of the data given the network.

7.7 Conclusion

`bnclassify` implements multiple state-of-the art algorithms for learning Bayesian network classifiers. It also provides features such as model analysis and evaluation. It is reasonably efficient and can handle medium-sized data sets. We hope that `bnclassify` will be useful to practitioners as well as to researchers wishing to compare their methods to existing ones. So far, `bnclassify` has been downloaded over 11 thousand times from the RStudio mirror of the Comprehensive R Archive Network (CRAN). There are roughly a thousand downloads per package update, suggesting that there are that many existing installations.

Future work includes handling real-valued features via conditional linear Gaussian models. Straightforward extensions include adding flexibility to the hill-climbing algorithm, such as restarts to avoid local minima.

Part III

CONTRIBUTIONS TO INTERNEURON CLASSIFICATION

Chapter 8

Discrete Bayesian network classifiers with majority labels

8.1 Introduction

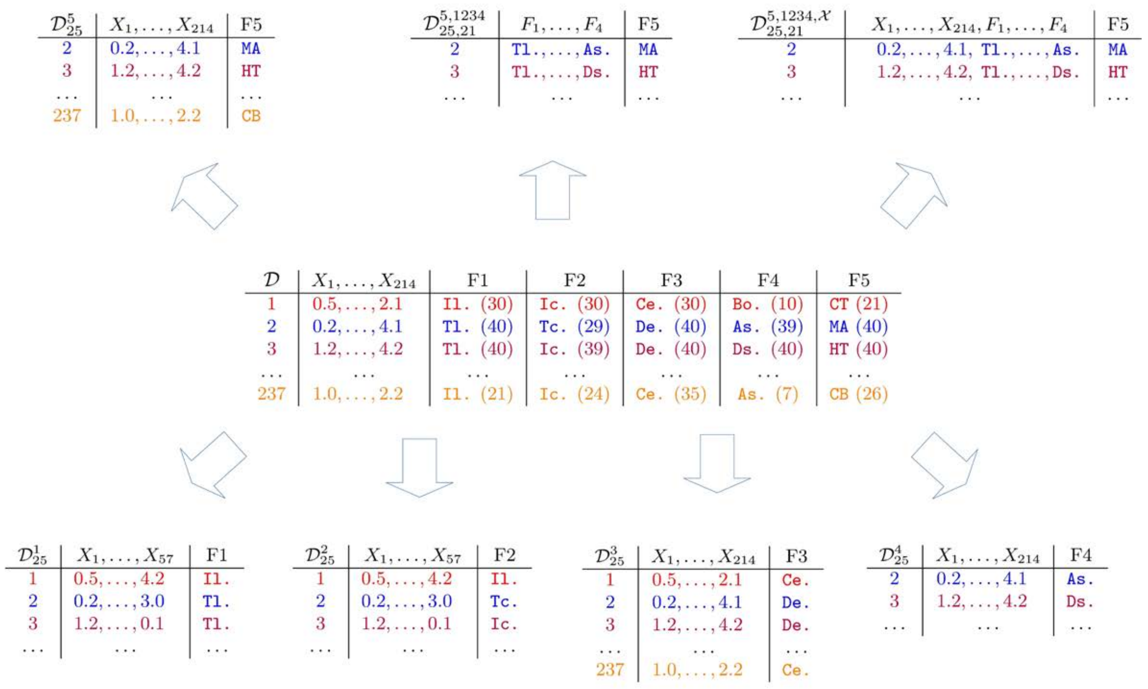
In this chapter, we approach the supervised classification of Gardener's interneurons by using the majority vote label (see Section 3.6). Since this approach resulted in limited accuracy for the interneuron type in DeFelipe et al. [2013], we are interested whether at least the reliably categorized interneurons can be accurately classified by a model. We thus form data subsets by increasing the threshold on label reliability, which we define as the minimal number of neuroscientists agreeing on the majority type. We apply supervised classification on each obtained data subset.

We separately categorize the interneurons according to interneuron type and four high-level axonal features, termed F1, F2, F3, and F4 (see Section 8.2 for details). We measure 214 parameters of axonal and dendritic arborizations and use all or some of them as predictive variables, depending on the axonal feature/type to be predicted. Additionally, we use axonal features F1–F4 as predictors of the interneuron type, both on their own and together with the morphological parameters. We estimate their values with majority vote and reliability threshold as well, thus discarding, when using them as predictors, interneurons unreliably categorized according to at least one of them. Figure 8.1 shows an overview of the described approach.

We tackle each classification task with four different Bayesian network classifiers (see Chapter 4). These are competitive performance classifiers [Morales et al., 2013, Friedman et al., 1997] that allow for analyzing probabilistic relationships among the variables of a domain.

The research covered in this chapter has been published in Mihaljević et al. [2015a].

The rest of this chapter is organized as follows. Section 8.2 describes the data and the practical approach to interneuron classification and then elaborates on the methodology — the formation of data sets according to label reliability; the Bayesian network classifiers



$\mathcal{D}^{5}_{25}$	$X_1,\dots,X_{214}$	F5
2	$0.2,\dots,4.1$	MA
3	$1.2,\dots,4.2$	HT
…	…	…
237	$1.0,\dots,2.2$	CB

$\mathcal{D}^{5,1234}_{25,21}$	$F_1,\dots,F_4$	F5
2	Tl.,…,As.	MA
3	Tl.,…,Ds.	HT
…	…	…

$\mathcal{D}^{5,1234,\mathcal{X}}_{25,21}$	$X_1,\dots,X_{214},F_1,\dots,F_4$	F5
2	$0.2,\dots,4.1$, Tl.,…,As.	MA
3	$1.2,\dots,4.2$, Tl.,…,Ds.	HT
…	…	…

$\mathcal{D}$	$X_1,\dots,X_{214}$	F1	F2	F3	F4	F5
1	$0.5,\dots,2.1$	Il. (30)	Ic. (30)	Ce. (30)	Bo. (10)	CT (21)
2	$0.2,\dots,4.1$	Tl. (40)	Tc. (29)	De. (40)	As. (39)	MA (40)
3	$1.2,\dots,4.2$	Tl. (40)	Ic. (39)	De. (40)	Ds. (40)	HT (40)
…	…	…	…	…	…	…
237	$1.0,\dots,2.2$	Il. (21)	Ic. (24)	Ce. (35)	As. (7)	CB (26)

$\mathcal{D}^{1}_{25}$	$X_1,\dots,X_{57}$	F1
1	$0.5,\dots,4.2$	Il.
2	$0.2,\dots,3.0$	Tl.
3	$1.2,\dots,0.1$	Tl.
…	…	…

$\mathcal{D}^{2}_{25}$	$X_1,\dots,X_{57}$	F2
1	$0.5,\dots,4.2$	Il.
2	$0.2,\dots,3.0$	Tc.
3	$1.2,\dots,0.1$	Ic.
…	…	…

$\mathcal{D}^{3}_{25}$	$X_1,\dots,X_{214}$	F3
1	$0.5,\dots,2.1$	Ce.
2	$0.2,\dots,4.1$	De.
3	$1.2,\dots,4.2$	De.
…	…	…
237	$1.0,\dots,2.2$	Ce.

$\mathcal{D}^{4}_{25}$	$X_1,\dots,X_{214}$	F4
2	$0.2,\dots,4.1$	As.
3	$1.2,\dots,4.2$	Ds.
…	…	…

Figure 8.1: A schematic overview of our automatic categorization of interneurons according to type and high-level axonal features F1–F4. The full data set, $\mathcal{D}$ (center), is used to form data subsets (shown around $\mathcal{D}$) with different class variables (indicated by the superscript; e.g., F1 is the class variable in $\mathcal{D}^{1}_{25}$), predictor variables (second and third superscript; e.g., $\mathcal{D}^{5,1234}_{25,21}$ has features F1–F4 as predictors of the type), label reliability threshold (first subscript; e.g., $\mathcal{D}^{1}_{25}$ has label reliability 25), and predictor reliability threshold (second subscript; e.g., $\mathcal{D}^{5,1234}_{25,21}$ has reliability 21 for predictors F1–F4). In $\mathcal{D}$, all instances are quantified with 214 morphological parameters and labeled with the majority vote for the interneuron type and each high-level axonal feature, with the label reliability (number of agreeing experts) shown in parentheses. The predictive variables in the classification tasks are indicated by the columns of the corresponding data sets (e.g., the predictors for F1 are the morphological parameters X_1–X_{57}; see $\mathcal{D}^{1}_{25}$). Note how label and predictor reliability determine which instances are included in a data set: for example, the first instance in $\mathcal{D}$ (shown in red) is omitted from $\mathcal{D}^{5}_{25}$ because its label reliability is 21, i.e., it is not above 25, the label reliability threshold in $\mathcal{D}^{5}_{25}$. Likewise, instance 237 in $\mathcal{D}$ (shown in orange) is omitted from $\mathcal{D}^{5,1234}_{25,21}$ and $\mathcal{D}^{5,1234,\mathcal{X}}_{25,21}$ because its reliability for F1 is not above 21. Besides the label reliability thresholds depicted here, many others were considered for each categorization task, e.g., $\mathcal{D}^{1}_{24}$, $\mathcal{D}^{1}_{28}$, $\mathcal{D}^{5,1234,\mathcal{X}}_{30,21}$, etc.

used; data preprocessing; and the experimental setting. Section 8.3 presents the results and Section 8.4 rounds off with conclusions.

8.2 Materials and methods

8.2.1 Morphology reconstructions and class labels

We had the digital reconstructions of 241 cells which DeFelipe et al. [2013] had obtained from NeuroMorpho.Org [Ascoli et al., 2007]. 42 leading neuroscientsts had classified these cells, by looking at 2D images and 3D reconstructions of their morpologies, into one of the ten classes of the Gardener's scheme and according to four categorical features of axonal morphology (see Section 5.4.2). 40 cells had an interrupted axonal process. For 36 of them we drew the small missing fragments with the Neurolucida workstation [Glaser and Glaser, 1990] and omitted the remaining four cells with large missing parts, reducing our sample to 237 cells. The cells come from different cortical areas of the rat, mouse, and monkey. Recall from Section 5.4.2, that the axonal features take the following values:

- Feature 1 (F1): `intralaminar` and `translaminar`
- Feature 2 (F2): `intracolumnar` and `transcolumnar`
- Feature 3 (F3): `centered` and `displaced`
- Feature 4 (F4): `ascending`, `descending`, and `both`
- Feature 5 (F5): `arcade` (`AR`), `Cajal-Retzius` (`CR`), `chandelier` (`CH`), `common basket` (`CB`), `common type` (`CT`), `horse-tail` (`HT`), `large basket` (`LB`), `Martinotti` (`MA`), `neuroglia-form` (`NG`), and `other` (`OT`)
- Feature 6 (F6): `characterized` and `uncharacterized`

8.2.2 Expert categorization reliability

Each cell was categorized according to every feature by up to 42 experts. To crisply categorize a cell according to an axonal feature f, we reduced the vector of experts' choices for f to its mode (majority vote). We used such crisp categorizations as values of the class variable (labels) and of the predictor variables —high-level axonal features F1–F4 were used as predictors of interneuron type. Cells with no unique majority vote for a feature f were discarded from classification tasks that involved f, either as the class or as a predictor variable (e.g., a cell without a unique majority vote for F4 was omitted when predicting F4 and when using F4 as a predictor of the type; it was used in all other classification tasks, e.g., when predicting F2 from the morphological variables ($\mathcal{D}^2_{25}$ in Figure 8.1)). Furthermore, we formed data subsets with different label reliability thresholds, i.e., such that each instance's label was backed by at least a certain (threshold) number of experts. Thus, a data set $\mathcal{D}^f_t$, for predicting f, was formed of cells with label reliability larger than t for feature f, with $t \in \{0, \ldots, 41\}$. When

	F1	F2	F3	F4
Expert 1	translaminar	...	displaced	descending
Expert 2	intralaminar	...	displaced	none
Expert 3	translaminar	...	centered	none
Consensus	translaminar	...	displaced	none

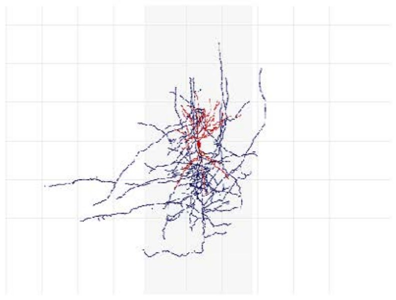


Figure 8.2: An example of a theoretically invalid characterization arising due to majority vote categorization —a cell might be categorized as `translaminar` and `displaced` in F1 and F3 but neither as `ascending`, `descending`, nor `both`, but instead as `none`, in F4. The table shows a hypothetical categorization of a cell by three experts. While `translaminar` and `displaced` are the modes for F1 and F3, only one expert selected them simultaneously, and therefore only he/she also categorized the cell according to F4. The rest found that the cell was not characterisable according to F4, which was registered with the value `none`. `none` was, therefore, the mode for F4. In our data, there were only three such improperly categorized cells with reliability threshold 21 applied to axonal features F1–F4. One of them is the `CT` cell (according to 21 experts) shown on the right, characterized as `translaminar`, `intracolumnar`, `displaced`, and `none`.

using features F1–F4 as predictors, cells were additionally filtered according to reliability for F1–F4. Thus, a data set $\mathcal{D}_{t,r}^{5,1234}$, for predicting the type with features F1–F4 as predictors, is formed of cells with reliability greater than r for each of the features F1–F4, and reliability greater than t for the interneuron type (the label). A data set $\mathcal{D}_{t,r}^{5,1234,\mathcal{X}}$, with both the morphological parameters and high-level axonal features as predictors of the type, is formed in the same way as $\mathcal{D}_{t,r}^{5,1234}$. When using F1–F4 as predictors, we augmented F4 with a category called `none`, to describe the cells which most experts considered as not categorizable according to F4 —these cells would have otherwise been discarded due to few experts having categorized them according to F4, yielding a low reliability for F4. Although this might lead to incorrect categorizations —a cell being `translaminar`, `displaced` and neither `ascending`, `descending` or `both` but instead `none`, in F1, F3, and F4, respectively— such combinations barely appeared in our data (see Figure 8.2).

8.2.3 Morphological variables

We used Neurolucida Explorer, the data analysis companion to Neurolucida, to compute 214 parameters of dendritic and axonal morphology using, among others, the following morphological analyses:

- Vertex analysis (described in Sadler and Berry [1983]): the count of three types of bifurcations —those with two terminal branches attached (V_a); with one terminal branch attached (V_b); and with two bifurcating branches attached (V_c).
- Convex hull analysis: various measures of how much space the arbor occupies.
- Sholl analysis: a histogram of intersections of the arbor and a series of concentric

spheres centered at the soma. Besides the intersections, we computed histograms of the endings, nodes, and arbor length between two contiguous spheres.

- Fractal analysis (described in Panico and Sterling [1995]): box-counting k-dimension —a measure of how well the arbor fills the space.
- Fan-in analysis [Glaser and McMullen, 1984]: torsion ratio —a measure indicative of any preferred orientation of the arbor.
- Polar histogram [McMullen et al., 1984]: a round directional histogram of total arborization length corresponding to an angle interval.

Table 8.1 shows all the parameters that we computed, grouped by morphological analysis. We applied each analysis, except for 'Dendritic analysis', to both the axon and the dendrites. So, for example, we computed the torsion ratio of the axon and the torsion ratio of the dendrites. In total, we computed 128 axonal and 86 dendritic parameters.

We used all the computed parameters as predictive variables for predicting axonal features F3, F4, and F5. For predicting features F1 and F2 we used only the following 57 axonal parameters: total length ($\mathbf{B}_3$), number of endings ($\mathbf{B}_1$), mean branch length ($\mathbf{B}_6$), torsion ratio ($\mathbf{FI}_1$), convex hull parameters ($\mathbf{C}_1$–$\mathbf{C}_4$), Sholl analysis of intersections starting from radius 240 μm ($\mathbf{S}_4$–$\mathbf{S}_{16}$), and polar histogram ($\mathbf{P}_1$–$\mathbf{P}_{36}$).

8.2.4 Discrete Bayesian network classifiers

We used four Bayesian network classifiers. Three are variants of the naive Bayes: plain naive Bayes [Minsky, 1961] (NB), forward sequential selection naive Bayes [Langley and Sage, 1994] (NB-FSS), and the attribute-weighted naive Bayes Hall [2007] (AWNB). The fourth is the tree augmented Bayes[Friedman et al., 1997] (TAN). See Chapter 4 for details on these classifiers.

8.2.5 Discretization and dimensionality reduction

Before classifier induction, we converted all numeric variables (i.e. the morphological parameters) to categorical ones. This process, known as discretization [Yang et al., 2010], often yields more accurate naive Bayes classifiers than when assumptions such as that of normality are made about the underlying probability distributions [Dougherty et al., 1995]. We used the equal-frequency discretization technique and determined the number of intervals as a function of data set size, following the weighted proportional k-interval discretization (WPKID) method [Yang and Webb, 2003]. The discretization process did not bias accuracy estimates as it was guided only by training data (within a cross-validation scheme): the test data were mapped, upon classification, to the intervals learned from training data.

The number of predictor variables (up to 218) was possibly too high for NB and TAN to perform well, as they do not perform predictor selection. Thus, after discretizing the training data and before inducing the classifiers (i.e., on the training set within a cross-validation scheme), we reduced the predictor set to the 100 variables with the highest mutual

Table 8.1: Axonal and dendritic parameters, grouped by morphological analysis. The parameter abbreviations (shown in bold) indicate the number of parameters corresponding to each analysis (e.g., there were nine branching parameters); the exception is Sholl analysis for which there were 20 dendritic and 64 axonal parameters, because more spheres were considered for the axon. The 'dendritic' analysis (lowermost column) applies only to the dendrites. The rightmost column shows examples of applying the analyses to a cell's axon.

Analysis	Parameters	Axon Examples
Branching	$\mathbf{B}_1$–$\mathbf{B}_2$ Number of endings and bifurcations	$\mathbf{B}_1 =$ 962.00
	$\mathbf{B}_3$–$\mathbf{B}_4$ Total and mean arbor length	$\mathbf{B}_3 =$ 41,697.10 μm
	$\mathbf{B}_5$–$\mathbf{B}_8$ Total, mean, median and std. deviation of branch length	$\mathbf{B}_6 =$ 21.75 μm
	$\mathbf{B}_9$ Highest branch order	$\mathbf{B}_9 =$ 103.00
Convex hull	$\mathbf{C}_1$–$\mathbf{C}_2$ Area and perimeter of 2D convex hull	$\mathbf{C}_1 =$ 174,185.00 μm^2
	$\mathbf{C}_3$–$\mathbf{C}_4$ Volume and surface of 3D convex hull	$\mathbf{C}_3 =$ 18,864,800.00 μm^3
Sholl	Spheres of radii $\{60\ \mu m, 2 \times 60\ \mu m, \ldots, R \times 60\ \mu m\}$, with $R = 16$ for the axon and $R = 5$ for the dendrites.	
	$\mathbf{S}_1$–$\mathbf{S}_R$ Intersections with the R spheres	$\mathbf{S}_1 =$ 57.00
	$\mathbf{S}_{R+1}$–$\mathbf{S}_{2R}$ Endings within the R spheres	$\mathbf{S}_{17} =$ 84.00
	$\mathbf{S}_{2R+1}$–$\mathbf{S}_{3R}$ Nodes within the R spheres	$\mathbf{S}_{33} =$ 91.00
	$\mathbf{S}_{3R+1}$–$\mathbf{S}_{4R}$ Arbor length within the R spheres	$\mathbf{S}_{49} =$ 3,528.00 μm
Fractal	$\mathbf{F}_1$ Box-counting k-dimension	$\mathbf{F}_1 =$ 1.49
Vertex	$\mathbf{V}_1$–$\mathbf{V}_4$ V_a, V_b, V_c, and $\frac{V_a}{V_b}$	$\mathbf{V}_1 =$ 225.00
Branch angle	$\mathbf{BA}_1$–$\mathbf{BA}_9$ Mean, median, and std. deviation of planar, local and spline bifurcation angles	$\mathbf{BA}_1 =$ 1.11 rad
Fan-in	$\mathbf{FI}_1$ Torsion ratio	$\mathbf{FI}_1 =$ 1.16
Polar histogram	36 angle intervals of width 0.17 radians, starting with [0 rad, 0.17 rad).	$\mathbf{P}_1 =$ 625.40 μm
	$\mathbf{P}_1$–$\mathbf{P}_{36}$ Length corresponding to the angle intervals	
Dendritic	$\mathbf{D}_1$ Number of primary dendrites	Does not apply
	$\mathbf{D}_2$ Number of bifurcations of primary dendrites	

information with the class variable[1]. Since predictors were selected from the training set alone, this did not bias the cross-validated accuracy estimates [Smialowski et al., 2010].

8.2.6 Experimental setting

8.2.6.1 Label reliability thresholds

The number of cells, naturally, decreased with higher label reliability. We only considered label reliability thresholds with no fewer than five instances of at least two classes, which provided the upper bounds for the reliability thresholds used: the bound was 40 for axonal features F1–F4 (see Figure 8.3a–Figure 8.3d), and 28 for F5 (Figure 8.3e). The lower bound in all classification tasks was ten, roughly corresponding to one quarter of the experts.

There were seven interneuron types up to threshold 24; no `NG` cells remained on higher thresholds. Regardless of the threshold there were fewer than five `CH`, `HT`, and `NG` cells, making these types especially hard to identify. Regarding F4, no `both` cells remained above threshold 28. The predictions of F1, F2, and F3 were binary tasks regardless of label reliability threshold, i.e., the same classes were present at all thresholds considered.

When using F1–F4 as predictors of the interneuron type (F5) we fixed the reliability threshold for F1–F4 to 21 —corresponding to 50% of the experts— while considering all thresholds from 10 to 28 for F5 (see Figure 8.3f), following the above-described criteria.

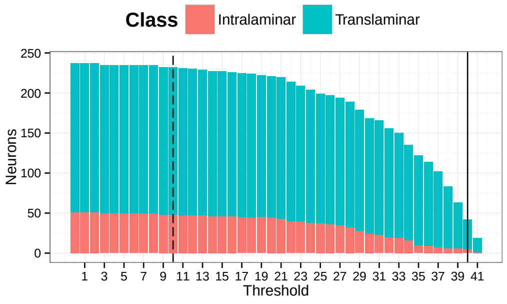
8.2.6.2 Classifier parametrization and accuracy estimation

For NB-FSS we used resubstitution accuracy as the objective function; we halted its search process when new accuracy improved current accuracy by no more than 10% (i.e., $acc_{\text{new}} - acc_{\text{current}} \leq 10$, $acc \in [0, 100]$). For the AWNB classifier, we built classification trees from 10 bootstrap samples half the size of the data set ($\frac{N}{2}$). We estimated the parameters of the Bayesian networks with Laplace correction for maximum likelihood. We estimated predictive accuracy of the classifiers with five repetitions of five-fold stratified cross-validation.

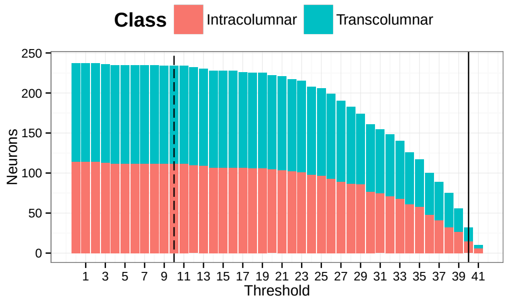
8.2.6.3 Software

The Bayesian network classifiers used are implemented in the `bnclassify` [Mihaljević et al., 2018] package for the `R` [R Core Team, 2015] statistical software environment. We used `Weka` [Hall et al., 2009] for discretization (through the `RWeka` [Hornik et al., 2009] interface for `R`) and the `caret` `R` package [Kuhn, 2008] for cross-validation estimation of accuracy.

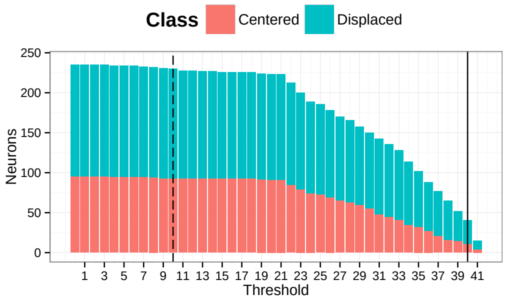
[1]This was not applied in classification tasks with less than 100 predictors, e.g., when predicting the interneuron type with only F1–F4 as predictor variables.



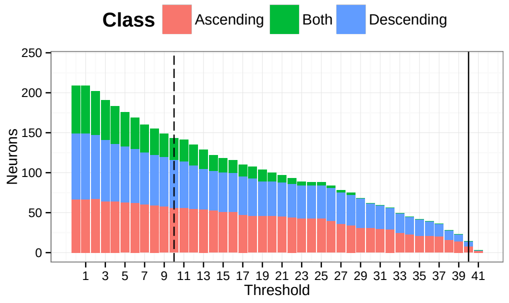
(a) F1



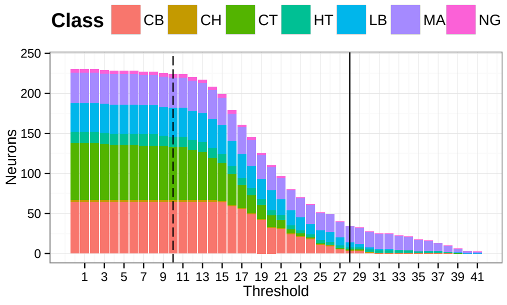
(b) F2



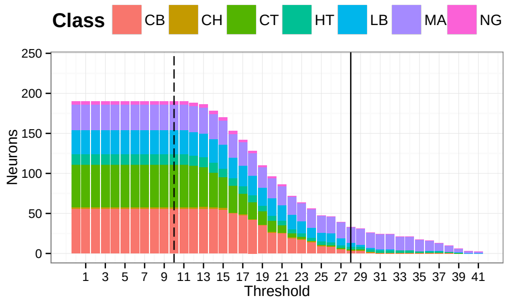
(c) F3



(d) F4

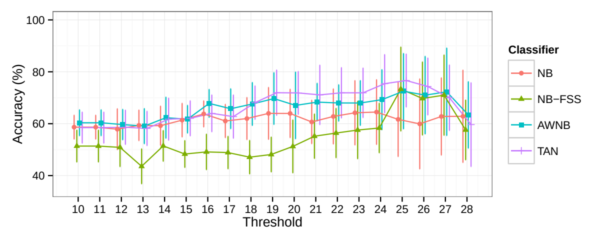


(e) F5

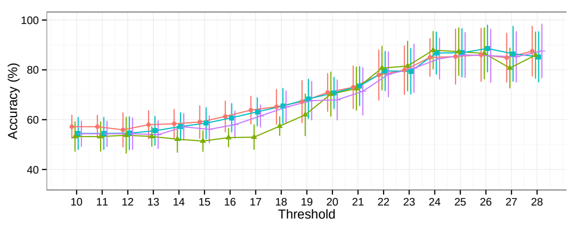


(f) F5 with threshold 21 applied to F1–F4

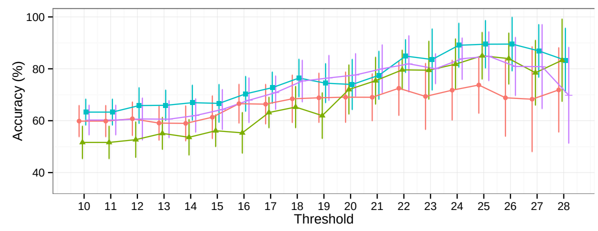
Figure 8.3: Number of cells of different classes of F1–F5, versus label reliability threshold. Note that F1 is unbalanced: there are many more `translaminar` than `intralaminar` cells. Regarding F4, there are no `both` cells above threshold 28 whereas for F5 there are no `NG` cells beyond threshold 24 in both (e) and (f). Solid vertical lines indicate the highest label reliability threshold with no fewer than five instances of at least two classes. Dashed vertical lines indicate the lowest label reliability threshold considered for classification.



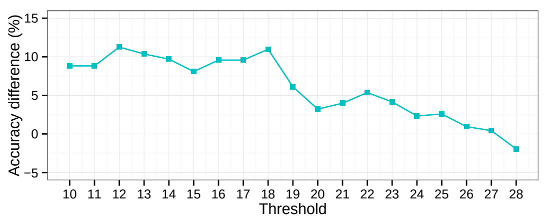
(a) $\mathcal{D}^{5}_{10-28}$



(b) $\mathcal{D}^{5,1234}_{10-28,21}$



(c) $\mathcal{D}^{5,1234,\mathcal{X}}_{10-28,21}$



(d) AWNB at (c) - AWNB at (b)

Figure 8.4: Interneuron type classification accuracy versus label reliability threshold. (a) With morphological variables as predictors ($\mathcal{D}^{5}_{10-28}$); (b) with axonal features F1-F4 as predictors ($\mathcal{D}^{5,1234}_{10-28,21}$); and (c) with axonal features F1-F4 and morphological variables as predictors ($\mathcal{D}^{5,1234,\mathcal{X}}_{10-28,21}$). For (b) and (c), values of F1-F4 were obtained with a label reliability threshold 21, the used data thus being a subset of data used in (a). (b) and (c) were produced by identical cross-validation produces (i.e., no differences between them due to chance). (d) plots the accuracy of AWNB at $\mathcal{D}^{5,1234,\mathcal{X}}_{10-28,21}$ minus its accuracy at $\mathcal{D}^{5,1234}_{10-28,21}$. Error bars in (a), (b), and (c) show the standard deviation of accuracy from five runs of five-fold cross-validation.

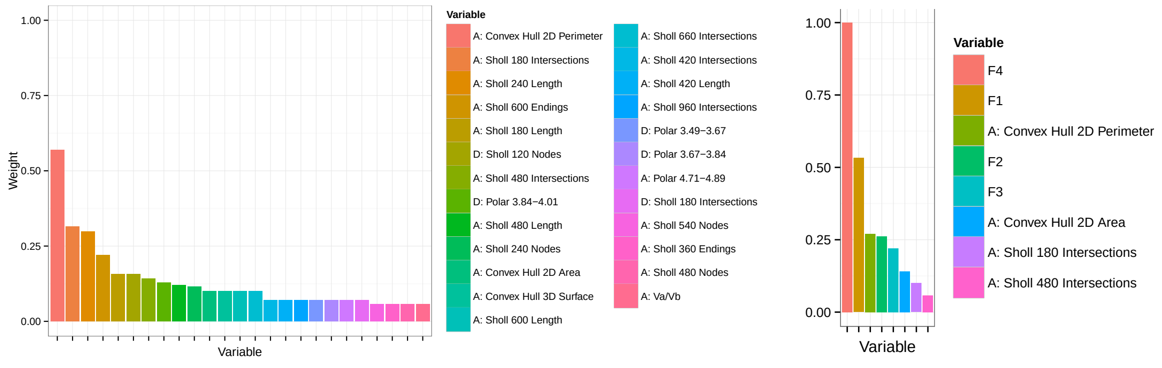
8.3 Results

8.3.1 Predicting interneuron type

8.3.1.1 From morphological variables

In general, accuracy improved with label reliability (see Figure 8.4a). Best accuracy —76.63%— was achieved by TAN at label reliability threshold 25 ($\mathcal{D}^{5}_{25}$). While a TAN model incorporates all predictive variables[2], AWNB and NB-FSS achieved comparable accuracy, at this threshold, with few variables: AWNB with 24 (19 axonal and five dendritic) and NB-FSS with two axonal variables: the 2D convex hull perimeter ($\mathbf{C}_2$) and Sholl intersections at 180 μm from soma ($\mathbf{S}_3$); these were also the most relevant variables according to AWNB —see Figure 8.5a. At thresholds 25–27 NB-FSS was very accurate by using these two variables alone, indicating that they suffice for discriminating among `CB`, `LB`, and `MA` cells, which are the interneuron types that NB-FSS was able identify at these thresholds (see Table 8.2). Unlike NB-FSS, TAN also managed to identify `HT` cells at threshold 25, thus accurately discriminating among `CB`, `HT`, `LB`, and `MA` cells (see Table 8.3).

[2]The 100 variables that were selected previous to classifier induction.



(a) (b)

Figure 8.5: AWNB weights for predictor variables at (a): $\mathcal{D}_{25}^{5}$ and (b): $\mathcal{D}_{26,21}^{5,1234,\mathcal{X}}$. Only variables with weights greater than zero are shown. 'A:' denotes an axonal variable whereas a 'D:' denotes a dendritic variable. The numbers in Sholl variables refer to distances of spheric rings from soma, in μm, e.g., 'A: Sholl 600 Length' is the axonal arborization length within the spheric ring at 540 μm–600 μm from soma. The numbers in the polar histogram variables refer to radian intervals, e.g., 'A: Polar 4.71-4.89' refers to axonal arborization length corresponding to the angle between 4.71 and 4.89 radians. The weights were computed by learning an AWNB model from the full data set, after selecting the 100 variables with the highest mutual information with the class variable.

Table 8.2: NB-FSS' confusion matrix for $\mathcal{D}_{25}^{5}$ (i.e., for predicting feature F5 at threshold 25; Figure 8.1 explains this notation). Columns denote the true classes whereas rows denote the predicted classes. Zeros were omitted. Bottom: NB-FSS' sensitivity and specificity per class. All values in the table were computed from a single run of stratified five-fold cross-validation and might not, therefore, exactly match the accuracy reported in Figure 8.4a (yielded by five runs).

	CB	CH	CT	HT	LB	MA
CB	10	1			2	1
CH						
CT						
HT						
LB					9	2
MA	1		1	4	1	19
Sensitivity	0.91	0.00	0.00	0.00	0.75	0.86
Specificity	0.90	1.00	1.00	1.00	0.95	0.76

Table 8.3: TAN's confusion matrix for $\mathcal{D}^{5}_{25}$. Columns denote the true classes whereas rows denote the predicted classes. Zeros were omitted. Bottom: TAN's sensitivity and specificity per class. All values in the table were computed from a single run of stratified five-fold cross-validation and might not, therefore, exactly match the accuracy reported in Figure 8.4a (yielded by five runs).

	CB	CH	CT	HT	LB	MA
CB	10	1			1	
CH						
CT						
HT				3		
LB					8	3
MA	1		1	1	3	19
Sensitivity	0.91	0.00	0.00	0.75	0.67	0.86
Specificity	0.95	1.00	1.00	1.00	0.92	0.79

Table 8.4: AWNB's confusion matrix for $\mathcal{D}^{5,1234}_{26,21}$. Columns denote the true classes whereas rows denote the predicted classes. Zeros were omitted. Bottom: AWNB's sensitivity and specificity per class. All values in the table were computed from a single run of stratified five-fold cross-validation and might not, therefore, exactly match the accuracy reported in Figure 8.4b (obtained by five runs).

	CB	CH	CT	HT	LB	MA
CB	6		1		1	
CH						
CT						
HT		1		4		
LB	2				9	
MA					1	21
Sensitivity	0.75	0.00	0.00	1.00	0.82	1.00
Specificity	0.95	1.00	1.00	0.98	0.94	0.96

8.3.1.2 From axonal features F1–F4

In general, accuracy improved with label reliability (see Figure 8.4b) and NB, AWNB, and TAN were similarly accurate at all thresholds. The best accuracy —88.58%— was achieved by AWNB at threshold 26 ($\mathcal{D}^{5,1234}_{26,21}$). All classifiers could accurately discriminate between reliable examples of the `CB`, `HT`, `LB`, and `MA` types (see Table 8.4 for AWNB). Not only were they similarly accurate but they actually classified in a similar way —for example, TAN and NB-FSS had identical confusion matrices at threshold 26. Prediction was more accurate than with morphological predictors alone (note that, although different, the data sets from the two settings were actually similar at high reliability thresholds, see Figure 8.3e and Figure 8.3f).

F4 seemed to be the most useful high-level axonal feature for predicting the interneuron type. Regardless of label reliability, AWNB always assigned most importance to F4, then to F2, and least to F1 and F3 (see Figure 8.6). Accordingly, the NB-FSS classifier selected F4, while omitting F1 and F3, at all thresholds; when it selected F2 along with F4 —at thresholds 20–27— it was more or similarly accurate as the remaining classifiers. Indeed, features F4 and F2 alone could separate reliable examples of the `CB`, `HT`, `LB`, and `MA` types

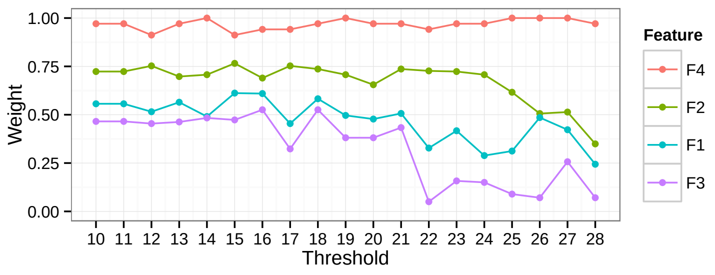


Figure 8.6: AWNB predictor weights for predicting F5 from axonal features F1–F4 versus label reliability threshold. Note that the order of importance of the features is constant across the thresholds: most weight is given to F4, least to F3, with F2 and F1 in between.

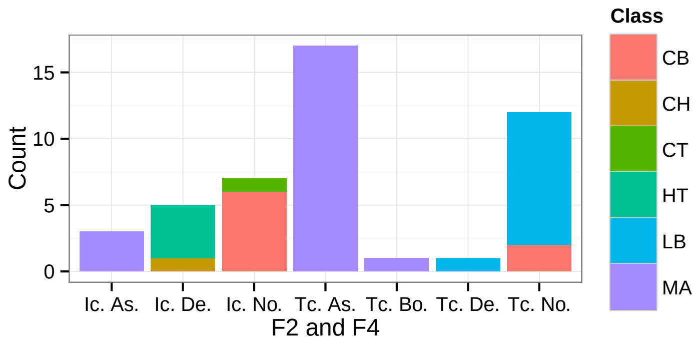


Figure 8.7: Interneuron type (colours) versus combination of F2 and F4 values at threshold 26 for interneuron type and threshold 21 for F1–F4 ($\mathcal{D}^{5,1234}_{26,21}$). F2 and F4 combined could discriminate between `CB`, `HT`, `LB`, and `MA` and cells rather precisely (e.g., all `transcolumnar` and `ascending` cells were `MA`). Feature F4 alone separated `MA` from `HT`: all `HT` were `descending` whereas `MA` were either `ascending` or `both`, whereas F2 largely separated `CB` from `LB` cells: all `LB` were `transcolumnar` while most `CB` were `intracolumnar`. Abbreviations: Ic. = `intracolumnar`; Tc. = `transcolumnar`; As. = `ascending`; De. = `descending`; Bo. = `both`; and No. = `none`.

(see Figure 8.7). The omission of F1 and F3, in favour of F4, by NB-FSS, and their lower importance in AWNB, is reasonable since F4 by definition carries information about F1 and F3 —a cell that is `none` in F4 is not, by definition, `translaminar` and `displaced` (in F1 and F3, respectively), whereas a cell with a different F4 value (`ascending`, `descending`, or `both`) is `translaminar` and `displaced` (in F1 and F3, respectively). This redundancy of F4 on the one hand and F1 and F3 on the other might suggest that predictor weighting is an adequate approach.

