DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

Facultad de Informática
Universidad Politécnica de Madrid

PhD THESIS

Multi-dimensional classification using Bayesian networks for stationary and evolving streaming data

Author

Hanen Borchani
MS Computer Science Applied to Management

Supervisors

Concha Bielza
PhD Computer Science

Pedro Larrañaga
PhD Computer Science

2012

Thesis Committee

President: José A. Gámez

External member: João Gama

Member: Iñaki Inza

Member: Jesse Read

Secretary: Luis P. Guerra

Acknowledgements

It has been a journey shared with many people who supported and inspired me in many ways.

First and foremost, I am grateful to my supervisors Concha Bielza and Pedro Larrañaga, for their generosity and help since the first day I came to Madrid. I am deeply indebted to them for their insightful suggestions, constructive corrections, and endless support.

I would also like to thank my colleagues in the Computational Intelligence Group with whom I have shared this journey: Pedro Luis López-Cruz, Hossein Karshenas, Luis Guerra, Alfonso Ibáñez, Diego Vidaurre, and Rubén Armañanzas. My gratitude goes as well to colleagues that spent some time with us in the laboratory: Roberto Santana, Dinora Morales, Rubén Granados, and Guangdi Li.

Sincere and special thanks to Pavel Brazdil and João Gama for their hospitality during my three months research stay at the Laboratory of Artificial Intelligence and Decision Support (LIAAD) of the University of Porto, Portugal. I am grateful for their constant help and invaluable advices and discussions. Warm thanks to all people in LIAAD especially Márcia Oliveira, Ezilda Almeida, Odair Tavares, Petr Kosina, Hadi Fanaee, as well as to friends I've met there: Elaine Faria, José G. Paiva, Zaigham F. Siddiqui, and Iñigo Mendialdua.

Moreover, I am grateful to the Spanish Ministry of Economy and Competitiveness for the offered FPI fellowship (BES-2008-003901) under TIN2007-62626 and for my participation in TIN2010-20900-C04-04 projects. Thanks also to many people for their collaboration in this work and for their financial support under several research projects, namely: 1) Panda Security, special thanks to Etor Llona for his help in dealing with the malware detection problem; 2) Dynamo Project (FONCICYT, European Union and México), special thanks to Enrique Sucar and Alma Ríos for their collaboration in understanding the HIV-1 problem; and 3) Abbott Products Operations AG and CIEN Foundation, special thanks to Birgit Gradl and Pablo Martínez-Martín for their support in dealing with the Parkinson's disease problem.

Finally, my deepest gratitude goes to my mother and my father, who raised me well. Though being far away from them during all these years, their permanent encouragement and trust have been of great support. I dedicate this dissertation to them, as well as to my sister, my brothers, my friends all over the world, and for sure to my homeland Tunisia.

Abstract

Nowadays, with the ongoing and rapid evolution of information technology and computing devices, large volumes of data are continuously collected and stored in different domains and through various real-world applications. Extracting useful knowledge from such a huge amount of data usually cannot be performed manually, and requires the use of adequate machine learning and data mining techniques.

Classification is one of the most important techniques that has been successfully applied to several areas. Roughly speaking, classification consists of two main steps: first, learn a classification model or classifier from an available training data, and secondly, classify the new incoming unseen data instances using the learned classifier. Classification is *supervised* when the whole class values are present in the training data (i.e., fully labeled data), *semi-supervised* when only some class values are known (i.e., partially labeled data), and *unsupervised* when the whole class values are missing in the training data (i.e., unlabeled data). In addition, besides this taxonomy, the classification problem can be categorized into *uni-dimensional* or *multi-dimensional* depending on the number of class variables, one or more, respectively; or can be also categorized into *stationary* or *streaming* depending on the characteristics of the data and the rate of change underlying it.

Through this thesis, we deal with the classification problem under three different settings, namely, supervised multi-dimensional stationary classification, semi-supervised uni-dimensional streaming classification, and supervised multi-dimensional streaming classification. To accomplish this task, we basically used Bayesian network classifiers as models.

The first contribution, addressing the supervised multi-dimensional stationary classification problem, consists of two new methods for learning multi-dimensional Bayesian network classifiers from stationary data. They are proposed from two different points of view. The first method, named `CB-MBC`, is based on a wrapper greedy forward selection approach, while the second one, named `MB-MBC`, is a filter constraint-based approach based on Markov blankets. Both methods are applied to two important real-world problems, namely, the prediction of the human immunodeficiency virus type 1 (HIV-1) reverse transcriptase and protease inhibitors, and the prediction of the European Quality of Life-5 Dimensions (EQ-5D) from 39-item Parkinson's Disease Questionnaire (PDQ-39). The experimental study includes comparisons of `CB-MBC` and `MB-MBC` against state-of-the-art multi-dimensional classification methods, as well as against commonly used methods for solving the Parkinson's disease prediction problem, namely, multinomial logistic regression, ordinary least squares, and censored least absolute deviations. For both considered case studies, results are promising in terms of classification

accuracy as well as regarding the analysis of the learned MBC graphical structures identifying known and novel interactions among variables.

The second contribution, addressing the semi-supervised uni-dimensional streaming classification problem, consists of a novel method (`CPL-DS`) for classifying partially labeled data streams. Data streams differ from the stationary data sets by their highly rapid generation process and their concept-drifting aspect. That is, the learned concepts and/or the underlying distribution are likely changing and evolving over time, which makes the current classification model out-of-date requiring to be updated. `CPL-DS` uses the Kullback-Leibler divergence and bootstrapping method to quantify and detect three possible kinds of drift: feature, conditional or dual. Then, if any occurs, a new classification model is learned using the expectation-maximization algorithm; otherwise, the current classification model is kept unchanged. `CPL-DS` is general as it can be applied to several classification models. Using two different models, namely, naive Bayes classifier and logistic regression, `CPL-DS` is tested with synthetic data streams and applied to the real-world problem of malware detection, where the new received files should be continuously classified into malware or goodware. Experimental results show that our approach is effective for detecting different kinds of drift from partially labeled data streams, as well as having a good classification performance.

Finally, the third contribution, addressing the supervised multi-dimensional streaming classification problem, consists of two adaptive methods, namely, `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) and `Globally Adaptive-MB-MBC` (`GA-MB-MBC`). Both methods monitor the concept drift over time using the average log-likelihood score and the Page-Hinkley test. Then, if a drift is detected, `LA-MB-MBC` adapts the current multi-dimensional Bayesian network classifier locally around each changed node, whereas `GA-MB-MBC` learns a new multi-dimensional Bayesian network classifier from scratch. Experimental study carried out using synthetic multi-dimensional data streams shows the merits of both proposed adaptive methods.

Resumen

Hoy en día, con la evolución continua y rápida de las tecnologías de la información y los dispositivos de computación, se recogen y almacenan continuamente grandes volúmenes de datos en distintos dominios y a través de diversas aplicaciones del mundo real. La extracción de conocimiento útil de una cantidad tan enorme de datos no se puede realizar habitualmente de forma manual, y requiere el uso de técnicas adecuadas de aprendizaje automático y de minería de datos.

La clasificación es una de las técnicas más importantes que ha sido aplicada con éxito a varias áreas. En general, la clasificación se compone de dos pasos principales: en primer lugar, aprender un modelo de clasificación o clasificador a partir de un conjunto de datos de entrenamiento, y en segundo lugar, clasificar las nuevas instancias de datos utilizando el clasificador aprendido. La clasificación es *supervisada* cuando todas las etiquetas están presentes en los datos de entrenamiento (es decir, datos completamente etiquetados), *semi-supervisada* cuando sólo algunas etiquetas son conocidas (es decir, datos parcialmente etiquetados), y *no supervisada* cuando todas las etiquetas están ausentes en los datos de entrenamiento (es decir, datos no etiquetados). Además, aparte de esta taxonomía, el problema de clasificación se puede categorizar en *unidimensional* o *multidimensional* en función del número de variables clase, una o más, respectivamente; o también puede ser categorizado en *estacionario* o *cambiante con el tiempo* en función de las características de los datos y de la tasa de cambio subyacente.

A lo largo de esta tesis, tratamos el problema de clasificación desde tres perspectivas diferentes, a saber, clasificación supervisada multidimensional estacionaria, clasificación semi-supervisada unidimensional cambiante con el tiempo, y clasificación supervisada multidimensional cambiante con el tiempo. Para llevar a cabo esta tarea, hemos usado básicamente los clasificadores Bayesianos como modelos.

La primera contribución, dirigiéndose al problema de clasificación supervisada multidimensional estacionaria, se compone de dos nuevos métodos de aprendizaje de clasificadores Bayesianos multidimensionales a partir de datos estacionarios. Los métodos se proponen desde dos puntos de vista diferentes. El primer método, denominado `CB-MBC`, se basa en una estrategia de envoltura de selección de variables que es voraz y hacia delante, mientras que el segundo, denominado `MB-MBC`, es una estrategia de filtrado de variables con una aproximación basada en restricciones y en el manto de Markov. Ambos métodos han sido aplicados a dos problemas reales importantes, a saber, la predicción de los inhibidores de la transcriptasa inversa y de la proteasa para el problema de infección por el virus de la inmunodeficiencia humana tipo 1 (HIV-1), y la predicción del European Quality of Life-5 Dimensions (EQ-5D) a

partir de los cuestionarios de la enfermedad de Parkinson con 39 ítems (PDQ-39). El estudio experimental incluye comparaciones de `CB-MBC` y `MB-MBC` con los métodos del estado del arte de la clasificación multidimensional, así como con métodos comúnmente utilizados para resolver el problema de predicción de la enfermedad de Parkinson, a saber, la regresión logística multinomial, mínimos cuadrados ordinarios, y mínimas desviaciones absolutas censuradas. En ambas aplicaciones, los resultados han sido prometedores con respecto a la precisión de la clasificación, así como en relación al análisis de las estructuras gráficas que identifican interacciones conocidas y novedosas entre las variables.

La segunda contribución, referida al problema de clasificación semi-supervisada unidimensional cambiante con el tiempo, consiste en un método nuevo (`CPL-DS`) para clasificar flujos de datos parcialmente etiquetados. Los flujos de datos difieren de los conjuntos de datos estacionarios en su proceso de generación muy rápido y en su aspecto de cambio de concepto. Es decir, los conceptos aprendidos y/o la distribución subyacente están probablemente cambiando y evolucionando en el tiempo, lo que hace que el modelo de clasificación actual sea obsoleto y deba ser actualizado. `CPL-DS` utiliza la divergencia de Kullback-Leibler y el método de *bootstrapping* para cuantificar y detectar tres tipos posibles de cambio: en las predictoras, en la a posteriori de la clase o en ambas. Después, si se detecta cualquier cambio, un nuevo modelo de clasificación se aprende usando el algoritmo EM; si no, el modelo de clasificación actual se mantiene sin modificaciones. `CPL-DS` es general, ya que puede ser aplicado a varios modelos de clasificación. Usando dos modelos diferentes, el clasificador naive Bayes y la regresión logística, `CPL-DS` se ha probado con flujos de datos sintéticos y también se ha aplicado al problema real de la detección de código *malware*, en el cual los nuevos ficheros recibidos deben ser continuamente clasificados en *malware* o *goodware*. Los resultados experimentales muestran que nuestro método es efectivo para la detección de diferentes tipos de cambio a partir de los flujos de datos parcialmente etiquetados y también tiene una buena precisión de la clasificación.

Finalmente, la tercera contribución, sobre el problema de clasificación supervisada multidimensional cambiante con el tiempo, consiste en dos métodos adaptativos, a saber, `Locally Adpative-MB-MBC` (`LA-MB-MBC`) y `Globally Adpative-MB-MBC` (`GA-MB-MBC`). Ambos métodos monitorizan el cambio de concepto a lo largo del tiempo utilizando la log-verosimilitud media como métrica y el test de Page-Hinkley. Luego, si se detecta un cambio de concepto, `LA-MB-MBC` adapta el actual clasificador Bayesiano multidimensional localmente alrededor de cada nodo cambiado, mientras que `GA-MB-MBC` aprende un nuevo clasificador Bayesiano multidimensional. El estudio experimental realizado usando flujos de datos sintéticos multidimensionales indica los méritos de los métodos adaptativos propuestos.