Nevertheless, not even the combination of the four high-level axonal features is expressive enough to separate the types well at low reliability thresholds. Many cells had identical axonal features F1–F4 but nevertheless belonged to different types; since a single combination of features can only be assigned to one type, many cells cannot be correctly classified. Poor accuracies at low thresholds are partially due to this limited expressiveness; in fact, they are close to the accuracies achievable by assigning each instance to its majority class (see Figure 8.8). This suggests that a richer predictor space (i.e., beyond F1–F4) might be necessary to better discriminate among interneuron types at low thresholds. For this purpose, we

Table 8.5: Sensitivity, specificity and accuracy of the different classifiers at $\mathcal{D}^1_{37}$. `intralaminar` is considered as the positive class.

	Sensitivity	Specificity	Accuracy
NB	1	0.88	88.43%
NB-FSS	0	1	93.19%
AWNB	0	1	93.19%
TAN	0.11	0.97	90.78%

augmented the predictor set with the 214 morphological variables; the obtained results are presented in the following section.

8.3.1.3 From morphological variables and axonal features F1–F4

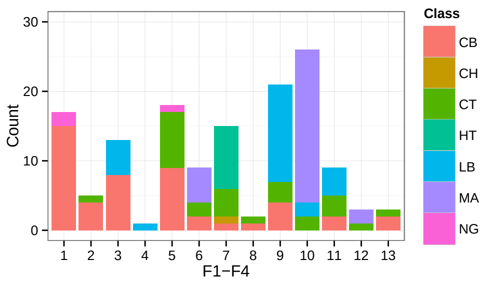
Using the morphological variables together with the high-level axonal features improved AWNB's accuracy at all thresholds except for 28 (see Figs. 8.4c and 8.4d). AWNB achieved the highest overall accuracy (considering all three predictor sets) for predicting the interneuron type —89.52%— at threshold 26 ($\mathcal{D}^{5,1234,\mathcal{X}}_{26,21}$). At this threshold, it assigned non-zero weights to eight predictors: the features F1–F4 and four morphological parameters of the axon (see Figure 8.5b), with most weight assigned to F4. These morphological parameters were also used by AWNB in Section 8.3.1.1 (see Figure 8.5a). In both settings most weight was assigned to the 2D convex hull perimeter ($\mathbf{C}_2$).

Augmenting the predictor set with morphological variables also improved the accuracies of TAN and NB —which do not prune the predictor set— at low thresholds (up to thresholds 22 and 19 for TAN and NB, respectively; see Figure 8.4c). This seems to confirm that poor accuracies at low thresholds in the previous setting were partially due to the limited expressiveness of features F1–F4. NB and TAN performed worse at high thresholds —where features F2 and F4 suffice to discriminate among the types—, possibly because the high-level axonal features were dominated by the many morphological variables, only some of which seem to be useful for class prediction.

8.3.2 Predicting axonal features F1–F4

The highest accuracy for predicting the axonal feature F1 —93.19%— was achieved by the AWNB and NB-FSS classifiers at threshold 37 ($\mathcal{D}^1_{37}$; see Figure 8.9a). These classifiers, however, assigned all cells to the predominant `translaminar` class (at threshold 37 there were 95 `translaminar` and 7 `intralaminar` cells). NB was best at correctly identifying `intralaminar` cells but had a lower accuracy that the remaining classifiers (88.43%; see Table 8.5).

Regarding the prediction of the axonal feature F2, the highest accuracy —93.75%— was achieved by AWNB at label reliability threshold 39 ($\mathcal{D}^2_{39}$; see Figure 8.9b). It was similarly good at identifying both categories of interneurons (0.95 sensitivity and 0.91 specificity, with `intracolumnar` being the positive class). Generally, AWNB was most accurate at classifying



(a)

	CB	CH	CT	HT	LB	MA	NG
CB	39		11		5		3
CH							
CT							
HT	1	1	4	9			
LB	6		6		19		
MA	2		5		2	29	
NG							

Accuracy: 67.60%

(b)

	CB	CH	CT	HT	LB	MA	NG
CB	36		9		6		3
CH							
CT	1		1		3		
HT	1	1	4	9			
LB	7		6		15		
MA	3		6		2	29	
NG							

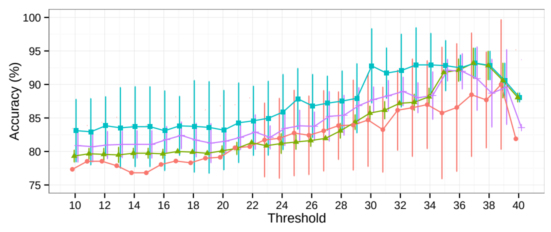
Accuracy: 63.38%

(c)

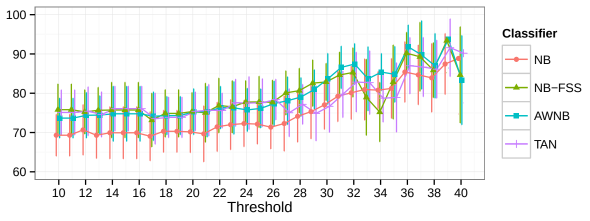
	CB	CH	CT	HT	LB	MA	NG
CB	3		2		-1		
CH							
CT	-1		-1		-3		
HT							
LB	-1				4		
MA	-1		-1				
NG							

(d)

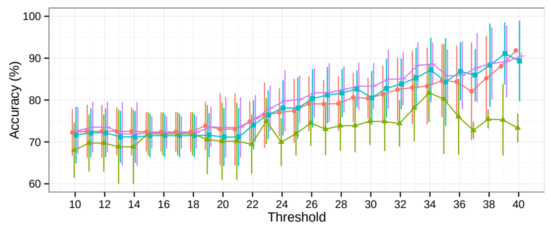
Figure 8.8: A combination of features F1–F4 does not clearly identify the interneuron type at a low reliability threshold. (a) plots the interneuron type against the combinations of features F1–F4 at $\mathcal{D}^{5,1234}_{17,21}$; (b) is the confusion matrix of an 'ideal' classifier which would assign every combination of F1–F4 values in (a) to its most common class. Thus, for example, the five LB cells corresponding to combination 3 (blue part of the third bar in (a)) would be assigned to CB (salmon-colored part of the same bar), since CB cells are predominant in combination 3. The confusion matrix of AWNB is shown in (c) whereas the difference between the 'ideal' confusion matrix and that of AWNB, (b) - (c), is shown in (d). AWNB was only slightly worse that the 'ideal' classifier: it misclassified three CB and four LB cells more than 'ideal' classifier (shown in red in (d)) but correctly classified one CT cell more (shown in blue). The latter is possible due to random permutations in cross-validation. The columns in confusion matrices denote the true classes whereas rows denote the predicted classes. Zeros were omitted. The confusion matrix of the AWNB was obtained from a single run of cross-validation. F1–F4 combinations: 1 = (intralaminar, intracolumnar, centered, none); 2 = (intralaminar, intracolumnar, displaced, none); 3 = (intralaminar, transcolumnar, centered, none); 4 = (intralaminar, transcolumnar, displaced, none); 5 = (translaminar, intracolumnar, centered, none); 6 = (translaminar, intracolumnar, displaced, ascending); 7 = (translaminar, intracolumnar, displaced, descending); 8 = (translaminar, transcolumnar, displaced, both); 9 = (translaminar, transcolumnar, centered, none); 10 = (translaminar, transcolumnar, displaced, ascending); 11 = (translaminar, transcolumnar, displaced, descending); 12 = (translaminar, transcolumnar, displaced, both); and 13 = (translaminar, transcolumnar, displaced, none).



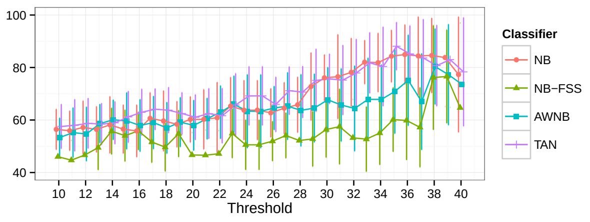
(a) F1



(b) F2



(c) F3

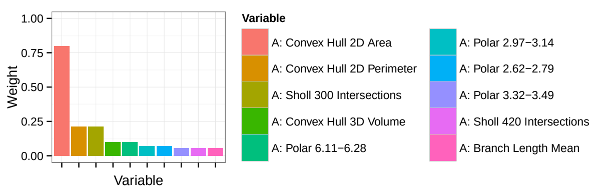


(d) F4

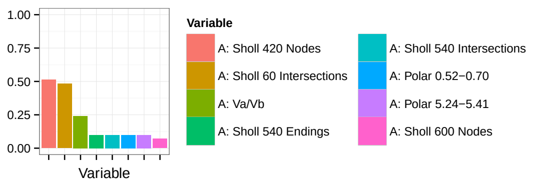
Figure 8.9: Classification accuracy for axonal features F1–F4 ((a)–(d), respectively) versus label reliability threshold. Error bars show standard deviation of accuracy from five runs of five-fold cross-validation.

reliably labeled cells (it was the best at thresholds 17 and 31–39) whereas NB was most often the least accurate, indicating that the predictor variables were redundant to some degree. AWNB used ten variables at threshold 39 (see Figure 8.10a), with most weight assigned to convex hull 2D area ($\mathbf{C}_1$), which was the only variable selected by NB-FSS —similarly accurate as AWNB— at this threshold. Indeed, NB-FSS selected a single convex hull variable at each threshold —either $\mathbf{C}_1$, $\mathbf{C}_3$, or $\mathbf{C}_4$— and it was nonetheless relatively accurate (scoring within 2% of the highest accuracy at all but three thresholds).

Regarding the prediction of F3, the highest accuracy —91.83%— was achieved by NB at label reliability threshold 40 ($\mathcal{D}^3_{40}$; see Figure 8.9c). NB was slightly more accurate, at this threshold, at identifying the more numerous `displaced` cells (0.87 sensitivity and



(a)



(b)

Figure 8.10: AWNB weights for predictor variables at (a): $\mathcal{D}^2_{39}$ and (b): $\mathcal{D}^3_{40}$. Only variables with weights greater than zero are shown. 'A:' denotes an axonal variable whereas a 'D:' denotes a dendritic variable. The numbers in Sholl variables refer to distances of spheric rings from soma, in μm, e.g., 'A: Sholl 600 Length' is the axonal arborization length within the spheric ring at 540 μm–600 μm from soma. The numbers in the polar histogram variables refer to radian intervals, e.g., 'A: Polar 4.71-4.89' refers to axonal arborization length corresponding to the angle between 4.71 and 4.89 radians. The weights were computed by learning an AWNB model from the full data set, after selecting the 100 variables with the highest mutual information with the class variable.

0.93 specificity, with `centered` being the positive class). Although F4 refers to the relative distribution of axonal and dendritic arbors, AWNB was similarly accurate as NB at this threshold by using only eight axonal variables (see Figure 8.10b). Although AWNB selected no more than 48 variables at a single threshold (with the number of variables inversely proportional to label reliability), it achieved similar accuracy as NB at most thresholds, which might suggest that only a subset of variables is useful for predicting the axonal feature F3.

The highest accuracy for predicting the axonal feature F4 —88.10%— was achieved by TAN at threshold 35 ($\mathcal{D}^4_{35}$; see Figure 8.9d). At this threshold there were no `both` cells and the classifiers thus had to distinguish between `ascending` and `descending` cells alone. TAN was equally good at identifying both classes (0.88 sensitivity and specificity). When distinguishing between `ascending` and `descending` cells, NB and TAN outperformed the classifiers that prune the predictor set —AWNB and NB-FSS— (see Figure 8.9d), which may suggest that many of the morphological variables used are useful for distinguishing among these two categories.

8.4 Conclusions

We used discrete Bayesian network classifiers to classify GABAergic interneurons according to their type and features of axonal arborization called F1, F2, F3, and F4. We trained the classifiers with the categorization of 237 interneurons according to the mentioned features and the interneuron type provided by 42 expert neuroscientists. We quantified the neurons with up to 214 morphological parameters (depending on the target categorization) and, when predicting the type, also with the high-level axonal features F1–F4. Due to little inter-expert agreement on the categorization of some cells, we separately analyzed data subsets with different expert categorization reliability thresholds.

We found that the interneurons that were categorized with more inter-expert agreement were more accurately classified by our models. The models accurately distinguished between reliable examples of the common basket, horse-tail, large basket and Martinotti interneuron types. Analyzing the Bayesian network classifiers, we identified two axonal variables —the convex hull 2D perimeter and number of branches at 180 μm from soma— and the high-level axonal features F1–F4 as especially useful for discriminating among these interneuron types. Indeed, axonal features F2 and F4 alone were able to accurately separate reliable examples of these types. Besides the interneuron type, we were also able to accurately categorize interneurons according to the axonal features F1–F4.

Since we show that the high-level axonal features can be accurately predicted, it might be possible to avoid resorting to experts for future neuron labellings according to these features and instead use the values provided by the models. A flexible alternative is to replace categorical high-level axonal features with real-valued measures: we could, for example, measure the percentage of the axon that leaves the soma's layer instead of categorically distinguishing between 'intralaminar' and 'translaminar' axons.

Chapter 9

Multi-dimensional classification with Bayesian network-modeled soft labels

9.1 Introduction

As mentioned in the introduction, 42 leading neuroscientists classified a set of interneurons according to their type and additional categorical axonal features of the Gardener's classification scheme (see Section 5.4.2). In this chapter, we predict interneuron type and four axonal features simultaneously. We asume that each neuroscientist's opinion is equally valid and that the class membership of an interneuron is probabilistic, given by the empirical distribution over the type and the axonal features in the annotators' input for that interneuron.

The joint distribution over our five class variables has $2 \times 2 \times 2 \times 4 \times 10 = 320$ entries (see Section 9.3.1) and can be handled directly. In a general setting with more variables, the joint distribution has too many entries and can only be compactly encoded with a tool such as a Bayesian network [Pearl, 1988, Koller and Friedman, 2009]. Thus, we develop a method for multi-dimensional classification with probabilistic labels which assumes that the labels are given as Bayesian networks rather than as a fully expanded probability distribution. We call such labels *label Bayesian networks* (LBNs). When applying our method to interneuron classification, we first obtain LBNs from the experts' input and then train and evaluate our model using LBNs. Thus, we replace the annotators' labels for an interneuron with a probability distribution encoded with the LBN. In a real-world scenario, this might be useful for hiding the actual labels for reasons such as confidentiality, or for representing them compactly when there are many annotators.

To the best of our knowledge, this is the first study tackling multi-dimensional classification (i.e., with multiple class variables; [van der Gaag and de Waal, 2006, Bielza et al., 2011]) with probabilistic labels. Multi-dimensional classification with a separate model for each class variable is suboptimal when the classes are not independent. We predict the LBN of an interneuron by forming a consensus Bayesian network (e.g., Matzkevich and Abramson [1992]) among the LBNs of its k nearest neighbors [Fix and Hodges, 1989] in the space of predictor variables. We form the consensus with the method by López-Cruz et al. [2014], sampling from the neighbouring networks and learn the consensus model from the sampled data. To account for distances among neighbours, we sample more instances from the labels of cells that are closer to the instance being classified.

We introduce 13 axonal morphometrics to be used as predictor variables. We defined these morphometrics seeking to capture the concepts represented by the four axonal features (other than neuronal type) and implemented software that computes them from digital reconstructions of neuronal morphology. In addition, we used five other axonal morphometric parameters, computed with NeuroExplorer [Glaser and Glaser, 1990], which were already used as predictors of neuronal type by DeFelipe et al. [2013]. In total, we used 18 axonal morphometrics as predictor variables.

The research covered in this chapter has been published in Mihaljević et al. [2015a].

The rest of this chapter is structured as follows. Section 9.2 disusses some related methods. Section 9.3 describes the data set, the morphometrics, including the ones we introduce in this study, and the extraction of LBNs from expert-provided labels; it also describes the proposed method —the distance-weighted consensus of k nearest label Bayesian networks—, the metrics for assessing our method's predictive performance, and, finally, specifies the experimental setting. We provide our results in Section 9.4, discuss them in Section 9.5, and conclude in Section 9.6.

9.2 Related methods

A suitable tool for multi-dimensional classification is a Bayesian network over both the features and the class variables [Bielza et al., 2011, Borchani et al., 2013], as it can account for dependencies among the class variables. We can easily learn Bayesian networks with probabilistic class labels, by using probabilistic counts in Equation 4.2. A Bayesian network commonly assumes a parametric conditional distribution, such as the Gaussian, for a real-valued feature X_i. Our model, on the other hand, is the non-parametric k-nearest neighbors (k-nn) classifier (see Chapter 3) and does not assume any particular distribution for the predictors. While the k-nn can readily handle probabilistic labels, we are not aware that it has been used with multi-dimensional probabilistic class labels.

Combining multiple Bayesian networks into a consensus Bayesian network is a recurring topic of interest. The standard methods for combining the parameters of a joint distribution, disregarding its underlying graphical structure (i.e., the conditional independencies), can yield undesirable results: for example, combining distributions with identical structures may render a consensus distribution with a different structure [Pennock and Wellman, 1999]. It is therefore common to first combine network structures (e.g., Pennock and Wellman [1999], Matzkevich and Abramson [1992], Peña [2011], Del Sagrado and Moral [2003]) and combine the parameters afterwards (e.g., Pennock and Wellman [1999], Etminani et al. [2013]). The cited structure-combining methods produce distributions which only contain independencies that are common to all networks, rendering them too complex (i.e., having too many parameters) to be useful in practice. An alternative is to draw samples from the different Bayesian networks and learn the consensus network from the generated data, using standard methods for learning Bayesian networks from data [Neapolitan, 2004, Koller and Friedman, 2009], as proposed by López-Cruz et al. [2014]. López-Cruz et al. [2014] weighted the influence of each Bayesian network on the consensus by sampling from it a number of instances proportional to its weight. We can readily adapt this method to weigh the effect of a neighbor's LBNs in proportion to how close it is to the instance being classified, $\mathbf{x}^{(u)}$.

9.3 Materials and methods

9.3.1 Morphology reconstructions and class labels

We had the digital reconstructions of 241 cells which DeFelipe et al. [2013] had obtained from NeuroMorpho.Org [Ascoli et al., 2007]. 42 leading neuroscientsts had classified these cells, by looking at 2D images and 3D reconstructions of their morpologies, into one of the ten classes of the Gardener's scheme and according to four categorical features of axonal morphology (see Section 5.4.2). 40 cells had an interrupted axonal process. For 36 of them we drew the small missing fragments with the Neurolucida workstation [Glaser and Glaser, 1990] and omitted the remaining four cells with large missing parts, reducing our sample to 237 cells. The cells come from different cortical areas of the rat, mouse, and monkey.

Recall from Section 5.4.2 that the values of the class variables $\mathbf{C}$ are the following:

- Axonal feature 1 (C_1): `intralaminar` and `translaminar`
- Axonal feature 2 (C_2): `intracolumnar` and `transcolumnar`
- Axonal feature 3 (C_3): `centered` and `displaced`
- Axonal feature 4 (C_4): `ascending`, `descending`, `both`, and `no`
- Axonal feature 5 (C_5): `arcade` (`AR`), `Cajal-Retzius` (`CR`), `chandelier` (`CH`), `common basket` (`CB`), `common type` (`CT`), `horse-tail` (`HT`), `large basket` (`LB`), `Martinotti` (`MA`), `neurogliaform` (`NG`), and `other` (`OT`)
- Axonal feature 6 (C_6): `characterized` and `uncharacterized`

Axonal feature C_6 is not a 'proper' morphological feature but more of a 'filter feature' which indicates whether the remaining axonal features can be reliably identified given a reconstructed interneuron. We therefore omitted C_6 from consideration in this study. Consequently, we removed from our data set 11 interneurons considered as `uncharacterized` by a majority (i.e., at least 21) of neuroscientists, considering that these interneurons cannot be reliably classified according to C_1–C_5, thereby reducing our data sample to 226 interneurons.

Thus, we have $N = 226$ interneurons, each of them quantified by a vector $\mathbf{X}$ of $m = 18$ real-valued predictor variables (i.e., $\mathbf{x} \in \mathbb{R}^{18}$). We also have $d = 5$ discrete class (i.e., target) variables $\mathbf{C} = (C_1, \ldots, C_5)$, with $\mathbf{c} \in \Omega_{C_1} \times \ldots \times \Omega_{C_d}$. Each interneuron, $\mathbf{x}^{(j)}$, is associated with a $N_j \times 5$ ($N_j \leq 42$) matrix $\mathcal{C}^{(j)}$ in which each row is an observation of $\mathbf{C}$ due to one annotator neuroscientist, i.e., $c_{i,a}^{(j)}$ is the label for class variable C_i assigned to interneuron $\mathbf{x}^{(j)}$ by expert neuroscientist a.[1]

9.3.2 Predictor variables

We used 18 axonal morphometrics as predictor variables. Five of these morphometrics were computed with NeuroExplorer and were already used to predict interneuron types by DeFelipe et al. [2013] and in Chapter 8. In addition, we introduce 13 axonal morphometrics, seeking to the capture the concepts represented by axonal features C_1–C_4. We computed these morphometrics from 3D interneuron reconstructions files in Neurolucida's ASCII (*.asc) format.

The five morphometrics we computed with NeuroExplorer are:

- X_1: 2D convex hull perimeter (in the XY projection)
- X_2: Axon length
- X_3: Axon length at less than 150 μm from the soma
- X_4: Axon length at more than 150 and less than 300 μm from the soma
- X_5: Axon length at more than 300 μm from the soma

Morphometrics X_3–X_5 are meant to measure axonal arborization with respect to the cortical column. Namely, morphometric X_3 approximates arborization length within a (300 μm wide) cortical column (at less than 150 μm from the soma); X_4 approximates the length outside but not far from the column (more than 150 and less than 300 μm from the soma); and X_5 approximates axonal length far from the column (more than 300 μm from the soma). X_1 and X_2 were used by DeFelipe et al. [2013] while in Chapter 8 we used X_3–X_5.

We introduce the following axonal morphometrics:

- X_6: Axon length within soma's layer
- X_7: Axon length outside soma's layer
- X_8: Proportion of axon length contained within soma's layer, $\frac{X_6}{X_6+X_7}$
- X_9: Axon length within soma's cortical column
- X_{10}: Axon length outside soma's cortical column
- X_{11}: Proportion of axon length within soma's cortical column, $\frac{X_9}{X_9+X_{10}}$

[1]Only neuroscientists who considered that $\mathbf{x}^{(j)}$ was `characterized` (axonal feature C_6) labeled $\mathbf{x}^{(j)}$ according to C_1–C_5. Therefore, N_j may be less than 42 and varies across interneurons.

- X_{12}: Distance, in dimensions X and Y, from axon's centroid to the soma
- X_{13}: Distance from the centroid of the above-the-soma part of the axon to the soma
- X_{14}: Distance from the centroid of the below-the-soma part of the axon to the soma
- X_{15}: Proportion of distances X_{13} and X_{14}, $\frac{X_{13}}{X_{13}+X_{14}}$
- X_{16}: Axon length above the soma
- X_{17}: Axon length below the soma
- X_{18}: Proportion of axon length above soma, $\frac{X_{16}}{X_{16}+X_{17}}$

We computed these morphometrics following assumptions made by DeFelipe et al. [2013], namely: a) cortical layer thickness is (roughly) determined by species and cortical area (see following paragraphs for details); b) the cortical column is a cylinder whose axis passes through the soma and has a diameter of 300 μm; and c) that the soma is equidistant from the top and bottom confines of the layer (i.e., a 250μm thick layer reaches 125μm above and 125μm below the soma) and the lateral 'borders' of the column. We measured the distance to soma as the distance to its centroid.

When computing morphometrics X_6 and X_7 we looked up the approximate layer thickness according to the neuron's species and cortical area. DeFelipe et al. [2013] defined an approximate layer thickness for every species/area/layer combination present in their data, and provided it as additional information for experts who classified the interneurons. DeFelipe et al. [2013] specified the approximate thickness in the form of an interval —e.g., stating that layer II/III of the mouse's visual cortex is 200–300μm thick—; we used the interval's midpoint (250μm for the previous example) as an estimate of layer thickness.

For 16 mouse interneurons, seven of them from the somatosensory and nine from the visual cortex, the cortical layer was not provided. In order to compute variables X_6 and X_7 for these cells, we assumed them to belong to a hypothetical 'average layer' for which we assumed a 197 μm thickness in the visual cortex and a 237 μm thickness in the somatosensory cortex. Although only an approximation, we consider this a more informed approximation to the 'true' values of these variables than one that could be performed by a distance-computing rule (see subsections 9.3.4 and 9.3.5) if we had left these values unspecified.

9.3.3 From multi-annotator labels to label Bayesian networks

Instead of the provided multi-annotator label matrix $\mathcal{C}$, our method requires each interneuron to be associated with an LBN. Thus, for each interneuron, we learned its LBN from its $\mathcal{C}$.

Formally, a label Bayesian network is a Bayesian network over the class variables $\mathbf{C}$. A Bayesian network [Pearl, 1988, Koller and Friedman, 2009] $\mathcal{B}$ is a pair $\mathcal{B} = (\mathcal{G},\mathbf{\Theta})$ where $\mathcal{G}$, the structure of the network, is a directed acyclic graph whose vertices correspond to the class variables $\mathbf{C}$ and encodes the conditional independencies in the joint distribution over $\mathbf{C}$, while $\mathbf{\Theta}$ are the parameters of the conditional probability distributions that the joint distribution is factorized into (see Chapter 4). Learning a Bayesian network $\mathcal{B}$ from data consists in two steps: learning network structure, $\mathcal{G}$ (i.e., the conditional independencies it encodes), and, having obtained the structure, learning its parameters (see Chapter 4). The second step is generally straightforward whereas many methods exist for performing the first step. We used a search and score structure learning method (see subsection 9.3.5).

Finally, having learned the LBNs, our final data set was $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathcal{B}^{(j)})\}_{j=1}^{N}$. Figure 9.1 depicts the LBNs for interneurons shown in Figure 5.4, along with the predicted LBNs for those interneurons.

9.3.4 Multi-dimensional classification with label Bayesian networks

We have m predictor variables $\mathbf{X}$, with $\mathbf{x} \in \mathbb{R}^m$, that describe the domain under study, and d discrete class (or target) variables $\mathbf{C}$, with $\mathbf{c} \in \Omega_{C_1} \times \ldots \times \Omega_{C_d}$, that we wish to predict on the basis of observations of $\mathbf{X}$. We observe a data set, $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathcal{B}^{(j)})\}_{j=1}^{N}$, where $\mathcal{B}$ is a label Bayesian network encoding a joint probability distribution over the multi-dimensional class variable $\mathbf{C}$. We predict the LBN of an unseen instance $\mathbf{x}^{(u)}$ by forming a consensus Bayesian network among the LBNs of its k nearest neighbors ($1 \leq k < N$) in the space of predictor variables. To form the consensus we generate a data set $\mathcal{D}_u$ by sampling from k Bayesian networks $\{\mathcal{B}^{(j)}\}_{j=1}^{k}$ associated to k instances at distances $d_1, \ldots, d_k$ from the unseen instance $\mathbf{x}^{(u)}$ and learn the consensus Bayesian network, $\mathcal{B}^{*(u)}$, from $\mathcal{D}_u$. We want the number of samples in $\mathcal{D}_u$ that are drawn from $\mathcal{B}^{(j)}$ to be proportional to the how close $\mathbf{x}^{(j)}$ is to $\mathbf{x}^{(u)}$, relative to the remaining $k-1$ neighbours. With M being the desired size of $\mathcal{D}_u$, we sample $w_j \times M$ instances from $\mathcal{B}^{(j)}$, where

$$w_j = \frac{(\sum_{i=1}^{k} d_i) - d_j}{(k-1)(\sum_{i=1}^{k} d_i)},$$

and $\sum_{j=1}^{k} w_j = 1$ and $w \geq 0$ hold. Fig. 9.2 summarizes our approach.

9.3.5 Experimental setting

We identified the nearest interneurons by measuring Euclidean distance. Thus, for a pair of interneurons $\mathbf{x}^j$ and $\mathbf{x}^o$, the distance d_{jo} is given by

$$d_{jo} = (\sum_{i=1}^{m} (x_i^{(j)} - x_i^{(o)})^2)^{\frac{1}{2}}.$$

Prior to computing distances, we standardized all predictor variables $X_1, \ldots, X_m$ (i.e., for each X_i, we subtracted its mean and divided by standard deviation).

We sought to draw enough samples from each distribution so to represent it correctly. We therefore set M, the total number of samples drawn from the k nearest neighbors' distributions, as $k * 500 * c$, where c was the maximal number of free parameters among the k networks whose consensus is being sought. The number of free parameters of a Bayesian network is

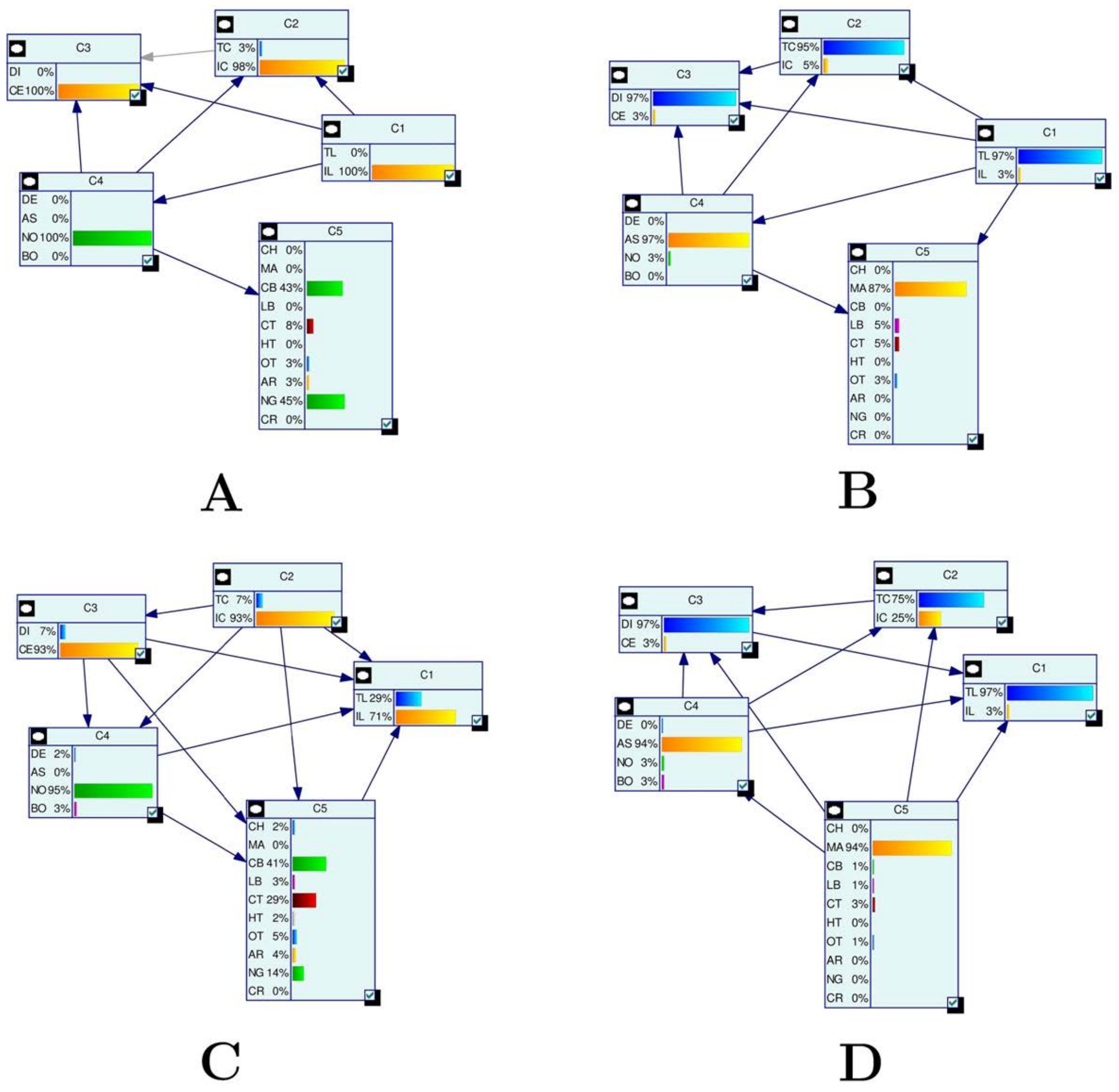


Figure 1. Examples of true ((A) and (B)) and predicted ((C) and (D)) label Bayesian networks (LBNs) for neurons shown in Figure 5.4. The leftmost networks ((A) and (C)) correspond to interneuron (A) in Figure 5.4 whereas the right-hand ones ((B) and (D)) correspond to neuron (B) in Figure 5.4. The Bayesian networks are depicted with their nodes (shown as rectangles), arcs, and each node's marginal probability distribution. The predicted distributions are similar to the true ones for many nodes —e.g., 93% vs. 98% for `IC` (node C_2) for interneuron (A). Some marginal probabilities do differ, such as that of the `NG` type for neuron (A) —14% predicted vs. 45% true; a lot of its probability mass was assigned to the more numerous `CT` type.

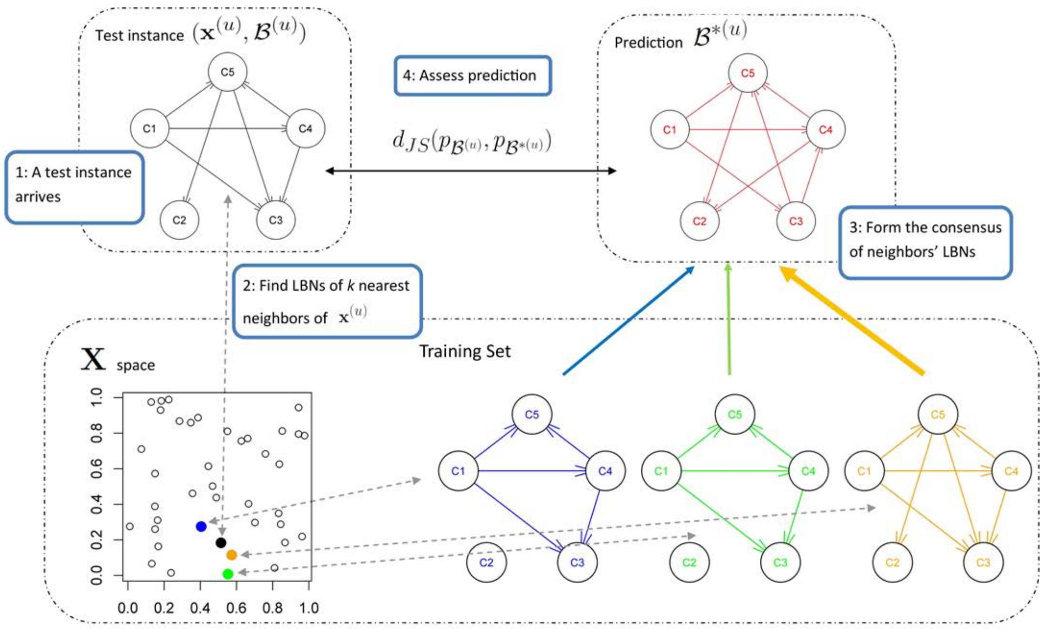

Test instance $(\mathbf{x}^{(u)}, \mathcal{B}^{(u)})$
Prediction $\mathcal{B}^{*(u)}$
4: Assess prediction
$d_{JS}(p_{\mathcal{B}^{(u)}}, p_{\mathcal{B}^{*(u)}})$
1: A test instance arrives
2: Find LBNs of k nearest neighbors of $\mathbf{x}^{(u)}$
3: Form the consensus of neighbors' LBNs
Training Set
$\mathbf{X}$ space
C1
C2
C3
C4
C5

Figure 2. A schematic representation of multi-dimensional classification with label Bayesian networks (LBNs). The figure depicts the assessment of our method's predictive performance. First (step 1; upper left), an instance $\mathbf{x}^{(u)}$ with LBN $\mathcal{B}^{(u)}$ is retrieved from the test set. Then (step 2; lower part), we identify k ($k = 3$ in this example) nearest neighbors of $\mathbf{x}^{(u)}$ and record their distances to $\mathbf{x}^{(u)}$; the blue, green, and orange Bayesian networks (lower right) depict the LBNs of the three nearest neighbors of $\mathbf{x}^{(u)}$. Then (step 3; upper right), we obtain the predicted Bayesian network labels, $\mathcal{B}^{*(u)}$, by forming a consensus Bayesian network from the LBNs of the three nearest neighbors. Here, arrow thickness denotes the weight of a neighbor's LBN in the consensus: the orange arrow is thicker than the blue and green arrows (orange is the closest neighbor of $\mathbf{x}^{(u)}$, see lower left). Finally (step four; upper middle), we compare true and predicted probability distributions, $p_{\mathcal{B}^{(u)}}$ and $p_{\mathcal{B}^{*(u)}}$, with Jensen-Shannon divergence.

the number of parameters that suffice to fully specify the network's probability distribution (recall that a network consists of a structure, $\mathcal{G}$, and parameters $\mathbf{\Theta}$; see Chapter 4).

Once we had generated the data set of sample points, we applied a Bayesian network learning algorithm to obtain the consensus probability distribution.

9.3.5.1 Learning Bayesian networks from data

There were two instances in which we learned Bayesian networks from data: when learning LBNs from expert-provided class label matrices (see subsection 9.3.3) and when learning the consensus network from sampled data points (subsection 9.3.4). We considered three options for the learning procedure and chose the one that we considered most adequate for learning LBNs, according to the criterion described in subsection 9.3.3. We then applied this chosen procedure in both instances of network learning.

We followed the search and score approach for learning Bayesian network structure (see Chapter 4). We searched the structure space with the tabu metaheuristic [Glover and Laguna, 2013] and considered three networks scores: Bayesian Information Criterion (BIC; [Schwarz, 1978]), K2 [Cooper and Herskovits, 1992] and Bayesian Dirichlet equivalence (BDe; Heckerman et al. [1995]). We fit parameters by maximum likelihood estimation.

9.3.5.2 Software and Assessment

We implemented the computation of the 13 here introduced axonal morphometrics from scratch. We performed Bayesian network learning and sampling with the `bnlearn` [Scutari, 2010, Nagarajan et al., 2013] package for the `R` statistical software environment [R Core Team, 2015].

In traditional uni-dimensional classification, it is common to perform stratified cross-validation, that is, to have similar class proportions in train and test sets. However, such stratification is problematic in the multi-dimensional setting, due to the high number of combinations of class variables. Therefore, instead of stratified cross-validation, we evaluated our model with 20 repetitions of plain (unstratified) 10-fold cross-validation.

9.3.6 Assessing results

We were primarily interested in predicting LBNs. We assessed this prediction with Jensen-Shannon divergence, a metric which we describe below. However, for comparison with related work on interneuron classification, we also assessed how well our method predicted crisp (i.e., non-probabilistic) labels. Such an evaluation is negatively biased against our method since we take label ambiguity into account to learn the model while it is evaluated as though a true crisp label existed (i.e., as if there was no ambiguity). Below we describe how we obtained crisp labels and present accuracy metrics for multi-dimensional classification.