Résumé

De nos jours, avec l'évolution continue et rapide de la technologie de l'information et des dispositifs informatiques, de grandes quantités de données sont continuellement collectées et stockées dans plusieurs domaines et à travers diverses applications réelles. Généralement, l'extraction de connaissances utiles à partir d'une quantité si énorme de données ne peut pas être réalisée manuellement, et exige l'utilisation de techniques adéquates d'apprentissage automatique et de fouille de données.

La classification est l'une des techniques les plus importantes qui a été appliquée avec succès à plusieurs domaines. En général, la classification se compose de deux étapes principales : premièrement, apprendre un modèle de classification ou classifieur à partir des données d'apprentissage disponibles, et deuxièmement, classifier les nouvelles instances reçues en utilisant le classifieur obtenu. La classification est *supervisée* quand toutes les étiquettes sont présentes dans les données d'apprentissage (c'est-à-dire, données totalement étiquetées), *semi-supervisée* quand seulement quelques étiquettes sont connues (c'est-à-dire, données partiellement étiquetées), et *non-supervisée* quand toutes les étiquettes sont absentes dans les données de d'apprentissage (c'est-à-dire, données totalement non-étiquetées). En outre, en plus de cette taxonomie, le problème de classification peut être catégorisé en *unidimensionnel* ou *multidimensionnel* selon le nombre de variables classe, un ou plusieurs, respectivement; ou peut être également catégorisé en *stationnaire* ou *non-stationnaire* selon les caractéristiques des données et le taux de changement sous-jacent.

A travers cette thèse, nous abordons le problème de classification à partir de trois perspectives différentes, à savoir, classification supervisée multidimensionnelle stationnaire, classification semi-supervisée unidimensionnelle non-stationnaire, et classification supervisée multidimensionnelle non-stationnaire. Pour atteindre ce but, nous avons utilisé fondamentalement les classifieurs Bayésiens comme modèles.

La première contribution, traitant le problème de classification supervisée multidimensionnelle stationnaire, consiste en deux méthodes nouvelles pour l'apprentissage des classifieurs Bayésiens multidimensionnels à partir de données stationnaires. Ces méthodes sont proposées selon deux points de vue différents. La première méthode, nommée `CB-MBC`, est basée sur une approche gloutonne *wrapper* avec sélection incrémentale, alors que la seconde, nommée `MB-MBC`, est une approche *filter* d'apprentissage sous contraintes basé sur les couvertures de Markov. Les deux méthodes sont appliquées à deux problèmes réels importants, à savoir, la prédiction des inhibiteurs de la transcriptase inverse et de la protéase du virus de l'immunodéficience humaine de type 1 (HIV-1), et la prédiction du score Européen de Qualité de vie en 5 dimensions (EQ-5D) à partir du questionnaire d'évaluation de la mal-

adie de Parkinson avec 39 items (PDQ-39). L'étude expérimentale inclut des comparaisons de `CB-MBC` et `MB-MBC` avec des méthodes de l'état de l'art de la classification multidimensionnelle, ainsi qu'avec des méthodes communément utilisées pour la résolution du problème de la prédiction de la maladie de Parkinson, à savoir, la régression logistique multinomiale, les moindres carrés ordinaires, et les moindres déviations absolues censurées. Pour les deux études de cas considérées, les résultats sont prometteurs en termes d'exactitude de classification ainsi que par rapport à l'analyse des structures graphiques qui permettent l'identification d'interactions connues et nouvelles entre les variables.

La deuxième contribution, traitant le problème de classification semi-supervisée unidimensionnelle non-stationnaire, consiste en une nouvelle méthode (`CPL-DS`) pour classifier les flux de données partiellement étiquetés. Les flux de données diffèrent des bases de données stationnaires par leur processus de génération très rapide et leur aspect de changement de concept. C'est-à-dire, les concepts appris et/ou la distribution sous-jacente changent et évoluent avec le temps, ce qui fait que le modèle de classification actuel devient obsolète et doit être mis à jour. `CPL-DS` utilise la divergence de Kullback-Leibler et la méthode de bootstrapping à fin de quantifier et détecter trois types possibles de changement : changement de la distribution des attributs, changement de la distribution conditionnelle de la classe, ou les deux. Ensuite, si un changement quelconque se produit, un nouveau modèle de classification est appris en utilisant l'algorithme EM; si non, le modèle de classification actuel est maintenu sans modifications. `CPL-DS` est général et peut être appliqué à plusieurs modèles de classification. En utilisant deux modèles différents, à savoir, le classifieur Bayésien naïf et la régression logistique, `CPL-DS` est testé avec des flux de données synthétiques et appliqué aussi au problème réel de la détection de *malware*, où les nouveaux dossiers reçus doivent être continuellement classifiés en *malware* ou *goodware*. Les résultats expérimentaux prouvent que notre approche est efficace pour la détection de différents types de changement à partir de flux de données partiellement étiquetés, et dispose aussi d'une bonne exactitude de classification.

Finalement, la troisième contribution, se référant au problème de classification supervisée multidimensionnelle non-stationnaire, se compose de deux méthodes adaptatives, à savoir `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) et `Globally Adaptive-MB-MBC` (`GA-MB-MBC`). Les deux méthodes contrôlent le changement de concept au fil du temps en utilisant la log-vraisemblance moyenne et le test de Page-Hinkley. Ensuite, si un changement de concept est détecté, `LA-MB-MBC` adapte le classifieur Bayésien multidimensionnel actuel localement autour de chaque nœud changé, tandis que `GA-MB-MBC` apprend un nouveau classifieur Bayésien multidimensionnel. L'étude expérimentale effectuée avec des flux de données multidimensionnels synthétiques montre les apports des deux méthodes adaptatives proposées.

Contents

List of Figures

List of Tables

Acronyms

BN	Bayesian network
BS	Bayesian score
CLAD	Censored least absolute deviation
CPT	Conditional probability table
DAG	Directed acyclic graph
EQ-5D	European quality of life-5 dimensions
EM	Expectation-maximization
HIV-1	Human immunodeficiency virus type 1
GS	Greedy search
KL	Kullback-Leibler
k**DB**	k-dependence Bayesian classifier
LL	Log-likelihood
LR	Logistic regression
MAP	Maximum a posteriori
MB	Markov blanket
MBC	Multi-dimensional Bayesian network classifier
MDL	Minimum description length
MPE	Most probable explanation
NB	Naive Bayes
NNRTI	Non-nucleoside reverse transcriptase inhibitors
NRTI	Nucleoside and nucleotide reverse transcriptase inhibitors
OLS	Ordinary least squares
PC	Parents-children
PD	Parkinson's disease
PDQ	Parkinson's disease questionnaire
PH	Page-Hinkley
PI	Protease inhibitors
RTI	Reverse transcriptase inhibitors
SNB	Selective naive Bayes
TAN	Tree augmented naive Bayes

Notations

The list below presents a summary of the most frequently used notations:

$\mathcal{B}$	Bayesian network
$\mathcal{G}$	Directed acyclic graph
$\mathbf{V}$	Set of vertices
$\mathbf{A}$	Set of arcs
$\mathbf{\Theta}$	Set of parameters
C	Class variable
d	Number of class variables
r_C	Range of possible values of C
Ω_C	Sample space of variable C
w	Number of maximal connected components
X	Feature variable
m	Number of feature variables
r_X	Range of possible values of X
Ω_X	Sample space of variable X
n	Total number of variables, i.e., $n = d + m$
i	Feature variable index
j	Class variable index
l	Instance index
N	Total number of instances
$\mathcal{D}^s$	Data stream at step s
N^s_{fl}	Number of labeled instances at step s
N^s_{ul}	Number of unlabeled instances at step s
P	Probability distribution
$\mathbf{Pa}(.)$	Parent set
$\mathbf{Ch}(.)$	Children set

Part I

Introduction

Chapter 1

Introduction

This dissertation aims to contribute to the state-of-the-art of both multi-dimensional classification and data stream mining. It provides new algorithms for learning multi-dimensional Bayesian network classifiers from stationary data, and deals with mining data streams in both uni-dimensional and multi-dimensional classification settings. In the following sections, we briefly present the main contributions as well as the full thesis overview pointing to the chapters where each contribution is introduced and discussed.

1.1 Contributions of the dissertation

The main contributions of this dissertation are presented in two parts. The first part discusses the contributions related to learning multi-dimensional Bayesian network classifiers from stationary data, whereas the second part is dedicated to mining uni- and multi-dimensional concept-drifting data streams.

1.1.1 Learning multi-dimensional Bayesian network classifiers from stationary data set

Multi-dimensional classification is defined as an extension of the classical uni-dimensional classification where each instance is associated with not only one class variable but with a set of class variables. The aim is to learn a classifier from the available multi-dimensional stationary data set, then use it subsequently to predict the set of class variables for each unseen instance. In the case where all class variables are binary, the multi-dimensional classification problem is also known in the literature as multi-label classification.

The state-of-the-art approaches dealing with multi-dimensional classification are generally categorized into two main categories: *problem transformation* and *algorithm adaptation*. Problem transformation methods are algorithm independent and proceed by transforming the multi-dimensional classification task into one or more uni-dimensional classification tasks, whereas algorithm adaptation methods extend specific uni-dimensional learning algorithms in order to handle directly the multi-dimensional data sets.

For Bayesian networks, problem transformation methods can either opt for: (a) building a compound class variable that models all possible combinations of the class variables and then applying any existing uni-dimensional Bayesian network classifier; or (b) learning multiple uni-dimensional Bayesian network classifiers, one for each individual class variable as if all of them were independent. In the former case, the class variable can easily ends up with a huge number of values which may weakness the predictive performance of such a classifier. However, in the latter case, by building multiple classifiers, the interactions among the various class variables and their simultaneous interactions with feature variables cannot be detected, which does not represent accurately the multi-dimensional classification problem.

In order to deal properly with this problem, the so-called multi-dimensional Bayesian networks classifiers (MBCs) have been recently proposed and defined as an algorithm adaptation method that extends the general Bayesian networks to the multi-dimensional classification problem [222]. An MBC is a Bayesian network that includes one or more class variables and one or more feature variables. It models the probabilistic (in)dependence relationships between the variables through its *graphical structure*, partitioning the set of class and feature variables into three different subgraphs: *class subgraph* representing the dependence relationships between class variables, *bridge subgraph* representing the dependence relationships between class and feature variables, and *feature subgraph* representing the dependence relationships between feature variables; and through its second component, the *parameters*, it defines the conditional probability distribution of each variable given the set of its parents in the graphical structure [13, 222].

As Bayesian networks, the learning problem from stationary data sets consists of finding an MBC that best fits the available data. Various MBC learning methods have been recently defined [13, 61, 177, 188, 222, 237, 238] either based on filter, wrapper or hybrid approaches. Moreover, some of these proposed MBC learning methods include restrictions on the structure of the class and feature subgraphs, while others allow any kind of structure.

Our aim in this part is to tackle some shortcomings of the existing works found in the literature such as those related to the most probable explanation computational burden or the structure learning strategy, including, as mentioned above, the restrictions on the structure

of the class and feature subgraphs. Basically, we propose in this part two novel and different algorithms for learning MBCs from stationary data sets:

- The first one, named class-bridge decomposable multi-dimensional Bayesian network classifiers (`CB-MBC`), is based on a wrapper greedy forward selection approach, optimizing at each step the accuracy of the model given the training data set. `CB-MBC` firstly learns an initial bridge subgraph with a number of maximal connected components equal to the number of class variables, then it induces an initial feature subgraph. Next, as long as the number of maximal connected components is greater than one and there is an accuracy improvement, it iteratively and sequentially merges together two components and updates the bridge and feature subgraphs accordingly.

- The second algorithm, named Markov blanket-based multi-dimensional Bayesian network classifiers (`MB-MBC`), is a filter constraint-based approach and is motivated by the fact that the classification is unaffected by parts of the structure that lie outside the Markov blankets of the class variables. `MB-MBC` builds MBCs by firstly identifying the Markov blanket around each class variable using the HITON algorithm [4, 5], then specifying the directionality over the MBC subgraphs.

Both algorithms are compared against the state-of-the-art MBC learning methods using both synthetic and real-world stationary data sets. Specifically, both algorithms are applied to two important real-world multi-dimensional problems, namely:

- The prediction of the human immunodeficiency virus type 1 (HIV-1) reverse transcriptase and protease inhibitors: this application is supported by the Dynamo project (FONCICYT, European Union and México) and has as objective the prediction of antiretroviral combination therapies given an input set of resistance mutations, that a HIV-1 patient carries, to treat the HIV-1 infection. The analyzed multi-dimensional data consists of both reverse transcriptase and protease inhibitor data sets obtained from the Stanford HIV-1 database [185].