9.3.6.1 Comparing probability distributions

We measured the dissimilarity between two probability distributions, say $p_{\mathcal{B}^{(u)}}$ and $p_{\mathcal{B}^{*(u)}}$, with Jensen-Shannon divergence,

$$d_{JS}(p_{\mathcal{B}^{(u)}}, p_{\mathcal{B}^{*(u)}}) = \frac{1}{2}(d_{KL}(p_{\mathcal{B}^{(u)}}, p_r) + d_{KL}(p_{\mathcal{B}^{*(u)}}, p_r)),$$

where $p_r = \frac{1}{2}(p_{\mathcal{B}^{(u)}} + p_{\mathcal{B}^{*(u)}})$ and $d_{KL}(p_{\mathcal{B}^{(u)}}, p_{\mathcal{B}^{*(u)}})$ is the Kullback-Leibler divergence [Kullback and Leibler, 1951] between $p_{\mathcal{B}^{(u)}}$ and $p_{\mathcal{B}^{*(u)}}$,

$$d_{KL}(p_{\mathcal{B}^{(u)}}, p_{\mathcal{B}^{*(u)}}) = \sum_{\mathbf{c} \in \Omega_{\mathbf{c}}} p_{\mathcal{B}^{(u)}}(\mathbf{c}) \log(\frac{p_{\mathcal{B}^{(u)}}(\mathbf{c})}{p_{\mathcal{B}^{*(u)}}(\mathbf{c})}).$$

Unlike Kullback-Leibler divergence, Jensen-Shannon divergence is symmetric, it does not require absolute continuity (i.e., that $p_{\mathcal{B}^{*(u)}}(\mathbf{c}) = 0 \implies p_{\mathcal{B}^{(u)}}(\mathbf{c}) = 0$), its square root is a metric, and it is bounded: $0 \leq d_{JS} \leq 1$.

9.3.6.2 Obtaining crisp labels

In order to assess the prediction of crisp labels, we needed to obtain a 'true' crisp class label vector for each interneuron $\mathbf{x}^{(j)}$. We assumed that such 'true' labels were given by the choice of the majority of the experts. There were two alternative majority choices: a) the most commonly selected class label vector, i.e., the most common row in a class labels matrix $\mathcal{C}$; and b) the concatenation of per-class majority labels, i.e., the vector formed by the most common choice for C_1, the most common choice for C_2, and so on, until C_5. We refer to the former as the *joint truth* and to the latter as *marginal truth*; the latter was used by [DeFelipe et al., 2013] and in Chapter 8 when predicting the axonal features C_1–C_5 independently. We compared our predicted crisp labels to both 'truths'.

We also needed to extract crisp predictions from a predicted label Bayesian networks. The two straightforward methods are analogous to the above-described ones: a) choosing the *most probable explanation* (MPE), i.e., the most likely joint assignment to $\mathbf{C}$ according to LBN $\mathcal{B}^*$); and b) concatenating the marginally most likely assignments to each of the class variables. For simplicity, we only used the MPE as the predicted crisp class labels vector.

9.3.6.3 Multi-dimensional classification accuracy metrics

We assessed crisp labels prediction with accuracy metrics for multi-dimensional classification [Bielza et al., 2011]:

- The *mean accuracy* over d ($d = 5$ in our case) class variables:

$$\overline{Acc} = \frac{1}{d} \sum_{l=1}^{d} \frac{1}{N} \sum_{u=1}^{N} \delta(c_l^{*(u)}, c_l^{(u)}),$$

where $c_l^{*(u)}$ is the predicted value of C_l for u-th instance, $c_l^{(u)}$ is the corresponding true value, and $\delta(a,b) = 1$ when $a = b$ and 0 otherwise.

- The *global accuracy* over d class variables:

$$Acc = \frac{1}{N}\sum_{u=1}^{N} \delta(\mathbf{c}^{*(u)}, \mathbf{c}^{(u)}).$$

Note that global accuracy is demanding as it only rewards full matches between the predicted vector and the true one. We also measured uni-dimensional *marginal accuracy* per each class variable,

$$Acc_l = \frac{1}{N}\sum_{u=1}^{N} \delta(c_l^{*(u)}, c_l^{(u)}).$$

When computing global and mean accuracy, we used the 'joint truth' crisp labels. When computing per-class-variable marginal accuracy, we used the 'marginal truth' crisp labels vector.

9.4 Results

9.4.1 From multi-annotator labels to label Bayesian networks

We first studied whether any network score was particularly adequate for transforming multi-expert labels into label Bayesian networks. Different scores yielded networks of different degrees of complexity but were all good at approximating of the empirical probability distribution over the expert-provided labels, p_ϵ (see Table 9.1). We used the score that yielded the best approximation, BDe, in the remainder of this study. Namely, we used it to a) transform multi-expert labels into label Bayesian networks; and b) learn a consensus networks from the generated samples.

Table 1. Transforming multi-expert labels into label Bayesian networks using different network scores. Upper row: average Jensen-Shannon (JS) divergence between the empirical probability distribution over the labels, p_ϵ, and the one encoded by the learned Bayesian network labels, $p_{\mathcal{B}}$; lower row: average number of free parameters per learned network. Averaged across entire data set.

	BIC	K2	BDe
JS divergence	00.10 ± 0.05	00.07 ± 0.04	00.06 ± 00.04
Free parameters	18.22 ± 1.83	31.08 ± 20.58	60.34 ± 31.14

9.4.2 Predicting label Bayesian networks

We considered four different values of k (the number of nearest neighbors) —namely, 3, 5, 7, and 9—, and obtained best results with $k \in \{5, 7\}$. As Table 9.2 shows, we predicted the label Bayesian networks relatively accurately, with a Jensen-Shannon divergence of 0.29 for $k \in \{5, 7\}$.

Table 2. Predicting label Bayesian networks and crisp labels. The leftmost column shows Jensen-Shannon (JS) divergence between predicted label Bayesian networks, $p_{\mathcal{B}^*}$, and label Bayesian networks $p_{\mathcal{B}}$ learned from $\mathcal{C}$. Rightmost columns show global and mean accuracy for predicting joint truth crisp labels vector, i.e., the class label vector most often selected by the experts. Obtained with 20 runs of 10-fold cross-validation.

	JS	Global acc. (%)	Mean acc. (%)
$k = 3$	0.30 ± 0.00	41.29 ± 1.57	79.10 ± 0.74
$k = 5$	0.29 ± 0.00	43.84 ± 1.48	79.52 ± 0.79
$k = 7$	0.29 ± 0.00	43.99 ± 1.26	79.88 ± 0.34
$k = 9$	0.30 ± 0.00	39.46 ± 1.67	78.58 ± 0.52

Figure 9.1 depicts the true and predicted LBNs for two interneurons, one having barely ambiguous axonal features and another having an ambiguous type; as the figure suggests, the LBN of the former interneuron was accurately predicted, while in that of the latter, the type (C_5) marginal probability was predicted only moderately well.

9.4.3 Predicting crisp labels

We predicted the joint truth (the class label vectors selected by a majority of experts; see Section 9.3.6.2) relatively accurately —with a mean accuracy of 80% and global accuracy of 44% for $k \in \{5, 7\}$ (see Table 9.2). The latter result means that 44% of the MPEs of the predicted label Bayesian networks ($\mathcal{B}^*$) were equivalent to the joint truth vectors.

We also assessed the marginal accuracy for each axonal feature C_1–C_5. Here we compared the $\mathcal{B}^*$ MPE with the marginal truth, class variable by class variable. We predicted features C_1–C_4 with over 80% accuracy —up to 88% in case of C_1— and feature C_5 with 64% accuracy with $k = 7$ (see Table 9.3). Albeit it may seem low, the latter result is better than chance. Namely, DeFelipe et al. [2013] showed that even 40.25% accuracy for C_5 —obtained by a classifier they used— was better than chance. It should also be recalled that the ten neuronal types were often hard to distinguish for expert neuroscientists [DeFelipe et al., 2013]. Regarding the prediction of the individual types, accurately predicted ones included the `MA` and `HT` types, which were easy to identify for the experts, and the numerous but less clear types such as `CB` and `LB`. The least clear out of the numerous types, `CT`, was predicted with relatively low accuracy (see Table 9.4).

Table 3. Accuracy (in %) for each of the five axonal features C_1–C_5. Here we compared the marginal true labels to the most probable explanation of the predicted soft labels. Obtained with 20 runs of 10-fold cross-validation.

	C_1	C_2	C_3	C_4	C_5
$k = 3$	86.15 ± 1.12	83.17 ± 0.98	86.50 ± 0.88	83.11 ± 0.90	62.69 ± 1.24
$k = 5$	86.49 ± 0.98	83.25 ± 0.79	86.05 ± 0.79	84.18 ± 0.65	63.78 ± 1.11
$k = 7$	88.07 ± 1.01	83.12 ± 0.72	85.29 ± 0.55	84.06 ± 0.74	64.33 ± 1.52
$k = 9$	87.16 ± 1.03	83.06 ± 0.78	85.39 ± 0.78	83.88 ± 0.71	63.79 ± 1.59

Table 4. Confusion matrix for predicting C_5 with $k = 7$. Here we compared the marginal true label for C_5 (rows) to the C_5 value of most probable explanation of the predicted label Bayesian network (columns). The rightmost column shows per-type sensitivity. Types `AR`, and `CR` and `OT` are omitted since no cell's crisp label was of one of these types. Obtained from a single run of 10-fold cross-validation.

	CB	CH	CT	HT	LB	MA	NG	Per-type sensitivity
CB	41	0	10	0	6	3	0	0.68
CH	2	0	1	0	0	0	0	0.00
CT	10	0	25	4	8	11	0	0.43
HT	0	0	3	10	1	0	0	0.71
LB	9	0	3	0	25	3	1	0.61
MA	0	0	2	0	4	36	0	0.86
NG	8	0	0	0	0	0	0	0.00

9.5 Discussion

DeFelipe et al. [2013] used majority crisp labels, estimated for each axonal feature independently, to train and evaluate their models. DeFelipe et al. [2013] predicted axonal features C_1–C_5 with an independent model for each of them. There were non-methodological differences among that study and the present work and therefore results comparison ought to be taken with caution. DeFelipe et al. [2013] used 15 cells more than we did (see Section 9.3.1), had several of variables' values corrupted by imperfections in the reconstructions of 36 cells —which we corrected—, and used only three values for C_4 —`ascending`, `descending`, and `both`. Furthermore, they used different morphometric predictors (over 2000 of them), and applied a possibly more optimistic accuracy estimation technique —leave-one-out estimation.

Differences aside, in Table 9.5 we compare the accuracies from the present study with those from DeFelipe et al. [2013]. We outperformed DeFelipe et al. [2013] in predictive accuracy for every axonal feature, even though we used a single model to predict all features simultaneously. We especially outperformed their approach in predicting C_3 and, even more, in predicting C_4. The latter was likely affected by the use of the additional category `no` (see Section 9.3.1).

Table 5. Our best predictive accuracy (in %) versus best accuracy from DeFelipe et al. [2013], for each of the axonal features C_1–C_5. Our results were obtained with 20 runs of 10-fold cross-validation; those of DeFelipe et al. [2013] were obtained with leave-one-out cross-validation.

	C_1	C_2	C_3	C_4	C_5
Present study	88.07	83.25	86.50	84.18	64.33
DeFelipe et al. [2013]	85.48	81.33	73.86	60.17	62.24

Despite the non-methodological differences with the study by DeFelipe et al. [2013], the better accuracies that we achieved might suggest some or all of the following: a) the introduced morphometrics are useful for predicting interneuron type and axonal features; b) we adequately assigned the value `no` for cells to which the other values of C_4 did not apply; c) variables $C_1 - C_5$ are correlated; and d) our method is adequate for classifying interneurons.

Note that in our motivating setting with multiple annotators, we may have learned the predicted LBNs directly from the labels of all annotators of all neighbouring instances, appropriately weighted. Our method, based on having an LBN for every instance, is more general as it applies to all settings with probabilistic labels, regardless of whether or not they come from having multiple annotators.

9.6 Conclusion

We learned a model that classifes an interneuron, on the basis of a set of axonal morphometrics, into the interneuron type and four other categorical axonal features of the Gardener's classification scheme. We used multi-dimensional probabilistic class labels encoded as Bayesian networks over interneuron type and axonal features. We obtained these labels from the classification choices provided by 42 leading neuroscientists. We then proposed an instance-based classifier which can learn from such multi-dimensional probabilistic input, predicting the output by forming a consensus among a set of Bayesian network labels.

We accurately predicted the probabilistic labels over the interneuron type and the four remaining axonal features. We outperformed DeFelipe et al. [2013] in predicting the non-probabilistic labels for axonal feature F4 and, slightly, for the interneuron type. Unlike DeFelipe et al. [2013], we predicted the type and four axonal features with a single model.

We introduced 13 axonal morphometrics which we defined as quantitative counterparts of the four categorical axonal features. Our results suggest that these morphometrics are useful for predicting the type and the four axonal features. Thus, they might be considered as objective replacements, or surrogates, of the subjective categorical axonal features.

Chapter 10

Semi-supervised projected model-based clustering

10.1 Introduction

In this chapter, we consider the semi-supervised clustering of Gardener's interneurons. We first used majority voting (see Section 3.6) to obtain a crisp class label, from the labels by 42 neuroscientists, for each neuron. We then unlabeled the cells of a) one type at a time; b) two types at a time; and c) half the instances of each type at a time, and ran the semi-supervised projected model-based clustering algorithm (SeSProc) by Guerra et al. [2013b]. This algorithm starts with a cluster for each type, formed by the labelled cells that belong to it, and clusters the unlabelled cells. It then considers forming an additional cluster and recomputes the assignment of the unlabelled cells, repeating this as long as the model's score is improving. This way we sought to find subtypes of the existing types and assess the separation among the types.

SeSProc models the data with the mixture model with localized feature selection by Li et al. [2009]. It proposes an algorithm for learning this mixture in a semi-supervised setting, with clusters memberships of the labelled cells fixed a priori, and an search procedure to determine the number of clusters. We modified the underlying mixture by modeling a variable X_i irrelevant for cluster m with a marginal distribution independent of any cluster. Despite this change, we keep the term SeSProc for our method.

We quantified the neurons with nine simple axonal and dendritic morphological variables, such as the axonal length close to the soma, and labeled them according to the choices of the expert neuroscientists. Because some of our class labels were backed few neuroscientists, we considered three subsets of our neuron sample, each given by a different 'label reliability threshold' th, i.e., such that the label of each neuron in the subset was agreed upon by at least th experts.

The research covered in this chapter has been published in Mihaljević et al. [2015b] and is an extension of Guerra et al. [2013a]. We extended Guerra et al. [2013a] by adapting the

SeSProC algorithm, refining some of the predictor variables, and considering two additional experimental settings.

The remainder of this chapter is organized as follows: Section 10.2 describes the materials and methods we used; Section 10.3 reports and discusses the obtained results; while Section 10.4 provides conclusions.

10.2 Materials and methods

10.2.1 Morphology reconstructions and class labels

We had the digital reconstructions of 241 cells which DeFelipe et al. [2013] had obtained from NeuroMorpho.Org [Ascoli et al., 2007]. 42 leading neuroscientsts had classified these cells, by looking at 2D images and 3D reconstructions of their morpologies, into one of the ten classes of the Gardener's scheme and according to four categorical features of axonal morphology (see Section 5.4.2). 40 cells had an interrupted axonal process. For 36 of them we drew the small missing fragments with the Neurolucida workstation [Glaser and Glaser, 1990] and omitted the remaining four cells with large missing parts, reducing our sample to 237 cells. The cells come from different cortical areas of the rat, mouse, and monkey.

We formed subsets of our sample by imposing minima on the number of experts that agreed on the label of an included cell (i.e., a 'label reliability threshold'), considering that a higher threshold yields more confidence in the cells' labels. We used thresholds 18, 22 (half plus one out of the 42 experts), and 26 to build three databases: *th18*, *th22*, and *th26*, respectively. These data sets contained interneurons of four different types (classes): common basket (CB), horse-tail (HT), large basket (LB), and Martinotti (MA). Table 10.1 shows the distribution of different types at the three label reliability thresholds.

Table 10.1: Distribution of interneuron types with respect to label reliability threshold. Lowermost row shows total number of cells per dataset.

	th18	*th22*	*th26*
CB	49	24	9
HT	9	5	4
LB	27	19	12
MA	33	25	22
Total	118	73	47

We characterized each neuron using nine features of axonal and dendritic morphology that are related to how, in our opinion, an expert classifies an interneuron upon visual examination. Namely, we consider that an expert classifies an interneuron by estimating the distribution and the orientation of axonal and dendritic arborizations. We therefore measured the axonal and dendritic length according to the Sholl (5 features) and polar histogram (4 features) analyses from NeuroExplorer, the data analysis companion to Neurolucida Glaser and Glaser [1990]. Sholl analysis computes axonal and dendritic length at different distances from the soma whereas the polar histogram [McMullen et al., 1984] describes the overall direction of dendritic growth; we only distinguished between two halves of the histogram, namely, the bifurcation angles falling in the $[0, \pi)$ interval and those falling in the $[\pi, 2\pi)$ interval. See Table 10.2 and Fig. 10.1 and for further details on predictor variables. We standardized all variables (transformed them so to have zero mean and unit standard deviation) prior to classification.

Table 10.2: Predictor variables used in the present study. Predictors X_1–X_5 correspond to the axon whereas X_6–X_9 correspond to the dendrites.

Variable	**Arbor type**	**Description**
X_1	Axon	Polar histogram length (μm) for the $[0, \pi)$ radians interval
X_2		Polar histogram length (μm) for the $[\pi, 2\pi)$ radians interval
X_3		Sholl analysis length (μm) at less than 150 μm from the soma
X_4		Sholl analysis length (μm) at more than 150 and less than 300 μm from soma
X_5		Sholl analysis length (μm) at more than 300 μm from the soma
X_6	Dendrites	Polar histogram length (μm) for the $[0, \pi)$ radians interval
X_7		Polar histogram length (μm) for the $[\pi, 2\pi)$ radians interval
X_8		Sholl analysis length (μm) at less than 90 μm from the soma
X_9		Sholl analysis length (μm) at more than 90 μm from the soma

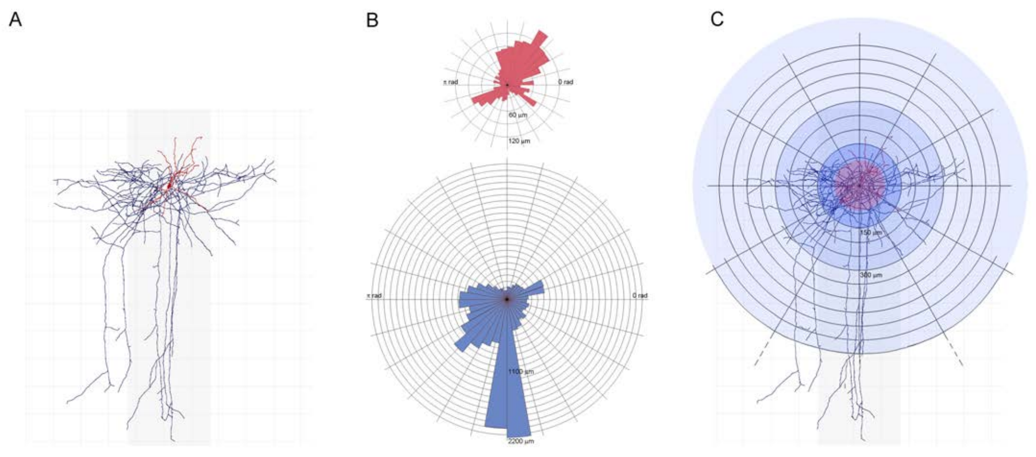


Figure 10.1: A schematic representation of the nine morphological features of axonal and dendritic morphology that we used as predictor variables. **A**, An example of a 3D reconstructed interneuron classified by expert neuroscientists in DeFelipe et al. [2013], showing its axonal (displayed in blue) and dendritic (red) arborizations. The grey vertical shadow indicates the extent of the cortical column (assumed to be 300 μm wide) whereas the dimensions of the squares are 100×100 μm. **B**, Schematic representation of the overall direction of dendritic growth (polar histogram) for the dendrites (above; red) and the axon (below; blue). Features X_1 and X_2 encode axonal growth length in the angle intervals $[0, \pi)$ and $[\pi, 2\pi)$, respectively, whereas features X_6 and X_7 capture the dendritic growth length in the same angle intervals. **C**, Schematic representation of the features encoding axonal and dendritic arborization lengths at different distances from the soma (Sholl analysis): features X_3, X_4, and X_5 encode, respectively, axonal lengths at 0–150 μm (marked with a dark blue circle), 150–300 μm (blue circle), and over 300 μm (outside of the blue circle) from the soma. Features X_8 and X_9 measure dendritic arborization at 0–90 μm (pink circle) and over 90 μm (outside of the pink circle) from the soma, respectively.

10.2.2 Semi-supervised projected model-based clustering

10.2.2.1 A mixture model with local feature selection

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be observed data, with $\mathbf{x}_i \in \mathbb{R}^F, \forall i \in \{1, \ldots, N\}$, where F denotes the number of features. Assuming that the data are generated from a finite mixture of K components, with the variables independent given the component, the density function for an instance $\mathbf{x}_i$ is,

$$f(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} f(x_{ij} \mid \theta_{mj}),$$

with $\pi_m \in [0,1]$ and $\sum_{m=1}^{K} \pi_m = 1$. Law et al. [2004] integrate probabilistic feature selection into the model,

$$f(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_j f(x_{ij} \mid \theta_{mj}) + (1 - \rho_j) f(x_{ij} \mid \lambda_j) \Big), \tag{10.1}$$

where ρ_j is feature salience, defined as the complement of the probability that X_j has a common density function $f(X_j \mid \lambda_j)$ rather than a conditional density $(x_{ij} \mid \theta_{mj})$ for each component m. SeSProC uses the model from Li et al. [2009] with localized feature salience, ρ_{mj}, for a variable X_j and component m,

$$f(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj} f(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj}) f(x_{ij} \mid \lambda_{mj}) \Big). \tag{10.2}$$

Unlike in Equation 10.1, a locally non-salient X_j has a component-conditional density $f(\cdot \mid \lambda_{mj})$ rather than a common one. We adapt Equation 10.2 so that a locally non-salient variable X_j has a common density $f(\cdot \mid \lambda_j)$ instead,

$$f(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj} f(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj}) f(x_{ij} \mid \lambda_j) \Big).$$

This reduces the number of parameters. We consider that it provides an intuitive meaning to the notion of local salience.

We encode the clustering with a hidden variable $\mathbf{Z}$, with $z_{im} = 1$ if $\mathbf{x}_i$ is assigned to component m and $z_{im} = 0$ otherwise. $\mathbf{Z}$ is partially observed because we fix its values for the labeled cells. Thus, $z_{im} = \mathbb{I}(c_i = m)$ for all $i \leq L$. We encode feature relevance with a hidden variable, $\mathbf{V}$, with $v_{mj} = 1$ if feature X_j is relevant for component m and $v_{mj} = 0$ otherwise. Feature salience if then $\rho_{mj} = P(v_{mj} = 1)$. Thus, learning the model involves: 1) learning the parameters; 2) filling-in the missing $\mathbf{Z}$, for the unlabeled cells, and filling-in $\mathbf{V}$ for all feature-component pairs.

10.2.2.2 Learning parameters

Except for estimating λ_j, learning is identical to that of the original SeSProC. We discuss the basics and refer the reader to Guerra et al. [2013b] for details.

Because $\mathcal{V}$ is hidden and $\mathcal{Z}$ only partially observed, we cannot maximize log-likelihood analytically. We approximate it with the expectation-maximization (EM) algorithm Dempster et al. [1977]. The EM algorithm produces a sequence of parameter estimates by alternating the expectation and the maximization steps. The expectation step fills the missing values with their expected values given the current parameters. In our case, this corresponds to missing rows in $\mathbf{Z}$ and all of $\mathbf{V}$. The maximization step uses the completed values to compute maximum likelihood parameter estimates, as if all data were observed. Repeated application of this procedure converges to a local optimum.

Assuming that both $f(\cdot \mid \theta_{mj})$ and $f(\cdot \mid \lambda_j)$ are Gaussian densities, and given parameters Θ from a previous iteration of the EM, and values filled-in in the expectation step, the maximization step updates the parameters as follows

$$\pi_m = \frac{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})}{N}, \tag{10.3}$$

$$\rho_{mj} = \frac{\sum_{i=1}^{N} \gamma(z_{im})\gamma(v_{mj})}{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})}, \tag{10.4}$$

$$\mu_{\theta_{mj}} = \frac{\sum_{i=1}^{N} \gamma(z_{im})\gamma(v_{mj})x_{ij}}{\sum_{i=1}^{N} \gamma(z_{im})\gamma(v_{mj})}, \tag{10.5}$$

$$\sigma^2_{\theta_{mj}} = \frac{\sum_{i=1}^{N} \gamma(z_{im})\gamma(v_{mj})(x_{ij} - \mu_{\theta_{mj}})^2}{\sum_{i=1}^{N} \gamma(z_{im})\gamma(v_{mj})}, \tag{10.6}$$

$$\mu_{\lambda_j} = \frac{\sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im})(1 - \gamma(v_{mj}))x_{ij}}{\sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im})(1 - \gamma(v_{mj}))}, \tag{10.7}$$

$$\sigma^2_{\lambda_j} = \frac{\sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im})(1 - \gamma(v_{mj}))(x_{ij} - \mu_{\lambda_j})^2}{\sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im})(1 - \gamma(v_{mj}))}, \tag{10.8}$$

$m = 1, \ldots, K; j = 1, \ldots, F$, where $\gamma(x) = P(x = 1)$. Note that the only difference with respect to the original SeSProC are Equation 10.7 and Equation 10.8.

10.2.2.3 Estimating the number of components

Our initial mixture has one component for each class in the data set. We learn the parameters of this mixture with the EM and compute its score (see below). Then, we add a component, and repeat the process. We repeat this until a mixture $\mathcal{M}^K$ with K components is better than a mixture $\mathcal{M}^{K+1}$, returning $\mathcal{M}^K$ as the final model. We score a model with the Akaike information criterion (AIC) score [Akaike, 1974],

$$AIC = -2 \log \mathcal{L} + 2R,$$

where R, the number of parameters in a model, is a function of the number of components K and the number of features F:

$$R = 2KF + 2F + (K-1) + KF,$$

The first addend corresponds to θ's, second to λ's, third to π's and fourth to ρ's.

When starting the EM procedure for a model $\mathcal{M}^K$ with K components, we use the class labels to initialize the θ parameters for the classes, i.e., we estimate $\boldsymbol{\theta}_m = (\theta_{m1}, \ldots, \theta_{mF})$, for $m \leq C$, from instances belonging to class m. If $K > C + 1$ (i.e., if there are already newly found components in the mixture), then we use the $\boldsymbol{\theta}_m, C < m < K$, from $\mathcal{M}^{K-1}$ as their initial values in $\mathcal{M}^K$, since there are no labels that could guide the estimation of their initial values. The initial $\boldsymbol{\theta}_K$ for the new component K are estimated from a number of unlabeled data points; the following paragraph describes how these data points are selected.

We have modified SeSProC to use the following heuristic for initializing new components. Starting from the previous mixture, $\mathcal{M}^{K-1}$, we consider the neighborhood of each unlabeled point as (a part of) a potential new cluster. Thus, we take the *ct* nearest unlabeled neighbors (according to the Euclidean distance in full dimensionality), where *ct* is a parameter of the algorithm, to a point $\mathbf{x}_i$ and assign them to a new cluster, by setting $z_{jK} = 1$ and $z_{jm} = 0, m \neq K$ for all $\mathbf{x}_j$ in the neighborhood of $\mathbf{x}_i$; we then update all parameters (this includes the $\boldsymbol{\theta}_K$) by maximum likelihood (i.e., with an M-step) and compute the likelihood of the thereby obtained model. The neighborhood that yields the most likely model is then used to initialize $\boldsymbol{\theta}_K$ in the new model $\mathcal{M}^K$. All assignments to z here described are then undone after $\boldsymbol{\theta}_K$ is initialized (i.e., these assignments were only temporary).

10.2.3 Experimental settting

We applied SeSProC in three experimental settings. First, we unlabeled all the cells of a single class and ran the algorithm once for each class. Here, there was initially a cluster for each of the other (labeled) types and the desired result was to assign unlabeled instances to a (one of) newly formed cluster(s), allowing us to explore the potential subtypes of each class separately. Second, we simultaneously unlabeled all cells of each pair of classes, yielding six clustering scenarios; this allowed to assess whether cells of different classes would be clustered together and whether, and to what extent, would the subtypes identified in the previous setting reappear. Finally, we simultaneously unlabeled a half the cells of each of the four classes. This allowed unlabeled cells to be placed in their 'true' cluster, other classes (i.e., be misclassified), or assigned to a new cluster, providing insight into the homogeneity of each interneuron class. We selected unlabeled cells by random sampling at each label reliability threshold (thus a cell might have been unlabeled at *th26* but not at *th18*, for example), and repeated the sampling ten times. We used these three unlabeling settings for each label reliability threshold, i.e., for *th18*, *th22*, and *th26*.

For the first two settings, we defined per-class discrimination accuracy as $\mathrm{acc}_t = \frac{c_t}{u_t}$, where u_t is the number of cells of the unlabeled class t and c_t the cardinality of the subset of u_t

assigned to a (one of) newly formed cluster(s) (and therefore not assigned to one of the other classes). In the second setting, we averaged acc_t across the three 'scenarios' in which t was unlabeled (e.g., `HT` was unlabeled together with `CB`, `LB`, and `MA`). For the third setting, we defined two measures —'error' and 'accuracy'—, as follows: $\text{err}_t = \frac{\sum_{t' \neq t} a_{tt'}}{u_t}$, where $a_{tt'}$ is the number of unlabeled cells of class t assigned to class t' and u_t the number of unlabeled cells of class t, and $\text{acc}_t = \frac{a_{tt}}{u_t}$. 'Accuracy' considers the proportion of unlabeled cells of class t classified as t whereas 'error' does not penalize assignments to newly formed clusters.

When starting the EM procedure for a mixture model $\mathcal{M}^K$, one has to choose how to initialize the parameters. Furthermore, when $K > C$ one can keep or adapt the parameters from the previous model, $\mathcal{M}^{K-1}$. We initialize the parameters with the following heuristics:

- For $m \leq C$, estimate $\boldsymbol{\theta}_m$ from cells labeled as belonging to class m.
- Keep the θ and ρ parameters from $\mathcal{M}^{K-1}$ for $C < m < K$. That is, $\boldsymbol{\theta}_m^K = \boldsymbol{\theta}_m^{K-1}, \boldsymbol{\rho}_m^K = \boldsymbol{\rho}_m^{K-1}$, $C < m < K$ where $\boldsymbol{\rho}_m = (\rho_{m1}, \ldots, \rho_{mF})$.
- Estimate $\boldsymbol{\theta}_K$ from the instances selected as described in paragraph 'Initializing a new component' in Section 10.2.2.3.
- For components $m \in \{1, \ldots, C\} \cup \{K\}$, i.e., those of fixed classes and the newly introduced one, make all features equally relevant and irrelevant: $\rho_{mj} = 0.5, \forall m \in \{1, \ldots, C\} \cup \{K\}, \forall j \in \{1, \ldots, F\}$.
- Adapt π from $\mathcal{M}^{K-1}$ to give more weight to newly discovered components $m > C$ than to class components $m' \leq C$: $\pi_m^K = 2\pi_{m'}^{K-1}, \forall m > C, \forall m' \leq C$.
- Estimate λ as if $\rho_{mj} = 0.5, \forall m, \forall j$, i.e., as if each feature was equally relevant and irrelevant for every component, in order to fully 'reset' the λ estimates.

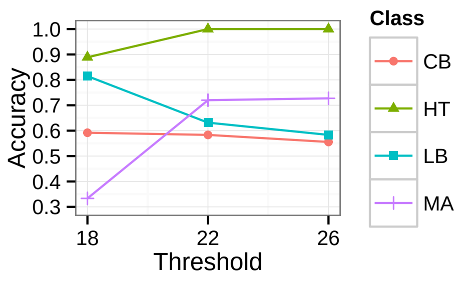
We halt mixture augmentation if $\mathcal{M}^K$ contains a component m with less than two instances (i.e., such that m is the most likely component for less than two instances), returning $\mathcal{M}^{K-1}$, unless $K = C$, in which case $\mathcal{M}^K$ is returned. The EM procedure iterates until log-likelihood converges or up to 25 iterations. We set $ct = 5$ for initializing new components, because this value produced the best results in preliminary experiments.

10.3 Results and discussion

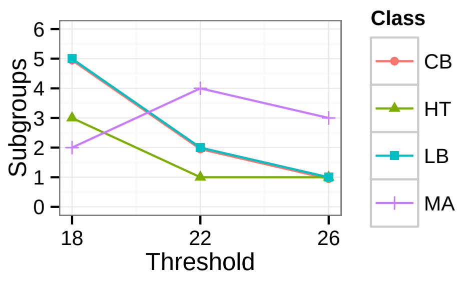
10.3.1 Discriminating among classes

10.3.1.1 Unlabeling a single class

When unlabeling a single class, `HT` and `MA` cells were better distinguished from other classes when label reliability increased; the opposite happened for `LB`, while the discrimination accuracy for `CB` cells was rather unaffected (see Fig. 10.2a). At *th26*, `HT` and `MA` cells were identified with rather high accuracy.

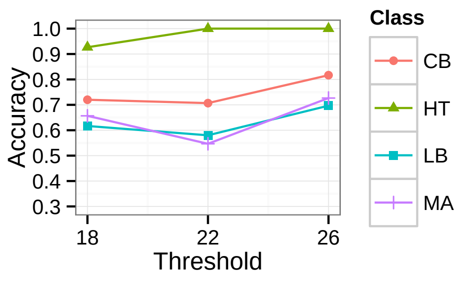


(a)

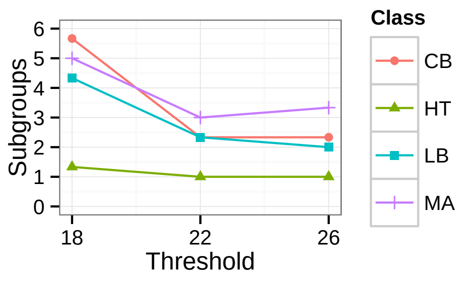


(b)

Figure 10.2: Per-class discrimination accuracy and number of subtypes when unlabeling a single class, versus label reliability threshold.



(a)



(b)

Figure 10.3: Average per-class discrimination accuracy and number of subtypes when unlabeling two classes, versus label reliability threshold. The averages are taken across the three different settings in which a class is hidden, e.g., the averages for `HT` come from the settings: `HT` and `LB` hidden; `HT` and `CB` hidden; and `HT` and `MA` hidden.

`HT` was the most easily identified class: with perfect accuracy at *th22* and *th26*, and high accuracy (0.89) at *th18*.

Although accurately identified at *th22* and *th26*, `MA` cells were confused with all the other classes, particularly with `HT` at *th18* and *th22*, and `CB` at *th26*. In this respect, the `MA` seemed to be the most heterogeneous interneuron type with respect to the used variables.

`LB` and `CB` cell types were often confused with each other but easily distinguished from other types (with the exception of `CB` at *th18*, where it was heavily confused with `HT`; see Table 10.3). This confusion is not surprising, as even expert neuroscientists often struggle to discern these two classes [DeFelipe et al., 2013].

10.3.1.2 Unlabeling two classes

When hiding two classes at a time, discrimination accuracy generally increased with label reliability (see Fig. 10.3a). `CB` cells were better discriminated than when hiding a single class, `HT` cells equally well, whereas `LB` and `MA` cells better on some label reliability thresholds and worse on others.

At *th26*, which contained the most reliably labeled cells, all classes were identified more

Table 10.3: Clustering of unlabeled instances when hiding a single class. Each row corresponds to one 'labeling scenario', e.g., `CB` is the hidden classes in the first row, whereas columns represent the classes to which the unlabeled cells were assigned. The cells assigned to newly formed clusters are considered as correctly classified and are thus displayed in blue color on the diagonal whereas the cells assigned to other classes are shown in red, outside the diagonal. Thus, for example, 16 `MA` cells were correctly classified, four were misclassified as `CB`, one as `HT`, and one as `LB` at *th26* (row four, rightmost table). Note that the number of formed subtypes is not shown —the 16 MA cells were placed in three clusters—; the formed clusters are discussed in Section 10.3.2. Zeros were omitted.

(a) *th18*

	CB	HT	LB	MA
CB	29	7	11	2
HT		8		1
LB	3		22	2
MA	1	13	8	11

(b) *th22*

	CB	HT	LB	MA
CB	14		9	1
HT		5		
LB	6		12	1
MA		5	2	18

(c) *th26*

	CB	HT	LB	MA
CB	5		3	1
HT		4		
LB	4		7	1
MA	4	1	1	16

accurately than when unlabeling a single class (see Figs. 10.2a and 10.3a). Furthermore, the classes were well separated in the formed clusters: only six out of the 20 clusters formed at *th26* (columns 'A', 'B', etc., in Tables 10.4a–10.4f) were 'mixed', i.e., contained instances of more than one class (shown in black). Thus, even though `MA` cells were misclassified as `CB` at *th26* (when unlabeling the `MA` type), these two types were neatly separated when unlabeled simultaneously (see Table 10.4a). `HT` cells were almost never placed in clusters containing other cell types (only a single `CB` cell was assigned to a `HT` cluster; see Table 10.4b). `LB` cells were least separated in the formed clusters: they were mixed with both `CB` and `MA` cells in two clusters (see Tables 10.4e and 10.4f).

Table 10.4: Clustering of unlabeled instances when hiding two classes a time at *th26*. Rows represent the hidden classes whereas columns denote the clusters the instances were ascribed to. '`A`', '`B`', etc., denote newly formed clusters. Misclassified instances are shown in red, instances in 'pure' (not-mixed) clusters are shown in blue, and instances in mixed clusters in black. Zeros were omitted.

(a) Unlabeling `MA` and `CB`

	HT	LB	A	B	C	D	E
MA	2		6	5	5	4	
CB		3		1			5

(b) `CB` and `HT`

	LB	MA	A	B
CB			1	8
HT			4	

(c) `HT` and `LB`

	MA	CB	A	B
HT			4	
LB		4		8

(d) `HT` and `MA`

	CB	LB	A	B	C	D
HT				4		
MA	2	5	6		4	5

(e) `LB` and `MA`

	CB	HT	A	B	C
LB	6		5	1	
MA	8	1	5	2	6

(f) `CB` and `LB`

	HT	MA	A	B	C	D
CB		2		4	2	1
LB		1	6		2	3

10.3.1.3 Partially unlabeling all classes

In this setting, discrimination accuracy generally improved with label reliability (see Fig. 10.4a), except for `MA` and `CB` at *th22*, where it decreased due to more instances being assigned to new clusters (indicated by their low error at *th22*; Fig 10.4b). Likewise, discrimination error generally decreased, except for `MA` and `LB` at *th26*.

At *th26*, the most accurately classified and most homogeneous types were `HT` and `CB`, as they had lowest error and highest accuracy. `MA` and `LB`, on the other hand, displayed low accuracies but relatively low errors, suggesting they were more heterogeneous than `HT` and `CB` at *th26*.

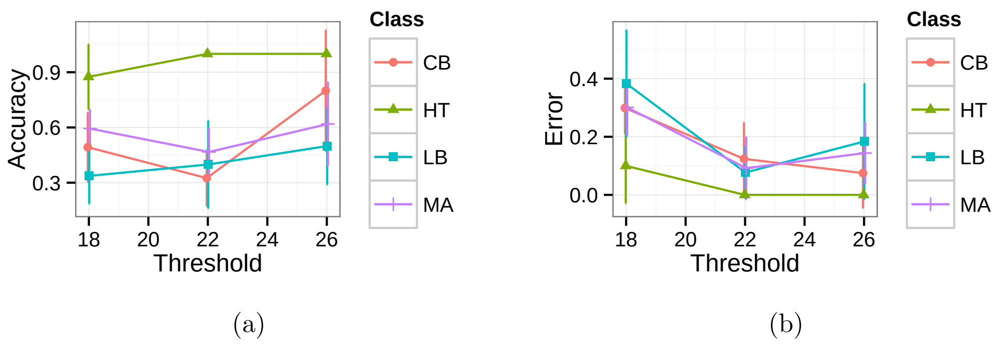


Figure 10.4: Per-class discrimination accuracy (acc_t) and error (err_t) when unlabeling half the instances of each class, versus label reliability threshold.