- The prediction of the European Quality of Life-5 Dimensions (EQ-5D) from 39-item Parkinson's Disease Questionnaire (PDQ-39): this application is carried out in a collaboration with Abbott Products Operations AG and CIEN Foundation. Basically, the aim is to use MBCs to predict the EQ-5D five class values from the 39-item Parkinson's disease questionnaire, and determine thereby the general health state of Parkinson's disease patients. The analyzed multi-dimensional Parkinson's disease data set was ob-

tained from an international multipurpose database collected by the National Center of Epidemiology, Carlos III Institute of Health, Madrid.

1.1.2 Mining uni- and multi-dimensional evolving data streams

A data stream environment has different characteristics from the stationary setting discussed in the previous section. In the traditional stationary classification problem, the whole stored training data set is used to induce a classifier or a decision model that will never be revised and will be valid to classify all the new incoming instances over time. The rationale behind this is that the distribution underlying the stationary data is assumed to be stable and does not change over time. However, the data stream classification problem presents more challenges due to its infinite length and evolving nature. In fact, the data is generally collected over an extended period of time, its underlying distribution is likely to change, and thus the decision models should be able to detect and adapt to such a change.

In recent years, the field of mining evolving data streams has received an increasing attention and a plethora of approaches have been developed and applied to a wide range of real-world applications [1, 14, 86, 130, 220, 231, 244]. However, most of these approaches are based on mining fully labeled uni-dimensional data streams. In fact, they assume that the true labels are entirely available in the data streams and they are limited to the uni-dimensional setting where only a single class variable needs to be predicted.

In this part of the thesis, the aim is to deal with these two limitations. On the one hand, a novel algorithm (`CPL-DS`) for classifying partially labeled uni-dimensional evolving data streams is proposed for the case where the assumption of entirely labeled data streams availability is violated and labels are either scarce and/or not readily available. On the other hand, streaming multi-dimensional classification is studied and new adaptive MBC learning algorithms are presented in order to deal with the case where more than one class variable are available in the data streams. Specifically:

- Classifying partially labeled uni-dimensional data streams:

  We propose here a new semi-supervised approach, named `CPL-DS`, for handling concept-drifting evolving data streams containing both labeled and unlabeled instances. First, contrary to most of the proposals found in the literature, `CPL-DS` monitors three possible kinds of drift: feature, conditional or dual drift. Drift detection is based on a hypothesis test comparing Kullback-Leibler divergence between old and recent data, whose distribution under the null hypothesis of coming from the same distribution is approximated via a bootstrap method. Then, if any drift occurs, a new classifier is

learned from the recent data using the expectation-maximization algorithm; otherwise, the current classifier is left unchanged.

`CPL-DS` is then applied to the real-world malware detection problem, where received files should be continuously classified into malware (e.g. viruses, spyware, trojans, phishing, spam, etc.) or goodware to ensure that users are protected against malicious code. This application is supported by Panda Security company. In the analyzed malware detection data, only a few true labels (i.e., malware or goodware) are available, mainly due to the time consuming and costly labeling process. Moreover, the malware detection data is imbalanced since the number of malware instances is generally much higher than goodware instances. Hence, in order to tackle both problems, i.e., partially labeled data and imbalanced data, `CPL-DS` is applied jointly with two approaches for mining skewed data streams, namely clustering-sampling [228] and SERA [39].

- Mining multi-dimensional data streams using MBCs:

  As commented above, most of the work within the field of mining data streams has focused on uni-dimensional data streams. However, the problem of mining multi-dimensional data streams has received less attention, and only few works have been recently proposed [131, 178, 181, 233]. To deal with this problem, we present two novel adaptive algorithms for mining multi-dimensional data streams based on Bayesian network classifiers. Basically, the so-called `Globally Adaptive-MB-MBC` (`GA-MB-MBC`) and `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) extend our previous `MB-MBC` algorithm to data streams. `GA-MB-MBC` uses the Page-Hinkley test to monitor the average global log-likelihood over time and detect the concept drift, then, in the case that a drift is detected, it learns a new MBC from scratch. `LA-MB-MBC` proceeds similarly but locally at the level of each node in the MBC network, i.e., it monitors the average local log-likelihood of each node over time, then, whenever a drift is detected, it learns a new local structure for each changed node.

1.2 Overview of the dissertation

The rest of this dissertation is divided into eight chapters, which are organized into four parts.

- **Part II: Background**

  This part consists of three chapters and is devoted to the introduction of the Bayesian network concepts used throughout the dissertation, the presentation of the different

types of classification with more focus on the multi-dimensional classification task, the exploration of the related state-of-the-art, as well as a brief review on data streams:

  - Chapter 2 reviews the basic concepts relevant to Bayesian networks and to the learning problem of these models from stationary data.
  - Chapter 3 firstly presents the uni-dimensional classification task and reviews some of the widely used uni-dimensional Bayesian network classifiers. Then, it focuses on the multi-dimensional classification problem, introduces multi-dimensional Bayesian network classifiers, and explores the state-of-the-art on existing multi-dimensional learning methods. Finally, it describes some performance metrics and estimation methods used for the evaluation of classifiers.
  - Chapter 4 briefly defines data streams, explains concept drift and its categorization, and reviews several state-of-the-art methods for data stream mining.

- **Part III: Multi-dimensional Bayesian network classifiers for stationary data**

  This part consists of two chapters and is dedicated to the presentation of two novel MBC learning methods for stationary data:

  - Chapter 5 introduces `CB-MBC`, a new algorithm for learning class-bridge decomposable MBCs based on a wrapper greedy forward selection approach.
  - Chapter 6 presents `MB-MBC`, a new Markov blanket-based algorithm for MBC learning, and discusses its application to two real-world problems, namely the prediction of the human immunodeficiency virus type 1 (HIV-1) reverse transcriptase and protease inhibitors and the prediction of the European Quality of Life-5 Dimensions (EQ-5D) from 39-item Parkinson's Disease Questionnaire (PDQ-39).