10.3.1.4 Summary

Summarizing the previous three sections, we can note that, despite some fluctuations — possibly due to small data samples— discrimination accuracy tended to increase with label reliability in all three experimental settings (e.g., accuracy was almost universally higher and error lower at *th26* than *th18*, see, e.g., Figs. 10.2a, 10.3a, 10.4a, and 10.4b). This might suggest that the degree of label noise decreased with label reliability threshold, and that therefore, the most reliable results were obtained at *th26*.

10.3.2 Potential subtypes

The number of subtypes generally decreased with label reliability (see Figs. 10.2b and 10.3b). This may indicate that there was more heterogeneity among less reliably labeled cells; nonetheless, this heterogeneity may simply be due to the higher number of instances at lower thresholds (especially for the `CB`, `LB`, and `HT` types). `HT` appeared as the most compact class as all of its cells were clustered together (in a single cluster) at *th22* and *th26*, in both the first and the second labeling scenario (see Figs. 10.2b and 10.3b and Tables 10.4b, 10.4c and 10.4d). `MA`, on the other hand, seemed to be the most heterogeneous —at *th26* `MA` cells were clustered into at least three subtypes (see Fig. 10.2b and Tables 10.4a, 10.4d and 10.4e). Thus, we focused on *th26* for analyzing the formed subtypes as it contained the most reliably labeled interneurons.

10.3.2.1 With a single hidden class

`MA` cells were clustered in three groups (see Fig. 10.5 for representative examples), with six, five, and five cells each, whereas a single distinct subtype was identified for the `CB` and `LB` classes, counting five and seven cells each, respectively. Since all `HT` cells were placed in a single cluster (see Fig. 10.2b), no potential subtypes of `HT` were identified.

Overall, `MA` subtypes showed relatively sparse axonal arbors, as indicated by their low or medium values for X_1 and X_2 (see Figs. 10.5a, 10.5b and 10.5c), with `MA-A` cells being less sparse than `MA-B` and `MA-C` ones. `MA-A` cells had plenty of axonal arborization far from soma (high X_5 values in Fig. 10.5a) and dendritic polar histogram length in the $[\pi, 2\pi)$ interval (X_7). `MA-B` cells exhibited medium values for all variables (Fig. 10.5b) whereas `MA-C` had the sparsest axons (low values for X_1 and X_2 in Fig. 10.5c) and, like `MA-A`, plenty of axon far from soma (X_5) and dendrites in the $[\pi, 2\pi)$ polar histogram interval (X_7). `CB-A` cells displayed relatively sparse axons (medium X_1 and low X_2 values in Fig. 10.5d), with little axon far from the soma (low X_5 values) and little dendritic arborization far from soma and in the $[\pi, 2\pi)$ polar histogram interval (low X_7 and X_9 values). `LB-A` cells exhibited the most dense axonal and dendritic arbors, with high or medium values for X_1, X_2, X_6 and X_7; the X_4 values (axonal length at medium distance from soma), was especially high (see Fig. 10.5e).

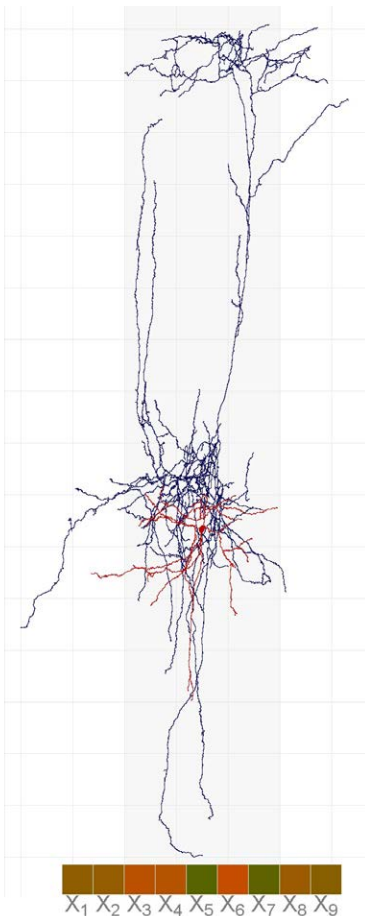
Validating the produced clusters While the discovered subtypes are relatively small —their sizes ranging from five to seven cells—, they may nonetheless be relevant in the domain of neuronal classification, where 3D neuronal reconstructions, and reliably classified reconstructions in particular, are scarce. Cluster quality indices [Halkidi et al., 2001, Handl et al., 2005] may thus help assess the goodness of the clustering solution.

One type of indices compares the obtained (crisp) clustering partition with the original one, given by class labels. We performed such an analysis in Section 10.3.1 when we computed accuracy, and now we report a measure more commonly used for this end, the Adjusted Rand index (ARI; Rand [1971], Hubert and Arabie [1985]),

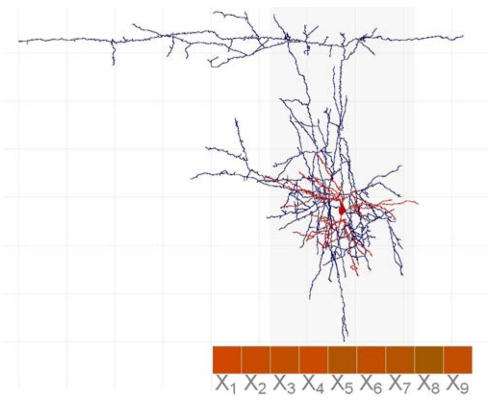
$$\mathrm{ARI} = \frac{\sum_{c=1}^{C}\sum_{m=1}^{K}\binom{N_{cm}}{2} - \frac{\sum_{c=1}^{C}\binom{N_{c.}}{2}\sum_{m=1}^{K}\binom{N_{.m}}{2}}{\binom{N}{2}}}{\frac{1}{2}[\sum_{c=1}^{C}\binom{N_{c.}}{2} + \sum_{m=1}^{K}\binom{N_{.m}}{2}] - \frac{\sum_{c=1}^{C}\binom{N_{c.}}{2}\sum_{m=1}^{K}\binom{N_{.m}}{2}}{\binom{N}{2}}}$$

where N_{cn} is the number of instances of class c assigned to cluster m. ARI rewards two types of agreements: a) clustering a pair of instances together (i.e., as members of a same cluster) in both partitions; and b) clustering a pair of instances separately (i.e., as members of different clusters) in both partitions. It reaches its maximum value, 1, when the two partitions agree perfectly, and this occurs when unlabeling `HT` cells (see Table 10.5). We also achieved high ARI values when unlabeling `CB` and `LB` cells whereas we obtained a low one when unlabeling `MA` cells; this is due to the discovery of three `MA` subtypes in the setting, which was not favored by ARI because it increased the difference among the two partitions.

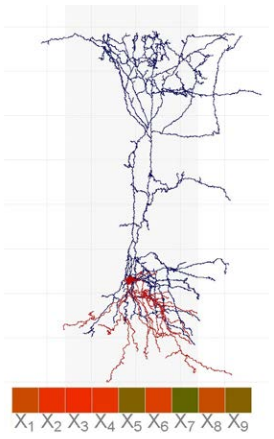
We cannot easily compute a second type of clustering quality indices, based on assessing



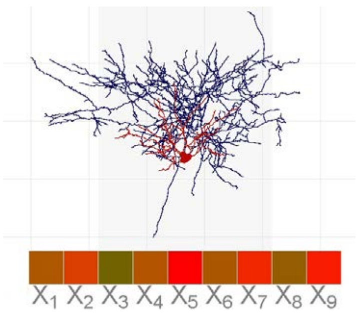
(a) MA-A



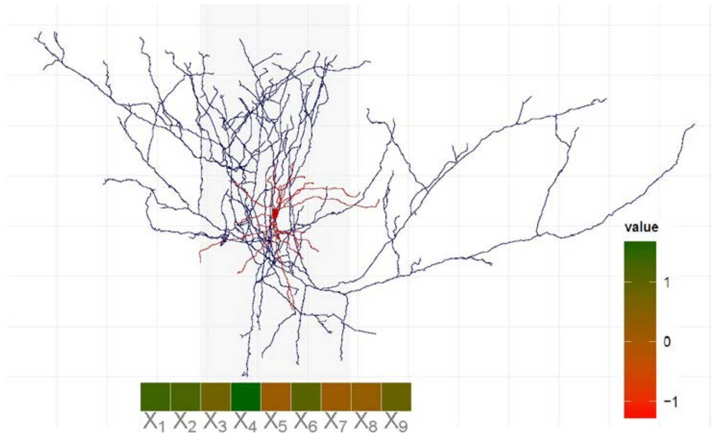
(b) MA-B



(c) MA-C



(d) CB-A



(e) LB-A

Figure 10.5: Representative members of potential subtypes identified at *th26*. The heatmaps show the subtypes' mean values for all variables. Due to standardization, values are not comparable among different variables. Thus, for example, although $X_1 > X_2$ in the case of CB-A, it does not necessarily mean that CB-A cells have (on average) more axonal length in the $[0, \pi)$ than in the $[\pi, 2\pi)$ polar histogram interval; it means that they have more of the former *relative* to the values of the remaining (i.e., non-CB-A) cells at *th26*. The five depicted subtypes contain 28 out of the 47 cells at *th26*; HT cells and misclassified MA, CB, and LB cells are not represented.

Table 10.5: Cluster validation indices when hiding a single class at *th26*. Columns denote the unlabeled class whereas rows correspond to different metrics.

	HT	MA	CB	LB
Adjusted Rand index	1.00	0.42	0.83	0.79
Silhouette	1.00	1.00	1.00	1.00

properties such intra-cluster compactness and inter-cluster separation, because that would require computing distances among data points, something which is unclear how to do when points are located in different feature subspaces. When then assessed our clustering in terms of probabilistic concordance, considering that a clustering is good if cluster membership probabilities, $p(\mathbf{z}_i)$, are similar among the members of a same cluster and different among members of different clusters. We measured the similarities among cluster membership probabilities of two data instances with Jensen-Shannon divergence,

$$d_{JS}(p(\mathbf{z}_i), p(\mathbf{z}_j)) = \frac{1}{2}(d_{KL}(p(\mathbf{z}_i), r) + d_{KL}(p(\mathbf{z}_j), r)),$$

where $r = \frac{1}{2}(p(\mathbf{z}_i) + p(\mathbf{z}_j))$ and $d_{KL}(p(\mathbf{z}_i), p(\mathbf{z}_j))$ is the Kullback-Leibler divergence [Kullback and Leibler, 1951] between $p(\mathbf{z}_i)$ and $p(\mathbf{z}_j)$,

$$d_{KL}(p(\mathbf{z}_i), p(\mathbf{z}_j)) = \sum_{m=1}^{K} p(z_{im}) \log(\frac{p(z_{im})}{p(z_{jm})}).$$

Note that $p(z_{jm})$ is simply $\gamma(z_{im})$ computed in the final step of the EM algorithm. Unlike Kullback-Leibler divergence, Jensen-Shannon divergence is symmetric, does not require absolute continuity (i.e., that $p(z_{im}) = 0 \implies p(z_{jm}) = 0$), its square root is a metric, and it is bounded: $0 \le d_{JS} \le 1$ [Lin, 1991].

Using Jensen-Shannon divergence as the measure of distance among data instances, we computed the Silhouette width [Rousseeuw, 1987] clustering index, thus measuring intra-cluster compactness and inter-cluster separation in terms of this distance. The Silhouette width is given by

$$\mathrm{SW} = \frac{1}{N} \sum_{i=1}^{N} \frac{b_i - a_i}{\max(b_i, a_i)},$$

where a_i is the average distance between $\mathbf{x}_i$ and other points in its cluster, while b_i is its average distance to the points in the closest cluster (defined as that yielding the lowest b_i). We achieved maximum Silhouette values (its values range from -1 to 1) for all labeling scenarios (see Table 10.5), showing that the cluster membership probabilities had converged reasonably.

A partial cause for high Silhouette widths was that we only unlabeled a subset of instances in each setting, and it was only those instances' $p(\mathbf{z}_i)$ that were estimated by our algorithm, while the rest instances' $p(\mathbf{z}_i)$ were fixed to degenerate distributions (probability 1 for the class labels' cluster; 0 for all other clusters) which were different among the clusters. Thus, e.g., when unlabeling the `HT` cells, we estimated the $p(\mathbf{z}_i)$ of only four instances, the overall Silhouette width therefore necessarily being high, due to the inter-cluster differences among the fixed $p(\mathbf{z}_i)$. Yet, as Table 10.6 shows, Silhouette widths were also high for the newly discovered clusters, not only for those corresponding to the known types (and thus containing many cells with fixed degenerate $p(\mathbf{z}_i)$).

Table 10.6: Average per-cluster Silhouette width. Rows correspond to the hidden (unlabeled) classes whereas the columns denote clusters. '`-A`', '`-B`', and '`-C`' denote newly formed clusters; for example, the intersection of the first row and fifth column (`-A`) corresponds to the `HT-A` cluster, whose average Silhouette width was 1.000.

	CB	HT	LB	MA	-A	-B	-C
CB		1.000	1.000	0.996	1.000		
HT	1.000		1.000	1.000	1.000		
LB	0.994	1.000		0.996	0.985		
MA	0.990	1.000	1.000		0.997	0.999	1.000

Finally, in the next section we validated the discovered subtypes with a different experimental setting —the hiding of two classes—, thus evaluating their 'stability', i.e., their robustness to different labeling scenarios.

10.3.2.2 With two hidden classes

The validity of the above-described subtypes of the `MA`, `CB`, and `LB` types, identified when hiding a single class, was confirmed when hiding two classes simultaneously. That is, in almost every labeling scenario in the second setting (i.e., for every pair of hidden classes), there was a cluster that greatly resembled the corresponding subtypes. So, for example, when hiding `CB` and `MA`, cluster E (Table 10.4a) was identical to subtype `CB-A`. Furthermore, four cells from the `CB-A` subtype were clustered together in every labeling scenario, that is, the intersection of `CB-A` and cluster E in Table 10.4a, cluster B in Table 10.4b, and cluster B in Table 10.4f consisted of four cells. This subset of `CB-A` cells emerged as its 'core' of highly similar instances, showing the robustness of this subtype (which totalled five cells).

A 'core' of the `MA-C` subtype (i.e., the `MA-C` cells that were always clustered together) emerged, consisting of four cells which formed the intersection of the pure `MA` clusters C, A, and C in Tables 10.4a, 10.4d, and 10.4e, respectively (the latter two clusters being identical). Regarding `MA-A`, a 'core' of three cells emerged, defined by the intersection of clusters A, C and A in Tables 10.4a, 10.4d, and 10.4e, respectively (the latter being a mixed cluster). Finally, clusters C and D in Tables 10.4a and 10.4d contained four and three `MA-B` cells, respectively.

A `LB-A` core of five cells emerged, contained in clusters A, B, and A in Tables 10.4e, 10.4c and 10.4f, respectively. The latter two contained a larger `LB-A` core of six cells.

10.3.2.3 Feature relevance

We focused on the first setting (i.e., hiding a single class) and *th26* to analyze the estimated relevance of predictor variables. Overall, all predictor variables seemed useful, as each one was very likely relevant (around 100% chance of being relevant; dark green boxes in Fig. 10.6) for at least two subtypes/classes identified at *th26*. Feature X_2 —axonal polar histogram in the $[\pi, 2\pi)$ interval— appeared to be the most useful as it was very likely relevant for all the subtypes/classes (see Fig. 10.6). While its relevance for the `HT` class and `MA` subtypes is clear —an `MA`'s axon grows predominantly upwards from the soma whereas the opposite holds for `HT`— it is interesting that it was relevant for the `CB` and `LB` subtypes as well. Features X_4 and X_5, which capture the length of axonal arborization at 150–300 µm and over 300 µm from soma, were relevant for five (out of six) subtypes. On the other hand, feature X_9 —length of dendritic arborization at over 90 µm from soma— appeared as least useful as it was very likely relevant for only two subtypes (`CB-A` and `MA-B`). In general, axonal features (X_1 to X_5) were more likely to be relevant than dendritic ones (X_6 to X_9). For example, an axonal feature was, on average, very likely relevant for 4.6 subtypes/classes in Fig. 10.6 whereas a dendritic one was for 2.5. Likewise, a dendritic feature was, on average, probably irrelevant (below 50% chance of being relevant; red and brown boxes in Fig. 10.6) for more subtypes than an axonal feature.

Moreover, as shown by Fig. 10.6, the relevance of axonal features differed among the subtypes/classes. All axonal features were very likely relevant for all `MA` subtypes (except X_1 for `MA-A`: `MA-A`, unlike `MA-B` and `MA-C`, did not stand out regarding X_1 —note the olive green box for X_1 in Fig. 10.5) and for the `HT` class. This suggests that the axonal arborizations of the `MA` subtypes and the `HT` class were distinct among themselves and from the `CB-A` and `LB-A` subtypes according to all axonal features. On the other hand, two and three (out of five) axonal features were relevant for the `CB-A` and `LB-A` subtypes, respectively, and, in particular, polar histogram in the $[0, \pi)$ interval and close to soma axonal arborization length (X_1 and X_3, respectively) were relevant with only 50% chance for both of those subtypes. This suggests that `CB-A` and `LB-A` were not particularly distinct according to those variables.

In summary, the results clearly show that the length of axonal polar histogram in the $[\pi, 2\pi)$ interval (X_2) is particularly relevant, followed by axonal length relatively near to the cell body (X_4) and the extent of the axon far from the cell body (X_5). In addition, the dendritic arborization was highly relevant for the characterization of `CB` cells and `MA` cells. These findings may be useful for generating an accurate automatic classifier of 3D reconstructed interneurons.

10.4 Conclusions

We introduced a semi-supervised approach to neuron classification. It can leverage prior knowledge —in the shape of class labels— to identify members of established types, while allowing for the discovery of new neuron types. Simultaneously, it is of descriptive value as it identifies variable relevance for the types and subtypes.

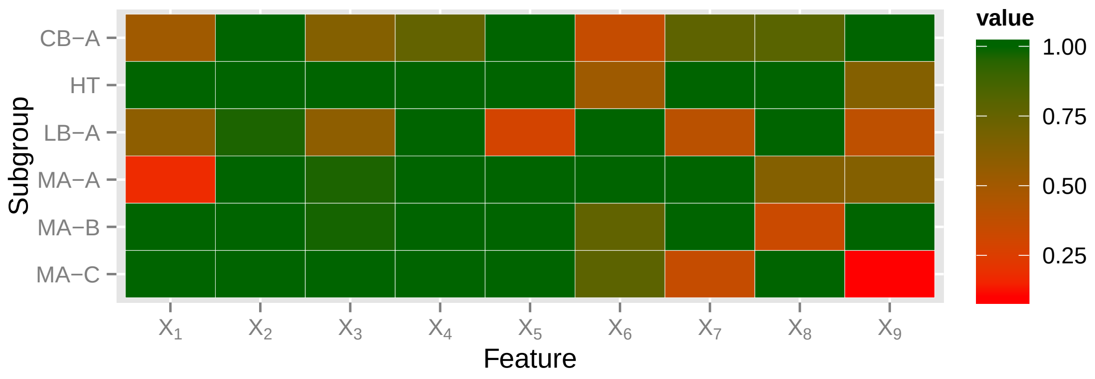


Figure 10.6: Relevance ($\boldsymbol{\rho}$) of features (X_1-X_9) for the `HT`, `LB`, `CB`, and `MA` subtypes identified at *th26* when hiding a single type. Each row corresponds to one subtype, e.g., the last three rows correspond to the three subtypes of `MA`. The displayed subtypes do not contain all the cells contained at *th26*, as some were misclassified (e.g., besides `CB-A`, `CB` cells at *th26* were also assigned to the `LB` and `MA` classes; the latter cells are not contained in any subtype represented in this graph).

We presented results on the classification of common basket, large basket, horse-tail, and Martinotti cells. We quantified the neurons with simple morphological features which describe the distribution and the orientation of axonal and dendritic arbors, seeking to mimic the way in which an expert visually classifies a neuron. The algorithm accurately discriminated among the different types when one and two types were unlabeled at a time and when half the instances of all types were unlabeled. It identified potential subtypes of common basket, large basket, and Martinotti cells, suggesting that these types are heterogeneous. While the identified subtypes are small, consisting of few cells, they may be indicative of the characteristics that differentiate cells belonging to the same interneuron type.

The proposed morphological variables seemed useful for discriminating among the types. Axonal features were more useful than dendritic ones, especially axonal polar histogram length in the $[\pi, 2\pi)$ interval. It is possible that more refined variables, such as those considering both the distance and position with respect to the soma (e.g., over 300 µm from soma and above it) could further improve discrimination accuracy.

Overall, the results suggest that a semi-supervised approach may be helpful in neuronal classification and characterization. Further studies, with more morphological features and neurons, are needed in order to obtain more conclusive results.

Instead of the majority label for each labeled instance, it would be straightforward to use probabilistic labels given by considering the choices of all the experts.

Chapter 11

Per-class supervised classifiers

11.1 Introduction

In this chapter, we learn models from 217 high-quality reconstructions, namely two-week-old male rat hind-limb somatosensory cortex interneurons, reconstructed at the Markram laboratory [Ramaswamy et al., 2015]. Each cell was pre-classified into one of eight morphological types described in Section 5.4.1 (see Figure 11.1 for abbreviations)[1]. With seven ChC and 15 bitufted (BTC) —yet 123 BA and 50 MC — cells, the sample was insufficient to accurately distinguish each of the eight types, yet the homogeneity and quality of the data, along with a careful selection of morphometrics and a comprehensive machine learning approach, allows for establishing a baseline classification.

Although the class labels were assigned following clear criteria, they came from a single laboratory, and we thus contrasted them, for 20 cells, to alternative labels provided by 42 leading neuroscientists that participated in DeFelipe et al. [2013]. We also looked for morphology reconstructions issues which might distort the morphometrics.

We trained a model for each type in a one-versus-all fashion [e.g., ChC or not ChC; see Rifkin and Klautau, 2004]. Importantly, we developed custom `R` [R Core Team, 2015] code to quantify a number of Petilla features, including those relative to arbor shape and direction, dendritic polarity, the presence of arborization patterns typical for the MC and ChC types, and translaminar extent [Helmstaedter et al., 2009c], which we estimated using metadata on laminar thickness and soma's laminar location (i.e., which layer contained it). We complemented them with standard axonal and dendritic morphometrics [Uylings and Van Pelt, 2002], such as the mean branching angle and mean terminal branch length, computed with the NeuroSTR library[2]. For each classification task (e.g., ChC or non-ChC), we ran seven state-of-the-art supervised classification algorithms [Murphy, 2012, Hastie et al., 2009], such

[1]While Markram et al. [2004] describe nine interneuron types in L2/3 to L6, we lacked enough bipolar and neurogliaform cells to learn classifiers for them. We also grouped small, nest, and large basket cells into a separate, basket type.

[2]NeuroSTR is an open source library developed in our research group in the context of the Human Brain Project [Markram, 2012]. Its online repository is at `https://github.com/ComputationalIntelligenceGroup/neurostr`.

as random forest [Breiman, 2001] and lasso-regularized logistic regression [Tibshirani, 1996]. As a prior step, we applied univariate and multivariate feature selection [Guyon et al., 2006, Saeys et al., 2007] and sampled the training data to deal with class imbalance [e.g., there were seven ChC and 210 non ChC cells; see He and Garcia, 2009, Chawla et al., 2004]. We validated the MC models against the classification by 42 neuroscientists from DeFelipe et al. [2013] and found that cells commonly misclassified by different models [Brodley and Friedl, 1999] may correspond to atypical MC morphologies[3]. The study can be easily reproduced [Ince et al., 2012, Leitner et al., 2016, Lowndes et al., 2017] as all code and data are available[4].

The research covered in this chapter has been submitted in Mihaljević et al. [2018c].

The rest of this chapter is organized as follows. Section 11.2 briefly discusses the data. Section 11.3 describes the methodology, including the computation and choice of morphometrics, supervised classifiers, feature selection, sampling and evaluation procedures, and the validation of the MC models. Section 11.4 presents the classification and feature selection results, including interpretable models for the BA and MC types. We discuss the results in Section 11.5 and conclude in Section 11.6. Appendix A provides the list and definitions of morphometrics while Appendix B gives additional results.

11.2 Data

We obtained 228 hind-limb somatosensory cortex interneurons from two-week-old male Wistar (Han) rats, which Markram et al. [2015] used for simulating the cortical column[5]. They corrected shrinkage along the Z-axis, while shrinkage along the X and Y axes was of approximately 10%. They classified the cells into 36 layer L2/3 to layer L6 morphological types (see Section 5.4.1) of inhibitory neurons, based on their soma's layer and anatomical features described in Markram et al. [2004], Wang et al. [2002, 2004], updating these criteria with a few laminar specificities: e.g., L6 MC cells were unique in that they did not reach L1, but 'had a second axonal cluster formed below L1' [Markram et al., 2015, page 2 in their supplementary material]. For each cell, we knew the layer that contained the soma and had estimates of mean and standard deviation of cortical layers' thickness (see Table A.3). We had no data on fine-grained features related to boutons and dendritic spines. We merged the interneuron types across layers (e.g., we considered L23_MC and L4_MC cells as members of a single MC class) into the nine morphological types defined by Markram et al. [2004].

We had an alternative classification for 79 of our cells provided by 42 neuroscientists that participated in the study by DeFelipe et al. [2013], who were shown 2D and 3D images of the cells and were told the layer containing the soma, and classified them following the scheme by DeFelipe et al. [2013]. Among these, we used the 20 cells[6] classified in our data —that

[3]We restricted this analysis to the MC type as only for MC we could compare it to an independent classification by neuroscientists in DeFelipe et al. [2013].

[4]Online repository at https://bitbucket.org/cbb-bojan/bbp-interneurons-classify.

[5]Markram et al. [2015] used 1009 digitally reconstructed cells; the 228 cells that we use are the interneurons that they classified on the basis of morphological parameters, as shown in Figure S2 of that paper.

[6]One of these 20 cells, C040600B2, was shown to the neuroscientists rotated upside-down, which may have

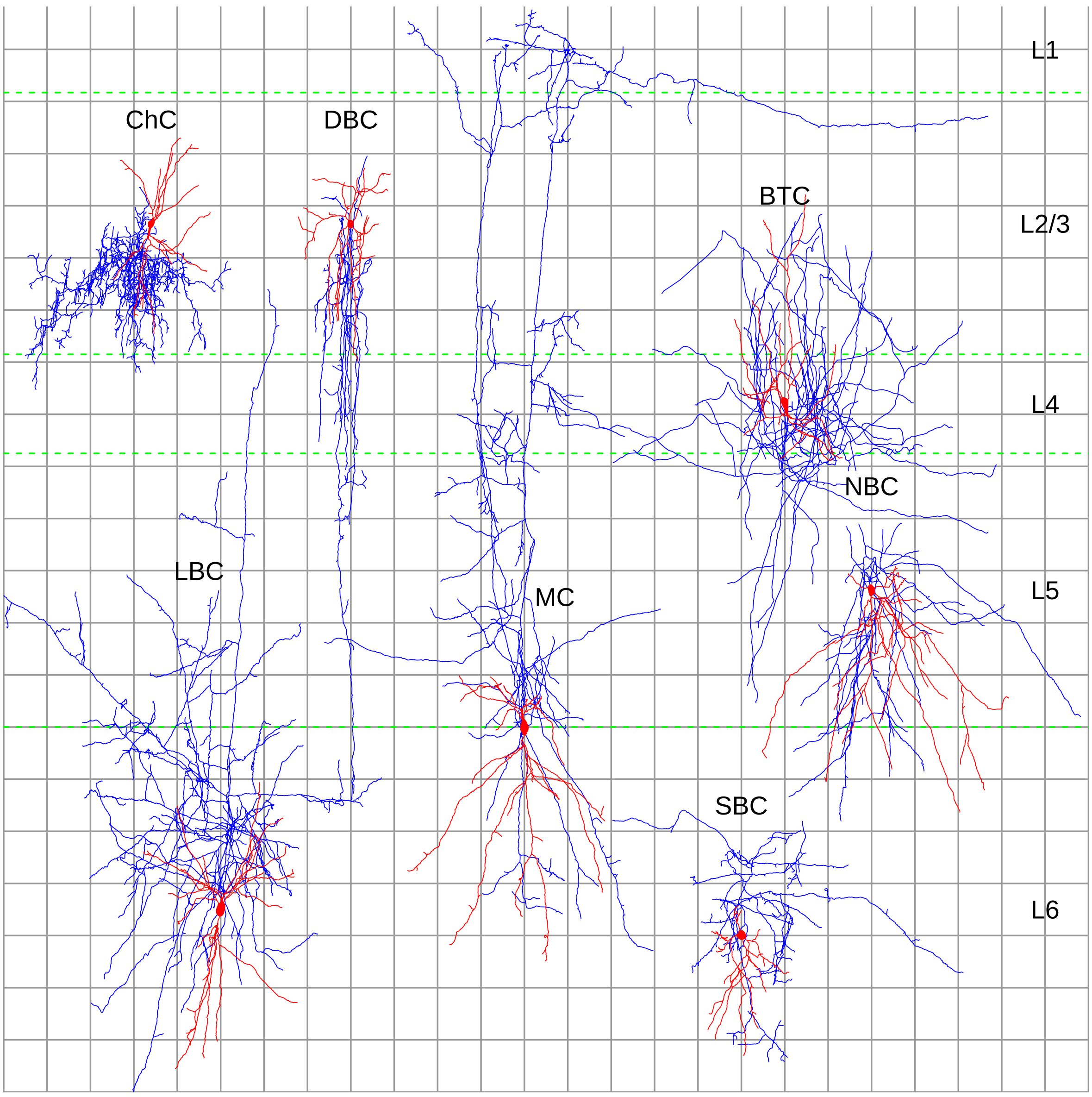


Figure 11.1: Examples of the eight morphological types from Markram et al. [2004] for which we learned supervised models. The types are: bitufted (BTC); chandelier (ChC); double bouquet (DBC); large basket (LBC); martinotti (MC); nest basket (NBC); small basket (SBC), and the compound basket (BA) type, composed of NBC, LBC, and SBC cells. Neurogliaform (NGC) and bipolar (BP) types not shown as we omitted them from supervised classification, because we had only three cells of each. Typical features, according to Markram et al. [2004], include bitufted dendrites (BTC), sharply branching axons and low bouton density (LBC), and axons with spiny boutons, reaching L1 (MC), and vertical rows of boutons (ChC). Axons are drawn in blue and dendrites and somata in red. Dashed green lines indicate layer boundaries from the rat hind-limb somatosensory cortex. There are 100 μm between consecutive grid lines.

is, by Markram et al. [2015]— as MC, ChC, and NGC —the three types common to both classification schemes— to contrast the neuroscientists' labels to ours, but we did not use

affected how they classified it.

them to train the models. We will reserve the term 'our labels' to the labels by Markram et al. [2015] which we trained the models with.

For supervised classification, we omitted the BP and NGC types, as we had only three examples of each and formed a compound type —basket (BA)— by merging the NBC, LBC, and SBC cells. We also omitted five cells with morphology issues: three cells whose axonal arborization was interrupted, and two with short axons (2500 μm and 2850 μm)[7], thus obtaining the final sample of 217 cells from eight interneuron types (seven 'base' types plus the compound BA type) used for supervised classification (see Figure 11.2).[8]

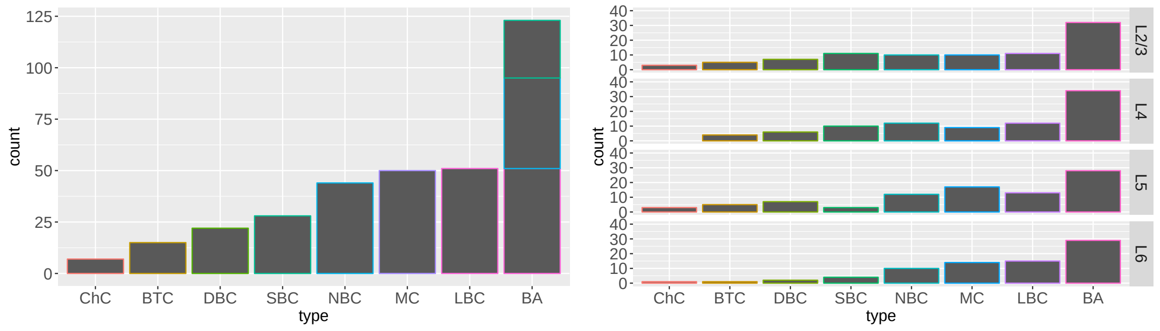


Figure 11.2: Frequencies of interneuron types in our data: overall (left) and per cortical layer (right). This figure shows the 217 cells used for supervised classification, with the SBC, NBC, and LBC types also shown in the bar corresponding to BA (i.e., the BA bar does not contribute to total cell count).

11.3 Method

11.3.1 Morphometrics

We computed a total of 103 axonal and dendritic morphometrics, 48 of which were custom-quantified Petilla [Ascoli et al., 2008] features. The custom-implemented morphometrics cover a) arbor shape, direction, density and size; b) laminar distribution; c) dendritic polarity and displacement from axonal arbor; and d) the presence of arborization patterns typical of the MC, ChC, and LBC types. We determined arbor orientation with principal component analysis, following Yelnik et al. [1983]. We quantified laminar distribution as the probability of the arbor reaching at least two layers (one being its soma's home layer), given that the soma's vertical position within its layer was unknown and that laminar thicknesses were random variables rather than precise values. We distinguished between bipolar/bitufted and multipolar dendrites by determining whether dendrite roots were located along a single axis (for an alternative metric see Helmstaedter et al. [2009b]). Finally, we quantified a number of complex, type-specific patterns with simple, ad-hoc morphometrics. For the MC type, we quantified the 'axonal collaterals that reach layer L1 and then ramify to form a fan-like spread

[7]We found that in the study by DeFelipe et al. [2013], the shortest axon which allowed at least half of the 42 neuroscientists involved to characterize an interneuron (i.e., to consider that the neuron can be classified) was 2805 μm, with the next shortest being 3197 μm.

[8]We considered all 228 cells when contrasting our class labels to those from DeFelipe et al. [2013].

of axonal collaterals' [Ascoli et al., 2008] pattern by considering the estimated probability of the axon reaching L1 and properties, such as width, of the upper part of the arbor. For ChC, we counted the number of 'short vertical terminal branches'. We did not estimate translaminar extent as, without knowing the soma's location within the column, it is poorly correlated to tangential arborization span [Helmstaedter et al., 2009c]. Figure 11.3 illustrates some of these morphometrics.

The remaining 55 morphometrics were standard metric and topological [Uylings and Van Pelt, 2002] ones, such as bifurcation angles and partition asymmetry [Van Pelt et al., 1992], including features of axon terminal branches such as length and curvature. We avoided morphometrics possibly sensitive to reconstruction granularity, such as those derived from axonal and dendritic diameter, local bifurcation angles, or segment length (e.g., the `Fragmentation` and `Length` analyses in L-Measure), as we had two groups of cells that differed sharply in mean diameter and segment length (see Section 11.4.1).

We computed the morphometrics with the open-source NeuroSTR library and custom `R` [R Core Team, 2015] code. NeuroSTR allowed us to handle multifurcations (e.g., we ignored angle measurements on multifurcating nodes) and compute arbitrary statistics, so that, for example, we computed the median branch length. Still, a number of potentially useful morphometrics available in Neurolucida Explorer, such as box counting fractal dimension [Panico and Sterling, 1995], were not available in NeuroSTR and thus were not considered in this study. Appendix A lists all the used morphometrics, with definitions and computation details.

11.3.2 Supervised classification

Rather than training models to distinguish among all interneuron classes at once, we considered eight settings where we discerned one class from all the others merged together (e.g., whether a cell is a ChC or a non-ChC cell). A benefit is that we can interpret such models, and look for relevant morphometrics, in terms of that particular type. On the other hand, training these models suffers from class imbalance [He and Garcia, 2009]; this was most pronounced for the ChC type (there were seven ChC cells and 210 non ChC cells), and least for BA (123 BA and 94 non-BA cells), which was the only setting in which the class of interest was the majority one (i.e., there were more BA than non-BA cells).

To each classification setting we applied seven supervised classification algorithms (see Table 11.1 for a list with abbreviations), such as random forest (RF) or support vector machines (SVM), representative of different machine learning paradigms. We handled class imbalance with a hybrid of random undersampling and SMOTE oversampling [e.g., Estabrooks et al., 2004], meaning that we removed some majority class and added minority class instances to the training data. We also pruned the set of morphometrics by keeping only those relevant according to the Kruskal-Wallis[9] (KW) statistical test [Kruskal and Wallis, 1952] and our

[9]In our binary classification settings the Kruskal-Wallis test corresponds to its special case for two samples, the Wilcoxon–Mann–Whitney test [Wilcoxon, 1945, Mann and Whitney, 1947]. We keep the term Kruskal-Wallis as that is the implementation that we used (`R` function `kruskal.test`).

adaptation of the RF variable importance ranking [Breiman, 2001] for imbalanced settings, termed balanced variable importance (RF BVI; see Section 11.3.2.1 for details), seeking to simplify the models. Also, in small-sample class-imbalance settings, univariate feature selection [Guyon et al., 2006], such as KW, can improve predictive performance more than over- and under-sampling [Wasikowski and Chen, 2010].

Most of the used classifiers, as well as the sampling and feature selection methods, require one to specify parameters, such as number of neighbors for the kNN classifier or the number of majority class instances to remove in undersampling. While learning these from data may improve performance, we opted to avoid additional learning complexity (i.e., increasing the probability of over-fitting) and instead pre-specified all parameters, using mostly the default values from the implementations of the corresponding methods (see Section 11.3.2.1) rather than fine-tuning them. For over- and under-sampling we devised a heuristic (see Section 11.3.2.1) to determine the sampling ratios; Figure 11.4 illustrates its effects on the class distributions in the different settings. Note that we used the same parameters in all eight classification settings.

The full learning sequence was then: 1) feature selection; followed by 2) data sampling; and finally 3) classifier induction, with steps 1 and 2 being optional (i.e., we also considered not selecting features and not sampling the training data). We evaluated the classification performance with F-measure (see Section 3.5) and estimated it with k-fold cross-validation. We ran all three steps of the learning sequence on the k training data sets alone, i.e., without using the test fold (that is, we selected features and sampled data within the cross-validation loop, not outside of it). Since data sampling is stochastic, and a large sampling ratio can change the training set class distribution, we repeated cross-validation ten times when including sampling within the learning sequence. Finally, we identified potentially atypical MC morphologies as those commonly misclassified by different models [Brodley and Friedl, 1999].

The next section provides details about the used methods, describes the sampling procedure and F-measure computation, and gives implementation details.

11.3.2.1 Details

We standardized all predictors to zero mean and unit variance. This gives equal weight to all predictors for the kNN classifier, and allows us to interpret the magnitude of the coefficients of the linear models, while it does not affect the remaining models.

We set the classifiers' parameters (see Table 11.1) on the basis of available recommendations [Boulesteix et al., 2012, Hsu et al., 2003] or we used the defaults in the software implementation. For kNN we used $k = 5$, and, similarly, for CART we set $|D^l| = 5$; while this might be too coarse-grained for ChC, as there are at most six ChC cells per training set, we sought to avoid overly complex models (with a lower k). Note that the $m = \sqrt{n}$ parameter for RF was recomputed on every training set; thus, it was adjusted each time feature selection reduced n. For RF, we set $T = 2000$ and chose the standard value of $m = \sqrt{n}$.

For KW feature selection, we set the significance level $\alpha = 0.05$, whereas for RF BVI ranking we selected features with BVI ≥ 0.01 and kept $m = \sqrt{n}$, although it can yield a

BVI ranking that prefers correlated features [Nicodemus et al., 2010], while we increased the number of trees, T, to 20000, as that produced stable BVI values (a higher T does not increase model variance nor presents any other drawback besides longer computation time).

Due to the imbalance, we did not use the standard VI metric, while we did not use the ROC-curve-based VI as it is only available for conditional inference trees (see Section 3.12.2). Instead, we used the arithmetic mean of the unscaled per-class VI values provided by the `randomForest` `R` package, and refer to this as the balanced VI (BVI)[10]. We used a simple heuristic and selected only features with a BVI above 0.01.

For undersampling, we randomly removed up to a half of the majority class instances, keeping at least three majority class instances per each one of the minority class (thus, for the imbalance ratios minority:majority above (i.e., less pronounced than) 1:3 we did not undersample). More precisely, after undersampling there were $N_M^{(u)} = \max\left(\min\left(3N_m, N_M\right), 0.5N_M\right)$ majority class instances, where N_M is the number of samples from the majority class and N_m that of the samples from the minority class. We then run SMOTE on the undersampled data set, adding up to three synthetic instances per each minority class example; thus, after oversampling there were $N_m^{(o)} = \min\left(N_M^{(u)}, 3N_m\right)$ minority class examples. Therefore, for large imbalances (e.g., a ratio 1:10) most balancing was due to undersampling, potentially reducing imbalance down to a ratio of to 1:3, with SMOTE oversampling then further reducing the ratio towards 1:1.

We evaluated the learning procedures with stratified 10-fold cross-validation, except for ChC versus rest, where we used seven folds (in order to have at least one ChC instance in each test set). When sampling the training data, we repeated CV 10 times and averaged the results. When computing F-measure, we always considered the minority class as the positive one, except for BA versus rest, when we considered BA, the majority class, as the positive one. Note that, unlike for classification accuracy, one cannot get an unbiased estimate of F-measure by averaging over the k test samples [Forman and Scholz, 2010]; thus, we computed the F-measure of a CV run from the full confusion matrix, obtained by aggregating the true labels and predictions from the k test folds. That is, the F-measure estimate for a run of cross-validation was not the average of k per-fold F-measure scores, but rather the single value computed from the aggregated confusion matrix. We looked for mislabelled cells by collected misclassifications over 30 runs of ten-fold cross validation. Table 11.1 and Table 11.2 list all the parameters.

[10]The VIs of the majority class are expected to be lower [Janitza et al., 2013] and using a harmonic or geometric mean, instead of the arithmetic, would further decrease the estimate, obfuscating possible effects in the minority class.

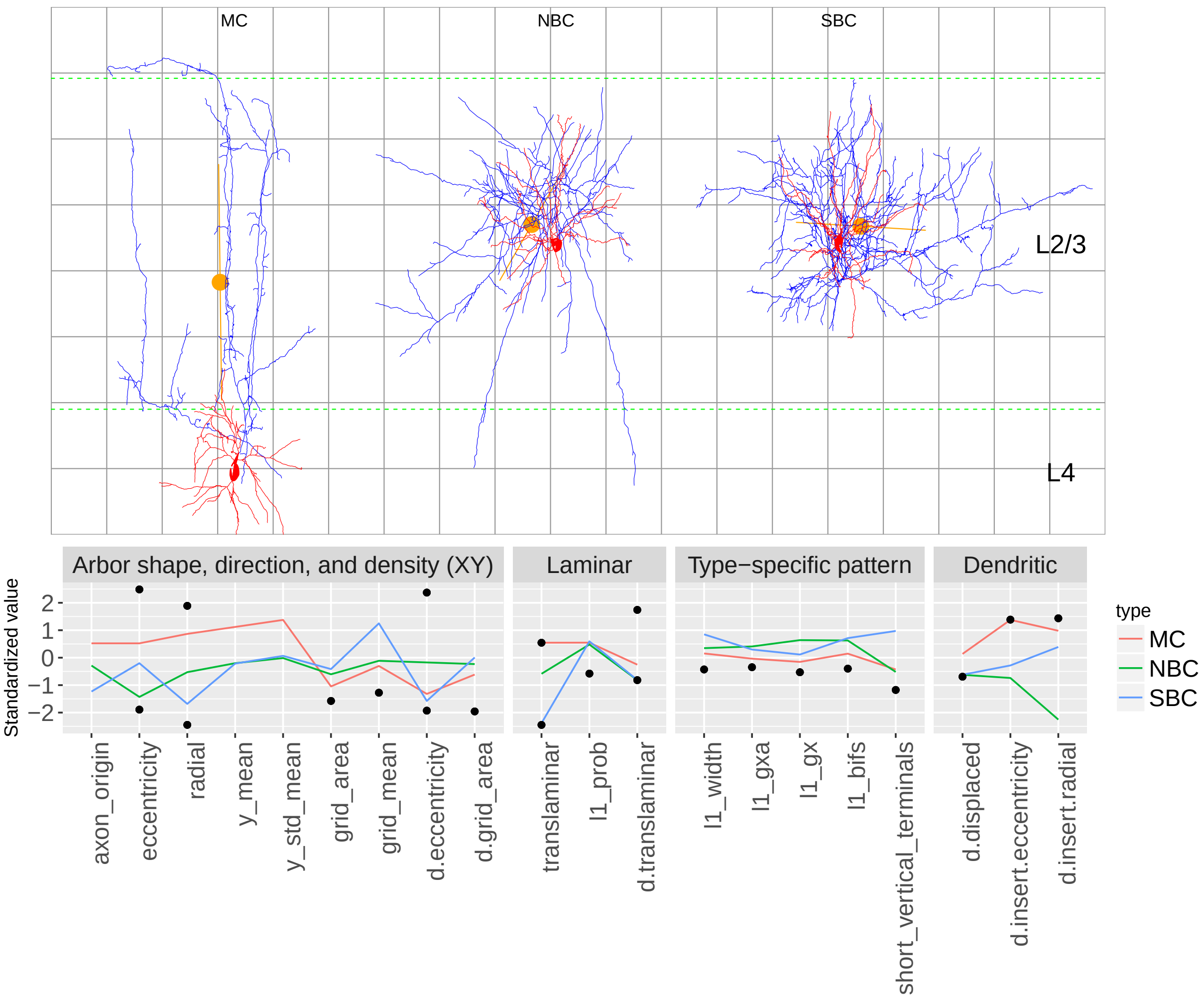


Figure 11.3: Custom-implemented morphometrics for an L4 MC (top: left; bottom: red), an L2/3 NBC (top: middle; bottom: green), and an L2/3 SBC (top: right; bottom: blue) interneuron. The bottom panel shows standardized values, with black dots indicating minima and maxima (extrema outside $(-2.5, 2.5)$ not shown). The axon of the MC cell originates from the upper part of the soma (`axon_origin`), grows along a radial axis (`eccentricity`, `radial`; axis drawn with the orange line), radially far from the soma (`y_mean`, center of mass shown with orange dot) and above it (`y_std_mean`), covers a small surface (`grid_area`), and its branches are not clustered together (`grid_mean`). It is translaminar (`translaminar`) and there is just a moderate (around 30%) probability it reaches L1 (`l1_prob`) because, even with its soma vertically in the middle of L4, it only touches the bottom of L1. Low `l1_prob` and arbor width produce a low estimate of width (`l1_width`), bifurcations count (`l1_bifs`), and horizontal fanning out (`l1_gxa`) in L1. The dendritic arbor of the MC cell is displaced (`d.displaced`) from the axon and the dendrites stem from opposite ends of the soma (`d.insert.eccentricity`), located along a radial axis (`d.insert.radial`). The NBC cell's axonal arbor is circular (`radial`), with closely grouped branches (`grid_mean`)) and a number of short vertical terminals (`short_vertical_terminals`). The axon of the SBC cell is intralaminar, tangentially oriented, with closely grouped branches, while both cells' dendrites are spread out (multipolar) and colocalized with the axons. Dashed green lines indicate layer boundaries from the rat hind-limb somatosensory cortex, assuming that the somas are located in the middle of their layer. Axon is shown in blue and dendrites and somata in red. The grid lines are at 100 μm from each other. Dendritic morphometrics are prefixed with `d.`. Axon terminal branch morphometrics, not shown here, are prefixed in the remainder of the text with `t.`.

Table 11.1: Classification algorithms and their parameterization. For kNN, $p = 2$ stands for Euclidean distance. RBF: radial basis function. Remaining parameters are defined in the background chapter. `R` package is the library implementing the method.

Classifier	Abbreviation	R Package	Prespecified Parameters
Classification and regression trees (Section 3.11.5)	CART	`rpart` [Therneau et al., 2015]	$\lvert\mathcal{D}^a\rvert = 10$, $\lvert\mathcal{D}^l\rvert = 5$
k nearest neighbors (Section 3.11.7)	kNN	`kknn` [Hechenbichler and Schliep, 2004]	$k = 5$, $p = 2$ unweighted
Linear discriminant analysis (Section 3.11.2)	LDA	`MASS` [Venables and Ripley, 2002]	
Gaussian naïve Bayes (Section 3.11.1)	NB	`e1071` [Meyer et al., 2015]	
Random forest (Section 3.11.6)	RF	`randomForest` [Liaw and Wiener, 2002]	$T = 2000$, $m = \sqrt{n}$
Lasso regularized logistic regression (Section 3.11.3)	RMLR	`glmnet` [Friedman et al., 2010]	$\lambda = 0.01$
Support vector machine (Section 3.11.4)	SVM	`e1071` [Meyer et al., 2015, Chang and Lin, 2011]	RBF: $\gamma = \frac{1}{n}$, $C = 1$

Table 11.2: Parameters for feature selection (KW and RF BVI), sampling (SMOTE) and cross-validation (CV). FDR stands for false discovery rate; r is the number of CV repetitions; k the number of folds. Learner parameters are the RF parameters used internally for RF BVI.

Method	R Package	Parameters	Learner parameters
KW (Section 3.12.1)	`stats` [R Core Team, 2015]	$\alpha = 0.05$ adjust = FDR	
RF BVI (Section 3.12.2)	`randomForest` [Liaw and Wiener, 2002]	$bvi > 0.01$	$T = 20000$, $m = \sqrt{n}$
SMOTE (Section 3.9)	`mlr` [Bischl et al., 2015]	$k = 5$	
CV	`mlr` [Bischl et al., 2015]	$r = 10$	k = 10, for ChC k=7

We implemented all the data analysis and classification in the R programming language [R Core Team, 2015]. We used the mlr [Bischl et al., 2015] for classifier learning and evaluation, feature selection, oversampling and undersampling, extending it to compute a global F-measure for an entire cross-validation run and adding the FDR p-value correction to the KW feature selection method. All code and data are available at https://github.com/ComputationalIntelligenceGroup/bbp-interneurons-classify.

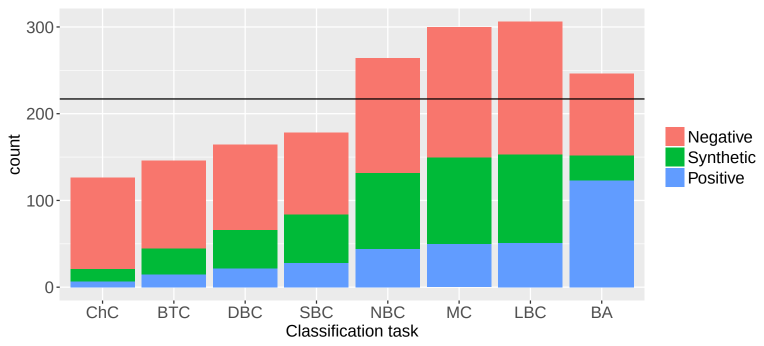


Figure 11.4: Effects of under- and over-sampling the full dataset with the chosen rates. Each bar represents a one-versus-all classification task (e.g., the leftmost bar is for ChC versus rest). 'Positive' denotes the examples of the class of interest (e.g., ChC in the leftmost bar), 'Synthetic' are the artificial SMOTE examples of the positive class (i.e., the class of interest), while 'Negative' are the kept examples of all remaining classes. The horizontal line shows the size of the original data set (217 examples). For ChC (leftmost bar), for example, applying our sampling method to the full data set holding seven ChC cells (red segment of the bar), would keep 105 (blue segment) out of 210 non-ChC cells and add 14 synthetic ChC cells (green segment), yielding a data set of size 126 (hence the bar is lower than the horizontal line at 217). Except for BA, in all cases the class of interest was the minority class. For NBC, MC, LBC and BA we performed no undersampling (note that the bar is higher than the horizontal line).

11.4 Results

In Section 11.4.1 we show that some class labels differed from those provided by the neuroscientists in DeFelipe et al. [2013] and illustrate reconstruction issues that require care when choosing and computing morphometrics. Section 11.4.2 presents the classification results, while Section 11.4.3 shows that accurate models classified MC cells in accordance with the independent classification by the neuroscientists from DeFelipe et al. [2013]. In Section 11.4.4 we provide quantitative descriptions of the types, in terms of only a few morphometrics or parsimonious CART and logistic regression models.

11.4.1 Validating class labels and morphology reconstructions

For eight out of 20 cells which were also classified by 42 neuroscientists in DeFelipe et al. [2013] our class label differed from that given by the majority of the neuroscientists (see Table 11.3 and Figure 11.5, left). There was no strong consensus on the actual type for these cells among the neuroscientists, although cells C050600B2, C091000D-I3, and C170998D-I3 were LBC, CB, and CB, respectively, according to at least 19 of them. For $\frac{5}{19} = 26\%$ of the considered cells no more than five neuroscientists agreed with our class label[11], suggesting that there might have been many such differing class labels had we been able to compare them for the entire data set.

Interestingly, the interneurons could be separated into two groups, one with its cells' arbors reconstructed at a finer level —with shorter and thinner segments— than those of the other (see Figure 11.5, right). We thus avoided using morphometrics sensitive to such fine-grained properties (e.g., the number of segments per branch). This difference may have, however, distorted metrics such as tortuosity, since finer reconstructed branches were more tortuous; see Section B.1.1. 84 cells had at least one multifurcation (a branching point splitting into three or more child branches; at most ten in a single neuron) yet their effect was minimal as we ignored these branching points when computing bifurcation morphometrics, such as mean partition asymmetry or mean bifurcation angle. Two cells seemed to be modified clones of other cells; see Section B.1.2 for details. We only found two reconstruction anomalies: a 285 μm long segment (whereas median length was 2 μm), and two axonal arbors extremely flat in the Z dimension (less than 80 μm deep while median depth was 215 μm; ratio of depth to axonal length was below $\frac{1}{100}$ while median ratio was $\frac{1}{62}$). We did not correct these issues nor we removed the corresponding neurons.

11.4.2 Classification

Table 11.4 shows the best F-measure results for the eight classification settings. The most accurately classified classes were BA, MC, and NBC (shown in green), each with an F-measure ≥ 0.80, while classifying ChC and BTC cells was hard (best F-measure 0.49 and 0.44, respectively). The best model for MC performed better than the average neuroscientist in DeFelipe et al. [2013] when identifying MC cells, as their average F-measure was 0.72[12]. In general, more numerous classes were classified more accurately (F-measure tends to increase towards the bottom rows of Table 11.4), with the exceptions of LBC, which was the third hardest to classify despite being the second most numerous, and BTC, which was the hardest type to classify yet only second least numerous.

Sampling improved the performance of most classifiers, although the largest increase in best F-measure was only 0.03, for the ChC, NBC, and MC types (see Table 11.4, rows 2, 18, and 24). Feature selection increased the best F-measure for BA, DBC, MC, and especially for BTC and SBC (Table 11.4, rows 7 and 15). RW BVI selected much smaller sets of morphometrics (e.g., 7 for SBC; Table 11.4, row 15) than KW (up to 68, for BA; Table 11.4, rows 31-32), allowing, for example, to accurately classify NBC cells using just 9 morphometrics (Table 11.4, row 15). Further feature pruning by the CART and RMLR models after KW produced parsimonious and accurate models, such as the RMLR model for

[11]We are ignoring cell C040600B2, which was shown to the neuroscientists rotated upside-down (this may have affected how they classified it), hence five out of 19 and not six out of 20.

[12]This value was not reported in DeFelipe et al. [2013]; instead we computed it from data from that study, taking into account only cells that could be clearly classified into a type. See Section C.1 details.

Table 11.3: Disagreement with our class labels by 42 neuroscientists who participated in DeFelipe et al. [2013]. Cell type is the label in our data, given according to the classification scheme from Markram et al. [2004] while DF (standing for DeFelipe) is the majority label chosen by the neuroscientists, according to the scheme from DeFelipe et al. [2013]. Agree is the number of neuroscientists that coincided with our label, while columns to the right show the number of neuroscientists who selected the corresponding DF label (all shown in boldface): AR - arcade; CB - common basket; CR - Cajal-Retzius; CT - common type; HT - horse-tail; OT - other; UN - uncharacterized, meaning that the axonal morphology reconstruction was not sufficient to distinguish the type. The table shows eight out of the 20 interneurons which were classified as ChC, MC, or NGC —the three types common to both classification schemes— in our data yet differently by the majority of neuroscientists (column DF); for the remaining twelve interneurons, the neuroscientists' majority label matched ours. Cell C040600B2, which was presented to the neuroscientists rotated upside-down, is marked in blue. ID can be used to look the neuron up at Neuromorpho.org.

	ID	Layer	Cell type	DF	Agree	**AR**	**CB**	**ChC**	**CR**	**CT**	**HT**	**LBC**	**MC**	**NGC**	**OT**	**UN**
1	C040600B2	2/3	MC	CT	0	3	9	0	0	15	2	5	0	0	5	3
2	C050600B2	2/3	MC	LBC	1	0	5	0	0	10	1	20	1	0	2	3
3	C150600B-I1	2/3	MC	CT	1	1	11	0	0	16	0	9	1	0	3	1
4	C091000D-I3	5	ChC	CB	3	3	19	3	0	6	0	6	0	2	2	1
5	C260199A-I3	4	MC	CT	3	0	5	0	0	17	0	6	3	0	4	7
6	C170998D-I3	2/3	NGC	CB	5	1	19	0	0	11	0	0	0	5	4	2
7	C070600B2	4	MC	LBC	11	2	1	0	0	8	0	15	11	0	2	3
8	C090997A-I2	4	MC	CT	12	1	6	0	0	14	0	4	12	0	1	4

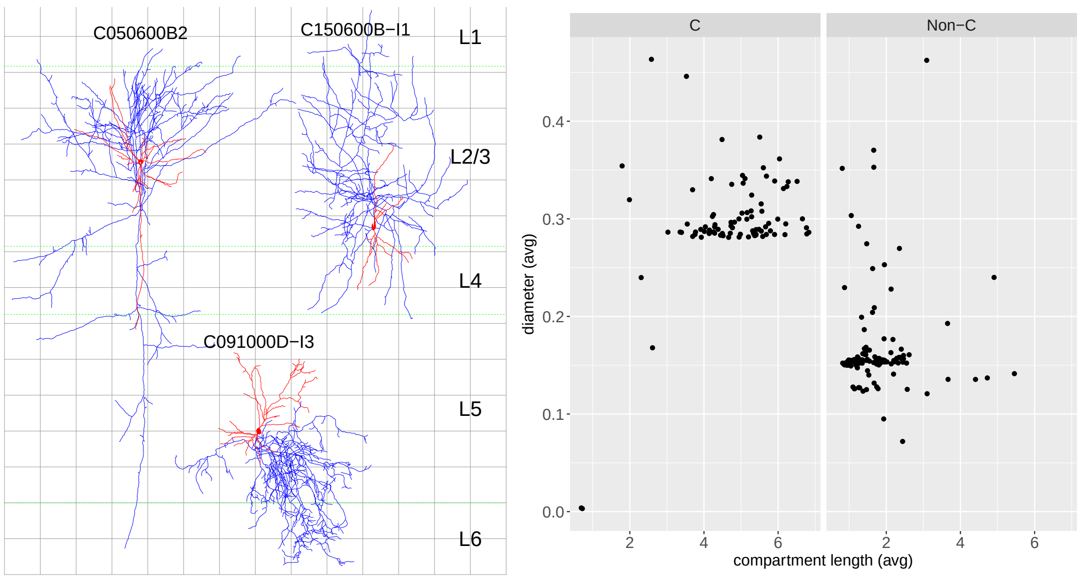


Figure 11.5: Possible class label and reconstruction issues. Left panel: cells C050600B2 (left), C091000D-I3 (middle), and C150600B-I1 (right) from Table 11.3, labelled as MC and ChC, respectively, yet only one, three, and one (out of 42) neuroscientists in DeFelipe et al. [2013], respectively, coincided with those labels, assigning them instead to the LBC, CB, and CT types. Note that we did not know the location of soma inside their layers; for the MC cells, a soma closer to L1 would mean more extensive axonal arborization in that layer. Axons are drawn in blue and dendrites and somata in red. Dashed green lines indicate layer boundaries from the rat hind-limb somatosensory cortex; L6 only partially shown. There are 100 μm between consecutive grid lines. Right panel: newer reconstructions, whose ids do not begin with a C, had thinner and shorter segments.

MC (with an F-measure of 0.80 and 22 morphometrics; Table 11.4, row 23). See Figure B.3 to Figure B.10 for detailed per-type graphs of classification performance, broken down by classification, feature selection and sampling method.

Table 11.4: F-measure one-versus-all classification. The table shows, for each type, the best F-measure in all four learning settings: with and without sampling, and with and without feature selection. TPR: true positive rate; TNR: true negative rate; the minority class is always the positive one, except for BA; Acc: classification accuracy; Morphom.: the number of morphometrics in the model. Types are sorted from least to most frequent (e.g., ChC, with only seven examples, is shown uppermost). The best F-measure for each type is typeset in bold. Types with their best F-measure ≥ 0.75 are shown in green; those with the F-measure ≥ 0.60 in orange; and the rest in red.

	Cell Type	Classifier	FSS	Sampling	F-measure	TPR	TNR	Acc	Morphom.
1	ChC	RMLR			0.40	2 / 7	209 / 210	0.97	11
2		RF		Yes	**0.49**	2.8 / 7	208.5 / 210	0.97	103
3		LDA	KW		0.46	3 / 7	207 / 210	0.97	15
4		RF	KW	Yes	0.46	3.5 / 7	205.2 / 210	0.96	15
5	BTC	NB			0.35	8 / 15	179 / 202	0.86	103
6		RMLR		Yes	0.35	6.7 / 15	185.8 / 202	0.89	23
7		LDA	KW		**0.44**	6 / 15	196 / 202	0.93	7
8		LDA	KW	Yes	0.40	8.8 / 15	181.8 / 202	0.88	7
9	DBC	RMLR			0.70	15 / 22	189 / 195	0.94	17
10		RF		Yes	0.70	14.6 / 22	189.8 / 195	0.94	103
11		RF	RF BVI		**0.72**	13 / 22	194 / 195	0.95	6
12		RF	KW	Yes	0.70	15.4 / 22	188.2 / 195	0.94	61
13	SBC	CART			0.63	16 / 28	182 / 189	0.91	5
14		RF		Yes	0.66	20.6 / 28	174.8 / 189	0.90	103
15		kNN	RF BVI		**0.73**	20 / 28	182 / 189	0.93	7
16		RF	RF BVI	Yes	0.69	22.5 / 28	173.8 / 189	0.90	7
17	NBC	CART			0.73	32 / 44	161 / 173	0.89	4
18		RF		Yes	**0.81**	36.2 / 44	164 / 173	0.92	103
19		RF	RF BVI		0.78	33 / 44	165 / 173	0.91	9
20		RF	RF BVI	Yes	0.76	36 / 44	158.6 / 173	0.90	9
21	MC	SVM			0.77	37 / 50	158 / 167	0.90	103
22		RF		Yes	0.81	40.2 / 50	158.4 / 167	0.92	103
23		RMLR	KW		0.80	38 / 50	160 / 167	0.91	22
24		RF	KW	Yes	**0.82**	40.9 / 50	157.8 / 167	0.92	62
25	LBC	RF			0.56	21 / 51	163 / 166	0.85	103
26		RF		Yes	**0.67**	29.8 / 51	157.4 / 166	0.86	103
27		CART	RF BVI		0.64	30 / 51	153 / 166	0.84	4
28		SVM	RF BVI	Yes	0.66	38.1 / 51	140 / 166	0.82	4
29	BA	RF			0.86	106 / 123	76 / 94	0.84	103
30		SVM		Yes	0.86	101.9 / 123	80.8 / 94	0.84	103
31		SVM	KW		**0.88**	105 / 123	84 / 94	0.87	68
32		SVM	KW	Yes	**0.88**	104.2 / 123	84.2 / 94	0.87	68

11.4.3 Validating the MC models

We validated the two most accurate models for MC —RF with sampling and RMLR, both preceded by KW feature selection (see Table 11.4, rows 22–24)—, by comparing their output to the classification by the neuroscientists from DeFelipe et al. [2013], which was not used to train the models.

As Table 11.5 shows, the models largely agreed with the neuroscientists in DeFelipe et al. [2013]. Cells that were considered MC by 13 or less neuroscientists (upper part of Table 11.5) were also rarely classified as MC by our models, with cells C050600B2, C260199A-I3, and C230998C-I4 never labelled as MC by either model. Both models disagreed with the neuroscientists on cells C040600B2 and C090997A-I2 —the former was, however, shown to the neuroscientists rotated upside-down, which may have yielded so few votes for MC— and RF disagreed on cell C150600B-I1, considering it MC 22 out of 30 times. On the other hand, cells that were MC according to 14 or more neuroscientists (lower part of Table 11.5) were always classified as MC by the models, except for C061000A3, which RMLR never classified as MC.

Figure 11.6 shows the four cells that were considered MC at most six (out of 30) times by both RF and RMLR. These include the cells C050600B2, C260199A-I3, C230998C-I4 (shown in red in Table 11.5), classified as MC by only one, three, and 13 neuroscientists, respectively. These cells may correspond to atypical MC morphologies.

11.4.4 Feature selection

For all types except for ChC and BTC, we achieved at least moderately accurate (F-measure ≥ 0.65) models using only few morphometrics (see Table B.2). Below we describe the BA, NBC, DBC, SBC, and SBC types in terms of the morphometrics selected with RF BVI, and the MC type in terms of those selected with KW followed by CART and RMLR embedded feature selection (this yielded more accurate models for MC than RF BVI). We also describe the BA and MC types in terms of accurate (F-measure ≥ 0.75) and parsimonious CART and logistic regression (RMLR) models. Finally, we complement each type description with some of the best-ranked morphometrics according to the KW test, and conclude in Section 11.4.4.6 with a summary of feature selection. We begin with the most accurately classified type, BA (Section 11.4.4.1), and proceed towards the least well discerned ones, ChC and BTC (Section 11.4.4.5). See Table B.4 and Table B.5, respectively, for the full list of KW- and RF BVI-selected morphometrics, along with the corresponding p-values and RF BVI values.

11.4.4.1 BA characteristics

Six axonal morphometrics selected by RF BVI (Figure 11.7) sufficed to accurately (with an F-measure of 0.86) distinguish BA cells. These morphometrics captured two properties only: remote branching angle and arborization distance from soma. Indeed, BA cells had sharper remote bifurcation angles and arborized closer to the soma, especially in terms of vertical distance (Figure 11.7). While LBC cells can extend vertically far from the soma (Markram et al. [2004], Wang et al. [2002]; their average height in our sample was 1020 $\mu m \pm$ 327 μm, versus 603 $\mu m \pm 190\,\mu m$ for the NBC and SBC together), it seems that most of their arbor is nonetheless located near the soma, with radially distant ramifications being rather sparse. The CART and RMLR models derived from the six RF BVI-selected morphometrics were accurate (F-measure of 0.85 and 0.83, respectively) and interpretable (e.g., Toledo-Rodriguez et al. [2005] used CART to relate mRNA expression to neuro-anatomical type). The CART model, for example, is a set of rules such as "all cells with `path_dist.avg` < 414 and `y_mean_abs` < 133 are BA cells". Both models are described in Figure 11.8.

The KW test identified additional 63 morphometrics, including 26 dendritic ones, that differed between the BA and non-BA cells, yet using them barely improved the F-measure achieved with the six RF BVI-selected morphometrics alone (from 0.86 to 0.88). Interestingly, the number of dendritic trees was among the most relevant morphometrics, with BA cells having more dendritic trees than non-BA ones (Figure 11.7). Although some basket cells have curved axon terminals [Ascoli et al., 2008], `t.tortuosity.avg` was only 47-th most relevant morphometric according to KW, suggesting that we may need a more appropriate morphometric to capture the curved property of basket terminal branches. On the other hand, axonal properties that did not differ for BA cells included average branch length, arbor length and initial direction (whether towards pia or the white matter).

11.4.4.2 MC characteristics

The six morphometrics selected by CART (following KW selection) allowed for classifying MC cells with an F-measure of 0.75. According to this model, a typical MC cell's axon arborized far above the soma (`y_mean`), widely in layer L1, and bifurcated in wide angles. The model is described in Figure 11.9. Using 22 morphometrics, including seven dendritic ones, KW + RMLR was more accurate (F-measure of 0.80) and uncovered additional MC properties, such as longer dendritic trees, displaced from axonal arbors, which in turn were moderately radial (see Figure 11.10). This agrees with Markram et al. [2004] and Wang et al. [2004], who reported elaborate dendrites, 1013 ± 503 μm axonal width in L1, and average tilt angles of 80 degrees. It also contrasts with the above description of BA cells, which arborized vertically close to soma, had shorter bifurcation angles, and many dendritic trees. This is illustrated in Figure 11.10, which plots MA, BA and all other types using the two most useful morphometrics for BA.

KW selected 40 additional morphometrics, including 17 dendritic ones, with the strongest difference for `path_dist.avg` and `y_mean` (see Table B.4). MC cells often had bitufted dendrites (also reported by Markram et al. [2004]) and axons originating above the soma.

11.4.4.3 NBC characteristics

Nine axonal morphometrics selected by RF BVI allowed an accurate (F-measure 0.78) classification of NBC cells (see Figure 11.11). Six of those morphometrics were related to arborization distance from soma; the rest to translaminar reach, branch length, and arbor density.

KW identified a larger and more diverse set of 48 morphometrics, including 21 dendritic ones, that differed for NBC cells (see Table B.4), yet using all of them slightly decreased performance with respect to using only the nine RF BVI-selected morphometrics (F-Measure from 0.78 down to 0.75). In addition to arborization distance from soma and translaminar reach, relevant morphometrics included axonal terminal degree, arbor eccentricity, partition asymmetry, terminal branch length, and whether the dendrites were bitufted.

11.4.4.4 DBC, SBC and LBC characteristics

DBC cells were classified with moderate accuracy (F-measure 0.72) with the five morphometrics selected by RF BVI, all related to axonal arbor eccentricity, distribution along the Y axis, and width (see Figure 11.12). While KW identified 61 significantly different morphometrics for DBC —more than for SBC, NBC, and LBC, even though these were more numerous than DBC— using all of those morphometrics did not improve DBC classification (F-measure dropped to 0.70). The most relevant ones were related to the radial arborization of both the axonal and the dendrites (Figure 11.12). Interestingly, KW selected more (26) dendritic morphometrics for DBC than for any other type.

For SBC we achieved an 0.73 F-measure with the seven RF BVI-selected morphometrics, related to mean branch length, arbor density, and arborization distance from soma (see Figure 11.12). KW selected 39 morphometrics, although using them did not improve with respect to just RF BVI-selected ones (F-measure from 0.73 down to 0.67). Relevant morphometrics included `y_sd`, related to radial arborization extent, and the maximal arborization distance from the soma (`euclidean_dist.max`).

LBC cells were classified with an F-measure of 0.66 with the four morphometrics selected with RF BVI, related only to remote bifurcation angles and arborization distance from soma (see Figure 11.12). According to KW, remote bifurcation angle was the most significant morphometric, with a p-value of 3.7×10^{-8}, followed by remote tilt angle, median terminal branch length, `grid_area` and the number of dendrites (see Table B.4). KW identified only 32 relevant morphometrics for LBC, much less that for other numerous types; using all these morphometrics reduced the best F-measure to 0.62.

11.4.4.5 BTC and ChC characteristics

For BTC, only seven morphometrics were relevant according to KW, with dendritic polarity and the standard deviation of branch length (`length.sd`), among the most significant ones. For ChC, the relevant properties according to KW included arbor density (`density_bifs`, `grid_mean`), mean branch length, the number of short vertical branches, and terminal degree.

11.4.4.6 Summary

KW identified more relevant morphometrics for the more numerous types, with the exceptions of LBC (second most numerous, yet only sixth most features) and DBC (sixth most numerous, yet third most features). Dendritic morphometrics represented 30-40% of the relevant ones, except for ChC (a single dendritic morphometric out of seven relevant ones; see Table B.4). 11 dendritic and four axonal morphometrics were not relevant for any type, and are possibly useless for interneuron classification: dendritic bifurcation angles, tortuosity, and radial and tangential arbor distribution, and axonal torque angle and tangential arbor distribution. Dendritic tree length and `d.displaced`, however, were relevant for six out of eight types. Custom-implemented morphometrics represented between 47% and 72% of the selected morphometrics. Only two custom-implemented morphometrics (`ratio_x` and `x_mean_abs`) were not useful for any type, while `translaminar` and `y_sd` were relevant for six types.

11.5 Discussion

We obtained accurate models for the NBC, MC, and BA types and moderately accurate ones for DBC, SBC, and LBC. The best MC model was better than the average neuroscientist in DeFelipe et al. [2013] and was outperformed by only three out of 42 of them (see Section C.1). The best BA model was even more accurate, correctly identifying 105 out of 123 BA cells (see Table 11.4). These models, along with the model for NBC, would probably be useful for the definitive automatic classifier envisioned by DeFelipe et al. [2013] to replace neuroscientists in this task. The remaining models were probably not good enough: the next best model correctly identified only 20 out of 28 SBC cells (see Table 11.4). The main limiting factor seems to have been sample size: with the exception of LBC, more numerous types were classified more accurately; indeed, we only had 28 SBC, 22 DBC, 15 BTC and seven ChC cells. Taking sample sizes into account, moderate F-measure values suggest that the DBC and SBC types are morphologically distinct and we expect that around 50 cells (a count close to that of NBC and MC cells) would suffice to accurately classify them. The LBC type was relatively hard to classify. Either we have missed to quantify its distinctive morphometrics —there were less relevant morphometrics for LBC than for other numerous types— or its morphology is not sufficiently distinct when contrasted to the other types merged together. Distinguishing across layers (e.g., L2/3 LBC, L4 LBC, etc.) might decompose it into morphologically distinct subtypes.

One explanation for the differences between our class labels and the classification from DeFelipe et al. [2013] shown in Table 11.3 is that ours were ultimately determined by the presence of spiny boutons and dendritic spines (MC), short vertical rows of boutons (ChC), or a high density of small boutons (NGC). Indeed, for Wang et al. [2004], spiny boutons, along with axonal spread in L1, are an essential (mandatory) characteristic of MC cells. Yet, ChC, MC and, to a lesser degree, NGC morphologies are often identifiable by axonal and dendritic geometry alone [DeFelipe et al., 2013] suggesting that their arborization patterns are distinct. Thus, while cells in Table 11.3 might be meeting fine-grained criteria for MC, ChC, and NGC membership, their high-level morphologies are atypical, as most of the 42 neuroscientists considered that they did not belong to those types. It is hard for a model to correctly classify such cells, unless some morphometrics are correlated with the fine-grained features. Thus, there might be a limit to how well the classification by Markram et al. [2004] could be replicated by a model trained on morphological reconstructions. However, even when the MC models failed to recover the class label, their output may have been sensible, as it was often consistent with the classification by the 42 neuroscientists (see Table 11.3). MC cells classified as not MC by accurate models might thus correspond to atypical MC morphologies.

An alternative, yet less likely, explanation for the difference is that some class labels had been wrongly assigned, without following the pre-specified criteria. In that case, wrong labels biased the models as well as their performance estimates [Lam and Stork, 2003]. Instead of assuming that all class labels are correct, as we did, they can be estimated together with classifier learning (Frénay and Verleysen, 2014), although this makes the learning problem harder.

Additional morphometrics might further improve the results. We consider that quantifying Petilla features related to arborization patterns would be useful, especially for scarce types such as ChC. Some of our custom-implemented morphometrics may have been too simple (e.g., only branches extending no more than 50 μm vertically were considered short and vertical) to adequately capture the complexity of these features, and could be elaborated. Type-specific morphometrics, such as the extent of axonal arborization in layer L1 for MC cells, incorporated prior knowledge about the types into the models. Note that such underlying knowledge may be disputed: e.g., DeFelipe et al. [2013] do not require an MC cell to reach layer L1, Wang et al. [2004] consider it an essential, mandatory feature, and so do Markram et al. [2015], except for L6 MC cells. It would be interesting to study the robustness of standard morphometrics to reconstruction issues such as inconsistent branch granularity and then develop robust alternatives. For example, `t.tortuosity.avg` might have better captured the 'curved terminal branches' feature of the BA type had not some cells' branches been reconstructed in finer detail than those of others, thus increasing their tortuosity (see Section B.1.1). While at least 21 analyses available in L-Measure would have not been robust to reconstruction granularity inconsistency in this data set, they are nonetheless used for neuron classification [e.g., Vasques et al., 2016]. Thus, a software tool that implements robust morphometrics could be useful for practitioners.

The small feature subsets and parsimonious models that allowed for (moderately) accurate classification serve as summaries of the types' morphological characteristics. Most types can be summarized in terms of simple morphometrics, related to arborization distribution with respect to the soma (e.g., `path_dist.avg`), its vertical direction (e.g., `y_std_mean`), branching angles (`remote_bifurcation_angle.avg`), or the number of dendrites(`d.N_stems`), and a few elaborate ones, such as the arborization extent in L1 (`l1_width`). It is possible that the simple morphometrics would not suffice if distinguishing multiple types at once, as the decision boundaries would presumably be more complex.

We have presented eight separate type-specific models. Combining them to classify a given interneuron involves choosing the type with the most confident one-versus-all model. An alternative is to learn a hierarchy of classifiers by grouping types into 'super types' such as BA: one would first classify a cell as BA or non-BA and then, if classified as BA, distinguish among LBC, NBC, and SBC types, and among the remaining types otherwise. Rather than learning the hierarchy from data, one might predefine it; useful 'super-types' could be formed, for example, by grouping according to axonal target area — a dendrite-targeting type would be composed of BP, BTC, DBC and NGC cells [Markram et al., 2004].

Note that we have learned the models from juvenile rat somatosensory cortex interneurons and they might be less effective if applied to classify other species' or brain area cells, especially because metric variables, such as those related to distances from the soma and arbor size, are affected by these factors. Doing so would also require appropriate laminar thickness metadata in order to quantify laminar extent. The presented supervised classification approach could easily be extended to allow the discovery of new types: since models such as logistic regression can quantify the confidence in their prediction, one could consider discovering types by clustering [Jain, 2010] cells that the model cannot reliably assign to any of the *a priori* known types.