- **Part IV: Mining uni- and multi-dimensional evolving streaming data**

  This part consists of two chapters and deals with mining evolving concept-drifting data streams. It tackles two main problems: First, the presence of some unlabeled instances in the uni-dimensional data streams, and second, the adaptive learning of MBCs from multi-dimensional data streams:

  - Chapter 7 proposes `CPL-DS`, a new method for classifying partially labeled data streams, then presents its application to the real-world malware detection problem where the data is imbalanced and only a few labels are available.

  - Chapter 8 addresses the problem of learning MBCs from evolving multi-dimensional streaming data, and proposes two new methods, namely, `LA-MB-MBC` and `GA-MB-MBC` for local and global adaptive MBC learning, respectively. Both methods monitor the concept drift over time using the Page-Hinkley test. If a drift is detected, `LA-MB-MBC` adapts the current MBC locally around each changed node, whereas `GA-MB-MBC` learns a new MBC from scratch.

- **Part V: Conclusions**

  The last part summarizes and concludes the dissertation:

  - Chapter 9 states the most important conclusions obtained throughout the dissertation and describes some open research lines for future work.

Part II

Background

Chapter 2

Bayesian networks

2.1 Introduction

Over the last years, probabilistic graphical models (PGMs) [34, 120, 144, 230] have become powerful tools for knowledge discovery and reasoning under uncertainty, and have gained considerable attention from artificial intelligence and machine learning communities. Different types of models have been studied within the PGM framework including Bayesian networks, undirected models, discrete and continuous models, temporal models, etc.

Roughly speaking, PGMs are composed of two main components: a graphical structure and parameters. The graphical structure represents the conditional (in)dependence relationships among the variables through several possible types of graphs such as directed, undirected, and chains; while the parameters describe the strength of the dependence relationships revealed by the graphical structure using marginal and/or conditional probabilities.

In this dissertation, we focus on Bayesian networks [118, 129, 145, 168], well-known PGMs that have been successfully applied in a wide range of fields and domains including medical diagnostics, bioinformatics, fraud detection, financial and marketing data analysis, pattern recognition, fault diagnosis, and many more [44, 53, 82, 105, 161, 167, 175, 194]. Two important issues can be identified when using Bayesian networks: The first is Bayesian network learning that aims to discover the set of probabilistic dependence relationships among the variables of interest. This modeling process can be either achieved by experts (by means of knowledge elicitation), or via learning algorithms applied to available data sets with or without the experts' assistance. The second issue is probabilistic inference that arises when a Bayesian network model is already built and deployed in a given application. Probabilistic inference [54, 118, 168] consists of estimating the posterior probability of a subset of query

variables given both the learned Bayesian network and the observed values taken by certain other variables (also called evidence).

Chapter outline

In Section 2.2, we introduce some notations and definitions. Then, in Section 2.3, we present Bayesian network models and their basic concepts in more detail. Section 2.4 provides a brief overview of the state-of-the-art approaches dealing with Bayesian network structure and parameter learning, and finally, Section 2.5 concisely describes probabilistic inference in Bayesian networks.

2.2 Notation and definitions

Notation

Throughout this dissertation the following syntactical conventions are used: $\mathbf{U} = \{X_1, \ldots, X_n\}$ denotes the universe defined as a finite set of n discrete random variables. A variable is denoted by an upper case letter (e.g. X, Y, X_i) and its state or value is denoted by the same lower-case letter (e.g. x, y, x_i). Moreover, a set of variables is denoted by a bold-face capitalized letter (e.g. $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{C}$) and the corresponding bold-face lower-case letter (e.g. $\mathbf{x}$, $\mathbf{y}$, $\mathbf{c}$) denotes an assignment for each variable in that given set.

Definitions

To begin, we present in what follows some preliminary definitions:

- A **directed acyclic graph** (DAG) is a pair $(\mathbf{V}, \mathbf{A})$ where $\mathbf{V}$ is a non-empty finite set of variables, called vertices or nodes, and $\mathbf{A}$ is a set of pairs of nodes (X, Y), $X, Y \in \mathbf{V}$, called edges. Note that "variables", "nodes", and "vertices"will be used interchangeably throughout the dissertation.

  There is a directed edge or arc between two nodes X and Y, denoted $X \rightarrow Y$, if $(X, Y) \in \mathbf{A}$ and $(Y, X) \notin \mathbf{A}$. However, if both $(X, Y) \in \mathbf{A}$ and $(Y, X) \in \mathbf{A}$, the edge is called undirected.

  A directed acyclic graph contains only directed edges and has no cycles.

- Two nodes linked by an edge are said to be **adjacent**.

- A **path** is a sequence of edges from one node to another.

- A **directed path** from X_1 to X_n is a sequence of directed edges $X_1 \to X_2 \to \ldots \to X_n$, with $n > 1$.
- A directed path is called a **cycle** if it begins and ends at the same variable.
- The following sets correspond to any node $X_i \in \mathbf{V}$: the *descendants* of X_i, $\mathbf{Desc}(X_i)$, representing the set of nodes reachable by a directed path from X_i, the *ancestors* of X_i, $\mathbf{Ancs}(X_i)$, representing the set of nodes that can reach X_i by a directed path of length one or more. The set of nodes $\mathbf{Ch}(X_i)$ reachable by a single direct edge from X_i are called *children* of X_i, and the set of nodes $\mathbf{Pa}(X_i)$ that can reach X_i by a single direct edge pointing into X_i are called *parents* of X_i.
- A **v-structure** in a DAG $\mathcal{G}$ is defined as an ordered triple of nodes (X_1, X_3, X_2) such that X_1 and X_2 are parents of X_3, and X_1 and X_2 are not adjacent in $\mathcal{G}$, i.e., we have $X_1 \to X_3 \leftarrow X_2$.
- A **tree** is a directed acyclic graph where each node has only one parent, except for the root node which has no parents.
- A **polytree** is a directed acyclic graph with the property that ignoring the directions of arcs yields a graph with no undirected cycles. In other words, there exists only one undirected path connecting any two nodes in the graph.

Example 2.1. *To illustrate these definitions, let us consider the directed acyclic graph shown in Figure 2.1(a). We have, the set of nodes* $\mathbf{V} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ *and the set of edges* $\mathbf{A} = \{(X_1, X_2), (X_1, X_3), (X_2, X_4), (X_2, X_5), (X_3, X_5), (X_3, X_6), (X_5, X_6)\}$.

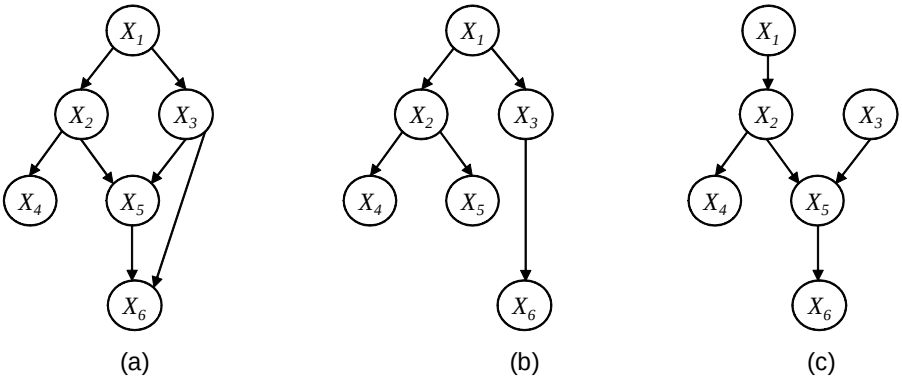


Figure 2.1: Examples of (a) a DAG, (b) a tree, and (c) a polytree.

The set of the descendants of node X_1 *is* $\mathbf{Desc}(X_1) = \{X_2, X_3, X_4, X_5, X_6\}$*, whereas the set of its children is* $\mathbf{Ch}(X_1) = \{X_2, X_3\}$*.* X_1 *has no parents.*
The set of the ancestors of node X_6 *is* $\mathbf{Ancs}(X_6) = \{X_1, X_2, X_3, X_5\}$*, whereas the set of its parents is* $\mathbf{Pa}(X_6) = \{X_3, X_5\}$*.* X_6 *has no children.*
Moreover, we have a unique v-structure (X_2, X_5, X_3)*, where* X_2 *and* X_3 *are parents of* X_5*, and* X_2 *and* X_3 *are not adjacent.*
Figures 2.1(b) and 2.1(c) illustrate examples of a tree and a polytree, respectively.

2.3 Bayesian networks

A Bayesian network (BN) [129, 168] is a PGM that encodes probabilistic (in)dependence relationships among variables based on directed acyclic graphs and probabilities. Formally, a BN $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ defined over a set of n discrete random variables consists of two components:

- The graphical structure $\mathcal{G}$: (a.k.a. the qualitative or graphical component) is a DAG that represents a set of conditional (in)dependence relationships among variables. The set of nodes represents the variables, and the set of arcs corresponds to dependence relationships between these variables. In fact, the presence of an arc between two nodes X and Y represents the existence of a dependence relationship between these variables. However, the absence of arcs between two nodes X and Y can be related to conditional independence relationships between these variables.

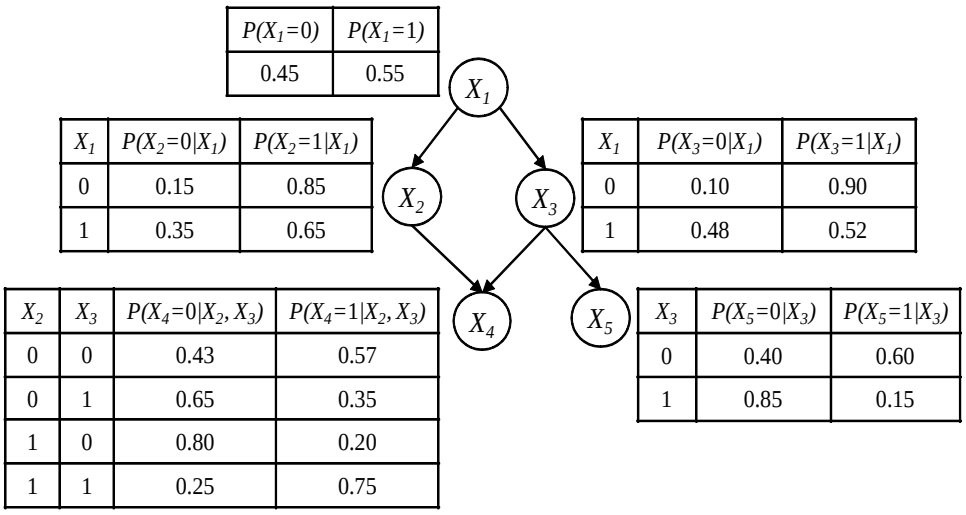
- The set of parameters $\mathbf{\Theta}$: (a.k.a. the quantitative or numerical component) specifies the set of conditional probability distributions. It represents the strength of the dependence relationships revealed by the graphical structure $\mathcal{G}$. That is, the set of parameters $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_i, \ldots, \mathbf{\Theta}_n\}$ such that each $\mathbf{\Theta}_i = P(x_i \mid \mathbf{pa}(x_i))$ denotes the conditional probability distribution of each node X_i given the values of its set of parents $\mathbf{Pa}(X_i)$. Let r_i be the number of distinct states of X_i and q_i be the number of different configurations of $\mathbf{Pa}(X_i)$, then $\mathbf{\Theta}_i = \{\theta_{ijk}, 1 \leq j \leq q_i, 1 \leq k \leq r_i\}$ such that $\theta_{ijk} = P(x_i = k \mid \mathbf{pa}(x_i) = j)$ denotes the conditional probability of the i^{th} variable X_i being in its k^{th} state, given that its parents $\mathbf{Pa}(X_i)$ are in their j^{th} configuration. All θ_{ijk} must verify the normalization constraint: $\sum_{k=1}^{r_i} \theta_{ijk} = 1$, $\forall$ i, j, and k. The conditional probability distributions are organized in tables, referred to as conditional probability tables (CPTs).