11.6 Conclusion

We used 217 high-quality morphology reconstructions of rat interneurons to learn models for eight interneuron types. We have proposed and implemented morphometrics that quantify relevant interneuron properties such as laminar distribution and arbor extent in L1, dendritic polarity, arbor orientation, and whether the dendrites are displaced from the axon. We carefully selected standard metric and topological morphometrics, omitting those not robust to reconstruction granularity. We applied state-of-the-art classification algorithms and learned accurate models for the BA, MC, and NBC types, with F-measure values above 0.80, which can compete with neuroscientists, and moderately accurate (F-measure above 0.70), for the DBC and SBC types, although we had less than 30 cells of the latter two types. We characterized the types in terms of parsimonious CART (for BA and MC) and logistic regression (for BA) models that can be interpreted by neuroscientists, and of small sets of relevant morphometrics: no more than nine morphometrics sufficed for an at least moderately accurate classification of the DBC, SBC, NBC, MC and BA types. Most relevant morphometric were related to axonal arborization distance from the soma and bifurcation angles while most dendritic ones were not. Differences between our class labels and those by 42 leading neuroscientists from DeFelipe et al. [2013] suggest that it might be hard to perfectly replicate the classification by Markram et al. [2004] without access to fine-grained morphological features. However, even when failing to recover the original label, the models' output seemed sensible as it often matched the classification by 42 leading neuroscientists. We computed all the morphometrics with open-source software and our code and data are publicly available. This study showed that with quality reconstructions, a careful selection of morphometrics and an informed machine learning approach, accurate models can be learned from relatively few examples. We speculate that 50 cells could suffice for learning accurate models for the DBC and SBC types. This study also illustrated minor reconstruction present in a curated set of high-quality morphologies.

Achieving accurate automatic classification for the all established morphological types will require more labeled interneurons to train the models with, especially for scarce types such as ChC. In the short term, this may require leveraging the reconstructions from Neuromorpho.org. Automated checks of morphology, such as those performed by NeuroSTR (e.g., whether a bifurcation angle is too wide to be plausible), could help filter useful reconstructions, while developing morphometrics robust to different types of variability (e.g., in reconstruction granularity) might facilitate combining diverse data. Aggregating cells labeled in different laboratories could be problematic if these class labels had been assigned following different criteria, and the labels might need to be validated by multiple neuroscientists. Classification criteria that give importance to fine-grained morphological features, such as bouton distribution, would imply a limit to attainable classification accuracy, unless we can discover morphometric correlates of such features. Finally, morphometrics that quantify complex arborization patterns could be especially useful for the less numerous types. In the long run, we expect efforts by the Human Brain Project, the Allen Institute for Brain Research, and NeuroMorpho.Org to provide many high-quality morphologies. Given such data, we consider that the methodology presented in this chapter can provide an accurate automatic classification into established morphological types.

Table 11.5: Classification of MC cells by the neuroscientists in DeFelipe et al. [2013] and our two most accurate models, RF and RMLR. MC is the number of neuroscientists who classified the cell as MC, Non-MC the number of those who assigned it to another type, and UN the number of those who considered that the axonal morphology reconstruction was not sufficient to distinguish the type. RF and RMLR show the number of times (out of 30) that RF and RMLR classified the cell as MC. Cells that were never classified as MC by both models are marked in red. Cell C040600B2, which was presented to the neuroscientists rotated upside-down, is marked in blue. ID can be used to look the neuron up at Neuromorpho.org.

	ID	Layer	RF	RMLR	MC	Non-MC	UN
1	C040600B2	L2/3	29	23	0	39	3
2	C050600B2	L2/3	0	0	1	38	3
3	C150600B-I1	L2/3	22	1	1	40	1
4	C260199A-I3	L4	0	0	3	32	7
5	C070600B2	L4	12	0	11	28	3
6	C090997A-I2	L4	30	19	12	26	4
7	C230998C-I4	L4	0	0	13	25	4
8	C190997A-I1	L4	30	30	14	26	2
9	C290500B-I3	L2/3	30	30	18	20	4
10	C150501A-I3	L5	30	30	22	6	14
11	C060400C1	L2/3	30	30	24	18	0
12	C290500C-I4	L5	30	30	26	15	1
13	C061000A3	L4	30	0	34	6	2
14	C100501A3	L2/3	30	30	36	3	3
15	C050896A-I	L5	30	30	37	4	1
16	C070301B2	L6	30	30	37	4	1
17	C180298B-I3	L5	30	30	38	3	1

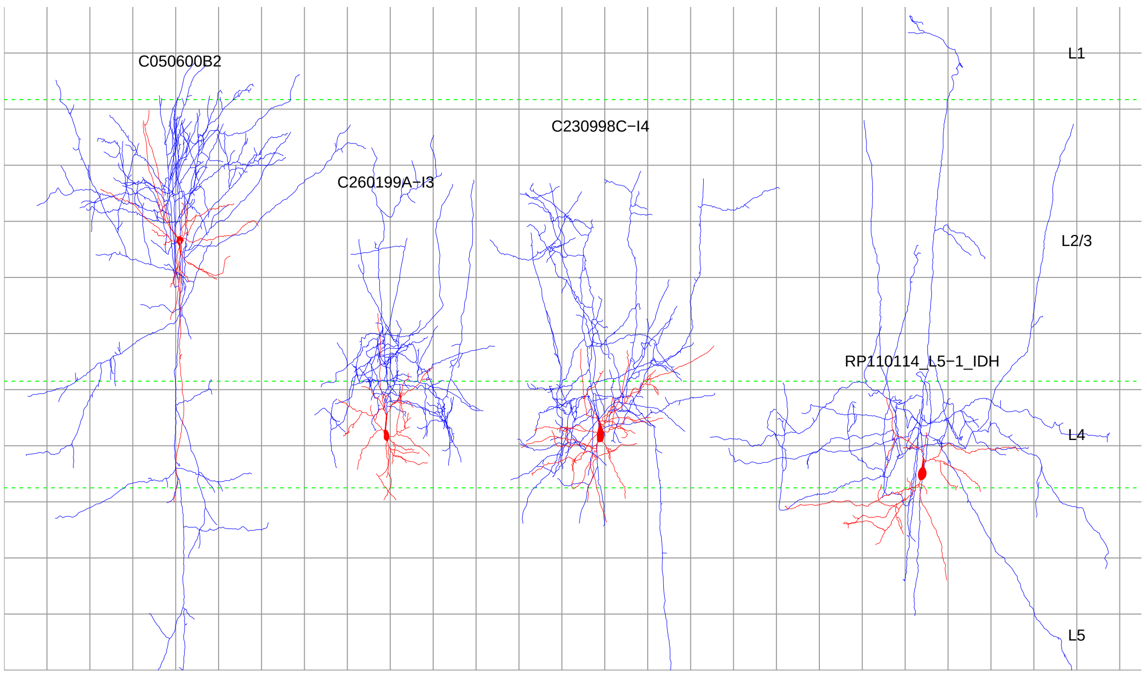


Figure 11.6: MC cells that were classified as non-MC by the two most accurate models. Cells C050600B2, C260199A-I3, and C230998C-I4 were classified as MC by only one, three, and 13 neuroscientists in DeFelipe et al. [2013], respectively. Cells C260199A-I3 and C230998C-I4 do not reach L1 unless their actual soma was located near the top of L4 cell, although tissue shrinkage may have reduced their height for around 10%. Axons are drawn in blue and dendrites and somata in red. Dashed green lines indicate layer boundaries from the rat hind-limb somatosensory cortex. There are 100 μm between consecutive grid lines.

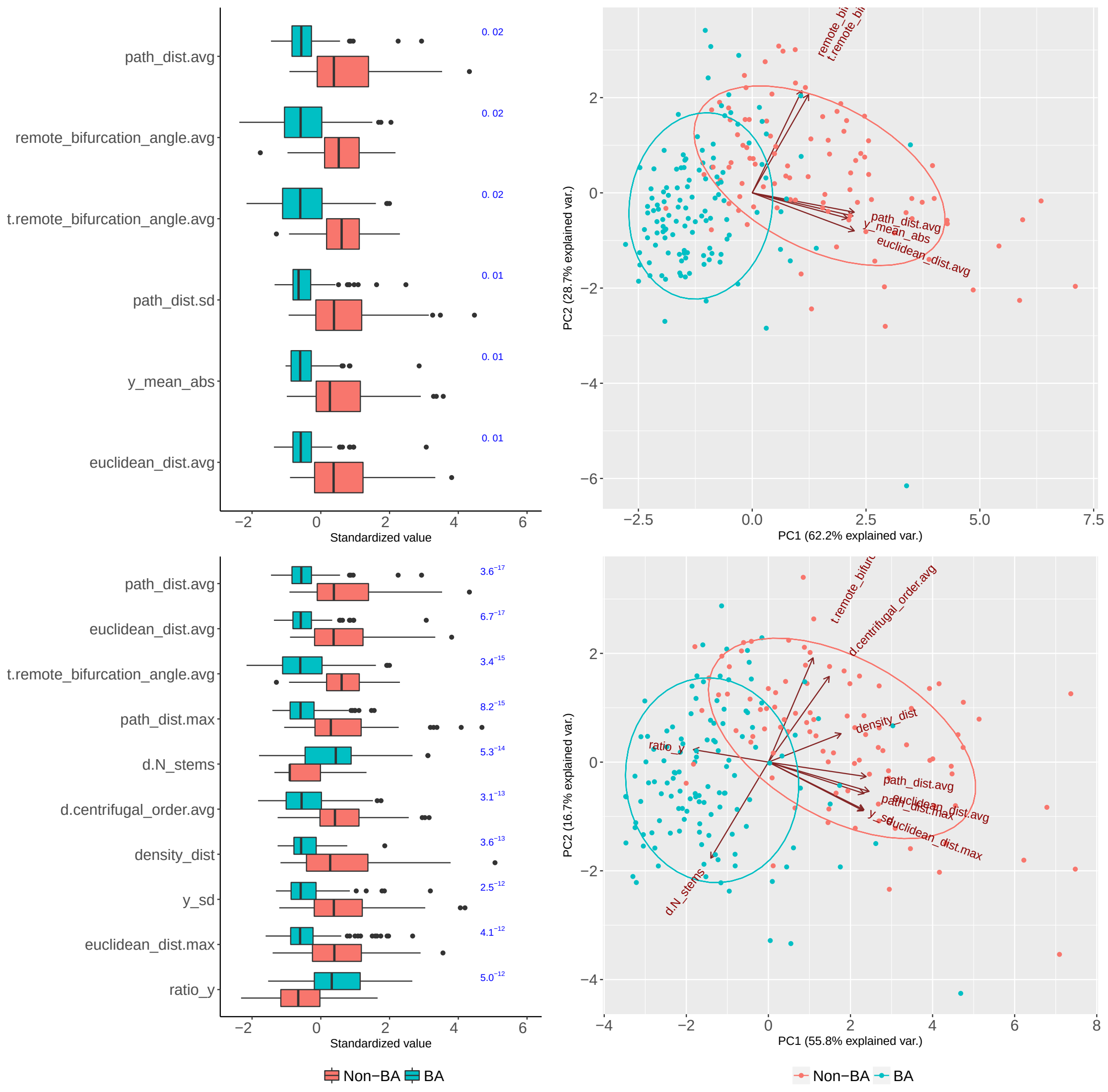


Figure 11.7: Relevant morphometrics for the BA type. Top left: per-type boxplots for the six morphometrics selected with RF BVI (RF BVI values shown, in blue, to the right). The most relevant morphometrics, mean arborization distance to soma (`path_dist.avg`), and mean remote bifurcation angle (`remote_bifurcation_angle.avg`), are shown in the upper part of the panel. Top right: a biplot of these six morphometrics, with the data projected onto the two principal components, found with principal component analysis (vectors represent morphometrics and the angles between them are indicative of their pairwise correlation). All morphometrics were correlated with either `path_dist.avg` or `remote_bifurcation_angle.avg`. Bottom left: the ten most relevant morphometrics according to KW, after removing those with absolute correlation ¿ 0.90 with a better ranked morphometric, with the KW p-values shown, in blue, to the right of the boxplot. These morphometrics included those relative to arborization distance from soma (e.g., `euclidean_dist.avg`, `path_dist.avg`), remote bifurcation angles (`t.remote_bifurcation_angle.avg`), the number of dendritic trees (`d.N_stems`), and axonal arborization along the radial direction (`ratio_y`). In addition to having sharper bifurcation angles and arborizing closer to the soma, especially in the radial direction, BA cells had more dendritic trees than non-BA cells.

Morphometric	β
`remote_bifurcation_angle.avg`	-2.1×10^{-1}
`euclidean_dist.avg`	-1.2×10^{-2}
`y_mean_abs`	-3.3×10^{-3}
`path_dist.sd`	-1.5×10^{-3}
`path_dist.avg`	-2.0×10^{-4}

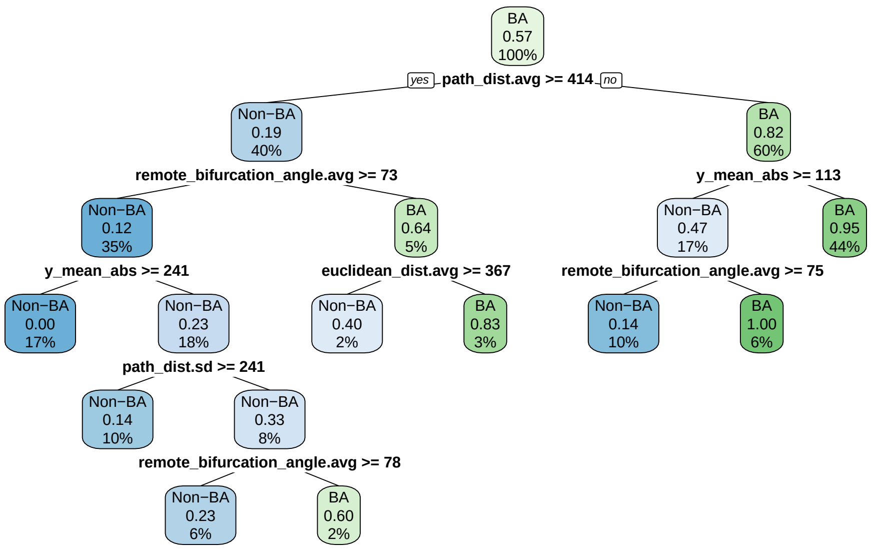


Figure 11.8: Logistic regression (top) and CART models (bottom) for BA derived from the six morphometrics selected with RF BVI. Top: the table shows the logistic regression (F-measure of 0.83), with the $\boldsymbol{\beta}$ estimated from the standardized data set, and BA being the positive class. Interpretation is straightforward; for example, according to the model, a 7.33° increase in the average bifurcation angle of a cell reduce the log-odds of BA by 0.21. Bottom: CART tree for BA, with an F-measure value of 0.85. Most of the BA cells (i.e., those contained in the two rightmost tree leaves) have a `path_dist.avg` ¡ 414 and either `y_mean_abs` ¡ 133 or `remote_bifurcation_angle.avg` ¡ 75°, meaning that they arborize close to the soma, especially vertically, whereas if they do arborize further vertically (as some LBC cells), they have sharper bifurcation angles. Each box represents a split in the data set, indicating its majority type (BA is the majority type overall and hence it is shown in the root node of the tree (i.e., the initial split)), proportion of positive examples (BA cells represent 57% of the data set and hence 0.57 in the root node; they present 95% of the samples in the rightmost node), and the percentage of the data set reaching the split (100% of the data passes through the root split; 44% of the data set reaches the rightmost node).

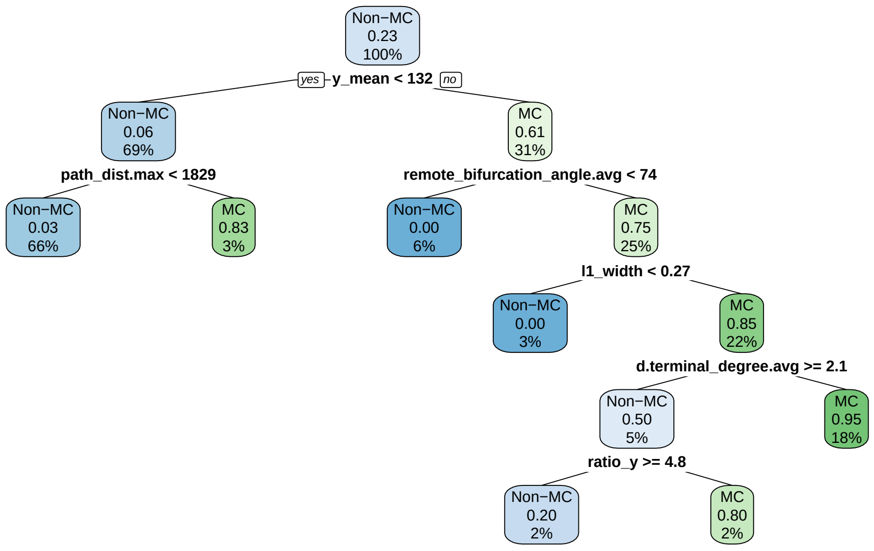


Figure 11.9: CART model for MC, with an F-measure value of 0.75. Most MC cells (rightmost leaf) have a `y_mean` ≥ 132 (their axons mainly arborize above the soma), `remote_bifurcation_angle.avg` $\geq 74°$, `l1_width` ≥ 0.27 and dendritic terminal degree < 2.1. Each box represents a split in the data set, indicating its majority type (Non-MC is the majority type overall and hence it is shown in the root node of the tree (i.e., the initial split), whereas MC is the majority type in the rightmost split), proportion of positive examples (MC cells represent 23% of the whole data set and hence 0.23 in the root node; they present 95% of the samples in the rightmost node), and the percentage of the data set reaching the split (100% of the data passes through the root split; 18% of the data set reaches the rightmost node).

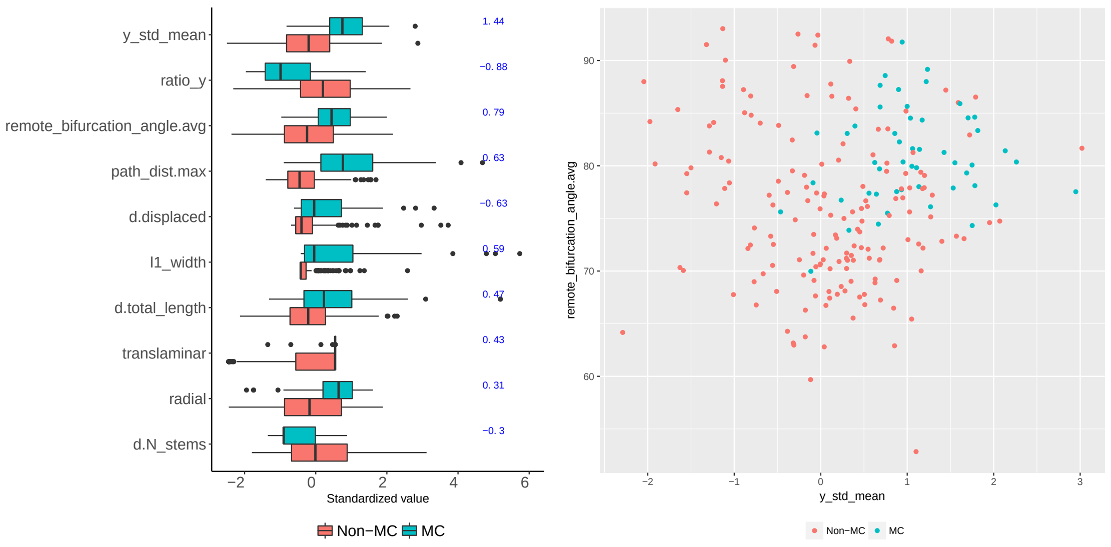


Figure 11.10: Relevant morphometrics for the MC type. Left: ten morhpometrics with strongest $\boldsymbol{\beta}$ in the KW + RMLR model ($\boldsymbol{\beta}$ shown, in blue, to the right of the boxplot; full model in Table B.3). Largely positive `y_std_mean` (top of the boxplot) indicates that MC cells preferentially arborized above the soma. Having longer dendritic arbors (`d.total_length`) yet less dendrites (`d.N_stems`) means that MC cells had longer individual dendritic trees; these arbors were displaced from the axonal ones (`d.displaced`), which were often radially oriented (`radial`). Right: MC cells mainly arborize above the soma (`y_std_mean`) and have wide bifurcation angles (`remote_bifurcation_angle.avg`).

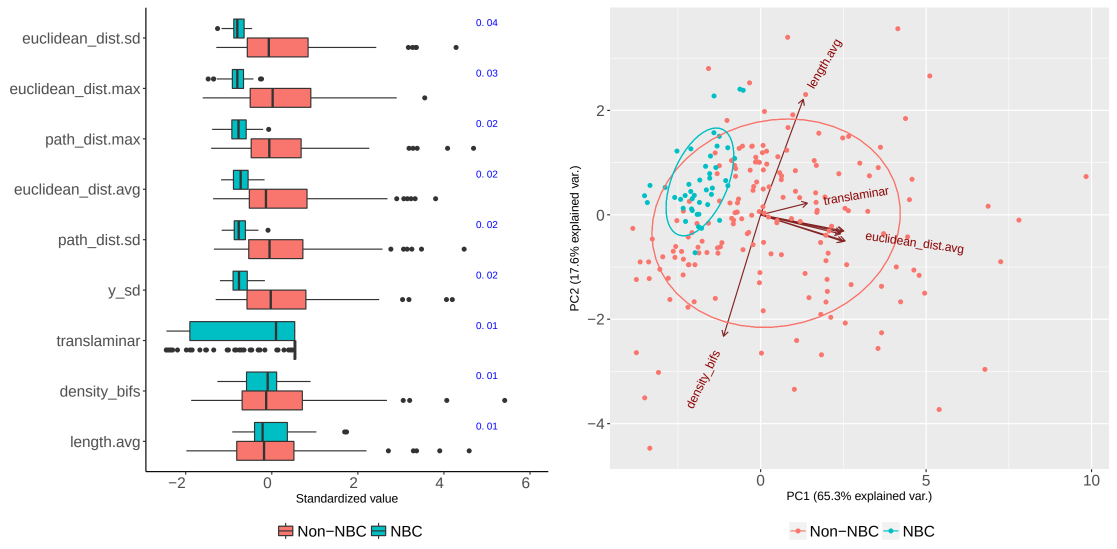


Figure 11.11: Relevant morphometrics for the NBC type. Left: per-type boxplots for the nine morphometrics selected with RF BVI (RF BVI values shown, in blue, to the right). For most NBC cells, the axon never arborized far from the soma (low `euclidean_dist.max`; top part of the panel) nor outside of its cortical layer (low `translaminar`). Although selected by RF BVI, `length.avg` and `density_bifs`, the box-plots (bottom part) show that these morphometrics were not univariately useful. Right: the nine selected morphometrics separate the NBC cells from non-NBC ones. The biplot shows the data projected onto the two principal components, found with principal component analysis, with the vectors representing the morphometrics and the angles between them indicative of their pairwise correlation. Besides branch length (`length.avg`), translaminar reach (`translaminar`), and arborization density (`density_bifs`), all selected morphometrics are related to arborization distance from soma. They correspond to the vectors pointing towards the right; only `euclidean_dist.avg` is annotated to avoid overlapping.).

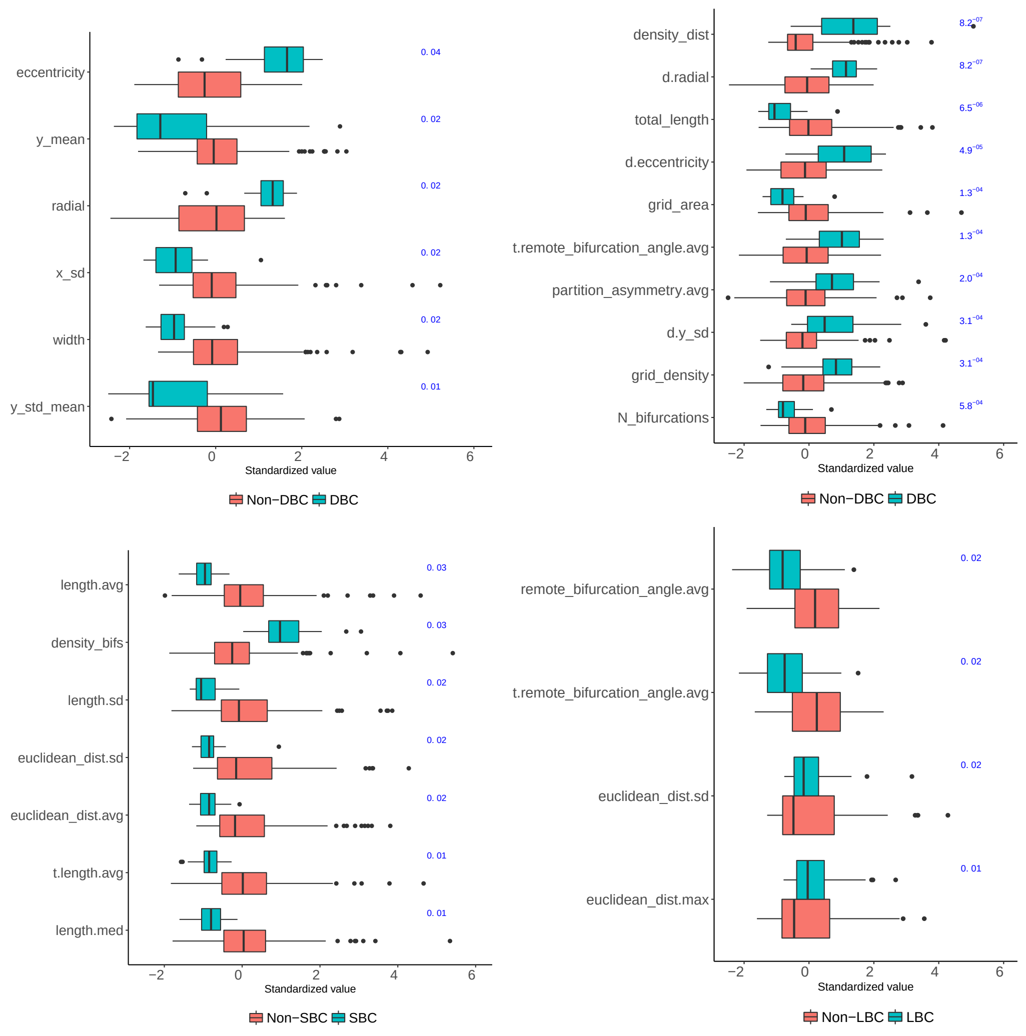


Figure 11.12: Relevant morphometrics for the DBC (above) and SBC and LBC (below) types. Top left: per-type boxplots for the morphometrics selected with RF BVI (RF BVI values shown, in blue, to the right). The axonal arbor of a typical DBC cell was radially oriented (high `radial` and `eccentricity` values), rather than circular, it did not spread far tangentially (low `x_sd` and `width`), and was mainly located below the soma (low `y_std_mean` and `y_mean`). Top right: the ten most relevant morphometrics according to KW, after removing those already shown in the left panel and those with an absolute correlation ¿ 0.90 with a better ranked morphometric (KW p-values shown, in blue, to the right). DBC cells's dendrites were bipolar/bitufted (`d.insert.radial`, not shown), arborized along the radial axis (`d.radial`) and reached far radially (`d.y_sd`), while their axonal arbors were short (`total_length`), with wide terminal bifurcation angles (`t.remote_bifurcation_angle.avg`). Bottom left: per-type boxplots for the morphometrics selected with RF BVI for SBC (RF BVI values shown, in blue, to the right). SBC cells had short branches (low `length.avg`) and dense, local arbors (low `density_bifs` and `euclidean_dist.avg`). Bottom right: per-type boxplots for the morphometrics selected with RF BVI for LBC (RF BVI values shown, in blue, to the right). LBC cells had sharp bifurcation angles.

Part IV

CONCLUSIONS

Chapter 12

Conclusions and future work

We now review our contributions to machine learning and neuroscience.

12.1 Machine learning

We have proposed a method for learning Bayesian network classifiers with equivalence classes. For this, we have specified the smallest DAG subspace that covers all possible class-conditional distributions, presented an algorithm to traverse the equivalence classes in this space, by adapting the greedy equivalence search algorithm, and specified how to efficiently compute the discriminative score of a CPDAG search operator. The algorithm showed promising results on preliminary experiments. We consider that this validates our starting hypothesis: completed partially DAGs can be usefully leveraged for learning Bayesian network classifiers.

We have implemented Bayesian network classifiers in the bnclassify package. The package has been downloaded over 11 thousand times from the RStudio mirror of the Comprehensive R Archive Network (CRAN). There are roughly a thousand downloads per package update, suggesting that there are that many existing installations.

We adapted the mixture model underlying the SeSProc method Guerra et al. [2013b] in order to use an, in our opinion, more intuitive notion of feature relevance.

Finally, we have provided a non-parametric method for multi-dimensional classification with probabilistic labels encoded as Bayesian networks.

12.2 GABAergic interneuron classification

When using all Gardener's interneurons to train and evaluate models, as with with the lowest label reliability threshold in Chapter 8 and in Chapter 9, we only slighly improved interneuron type prediction by DeFelipe et al. [2013] (from 62% to 64% accuracy). Prediction accuracy was also limited with probabilistic labels, independently of any possibly noise introduced by unreliabile majority vote labels. Besides the small size, this data set may be difficult because it is heterogeneous, with neurons from different species, cortical areas and animals of non-uniform age and gender, reconstructed by different laboratories. We did improve the prediction of axonal feature F4 of the Gardener's scheme with respect to DeFelipe et al. [2013] and did not degrade their already good results for features F1, F2, and F3.

Our supervised and semi-supervised models were accurate when using only reliably labelled cells of the Martinotti, horse-tail, common basket and large basket types. Thus, clear examples of these types can be identified on the basis of morphology. Prediction improved with axonal features F1-F4 as predictors. However, the training sets consisting of reliably labeled cells alone were small and not representative of these types' populations. We identified a number of relevant morphometrics, with the axonal ones more useful than dendritic ones.

With Markram neurons, we learned accurate and interpretable models for the Martinotti, basket, and nest basket types, and moderately accurate for the double boquet and small basket types. We learned parsimonious and interpretable classification tree and logistic regression models for Martinotti and basket, and identified predictive sets of up to nine morphometrics for the double boquet, small basket, nest basket types.

Our initial hypothesis was: supervised and semi-supervised learning can produce accurate and interpretable models of GABAergic interneuron types. We consider that our results suggest that this hypothesis is valid.

In Chapter 9, Chapter 10, and Chapter 11, we proposed and implemented custom morphometrics. We developed the neurostrplus `R` [R Core Team, 2015] package that quantifies a number of Petilla [Ascoli et al., 2008] features, including those relative to arbor shape and direction, dendritic polarity, translaminar extent [Helmstaedter et al., 2009c] estimate based on laminar thickness and soma's laminar location, as well as, motivated by our results, morphometrics that capture axonal features F1-F4 from the Gardener's scheme.

12.3 Publications

The following publications and submissions are associated with the research reported in this dissertation.

Peer-reviewed journals:

- B. Mihaljević, C. Bielza, and P. Larrañaga. bnclassify: Learning Bayesian network classifiers. *The R Journal*, 2018a. submitted

- B. Mihaljević, R. Benavides-Piccione, C. Bielza, J. DeFelipe, and P. Larrañaga. Bayesian network classifiers for categorizing cortical GABAergic interneurons. *Neuroinformatics*, 13(2):192–208, 2015a

- B. Mihaljević, R. Benavides-Piccione, L. Guerra, J. DeFelipe, P. Larrañaga, and C. Bielza. Classifying GABAergic interneurons with semi-supervised projected model-based clustering. *Artificial Intelligence in Medicine*, 65(1):49–59, 2015b

- B. Mihaljević, C. Bielza, R. Benavides-Piccione, J. DeFelipe, and P. Larrañaga. Multidimensional classification of GABAergic interneurons with Bayesian network-modeled label uncertainty. *Frontiers in Computational Neuroscience*, 8:150, 2014

- B. Mihaljević, P. Larrañaga, R. Benavides-Piccione, S. Hill, J. DeFelipe, and C. Bielza. Towards a supervised classification of neocortical interneuron morphologies. *BMC Bioinformatics*, 2018c. submitted

Congress contributions and communications:

- B. Mihaljević, C. Bielza, and P. Larrañaga. Learning Bayesian network classifiers with completed partially directed acyclic graphs. In *9th International Conference on Probabilistic Graphical Models*, 2018b. submitted
- B. Mihaljević, P. Larrañaga, and C. Bielza. Automatic classification of cortical interneuron morphologies. In *Proceedings of the Workshop on Advances and Applications of Data Science & Engineering, Real Academia de Ingenieria*, Madrid, 2016

12.4 Software

The following publicly available software tools have been produced to support the research carried out in this thesis.

- bnclassify An R package for learning discrete Bayesian network classifiers from data. It has been downloaded over 11 thousand times from the RStudio mirror of the Comprehensive R Archive Network (CRAN). There are roughly a thousand downloads per package update, suggesting that there are that many existing installations. Published on CRAN since 2015. `https://cran.r-project.org/web/packages/bnclassify/index.html`
- gabaclassifier An R package to classifies interneuron morphologies into established types. `https://github.com/ComputationalIntelligenceGroup/gabaclassifier`
- neurostrplus An R package to computes interneuron morphometrics. `https://github.com/ComputationalIntelligenceGroup/neurostrplus`
- neurostr An R wrapper for the NeuroSTR C++ library.
`https://github.com/ComputationalIntelligenceGroup/neurostrr`

12.5 Future work

Immediate future work regarding the learning of Bayesian network classifiers with equivalence classes includes evaluating our algorithm on additional synthetic and real-world data sets, to better assess its merits.

Regarding the Gardener data, it is desirable to use as much data as possible to train models. It might be possible to improve our results by dropping the assumption that all neuroscientists are equally good at classifying interneurons and applying multi-annotator methods such as that due to Raykar et al. [2010].

Achieving accurate classification for all established morphological types will require more labeled interneurons to train the models with, especially for scarce types such as ChC. In the short term, this may require leveraging the reconstructions from Neuromorpho.org. Developing automated checks of morphology quality and morphometrics robust to variability (e.g., in reconstruction granularity) could enable combining such diverse data. Morphometrics that quantify complex arborization patterns could be especially useful for the less numerous types. In the long run, we expect the Human Brain Project, the Allen Institute for Brain Research, and NeuroMorpho.Org to provide many high-quality morphologies. Given such data, we consider that the methods presented in this thesis can provide for an accurate automatic classification into established morphological types.

Part V

APPENDICES

Appendix A

Per-class supervised classifiers: Morphometrics

A.1 NeuroSTR morphometrics

We computed 'standard' morphometrics with the NeuroSTR neuroanatomy library (see Table A.1). These include branch length and bifurcation angles, arbor height and width, and topological features such as vertex ratio. We mainly summarized part-of-tree analyses (i.e., those computed for a section of an arbor, such as a branch or segment) by computing their average, using the median, standard deviation, or maximum statistics only when we deemed it justified (e.g., for maximum arbor distance to soma). We also computed some morphometrics specific to axonal terminal branches (e.g., mean terminal branch length).

Table A.1: NeuroSTR morphometrics. For part-of-tree morphometrics, suffixes avg, med, sd, and max denote the mean, median, standard deviation, and maximum, respectively. Detailed documentation for NeuroSTR features is available online: `https://computationalintelligencegroup.github.io/neurostr/doc/measures/prebuilt.html`.

Morphometric	Axon	Terminal	Dendrite
`centrifugal_order.avg`	✓		✓
`centrifugal_order.max`	✓		✓
`centrifugal_order.sd`	✓		✓
`euclidean_dist.avg`	✓		✓
`euclidean_dist.max`	✓		✓
`euclidean_dist.sd`	✓		✓
`height`	✓		✓
`length.avg`	✓	✓	✓
`length.med`	✓	✓	✓
`length.sd`	✓		✓
`N_bifurcations`	✓		✓
`N_stems`			✓
`partition_asymmetry.avg`	✓		✓
`path_dist.avg`	✓		✓
`path_dist.max`	✓		✓
`path_dist.sd`	✓		✓
`remote_bifurcation_angle.avg`	✓	✓	✓
`remote_tilt_angle.avg`	✓	✓	✓
`remote_torque_angle.avg`	✓	✓	✓
`terminal_degree.avg`	✓		✓
`tortuosity.avg`	✓	✓	✓
`tortuosity.med`	✓		✓
`total_length`	✓		✓
`tree_length.avg`			✓
`vertex_ratio`	✓		
`width`	✓		✓

A.2 Custom-implemented morphometrics

We used 48 axonal and dendritic custom-implemented morphometrics (see Table A.2).

Table A.2: Custom morphometrics.

Type	Morphometric	Axon	Dendrite
Arbor density	`density_area`	✓	✓
Arbor density	`density_bifs`	✓	✓
Arbor density	`density_dist`	✓	✓
ChC arborization pattern	`short_vertical_terminals`	✓	
Dendritic displaced	`displaced`		✓
Dendritic polarity	`insert.eccentricity`		✓
Dendritic polarity	`insert.radial`		✓
Laminar	`l1_prob`	✓	
Laminar	`translaminar`	✓	✓
MC arborization pattern	`l1_bifs`	✓	
MC arborization pattern	`l1_gx`	✓	
MC arborization pattern	`l1_gxa`	✓	
MC arborization pattern	`l1_width`	✓	
XY distribution / Axon origin	`axon_above_below`	✓	
XY distribution / Axon origin	`axon_origin`	✓	
XY distribution / Grid	`grid_area`	✓	✓
XY distribution / Grid	`grid_density`	✓	
XY distribution / Grid	`grid_mean`	✓	✓
XY distribution / Moments	`ratio_x`	✓	✓
XY distribution / Moments	`ratio_y`	✓	✓
XY distribution / Moments	`x_mean`	✓	✓
XY distribution / Moments	`x_mean_abs`	✓	✓
XY distribution / Moments	`x_sd`	✓	✓
XY distribution / Moments	`y_mean`	✓	✓
XY distribution / Moments	`y_mean_abs`	✓	✓
XY distribution / Moments	`y_sd`	✓	✓
XY distribution / Moments	`y_std_mean`	✓	✓
XY distribution / Moments	`y_std_mean_abs`	✓	✓
XY distribution / PCA	`eccentricity`	✓	✓
XY distribution / PCA	`radial`	✓	✓

A.2.1 Distribution along X and Y axes

Each neuronal reconstruction consisted of points with Euclidean coordinates, with the center of gravity of the soma located at coordinates $(0, 0, 0)$. Thus, computing, e.g., the standard deviation along the X axis provided an estimate of arborization extent in the horizontal direction.

A.2.1.1 PCA-derived

Following Yelnik et al. [1983] we used principal component analysis (PCA) to quantify possible preferential orientation of an arbor along either the X or Y dimension. We set the Z coordinates to zero and quantified such preference with the index of axialization measure of Yelnik et al. [1983], calling it `eccentricity`:

$$\mathrm{e} = 1 - \frac{s_2}{s_1},$$

where s_1 and s_2 are standard deviations of the first and second principal components, respectively (thus, $s_1 \geq s_2 \geq 0$). An e towards 1 indicates a strong preference for one axis, whereas an e towards 0 indicates a circular arbor. We used the angle θ of the main axis (i.e., the first principal component) to a positive X axis passing through the center of mass to quantify the degree of `radial` or tangential orientation of the arbor, namely,

$$\mathrm{r} = (|y| - |x|) \times \mathrm{e},$$

where y and x are the loadings of the first component on the Y and X axes, respectively, and correspond to $|\sin\theta|$ and $|\cos\theta|$. Thus, r is positive if the tree is ascending or descending, and close to -1 if it mainly arborizes horizontally. To reduce its magnitude for trees that did not have a preference for one of the two directions, we factored in the degree of eccentricity, e (which is always positive).

A.2.1.2 Moments along the axes (distribution around the soma)

We computed the mean, standard deviation, and the standardized mean (i.e., the ratio of the mean to the standard deviation) along the X and Y axes. The sign of `y_mean` and `y_std_mean`, for example, may help distinguish between arbors that ascend towards the pial surface or descend towards the white matter; unlike `y_mean`, `y_std_mean` is dimensionless and expresses the arborization preference in terms of the Y extent of the arbor. We also computed the means of $|x|$ and $|y|$ so as to not distinguish between arbors skewed towards the right or the left (or above or below) of the soma, but instead between those arborizing close and far from soma, both horizontally and vertically. The standard deviations indicate the extent along an axis and are very correlated with the `height` and `width` morphometrics. Finally, we computed the ratio of the range along an axis and the standard deviation along that axis (`ratio_x` and `ratio_y`).

A.2.1.3 Grid analysis

We split the X and Y plane into $20\,\mu m$ by $20\,\mu m$ squares, and computed the number of branches in each square. We recorded the number of non-empty squares (i.e., those containing at least one branch; `grid_area`), as an estimate of the arbor's area, and the mean (`grid_mean`) branch count per non-empty square. Finally, we computed the ratio of non-empty $100\,\mu m$ by $100\,\mu m$ squares and `grid_area`, to quantify arborization density (`grid_density`), i.e., arbors that tend to occupy a large of portion of a given $100\,\mu m$ by $100\,\mu m$ square.

A.2.1.4 Axon origin

In order to distinguish axons that originate from below the soma from those that originate above it, we recorded the Y coordinate of the first axonal bifurcation (`axon_origin`), as well as the difference between the minimal path distance from the soma among points more than $100\,\mu m$ below the soma ($Y < 100\ \mu m$) and those at least $100\,\mu m$ above the soma ($Y > 100\ \mu m$; `axon_above_below`); a positive value would suggest that the arborization begun on the upper side of the soma.

A.2.2 Laminar distribution

Since we did not know the exact location of the soma within a layer, we could only estimate axonal projection across the layers. For these estimates we relied on layer thickness data from Figure 3 of Markram et al. [2015], shown in Table A.3, assuming that the thickness T_l of layer l follows a Gaussian distribution, $\mathcal{N}(mt_l, st_l)$, where mt_l and st_l are the mean and standard deviation of T_l (given in Table A.3).

The probability of an axon reaching L1 depends on axonal height above the soma, h_a, and the distance D from the soma to the center of L1, c_1. We modelled D as a sum of two independent random variables, $D = D_l + P$, where D_l is the distance from c_l, the center of the soma's layer l, to c_1, and P the position of the soma with respect to c_l (considering, in both cases, only the Y dimension). Assuming layers' thicknesses are independent, $D_l \sim \mathcal{N}(md_l, sd_l)$, where

$$md_l = \frac{mt_1}{2} + \sum_{k=2}^{l-1} mt_k + \frac{mt_l}{2},$$

and

$$sd_l = \sqrt{\frac{st_1^2}{4} + \sum_{k=2}^{l-1} st_k^2 + \frac{st_l^2}{4}},$$

where the summation term is omitted for L2 (i.e., $l = 2$). Assuming that P follows $\mathcal{N}(0, \frac{mt_l}{4})$, the sum $D_l + P$ follows $\mathcal{N}(md_l, \sqrt{sd_l^2 + (\frac{mt_l}{4})^2})$ and the probability `l1_prob` of an axon reaching L1 is that of drawing a value equal or greater than h_a from this distribution. Thus, for example, for an L4 cell with its axon extending 500 μm above its soma, the probability of reaching L1 was 0.0005, whereas for one with length 700 μm was 0.6450, i.e., 65% ($md_4 = 679.5$; see Table A.3).

Table A.3: Layer thickness data from Markram et al. [2015] and the estimated distance from the layer's center to the center of L1.

Layer	Thickness	Distance to L1 ($md_l \pm sd_l$)
1	165 ± 13	
2/3	502 ± 27	333.5 ± 15
4	190 ± 7	679.5 ± 28
5	525 ± 33	1037.0 ± 33.1
6	700 ± 48	1649.5 ± 49.9

We estimated the probability p_a of an arbor extending into the layer above as the probability of drawing h_a from a Gaussian distribution $\mathcal{N}(\frac{mt_l}{2}, \frac{mt_l}{4})$, where h_a is the arbor's height above the soma, and mt_l, as above, the mean thickness of the soma's layer l. We computed the probability p_b of reaching the layer below analogously, setting it to 0 for layer L6. The probability of an arbor being `translaminar` (i.e., not confined to a single layer) was given by $\max\{p_a, p_b\}$.

A.2.3 MC arborization pattern

To estimate axonal width in L1 (`l1 width`; MC cells' axons tend to spread out horizontally in this layer), we computed its width in the upper 165 μm (i.e., the thickness of L1) of its arborization and multiplied it with the probability of having reached L1 (`l1_prob`). In an analogous way we estimated the total number of bifurcations in L1 (as a proxy for total arbor extent in that layer). We also estimated the extent to which this arborization grew horizontally (`l1_gx`) and away from the soma (`l1_gxa`), following the assumption that the axon rises vertically approximately above the soma, and ramifies in both horizontal directions in layer L1. `l1_gx` is the sum of all segments' X-axis projections, whereas `l1_gxa` equals `l1_gx` minus the X-axis projections of all segments directed towards the soma (i.e., their initial X coordinate is further from the soma than their terminal coordinate).

A.2.4 ChC arborization pattern

Since ChC cells' axons have short vertical terminals [Markram et al., 2004, Somogyi, 1977], we counted the number of terminal branches with an extent along the Y axis $< 50\,\mu m$ (Somogyi [1977] reports ChC vertical terminals from $10\,\mu m$ to $50\,\mu m$ long) and at least twice as large as the extent along the X axis (`short_vertical_terminals`).

A.2.5 Arbor density

We quantified arbor density with a number of ratios involving arbor length as the denominator: the ratio of the number of bifurcations and arbor length (`density_bifs`), proportional to the inverse of branch length; the ratio of `area` and arbor length (`density_area`), and, finally, the ratio of average Euclidean distance and total length (`density_dist`).

A.2.6 Dendritic bipolarity

We quantified whether the dendrites stemmed from opposite ends of the soma and whether those ends are located along a radial (i.e., parallel to the Y) axis, as is the case with bipolar and bitufted dendrites. We did this by applying the above-described PCA-derived analysis to the dendrite insertion points on the soma's surface, after having replicated every insertion point once for each whole μm of the corresponding dendrite's length, so as to give more weight to insertion points of longer dendrites, and having set the Z coordinates to 0. A high `insert.eccentricity` thus indicated insertion points along an axis, rather than spread-out across the soma's surface, whereas a high `insert.radial` suggested that the axis was parallel to the Y axis. For cells with a single dendrite insertion point we set `insert.eccentricity` and `insert.radial` to 0.

A.2.7 Displaced dendritic arbor

To quantify whether the dendritic arbor was displaced [DeFelipe et al., 2013] from the axonal one, we averaged the distance to the closest axonal reconstruction point for each point of a dendritic arbor (`displaced`).

A.3 Implementation

NeuroSTR is available at `https://github.com/ComputationalIntelligenceGroup/neurostr`.

Appendix B

Per-class supervised classifiers: Additional results

B.1 Morphology quality

B.1.1 Reconstruction differences

We found that the cells differed in mean axonal segment length and that this difference could be related to the internal ids (e.g., C010600B1 and MTC070301B_IDC) of the cells. Out of the seven different initial letters of these ids, cells whose id begun with a letter C (88 of them) had shorter, as well as thicker, axonal segments than the remaining 131 cells (see Figure 5 in main text). As branch and total arbor length did not differ between the two groups, this meant that the C-prefixed cells had fewer long and thick segments per branch, whereas the non-C cells contained more short and thin ones, suggesting that they were simply reconstructed at a finer granularity. We found that the C-prefixed cells were deposited at Neuromorpho.org repository [Ascoli et al., 2007] earlier than the non-C ones, meaning they may have been reconstructed at an earlier stage.

More morphometrics, such as axonal remote tilt angle (`remote_tilt_angle.avg`), arbor depth (`depth`), and tortuosity (`tortuosity.avg`), also differed between the two groups, albeit with much less statistical significance (see Table B.1) than thickness and segment length. We suspect that only some of these, such as possibly tortuosity (the non-C cells have lower tortuosity, i.e., they are less straight, which seems logical given that they are broken into more segments) had been affected by differences in branch reconstruction granularity; others might have differed due to others causes, such as different proportion of interneuron types in the two groups, or the different laminar distribution (see Figure B.1).

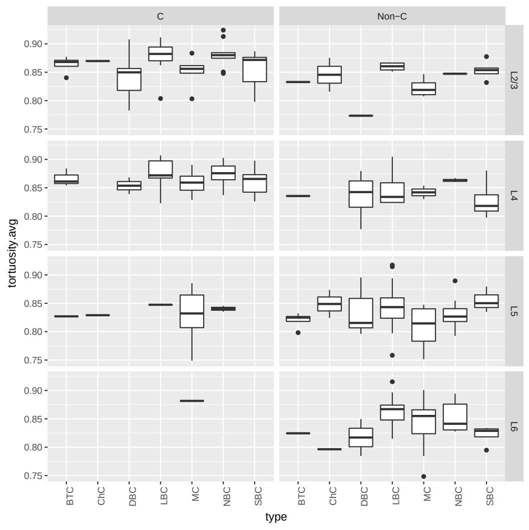


Figure B.1: The branches of non-C cells' (those whose ids do not begin with a C) were less straight (i.e., their tortuosity values were lower), even after accounting for interneuron type and layer.

Table B.1: Morphometrics that differed between the cells whose id begins with a C and the rest, according to a Kruskal-Wallis test at $\alpha = 0.05$, with the p-value corrected for multiple testing with the false discovery rate procedure [Benjamini and Hochberg, 1995].

Morphometric	Axon	Dendrite
`compartment_length.avg`	1.3×10^{-27}	
`diameter.avg`	5.3×10^{-26}	9.5×10^{-3}
`N_nodes`	6.2×10^{-22}	
`remote_tilt_angle.avg`	4.4×10^{-6}	2.6×10^{-2}
`depth`	2.5×10^{-5}	
`tortuosity.avg`	8.6×10^{-5}	
`tortuosity.med`	3.8×10^{-4}	8.9×10^{-3}
`l1_gx`	4.9×10^{-4}	
`x_sd`	1.4×10^{-3}	
`density_area`	3.2×10^{-3}	1.9×10^{-4}
`l1_bifs`	6.7×10^{-3}	
`local_tilt_angle.avg`	8.9×10^{-3}	
`height`	1.3×10^{-2}	8.0×10^{-4}
`width`	2.0×10^{-2}	2.0×10^{-2}
`l1_gxa`	2.1×10^{-2}	
`euclidean_dist.max`	2.3×10^{-2}	1.2×10^{-4}
`y_mean`	2.9×10^{-2}	
`grid_area`	3.8×10^{-2}	2.6×10^{-2}
`density_dist`		1.5×10^{-2}
`euclidean_dist.avg`		3.2×10^{-3}
`euclidean_dist.sd`		2.5×10^{-5}
`path_dist.avg`		3.8×10^{-2}
`path_dist.max`		1.6×10^{-3}
`path_dist.sd`		4.1×10^{-4}
`ratio_x`		2.6×10^{-2}
`remote_torque_angle.avg`		1.2×10^{-2}
`terminal_degree.avg`		1.6×10^{-3}
`x_mean_abs`		2.3×10^{-2}
`y_sd`		8.3×10^{-3}

B.1.2 Two cloned cells

We visually identified two cells as possible modified duplicates of another pair of cells (see Figure B.2). They differed in most axonal and dendritic morphometrics, including the number of branches or axonal length, but were similar in axonal height and total dendritic length and height, suggesting how cells which are similar to the eye are not so according to most of the morphometrics that we are using. We then ran hierarchical clustering on all cells using these variables (i.e., axonal height, dendritic length and height) but found no additional pairs of duplicated cells.

B.2 Feature selection results

Table B.2 shows the sizes of the feature subsets selected by the different methods and their performance.

Table B.2: Number of selected morphometrics with the different methods. The color indicates the best F-measure obtained with the corresponding feature selection method. Best F-measure ≥ 0.75 are shown in green; best F-measure ≥ 0.60 in orange; and the rest in red. CART and RMLR refer to the embedded feature selection performed by those models. Filter feature selection followed by embedded selection is denoted with a +, e.g., KW followed by CART is denoted with KW + CART. CART and RMLR are only considered in absence of prior sampling. There are no entries for RF BVI + RMLR for the BTC as RMLR could not be fit due to too few features being selected by the RF BVI.

Class	KW	KW + CART	KW + RMLR	RF BVI	RF BVI + CART	RF BVI + RMLR	CART	RMLR
ChC	15	2	5	3	1	1	2	11
BTC	7	5	3	2	2		4	22
DBC	61	3	15	6	2	4	3	17
SBC	39	5	9	7	5	4	5	24
NBC	57	5	19	9	3	5	4	27
MC	62	6	22	8	5	6	5	28
LBC	32	9	17	4	4	4	8	38
BA	68	11	27	6	5	5	10	31

Table B.3 shows the logistic regression model for MC.

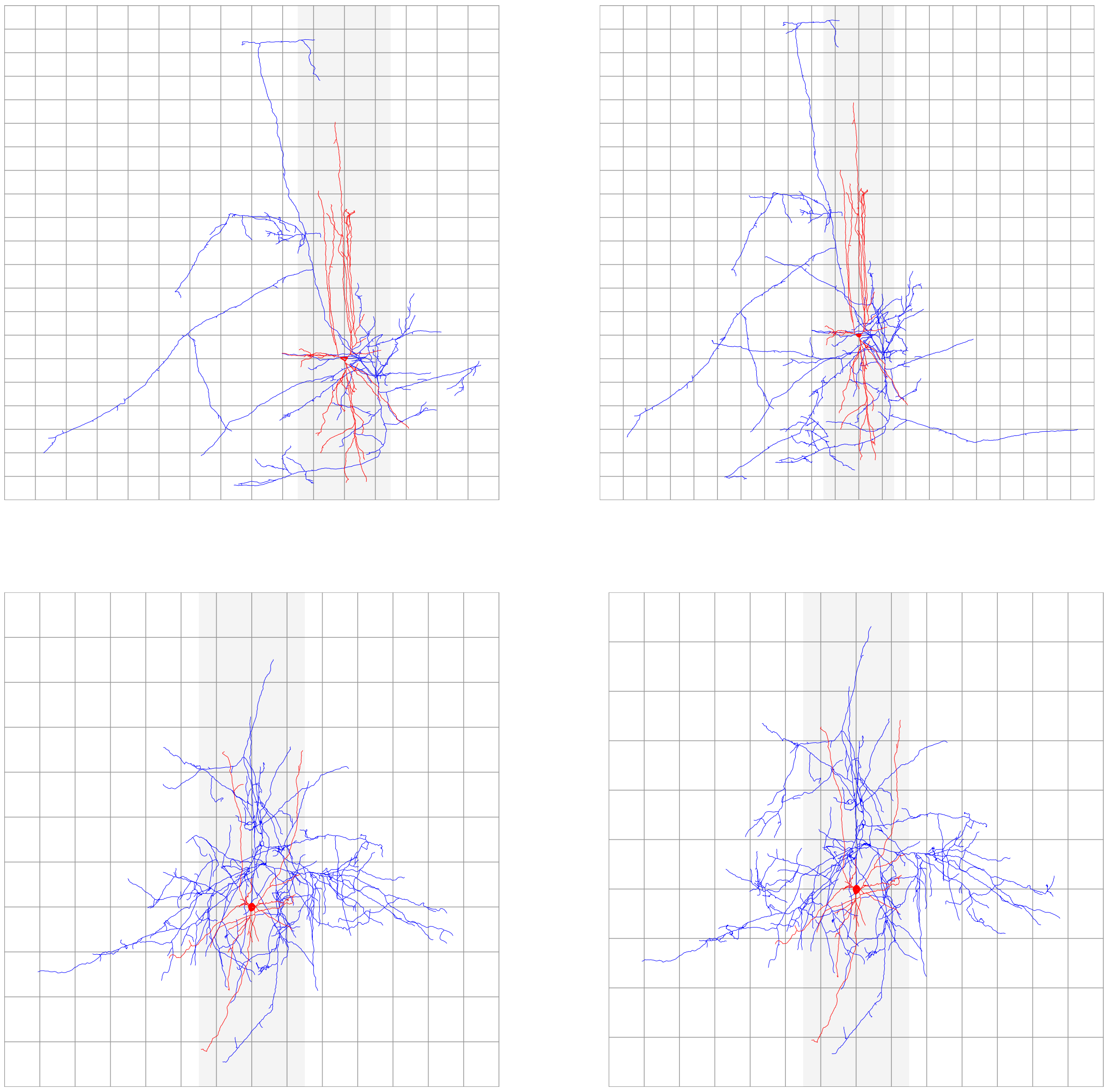


Figure B.2: Cells OG061201A1-8_IDA and OG061201A3_CH1_IN_H_ZK_60X_1 (above), and OG061201A1-8_IDE and OG061201A6_CH5_BC_H_ZK_60X_1 (below) which seemed very similar by visual inspection. Axons are drawn in blue and dendrites and somata in red. There are 100 μm between consecutive grid lines.

Table B.3: The logistic regression model for MC. The $\boldsymbol{\beta}$ were estimated from the standardized data set, after feature selection with KW.

Morphometric	β
`y_std_mean`	1.44
`ratio_y`	-0.88
`remote_bifurcation_angle.avg`	0.79
`path_dist.max`	0.63
`d.displaced`	-0.63
`l1_width`	0.59
`d.total_length`	0.47
`translaminar`	0.43
`radial`	0.31
`d.N_stems`	-0.30
`d.terminal_degree.avg`	-0.24
`density_bifs`	-0.23
`l1_bifs`	0.22
`d.path_dist.avg`	-0.22
`path_dist.avg`	0.14
`t.tortuosity.avg`	-0.14
`d.y_std_mean_abs`	0.13
`x_mean`	-0.11
`d.insert.radial`	-0.09
`t.length.med`	-0.09
`l1_prob`	0.01
`grid_density`	0.00

Table B.4 shows the 88 features selected by KW for at least one of the types, showing the corresponding p-values. Each column corresponds to a one-versus-all classification setting. Overall, the single most relevant feature was `path_dist.avg` for BA, with a p-value of 3.6×10^{-17}, and the strongest dendritic predictor was the number of dendrites (`d.N_stems`) also for BA, with p-value 5.3×10^{-14}. Table B.5 shows the features selected by RF BVI for the different classification settings, along with their RF BVI values. Overall, RF BVI selected only one dendritic morphometric, for the BTC type, and picked only axonal features for all remaining types.

Table B.4: Morphometrics that differed most between the given class and the remaining classes joined together, according to the Kruskal-Wallis test. Empty entries mean that the p-value was above 0.05. Morphometrics that were significant for most classes are shown in the upper rows.

	Morphometric	ChC	BTC	DBC	SBC	NBC	MC	LBC	BA
1	height	3.8×10^{-3}		8.3×10^{-3}	2.4×10^{-7}	5.5×10^{-7}	4.2×10^{-6}	8.1×10^{-3}	2.7×10^{-7}
2	d.displaced	5.0×10^{-2}		8.5×10^{-4}		1.3×10^{-4}	2.2×10^{-4}	4.7×10^{-2}	3.0×10^{-10}
3	d.insert.eccentricity		4.3×10^{-2}	1.5×10^{-3}		8.6×10^{-4}	2.6×10^{-3}	4.5×10^{-2}	7.9×10^{-10}
4	euclidean_dist.max			3.6×10^{-3}	2.7×10^{-8}	1.1×10^{-10}	3.4×10^{-10}	2.0×10^{-2}	4.1×10^{-12}
5	grid_density	4.8×10^{-2}		3.1×10^{-4}	2.2×10^{-2}	1.7×10^{-4}	1.3×10^{-4}		1.6×10^{-8}
6	grid_mean	2.1×10^{-3}	1.2×10^{-2}		8.6×10^{-9}	2.0×10^{-2}	2.5×10^{-6}		3.2×10^{-7}
7	terminal_degree.avg	4.5×10^{-2}		7.3×10^{-3}		3.5×10^{-8}	2.0×10^{-7}	4.5×10^{-2}	2.7×10^{-9}
8	t.length.avg	3.8×10^{-3}	7.0×10^{-3}	7.3×10^{-3}	1.5×10^{-7}	2.4×10^{-3}		1.1×10^{-2}	
9	t.length.med			5.8×10^{-4}	4.3×10^{-3}	5.6×10^{-7}	1.8×10^{-3}	2.9×10^{-4}	2.1×10^{-8}
10	translaminar	5.5×10^{-3}		9.4×10^{-4}	1.2×10^{-5}	3.1×10^{-9}	7.0×10^{-8}		8.2×10^{-10}
11	y_sd	3.8×10^{-3}		1.1×10^{-3}	9.5×10^{-8}	6.3×10^{-9}	9.6×10^{-11}		2.5×10^{-12}
12	d.centrifugal_order.avg			2.7×10^{-3}		1.7×10^{-5}	5.6×10^{-7}	2.2×10^{-2}	3.1×10^{-13}
13	d.centrifugal_order.sd			1.0×10^{-2}	2.8×10^{-2}	7.0×10^{-4}	6.9×10^{-4}		7.9×10^{-8}
14	d.density_bifs			3.7×10^{-3}		1.4×10^{-5}	2.2×10^{-4}	2.8×10^{-3}	3.3×10^{-11}
15	density_area	3.8×10^{-3}			3.2×10^{-6}	3.4×10^{-3}	1.4×10^{-5}		1.5×10^{-6}
16	density_bifs	2.1×10^{-3}	2.4×10^{-3}		3.5×10^{-10}		1.4×10^{-2}	4.5×10^{-2}	
17	density_dist			8.2×10^{-7}		3.1×10^{-6}	5.5×10^{-7}	5.0×10^{-2}	3.6×10^{-13}
18	d.insert.radial		2.8×10^{-3}	7.7×10^{-4}		9.0×10^{-7}	2.8×10^{-2}		4.0×10^{-9}
19	d.N_bifurcations			3.8×10^{-2}		1.9×10^{-4}	1.3×10^{-7}	2.8×10^{-2}	1.8×10^{-11}
20	d.N_stems			6.9×10^{-4}		2.3×10^{-5}	5.0×10^{-6}	3.5×10^{-4}	5.3×10^{-14}
21	d.path_dist.avg			5.2×10^{-3}	2.6×10^{-2}	9.1×10^{-3}	1.3×10^{-2}		3.0×10^{-5}
22	d.terminal_degree.avg			1.4×10^{-3}		3.0×10^{-4}	1.1×10^{-3}	2.8×10^{-2}	3.4×10^{-9}
23	d.tree_length.avg			9.4×10^{-3}		8.1×10^{-4}	3.3×10^{-7}	1.3×10^{-2}	1.2×10^{-11}
24	euclidean_dist.avg			1.5×10^{-3}	6.1×10^{-9}	1.2×10^{-10}	1.2×10^{-13}		6.7×10^{-17}
25	euclidean_dist.sd			7.8×10^{-4}	6.1×10^{-9}	4.0×10^{-11}	7.1×10^{-13}		3.5×10^{-15}
26	length.avg	2.1×10^{-3}	2.4×10^{-3}		3.5×10^{-10}		1.4×10^{-2}	4.5×10^{-2}	
27	length.med	2.4×10^{-3}		1.6×10^{-2}	1.5×10^{-7}	1.1×10^{-2}		2.8×10^{-3}	
28	length.sd	2.1×10^{-3}	4.9×10^{-4}		3.5×10^{-10}		4.9×10^{-6}		2.4×10^{-6}
29	path_dist.avg			7.5×10^{-3}	3.2×10^{-6}	3.8×10^{-10}	6.8×10^{-14}		3.6×10^{-17}
30	path_dist.max			4.1×10^{-2}	5.5×10^{-7}	1.1×10^{-10}	4.0×10^{-13}		8.2×10^{-15}
31	path_dist.sd			1.3×10^{-3}	2.3×10^{-7}	1.2×10^{-10}	1.4×10^{-13}		6.7×10^{-17}
32	radial			4.6×10^{-8}	4.3×10^{-2}	3.1×10^{-6}	1.8×10^{-4}		9.8×10^{-9}
33	remote_bifurcation_angle.avg			3.1×10^{-4}		3.4×10^{-4}	1.2×10^{-5}	3.7×10^{-8}	7.8×10^{-15}
34	t.remote_bifurcation_angle.avg			1.3×10^{-4}		1.3×10^{-4}	5.2×10^{-6}	6.9×10^{-8}	3.4×10^{-15}
35	y_mean_abs			1.2×10^{-3}	8.3×10^{-4}	5.3×10^{-8}	1.0×10^{-10}		2.1×10^{-15}
36	y_std_mean_abs			1.7×10^{-2}		5.9×10^{-4}	1.7×10^{-4}	4.7×10^{-2}	1.6×10^{-8}
37	centrifugal_order.max			2.9×10^{-2}		1.3×10^{-5}	2.5×10^{-3}		5.5×10^{-4}
38	centrifugal_order.sd			7.7×10^{-4}		5.5×10^{-7}	3.9×10^{-5}		1.3×10^{-7}
39	d.centrifugal_order.max			3.1×10^{-3}		1.7×10^{-4}	2.9×10^{-5}		5.4×10^{-10}
40	d.euclidean_dist.avg			5.1×10^{-3}	1.2×10^{-2}	1.6×10^{-2}			6.0×10^{-4}
41	d.path_dist.max			9.4×10^{-3}	1.8×10^{-2}	4.0×10^{-2}			3.7×10^{-3}
42	d.y_sd			3.1×10^{-4}	1.2×10^{-2}	8.5×10^{-3}			7.2×10^{-3}
43	eccentricity			4.6×10^{-8}		7.2×10^{-8}	1.4×10^{-3}		5.3×10^{-8}
44	grid_area			1.3×10^{-4}	1.1×10^{-6}		1.4×10^{-3}	3.9×10^{-5}	
45	N_bifurcations	1.1×10^{-2}		5.8×10^{-4}	4.4×10^{-2}				1.2×10^{-2}
46	partition_asymmetry.avg			2.0×10^{-4}		8.2×10^{-7}	3.9×10^{-5}		1.9×10^{-9}
47	ratio_y			3.3×10^{-3}		4.4×10^{-6}	1.1×10^{-10}		5.0×10^{-12}
48	vertex_ratio			9.8×10^{-4}		5.5×10^{-6}	5.7×10^{-4}		1.9×10^{-7}
49	width			1.9×10^{-6}	2.1×10^{-4}		2.0×10^{-3}	3.8×10^{-3}	
50	x_sd			1.9×10^{-6}	3.4×10^{-4}		3.1×10^{-3}	4.6×10^{-3}	
51	y_mean			5.8×10^{-4}		9.1×10^{-3}	9.7×10^{-14}		1.4×10^{-3}
52	centrifugal_order.avg					3.5×10^{-5}	2.7×10^{-4}		1.9×10^{-4}
53	d.eccentricity			4.9×10^{-5}		2.5×10^{-3}			1.8×10^{-2}
54	d.euclidean_dist.max			5.6×10^{-3}	1.1×10^{-2}				1.6×10^{-2}
55	d.grid_area				4.0×10^{-3}		3.3×10^{-3}		3.1×10^{-2}
56	d.grid_mean						3.8×10^{-2}	1.3×10^{-2}	5.8×10^{-3}
57	d.height			1.1×10^{-3}	1.2×10^{-2}	3.1×10^{-2}			
58	d.length.avg					4.4×10^{-2}		7.3×10^{-3}	7.4×10^{-4}
59	d.path_dist.sd			3.4×10^{-2}	2.3×10^{-2}				2.8×10^{-2}
60	d.radial			8.2×10^{-7}		6.9×10^{-4}			1.2×10^{-3}
61	d.ratio_y			1.5×10^{-3}		2.7×10^{-2}			3.6×10^{-5}
62	l1_bifs					4.2×10^{-3}	1.2×10^{-7}		1.0×10^{-3}
63	l1_gx					2.2×10^{-2}	1.3×10^{-6}		2.4×10^{-3}
64	l1_gxa					8.7×10^{-4}	3.2×10^{-9}		4.5×10^{-5}
65	l1_prob					7.0×10^{-4}	2.2×10^{-8}		4.9×10^{-5}

66	`l1_width`					4.1×10^{-4}	1.0×10^{-10}		5.7×10^{-6}
67	`remote_tilt_angle.avg`				1.4×10^{-3}			3.4×10^{-6}	3.9×10^{-2}
68	`short_vertical_terminals`	2.7×10^{-3}		5.2×10^{-3}	1.3×10^{-3}				
69	`tortuosity.avg`						4.1×10^{-2}	2.7×10^{-2}	5.0×10^{-4}
70	`t.remote_tilt_angle.avg`				1.2×10^{-2}			4.6×10^{-6}	1.6×10^{-3}
71	`t.tortuosity.avg`						2.2×10^{-2}	4.5×10^{-2}	6.7×10^{-4}
72	`y_std_mean`			5.6×10^{-4}			1.6×10^{-10}		2.7×10^{-2}
73	`axon_above_below`					4.0×10^{-2}	1.7×10^{-2}		
74	`axon_origin`			1.6×10^{-2}			5.0×10^{-6}		
75	`d.euclidean_dist.sd`			2.4×10^{-2}	1.8×10^{-2}				
76	`d.length.med`					4.4×10^{-2}			1.3×10^{-3}
77	`d.length.sd`				3.6×10^{-2}			4.7×10^{-2}	
78	`d.total_length`				3.3×10^{-2}		1.4×10^{-3}		
79	`d.y_mean_abs`						4.1×10^{-2}		1.8×10^{-2}
80	`tortuosity.med`							5.0×10^{-2}	4.0×10^{-3}
81	`total_length`			6.5×10^{-6}				4.0×10^{-3}	
82	`x_mean`						5.6×10^{-4}		2.1×10^{-2}
83	`d.density_dist`			1.9×10^{-3}					
84	`d.partition_asymmetry.avg`								2.4×10^{-3}
85	`d.translaminar`			4.5×10^{-3}					
86	`d.width`			2.9×10^{-3}					
87	`d.x_sd`			8.5×10^{-4}					
88	`d.y_std_mean_abs`						1.6×10^{-2}		
	Dendritic	1	2	26	12	21	17	11	26
	Total	15	7	61	39	57	62	32	68

Table B.5: Morphometrics that differed between the given class and the remaining classes joined together, according to the RF BVI ranking. Empty entries mean that the RF BVI for that class was above 0.01. Morphometrics that were relevant to most classes are shown in the upper rows.

	Morphometric	ChC	BTC	DBC	SBC	NBC	MC	LBC	BA
1	`density_bifs`	0.02			0.03	0.01			
2	`euclidean_dist.avg`				0.02	0.02			0.01
3	`euclidean_dist.sd`				0.02	0.04		0.02	
4	`length.avg`	0.02			0.03	0.01			
5	`path_dist.sd`					0.02	0.01		0.01
6	`euclidean_dist.max`					0.03		0.01	
7	`length.sd`	0.01			0.02				
8	`path_dist.avg`						0.02		0.02
9	`path_dist.max`					0.02	0.01		
10	`remote_bifurcation_angle.avg`							0.02	0.02
11	`t.length.avg`		0.01		0.01				
12	`t.remote_bifurcation_angle.avg`							0.02	0.02
13	`y_mean`			0.02			0.03		
14	`y_std_mean`			0.01			0.01		
15	`axon_origin`		0.01						
16	`d.insert.radial`		0.01						
17	`eccentricity`			0.04					
18	`l1_gxa`						0.01		
19	`l1_width`						0.01		
20	`length.med`				0.01				
21	`radial`			0.02					
22	`ratio_y`						0.01		
23	`translaminar`					0.01			
24	`width`			0.02					
25	`x_sd`			0.02					
26	`y_mean_abs`								0.01
27	`y_sd`					0.02			
	Dendritic	0	1	0	0	0	0	0	0
	Total	3	3	6	7	9	8	4	6

B.3 Classification results

Figures B.3 to B.10 show all classifiers' F-measure for all eight classification tasks. For ChC and BTC the results depended more strongly on sampling, with some samplings providing better and other worse results (e.g., for ChC, the F-measure of RF BVI + SVM ranged from 0.13 to 0.57; see Figure B.3), as in these settings the amount of removed instances was highest. Perhaps an informed, rather than random, undersampling scheme could have improved the results.

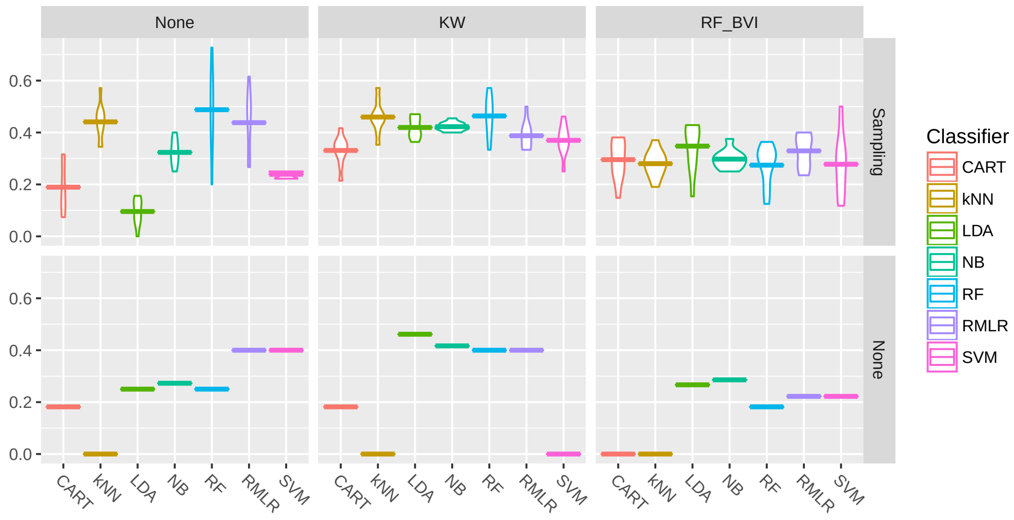


Figure B.3: ChC versus rest. Violin plot of 7-fold cross-validation estimates of F-measure. Above: seven CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

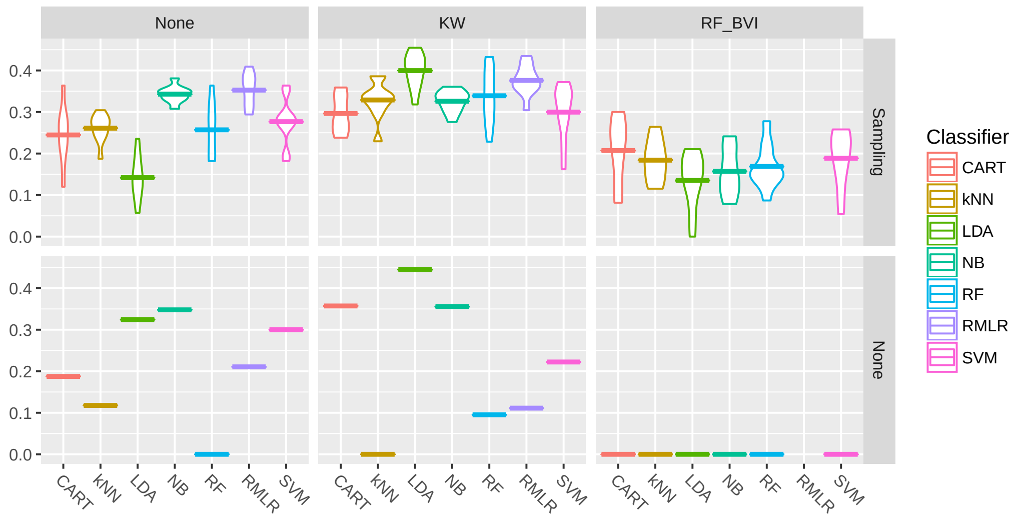


Figure B.4: BTC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

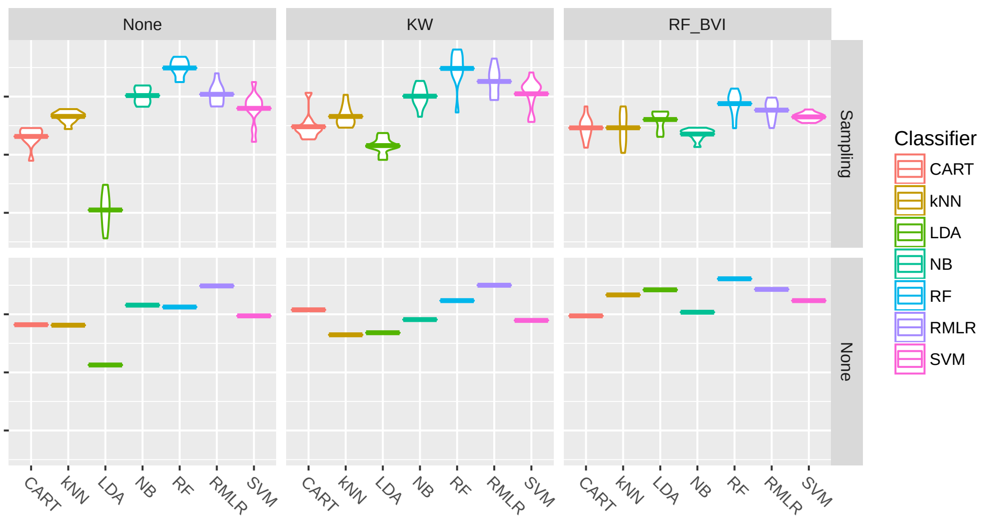


Figure B.5: DBC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

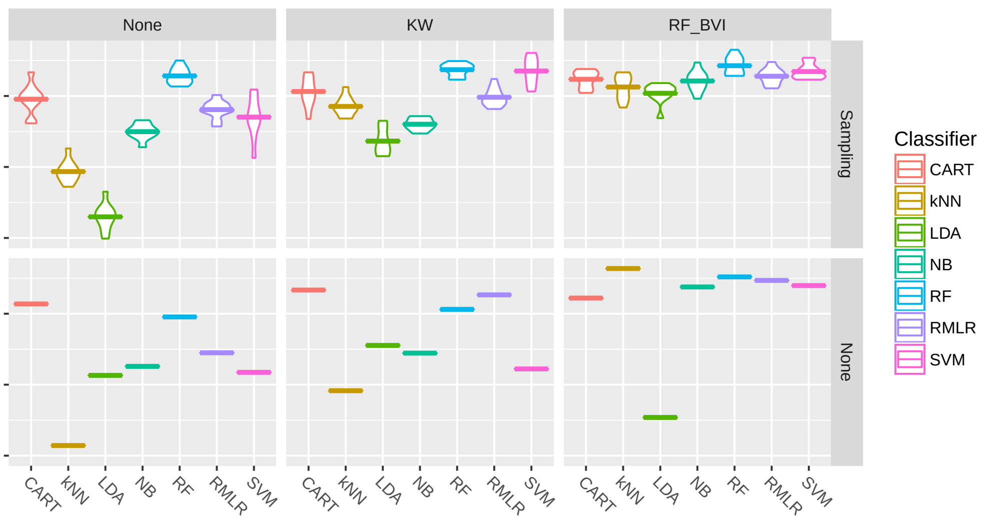


Figure B.6: SBC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

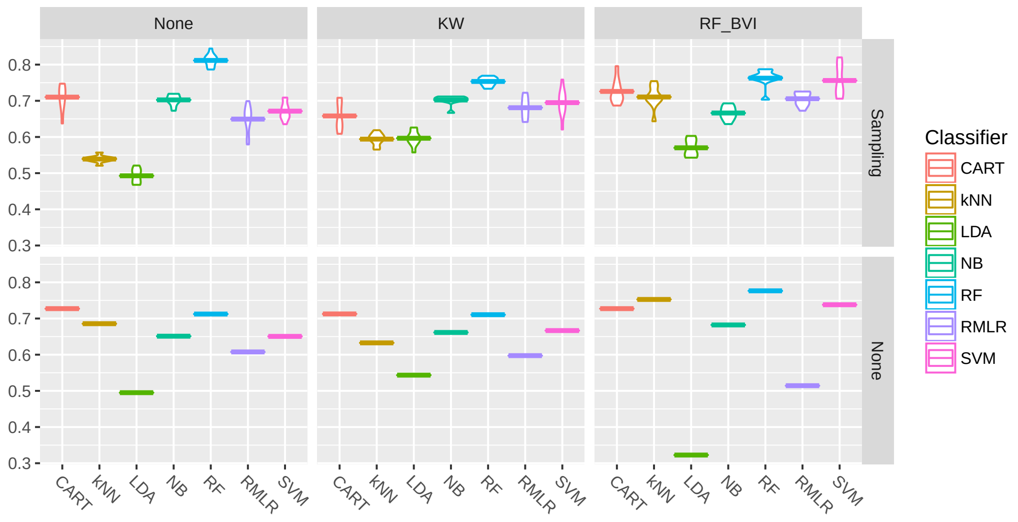


Figure B.7: NBC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

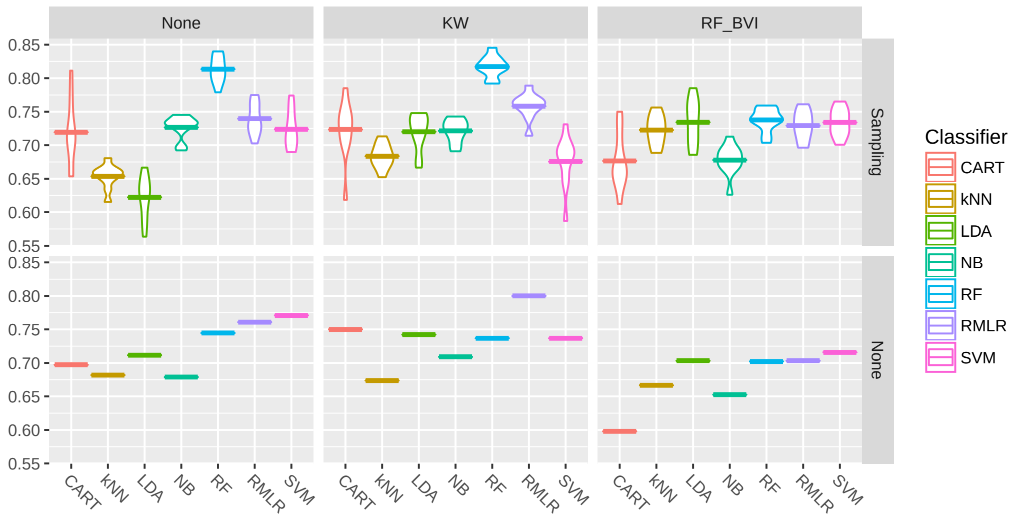


Figure B.8: MC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

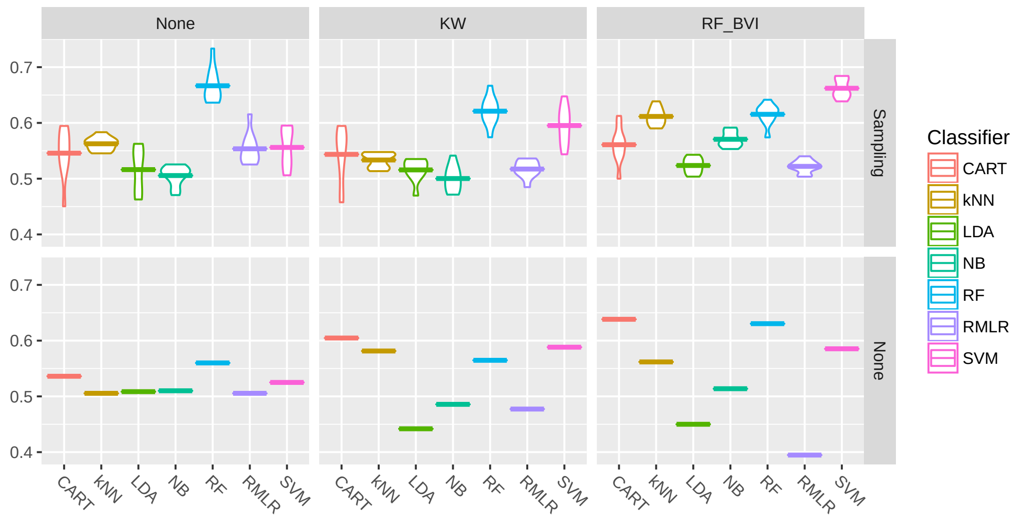


Figure B.9: LBC versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI.

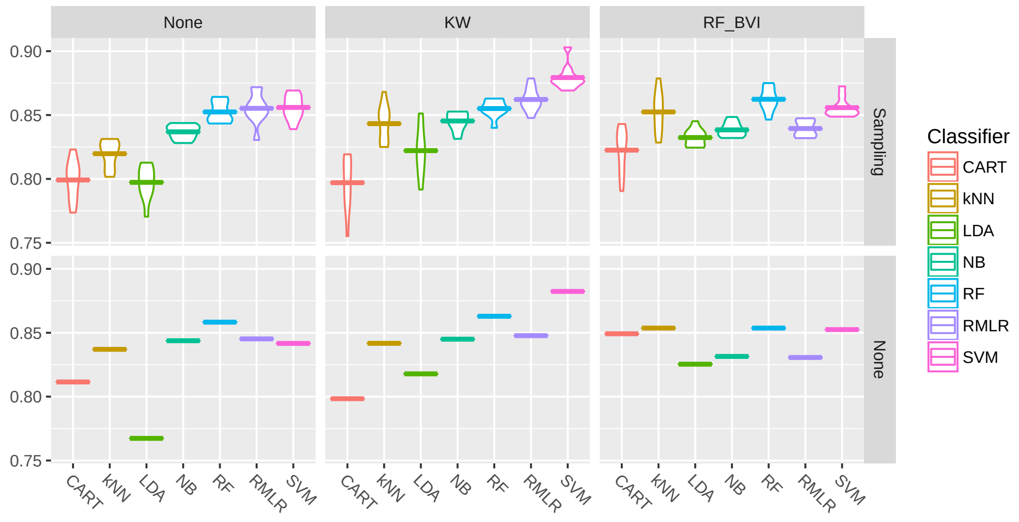


Figure B.10: BA versus rest. Violin plot of 10-fold cross-validation estimates of F-measure. Above: ten CV repetitions when under- and over-sampling training data; below: a single CV repetition with no data sampling. Vertical rows of panels correspond to the feature selection methods applied: none, KW, and RF BVI. KW feature selection improved the performance of multiple models, most notably kNN, LDA, and SVM.

Appendix C

Gardener's classification: Additional results

Here we present additional results regarding the classification of interneurons in DeFelipe et al. [2013] (see Section 5.4).

C.1 Neuroscientists' F-measure for the MC type

42 neuroscientists classified 320 cells in DeFelipe et al. [2013]. For 299 those cells, at least 22 (half + one) of them agreed on single type, which we then considered as the true type of that interneuron; 48 of those cells were MC and 251 non-MC. We computed the F-measure of each neuroscientists with respect to the determined true type. The average F-measure was 0.72, minimal 0.12 and maximal 0.89, with only three neuroscientists performing better than our best MC model (F-measure 0.81).

Bibliography

S. Acid and L. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

S. Acid, L. de Campos, and J. G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

D. W. Aha, editor. *Lazy Learning.* Springer, 1997.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

A. Altmann, L. Toloşi, O. Sander, and T. Lengauer. Permutation importance: A corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

K. Amunts, C. Ebell, J. Muller, M. Telefont, A. Knoll, and T. Lippert. The human brain project: Creating a European research infrastructure to decode the human brain. *Neuron*, 92(3):574–581, 2016.

I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM, 2000.

R. Armañanzas and G. A. Ascoli. Towards the automatic classification of neurons. *Trends in Neurosciences*, 38(5):307–318, 2015.

G. A. Ascoli. Mobilizing the base of neuroscience data: The case of neuronal morphologies. *Nature Reviews Neuroscience*, 7(4):318–324, 2006.

G. A. Ascoli, D. E. Donohue, and M. Halavi. Neuromorpho.org: A central resource for neuronal morphologies. *The Journal of Neuroscience*, 27(35):9247–9251, 2007.

G. A. Ascoli, L. Alonso-Nanclares, S. Anderson, G. Barrionuevo, R. Benavides-Piccione, A. Burkhalter, G. Buzsaki, B. Cauli, J. DeFelipe, A. Fairén, et al. Petilla terminology: Nomenclature of features of GABAergic interneurons of the cerebral cortex. *Nature Reviews Neuroscience*, 9(7):557–568, 2008.

G. A. Ascoli, P. Maraver, S. Nanda, S. Polavaram, and R. Armañanzas. Win-win data sharing in neuroscience. *Nature Methods*, 14(2):112–116, 2017.

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.

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(7):990–1001, 2006.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

C. Bielza and P. Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1), 2014a.

C. Bielza and P. Larrañaga. Bayesian networks in neuroscience: A survey. *Frontiers in Computational Neuroscience*, 8:131, 2014b.

C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

B. Bischl, M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus, and L. Kotthoff. *mlr: Machine Learning in R*, 2015. URL http://CRAN.R-project.org/package=mlr. R package version 2.4.

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, Z. Jones, G. Casalicchio, and M. Gallo. *mlr: Machine Learning in R*, 2017. URL https://CRAN.R-project.org/package=mlr. R package version 2.11.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007. ISBN 0387310738.

H. Borchani, C. Bielza, C. Toro, and P. Larrañaga. Predicting human immunodeficiency virus inhibitors using multi-dimensional Bayesian network classifiers. *Artificial Intelligence in Medicine*, 57(3):219–229, 2013.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

S. G. Bottcher and C. Dethlefsen. *deal: Learning Bayesian Networks with Mixed Variables*, 2013. URL https://CRAN.R-project.org/package=deal. R package version 1.2-37.

R. Bouckaert. Bayesian network classifiers in Weka. Technical Report 14/2004, 2004.

R. R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, Universiteit Utrecht, 1995.

A.-L. Boulesteix, S. Janitza, J. Kruppa, and I. R. König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507, 2012.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Wadsworth, New York, NY, USA, 1984.

C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

R. Cannon, D. Turner, G. Pyapali, and H. Wheal. An on-line archive of reconstructed hippocampal neurons. *Journal of Neuroscience Methods*, 84(1–2):49 – 54, 1998. ISSN 0165-0270.

A. Carvalho, T. Roos, A. Oliveira, and P. Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 12:2181–2210, 2011.

B. Cauli, E. Audinat, B. Lambolez, M. C. Angulo, N. Ropert, K. Tsuzuki, S. Hestrin, and J. Rossier. Molecular and physiological diversity of cortical nonpyramidal cells. *The Journal of Neuroscience*, 17(10):3894–3906, 1997.

B. Cauli, J. T. Porter, K. Tsuzuki, B. Lambolez, J. Rossier, B. Quenet, and E. Audinat. Classification of fusiform neocortical interneurons based on unsupervised clustering. *Proceedings of the National Academy of Sciences*, 97(11):6144–6149, 2000.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. The MIT Press, Cambridge, MA, 2006.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

J. Cheng, R. Greiner, J. Kelly, D. A. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

D. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002a.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002b.

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.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

C. Christin, H. C. Hoefsloot, A. K. Smilde, B. Hoekman, F. Suits, R. Bischoff, and P. Horvatovich. A critical assessment of feature selection methods for biomarker discovery in clinical proteomics. *Molecular & Cellular Proteomics*, 12(1):263–276, 2013.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

A. R. Crossman. *Neuroanatomy: An Illustrated Colour Text.* Churchill Livingstone/Elsevier, Edinburgh New York, 2010.

A. A. Curley and D. A. Lewis. Cortical basket cell dysfunction in schizophrenia. *The Journal of Physiology*, 590(4):715–724, 2012.

D. Dash and G. F. Cooper. Exact model averaging with naive Bayesian classifiers. In *19th International Conference on Machine Learning (ICML-2002)*, pages 91–98, 2002.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979.

J. DeFelipe. Neocortical neuronal diversity: Chemical heterogeneity revealed by colocalization studies of classic neurotransmitters, neuropeptides, calcium-binding proteins, and cell surface molecules. *Cerebral Cortex*, 3(4):273–289, 1993.

J. DeFelipe. Chandelier cells and epilepsy. *Brain*, 122(10):1807–1822, 1999.

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.

J. DeFelipe, H. Markram, and K. Rockland. The neocortical column. *Frontiers in Neuroanatomy*, 6(22):1–2, 2012.

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. Wang, 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.

J. Del Sagrado and S. Moral. Qualitative combination of Bayesian networks. *International Journal of Intelligent Systems*, 18(2):237–249, 2003.

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):1–38, 1977.

D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

R. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, 1995.

S. Druckmann, S. Hill, F. Schürmann, H. Markram, and I. Segev. A hierarchical structure of cortical interneuron electrical diversity revealed by automated statistical analysis. *Cerebral Cortex*, 23(12):2994–3006, 2013.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Edition.* Wiley-Interscience, New York, 2000.

D. Dumitriu, R. Cossart, J. Huang, and R. Yuste. Correlation between axonal morphologies and synaptic input kinetics of interneurons from mouse visual cortex. *Cerebral Cortex*, 17 (1):81–91, 2007.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979.

A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

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.

B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis.* Wiley, 5th edition, 2011.

A. Fairen, J. DeFelipe, and J. Regidor. Nonpyramidal neurons: General account. *Cerebral Cortex*, 1:201–253, 1984.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan Kaufmann, 1993.

D. Feldmeyer, G. Qi, V. Emmenegger, and J. F. Staiger. Inhibitory interneurons and their circuit motifs in the many layers of the barrel cortex. *Neuroscience*, 368(Supplement C): 132 – 151, 2018.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

E. Fix and J. L. Hodges. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3): pp. 238–247, 1989.

G. Forman and M. Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.

B. Frénay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

E. M. Glaser and N. T. McMullen. The fan-in projection method for analyzing dendrite and axon systems. *Journal of Neuroscience Methods*, 12(1):37–42, 1984.

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.

F. Glover and M. Laguna. Tabu Search*. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 3261–3362. Springer, New York, NY, 2013.

A. Golugula, G. Lee, and A. Madabhushi. Evaluating feature selection strategies for high dimensional, small sample size datasets. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 949–952, 2011.

S. Grillner, N. Ip, C. Koch, W. Koroshetz, H. Okano, M. Polachek, M. Poo, and T. J. Sejnowski. Worldwide initiatives to advance brain research, 2016.

D. Grossman and P. Domingos. Learning Bayesian Network classifiers by maximizing conditional likelihood. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 361–368, 2004.

L. Guerra, L. M. McGarry, V. Robles, C. Bielza, P. Larrañaga, and R. Yuste. Comparison between supervised and unsupervised classifications of neuronal cell types: A case study. *Developmental Neurobiology*, 71(1):71–82, 2011.

L. Guerra, R. Benavides-Piccione, C. Bielza, V. Robles, J. DeFelipe, and P. Larrañaga. Semi-supervised projected clustering for classifying GABAergic interneurons. In N. Peek, R. Marín Morales, and M. Peleg, editors, *Artificial Intelligence in Medicine*, volume 7885 of *Lecture Notes in Computer Science*, pages 156–165. Springer, Berlin, 2013a.

L. Guerra, C. Bielza, V. Robles, and P. Larrañaga. Semi-supervised projected model-based clustering. *Data Mining and Knowledge Discovery*, 28(4):1–36, 2013b.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction: Foundations and Applications*. Springer-Verlag, Berlin, Germany, 2006.

M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

M. Hall. A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, 20(2):120–126, 2007.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

D. J. Hand and K. Yu. Idiot's Bayes - not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

K. D. Hansen, J. Gentry, L. Long, R. Gentleman, S. Falcon, F. Hahne, and D. Sarkar. *Rgraphviz: Provides plotting capabilities for R graph objects*, 2017. R package version 2.20.0.

K. D. Harris and T. D. Mrsic-Flogel. Cortical connectivity and sensory coding. *Nature*, 503 (7474):51, 2013.

T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, New York, NY, USA, 2009.

H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

K. Hechenbichler and K. Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. Technical Report Discussion paper 399, SFB 386, Ludwig-Maximilians University, Munich, 2004.

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.

M. Helmstaedter, B. Sakmann, and D. Feldmeyer. L2/3 interneuron groups defined by multiparameter analysis of axonal projection, dendritic geometry, and electrical excitability. *Cerebral Cortex*, 19(4):951–962, 2009a.

M. Helmstaedter, 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, 2009b.

M. Helmstaedter, B. Sakmann, and D. Feldmeyer. Neuronal correlates of local, lateral, and translaminar inhibition with reference to cortical columns. *Cerebral Cortex*, 19(4):926–937, 2009c.

S. Højsgaard. Graphical independence networks with the `gRain` package for `R`. *Journal of Statistical Software*, 46(10):1–26, 2012.

K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: `R` meets `Weka`. *Computational Statistics*, 24(2):225–232, 2009.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

C.-W. Hsu, C.-C. Chang, and Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.

Z. J. Huang and L. Luo. It takes the world to understand the brain. *Science*, 350(6256): 42–44, 2015.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

R. F. Hunt, K. M. Girskis, J. L. Rubenstein, A. Alvarez-Buylla, and S. C. Baraban. GABA progenitors grafted into the adult epileptic brain control seizures and abnormal behavior. *Nature Neuroscience*, 16(6):692, 2013.

M. D. Hurd, P. Martorell, A. Delavande, K. J. Mullen, and K. M. Langa. Monetary costs of dementia in the united states. *New England Journal of Medicine*, 368(14):1326–1334, 2013.

S. Højsgaard. *gRain: Graphical Independence Networks*, 2016. URL `https://CRAN.R-project.org/package=gRain`. R package version 1.3-0.

M. Inan, T. J. Petros, and S. A. Anderson. Losing your inhibition: Linking cortical GABAergic interneurons to schizophrenia. *Neurobiology of Disease*, 53:36–48, 2013.

D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485, 2012.

T. R. Insel, S. C. Landis, and F. S. Collins. The NIH BRAIN initiative. *Science*, 340(6133): 687–688, 2013. doi: 10.1126/science.1239276.

D. Jaeger. Accurate reconstruction of neuronal morphology. In E. D. Schutter, editor, *Computational Neuroscience: Realistic Modeling for Experimentalists*, pages 159–178. CRC Press, Boca Raton, FL, USA, 2010.

M. Jaeger. Probabilistic classifiers and the concept they recognize. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, pages 266–273, 2003.

A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8): 651–666, 2010.

S. Janitza, C. Strobl, and A.-L. Boulesteix. An AUC-based permutation variable importance measure for random forests. *BMC Bioinformatics*, 14(1):119, 2013.

X. Jiang, S. Shen, C. R. Cadwell, P. Berens, F. Sinz, A. S. Ecker, S. Patel, and A. S. Tolias. Principles of connectivity among morphologically defined cell types in adult neocortex. *Science*, 350(6264):aac9462, 2015.

M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

D. Joshi, J. M. Fullerton, and C. S. Weickert. Elevated ErbB4 mRNA is related to interneuron deficit in prefrontal cortex in schizophrenia. *Journal of Psychiatric Research*, 53:125–132, 2014.

E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, 4th edition, 2000.

A. Karagiannis, T. Gallopin, C. Dávid, D. Battaglia, H. Geoffroy, J. Rossier, E. M. Hillman, J. F. Staiger, and B. Cauli. Classification of NPY-expressing neocortical interneurons. *The Journal of Neuroscience*, 29(11):3642–3659, 2009.

Y. Kawaguchi and Y. Kubota. GABAergic cell subtypes and their synaptic connections in rat frontal cortex. *Cerebral Cortex*, 7(6):476–486, 1997.

E. J. Keogh and M. J. Pazzani. Learning the structure of augmented Bayesian classifiers. *International Journal on Artificial Intelligence Tools*, 11(4):587–601, 2002.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97 (1):273–324, 1997.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, Cambridge, MA, USA, 2009.

J.-H. Kong, D. R. Fish, R. L. Rockhill, and R. H. Masland. Diversity of ganglion cells in the mouse retina: Unsupervised morphological classification and its limits. *Journal of Comparative Neurology*, 489(3):293–310, 2005.

W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

M. Kuhn. Building predictive models in `R` using the `caret` package. *Journal of Statistical Software*, 28(5):1–26, 2008.

M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2017. URL https://CRAN.R-project.org/package=caret. R package version 6.0-78.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

C. P. Lam and D. G. Stork. Evaluating classifiers by means of test data with noisy labels. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 513–518, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1630659.1630735.

P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann, 1994.

S. 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.

M. H. Law, M. A. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

F. Leitner, C. Bielza, S. L. Hill, and P. Larrañaga. Data publications correlate with citation impact. *Frontiers in Neuroscience*, 10:419, 2016.

D. A. Lewis. The chandelier neuron in schizophrenia. *Developmental Neurobiology*, 71(1): 118–127, 2011.

N. Li, T.-W. Chen, Z. V. Guo, C. R. Gerfen, and K. Svoboda. A motor cortex circuit for motor planning and movement. *Nature*, 519(7541):51, 2015.

Y. Li, M. Dong, and J. Hua. Simultaneous localized feature selection and model detection for Gaussian mixtures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):953, 2009.

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Chapman & Hall/CRC, Boca Raton, FL, 2007.

C. H. Lo. *Statistical methods for high throughput genomics*. PhD thesis, University of British Columbia, 2009.

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*, 55(1):3–22, 2014.

J. S. S. Lowndes, B. D. Best, C. Scarborough, J. C. Afflerbach, M. R. Frazier, C. C. O'Hara, N. Jiang, and B. S. Halpern. Our path to better science in less time using open data science tools. *Nature Ecology & Evolution*, 1:160, 2017.

H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, (1):50–60, 1947.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 2000.

E. C. Marin, G. S. Jefferis, T. Komiyama, H. Zhu, and L. Luo. Representation of the glomerular olfactory map in the drosophila brain. *Cell*, 109(2):243–255, 2002.

H. Markram. The Human Brain Project. *Scientific American*, 306(6):50–55, 2012.

H. Markram, M. Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg, and C. Wu. Interneurons of the neocortical inhibitory system. *Nature Reviews Neuroscience*, 5(10):793–807, 2004.

H. Markram, E. Muller, S. Ramaswamy, M. Reimann, M. Abdellah, C. Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever, G. Kahou, T. Berger, A. Bilgili, N. Buncic, A. Chalimourda, G. Chindemi, J.-D. Courcol, F. Delalondre, V. Delattre, S. Druckmann, R. Dumusc, J. Dynes, S. Eilemann, E. Gal, M. Gevaert, J.-P. Ghobril, A. Gidon, J. Graham, A. Gupta, V. Haenel, E. Hay, T. Heinis, J. Hernando, M. Hines, L. Kanari, D. Keller, J. Kenyon, G. Khazen, Y. Kim, J. King, Z. Kisvarday, P. Kumbhar, S. Lasserre, J.-V. Le Bé, B. Magalhães, A. Merchán-Pérez, J. Meystre, B. Morrice, J. Muller, A. Muñoz-Céspedes, S. Muralidhar, K. Muthurasa, D. Nachbaur, T. Newton, M. Nolte, A. Ovcharenko, J. Palacios, L. Pastor, R. Perin, R. Ranjan, I. Riachi, J.-R. Rodríguez, J. Riquelme, C. Rössert, K. Sfyrakis, Y. Shi, J. Shillcock, G. Silberberg, R. Silva, F. Tauheed, M. Telefont, M. Toledo-Rodriguez, T. Tränkler, W. Van Geit, J. Díaz, R. Walker, Y. Wang, S. Zaninetta, J. DeFelipe, S. Hill, I. Segev, and F. Schürmann. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456 – 492, 2015.

I. Matzkevich and B. Abramson. The topological fusion of Bayes nets. In *Proceedings of the Eighth international conference on Uncertainty in artificial intelligence*, pages 191–198, Stanford, CA, 1992. Morgan Kaufmann Publishers Inc.

M. J. McGeachie, H.-H. Chang, and S. T. Weiss. CGBayesNets: Conditional Gaussian Bayesian network learning and inference with mixed discrete and continuous data. *PLoS Computational Biology*, 10(6):e1003676, 2014.

N. T. McMullen, E. M. Glaser, and M. Tagamets. Morphometry of spine-free nonpyramidal neurons in rabbit auditory cortex. *Journal of Comparative Neurology*, 222(3):383–395, 1984.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. URL https://CRAN.R-project.org/package=e1071. R package version 1.6-7.

B. Mihaljević, C. Bielza, R. Benavides-Piccione, J. DeFelipe, and P. Larrañaga. Multi-dimensional classification of GABAergic interneurons with Bayesian network-modeled label uncertainty. *Frontiers in Computational Neuroscience*, 8:150, 2014.

B. Mihaljević, R. Benavides-Piccione, C. Bielza, J. DeFelipe, and P. Larrañaga. Bayesian network classifiers for categorizing cortical GABAergic interneurons. *Neuroinformatics*, 13 (2):192–208, 2015a.

B. Mihaljević, R. Benavides-Piccione, L. Guerra, J. DeFelipe, P. Larrañaga, and C. Bielza. Classifying GABAergic interneurons with semi-supervised projected model-based clustering. *Artificial Intelligence in Medicine*, 65(1):49–59, 2015b.

B. Mihaljević, P. Larrañaga, and C. Bielza. Automatic classification of cortical interneuron morphologies. In *Proceedings of the Workshop on Advances and Applications of Data Science & Engineering, Real Academia de Ingenieria*, Madrid, 2016.

B. Mihaljević, C. Bielza, and P. Larrañaga. bnclassify: Learning Bayesian network classifiers. *The R Journal*, 2018a. submitted.

B. Mihaljević, C. Bielza, and P. Larrañaga. Learning Bayesian network classifiers with completed partially directed acyclic graphs. In *9th International Conference on Probabilistic Graphical Models*, 2018b. submitted.

B. Mihaljević, P. Larrañaga, R. Benavides-Piccione, S. Hill, J. DeFelipe, and C. Bielza. Towards a supervised classification of neocortical interneuron morphologies. *BMC Bioinformatics*, 2018c. submitted.

B. Mihaljević, C. Bielza, and P. Larrañaga. *bnclassify: Learning Discrete Bayesian Network Classifiers from Data*, 2018. URL https://CRAN.R-project.org/package=bnclassify. R package version 0.4.0.

M. Minsky. Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, 49:8–30, 1961.

D. Morales, Y. Vives-Gilabert, B. Gómez-Ansón, E. Bengoetxea, P. Larrañaga, C. Bielza, J. Pagonabarraga, J. Kulisevsky, I. Corcuera-Solano, and M. Delfino. Predicting dementia development in Parkinson's disease using Bayesian network classifiers. *Psychiatry Research: NeuroImaging*, 213:92–98, 2013.

V. B. Mountcastle. The columnar organization of the neocortex. *Brain: A Journal of Neurology*, 120(4):701–722, 1997.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, Cambridge, MA, USA, 2012.

R. Nagarajan, M. Scutari, and S. Lébre. *Bayesian Networks in R: with Applications in Systems Biology (Use R!).* Springer, New York, 2013. ISBN 1461464455.

R. Neapolitan. *Learning Bayesian networks.* Prentice Hall, Upper Saddle River, 2004.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naïve bayes. In *Advances in Neural Information Processing Systems 14 (NIPS-2001)*, pages 841–848. MIT Press, 2001.

K. K. Nicodemus, J. D. Malley, C. Strobl, and A. Ziegler. The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinformatics*, 11(1):110, 2010.

J. Nielsen, T. Kocka, and J. Peña. On local optima in learning Bayesian networks. In *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence (UAI-2003)*, pages 435–442. Morgan Kaufmann, 2003.

J. Panico and P. Sterling. Retinal neurons and vessels are not fractal but space-filling. *Journal of Comparative Neurology*, 361(3):479–490, 1995.

R. Parekh and G. A. Ascoli. Neuronal morphology goes digital: A research hub for cellular and system neuroscience. *Neuron*, 77(6):1017–1038, 2013.

M. Pazzani. Constructive induction of Cartesian product attributes. In *Proceedings of the Information, Statistics and Induction in Science Conference*, pages 66–77, 1996.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Francisco, CA, USA, 1988.

J. M. Peña. Finding consensus Bayesian network structures. *Journal of Artificial Intelligence Research*, 42(1):661–687, 2011.

H. Peng, M. Hawrylycz, J. Roskams, S. Hill, N. Spruston, E. Meijering, and G. A. Ascoli. BigNeuron: Large-scale 3D neuron reconstruction from optical microscopy images. *Neuron*, 87(2):252–256, 2015.

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, Stockholm, 1999. Morgan Kaufmann Publishers Inc.

F. Pernkopf and J. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *Journal of Machine Learning Research*, 11:2323–2360, 2010.

F. Pernkopf and P. O'Leary. Floating search algorithm for structure learning of Bayesian network classifiers. *Pattern Recognition Letters*, 24(15):2839–2848, 2003.

A. Peters and E. G. Jones. *Cerebral Cortex: Volume 1: Cellular Components of the Cerebral Cortex*. Plenum Press, New York, NY, USA, 1984.

S. Polavaram, T. A. Gillette, R. Parekh, and G. A. Ascoli. Statistical analysis and data mining of digital reconstructions of dendritic morphologies. *Frontiers in Neuroanatomy*, 8: 138, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL http://www.R-project.org/.

S. Ramaswamy, J.-D. Courcol, M. Abdellah, S. R. Adaszewski, N. Antille, S. Arsever, G. Atenekeng, A. Bilgili, Y. Brukau, A. Chalimourda, G. Chindemi, F. Delalondre, R. Dumusc, S. Eilemann, M. E. Gevaert, P. Gleeson, J. W. Graham, J. B. Hernando, L. Kanari, Y. Katkov, D. Keller, J. G. King, R. Ranjan, M. W. Reimann, C. Rössert, Y. Shi, J. C. Shillcock, M. Telefont, W. Van Geit, J. Villafranca Diaz, R. Walker, Y. Wang, S. M. Zaninetta, J. DeFelipe, S. L. Hill, J. Muller, I. Segev, F. Schürmann, E. B. Muller, and H. Markram. The neocortical microcircuit collaboration portal: A resource for rat somatosensory cortex. *Frontiers in Neural Circuits*, 9:44, 2015. ISSN 1662-5110.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

C. R. Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

V. C. Raykar and S. Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13:491–518, 2012.

V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer, 2009.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

J. Rubenstein and M. M. Merzenich. Model of autism: Increased ratio of excitation/inhibition in key neural systems. *Genes, Brain and Behavior*, 2(5):255–267, 2003.

J. P. Sacha, L. S. Goodenday, and K. J. Cios. Bayesian learning for cardiac spect image interpretation. *Artificial Intelligence in Medicine*, 26(1):109–143, 2002.

M. Sadler and M. Berry. Morphometric study of the development of Purkinje cell dendritic trees in the mouse using vertex analysis. *Journal of Microscopy*, 131(3):341–354, 1983.

Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-1996)*, volume 96, pages 335–338, 1996.

R. Santana, L. M. McGarry, C. Bielza, P. Larrañaga, and R. Yuste. Classification of neocortical interneurons using affinity propagation. *Frontiers in neural circuits*, 7, 2013.

D. Schubert, R. Kötter, and J. F. Staiger. Mapping functional connectivity in barrel-related columns reveals layer- and cell type-specific microcircuits. *Brain Structure and Function*, 212(2):107–119, 2007.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

F. Schwenker and E. Trentin. Pattern classification and clustering: A review of partially supervised learning approaches. *Pattern Recognition Letters*, 37:4–14, 2014.

R. Scorcioni, M. T. Lazarewicz, and G. A. Ascoli. Quantitative morphometry of hippocampal pyramidal cells: Differences between anatomical classes and reconstructing laboratories. *Journal of Comparative Neurology*, 473(2):177–193, 2004.

R. Scorcioni, S. Polavaram, and G. A. Ascoli. L-Measure: A web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nature Protocols*, 3(5):866–876, 2008.

M. Scutari. Learning Bayesian networks with the `bnlearn` `R` package. *Journal of Statistical Software*, 35(3):1–22, 2010.

M. Scutari and R. Ness. *bnlearn: Bayesian Network Structure Learning, Parameter Learning and Inference*, 2018. URL `https://CRAN.R-project.org/package=bnlearn`. R package version 4.3.

D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 2003.

D. Silberberg, N. P. Anand, K. Michels, and R. N. Kalaria. Brain and other nervous system disorders across the lifespan [mdash] global challenges and opportunities. *Nature*, 527 (7578):S151–S154, 2015.

P. Smialowski, D. Frishman, and S. Kramer. Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440–443, 2010.

R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008.

P. Somogyi. A specific 'axo-axonal' interneuron in the visual cortex of the rat. *Brain Research*, 136(2):345–350, 1977.

A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *First IEEE Workshop on Internet Vision at CVPR*, pages 1–8, Anchorage, AK, 2008.

O. Sporns. *Networks of the brain.* MIT Press, Cambridge, Mass, 2011.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.

C. Strobl and A. Zeileis. Danger: High power!–exploring the statistical properties of a test for random forest variable importance. In *Proceedings of the 8th International Conference on Computational Statistics, Porto, Portugal*, 2008.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307, 2008.

V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: A classification and regression tool for compound classification and QSAR modeling. *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958, 2003.

J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

B. Tasic, V. Menon, T. N. Nguyen, T. K. Kim, T. Jarsky, Z. Yao, B. Levi, L. T. Gray, S. A. Sorensen, T. Dolbeare, D. Bertagnolli, J. Goldy, N. Shapovalova, S. Parry, C. Lee, K. Smith, A. Bernard, L. Madisen, S. M. Sunkin, M. Hawrylycz, C. Koch, and H. Zeng. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature Neuroscience*, 19(2): 335–346, 2016.

T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. URL `https://CRAN.R-project.org/package=rpart`. R package version 4.1-10.

C. Thiel, S. Scherer, and F. Schwenker. Fuzzy-input fuzzy-output one-against-all support vector machines. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4694 of *Lecture Notes in Computer Science*, pages 156–165. Springer, 2007.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

M. Toledo-Rodriguez, P. Goodman, M. Illic, C. Wu, and H. Markram. Neuropeptide and calcium-binding protein gene expression profiles predict neuronal anatomical type in the juvenile rat. *The Journal of Physiology*, 567(2):401–413, 2005.

R. Tremblay, S. Lee, and B. Rudy. GABAergic interneurons in the neocortex: From cellular properties to circuits. *Neuron*, 91(2):260–292, 2016.

I. Tsamardinos and C. F. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann Publishers: Key West, FL, USA, 2003.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Algorithms for large scale Markov blanket discovery. In *Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2003)*, pages 376–381. AAAI Press, 2003.

A. Tsiola, F. Hamzei-Sichani, Z. Peterlin, and R. Yuste. Quantitative morphologic classification of layer 5 neurons from mouse primary visual cortex. *Journal of Comparative Neurology*, 461(4):415–428, 2003.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

H. B. Uylings and J. Van Pelt. Measures for quantifying dendritic arborizations. *Network: Computation in Neural Systems*, 13(3):397–414, 2002.

L. C. van der Gaag and P. R. de Waal. Multi-dimensional Bayesian network classifiers. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models (PGM-2006)*, pages 107–114, Prague, 2006.

J. Van Pelt, H. B. Uylings, R. W. Verwer, R. J. Pentney, and M. J. Woldenberg. Tree asymmetry: A sensitive and practical measure for binary topological trees. *Bulletin of Mathematical Biology*, 54(5):759–784, 1992.

G. Varando, C. Bielza, and P. Larrañaga. Decision boundary for discrete Bayesian network classifiers. *Journal of Machine Learning Research*, (16):2725–2749, 2015.

X. Vasques, L. Vanel, G. Villette, and L. Cif. Morphological neuron classification using machine learning. *Frontiers in Neuroanatomy*, 10:102, 2016.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Springer, New York, NY, USA, 2002.

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(5): 629–641, Sep 1990.

E. R. Walker, R. E. McGee, and B. G. Druss. Mortality in mental disorders and global disease burden implications: a systematic review and meta-analysis. *JAMA Psychiatry*, 72 (4):334–341, 2015.

M. Wang, X. Chen, and H. Zhang. Maximal conditional chi-square importance in random forests. *Bioinformatics*, 26(6):831–837, 2010.

Y. Wang, A. Gupta, M. Toledo-Rodriguez, C. Z. 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.

Y. Wang, M. Toledo-Rodriguez, A. Gupta, C. Wu, G. Silberberg, J. Luo, and H. Markram. Anatomical, physiological and molecular properties of Martinotti cells in the somatosensory cortex of the juvenile rat. *The Journal of Physiology*, 561(1):65–90, 2004.

M. Wasikowski and X. Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1388–1400, 2010.

G. I. Webb, J. R. Boughton, and Z. Wang. Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, pages 2424–2432, 2010.

E. White. *Cortical Circuits: Synaptic Organization of the Cerebral Cortex Structure, Function, and Theory.* Birkhäuser, Boston, MA, USA, 1989.

J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, volume 22, pages 2035–2043, Vancouver, BC, 2009.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

Y. Yang and G. I. Webb. Weighted proportional k-interval discretization for naive-Bayes classifiers. In *Advances in Knowledge Discovery and Data Mining*, pages 501–512. Springer, 2003.

Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In *Data Mining and Knowledge Discovery Handbook*, pages 101–116. Springer, 2010.

J. Yelnik, G. Percheron, C. Francois, and Y. Burnod. Principal component analysis: A suitable method for the 3-dimensional study of the shape, dimensions and orientation of dendritic arborizations. *Journal of Neuroscience Methods*, 9(2):115–125, 1983.

N. A. Zaidi, J. Cerquides, M. J. Carman, and G. I. Webb. Alleviating naive Bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 14:1947–1988, 2013.

N. A. Zaidi, G. I. Webb, M. J. Carman, F. Petitjean, W. Buntine, M. Hynes, and H. De Sterck. Efficient parameter learning of Bayesian network classifiers. *Machine Learning*, 106(9): 1289–1329, 2017.

H. Zeng and J. R. Sanes. Neuronal cell-type classification: Challenges, opportunities and the path forward. *Nature Reviews Neuroscience*, 18(9):530–546, 2017.

F. Zheng and G. I. Webb. Efficient lazy elimination for averaged one-dependence estimators. In *Proceedings of the 23rd International Conference on Machine Learning*, volume 148, pages 1113–1120. ACM, 2006.

X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.