A Bayesian network $\mathcal{B}$ represents a joint probability distribution over $\mathbf{U}$ factorized according to structure $\mathcal{G}$ as follows:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \mathbf{pa}(x_i)) \tag{2.1}$$

Example 2.2. *Figure 2.2 shows a simple example of a Bayesian network including five binary variables, i.e., each variable has two possible values either 0 or 1. A conditional probability table is associated to each node in the network describing its conditional probability distribution given the set of its parents. For instance, node X_4 has two nodes X_2 and X_3 as parents, and hence, its probability distribution is conditioned only on the values of X_2 and X_3.*

The joint probability distribution corresponding to this BN can then be computed as follows: $P(x_1, x_2, x_3, x_4, x_5) = P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1) \cdot P(x_4 \mid x_2, x_3) \cdot P(x_5 \mid x_3)$.



$P(X_1=0)$	$P(X_1=1)$
0.45	0.55

X_1	$P(X_2=0\|X_1)$	$P(X_2=1\|X_1)$
0	0.15	0.85
1	0.35	0.65

X_1	$P(X_3=0\|X_1)$	$P(X_3=1\|X_1)$
0	0.10	0.90
1	0.48	0.52

X_2	X_3	$P(X_4=0\|X_2, X_3)$	$P(X_4=1\|X_2, X_3)$
0	0	0.43	0.57
0	1	0.65	0.35
1	0	0.80	0.20
1	1	0.25	0.75

X_3	$P(X_5=0\|X_3)$	$P(X_5=1\|X_3)$
0	0.40	0.60
1	0.85	0.15

Figure 2.2: Example of a Bayesian network.

In addition, the conditional independence property entailed by a probability distribution P over a Bayesian network $\mathcal{B}$ is defined as follows:

Definition 2.1. *Conditional independence [168]:*
Two set of variables $\boldsymbol{X}$ and $\boldsymbol{Z}$ are conditionally independent given some set of variables $\boldsymbol{Y}$, denoted as $I(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{Y})$, iff $P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{Y} = \boldsymbol{y}) = P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Y} = \boldsymbol{y})$ for any assignment of values $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ of $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$, respectively, such that $P(\boldsymbol{Y} = \boldsymbol{y}) > 0$.

2.3.1 D-separation criterion

The intuition behind the d-separation criterion [168] is as follows. Let us consider three disjoint variables X, Y, and Z. To test whether X and Z are d-separated by Y in a DAG $\mathcal{G}$, we need to test whether every path between X and Z is blocked by Y. Blocked means here that the flow of information is stopped between these variables connected by such paths.

To further define the d-separation criterion, we first present the three basic patterns of connection in Bayesian networks, namely:

1. Serial connection: In this case, information may be transmitted through the connection between X and Z unless the state of Y is known. In other words, the path between X and Z is blocked only if there is an evidence about Y (see Figure 2.3(a)).

2. Diverging connection: In this case, X and Z have a common parent Y, and information may be transmitted through the connection between X and Z unless the state of Y is known. In other words, the path between X and Z is blocked only if there is an evidence about Y (see Figure 2.3(b)).

3. Converging connection: In this case, Y has X and Z as parents, and information may be transmitted through the connection only if information about the state of Y or one of its descendants is available. In other words, any evidence about Y or one of its descendants results in not blocking the path between X and Z (see Figure 2.3(c)).

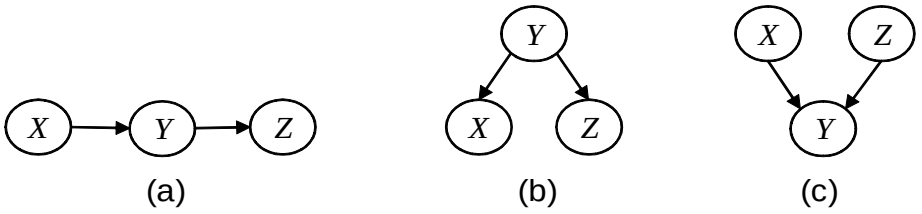


Figure 2.3: The (a) serial, (b) diverging and (c) converging connections.

Therefore, the d-separation criterion can be defined as follows:

Definition 2.2. *D-separation [168]:*
i) Let $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ be a DAG and let X, Y and Z be three distinct variables in $\mathbf{V}$. Then, X and Z are said to be d-separated by Y, if for all paths between X and Z, there is an intermediate variable Y such that either:

1. *the connection is serial or diverging and the state of Y is known, or*

2. *the connection is converging and neither the state of Y, nor any of its descendants, are known.*

ii) Let $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ be a DAG and let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ be three disjoint subsets of $\mathbf{V}$. Then, $\mathbf{X}$ and $\mathbf{Z}$ are said to be d-separated by $\mathbf{Y}$, if for every variable $X_i \in \mathbf{X}$ and $Z_i \in \mathbf{Z}$, X_i and Z_i are d-separated by any variable in $\mathbf{Y}$.

Using the d-separation criterion, one can gather graphically the conditional independencies in a Bayesian network as established in the following theorem [169].

Theorem 2.1. *If two sets of variables $\mathbf{X}$ and $\mathbf{Z}$ are d-separated by $\mathbf{Y}$ in a DAG $\mathcal{G}$, then $\mathbf{X}$ and $\mathbf{Z}$ are conditionally independent given $\mathbf{Y}$.*

2.3.2 Markov blanket

An additional important and basic concept of Bayesian networks is the *Markov blanket*. The Markov blanket of a variable X is the smallest set containing all variables carrying information about X, and which cannot be obtained from any other variable.

Definition 2.3. *Markov blanket [168]:*
A Markov blanket of a variable X, denoted as $MB(X)$, is a minimal set of variables with the following property: $I(X, \boldsymbol{S} \mid MB(X))$ holds for every variable subset $\boldsymbol{S}$ with no variables in $MB(X) \cup X$.

In other words, $MB(X)$ is the minimal set of variables conditioned by which X is conditionally independent of all subsets of the remaining variables.

Faithfulness

The faithfulness of a probability distribution is a well-known condition that guarantees the existence of a DAG $\mathcal{G}$ such that there is a one-to-one mapping between the graphical d-separation criterion and the probabilistic conditional independencies in the data set. Formally,

Definition 2.4. *Faithfulness [97, 205]:*
A DAG $\mathcal{G}$ is faithful to a joint probability distribution P over a set of variables $\mathbf{V}$ iff every independence present in P is entailed by $\mathcal{G}$. A probability distribution P is faithful iff there exists a DAG $\mathcal{G}$ such that $\mathcal{G}$ is faithful to P.

Under the faithfulness condition, the Markov blanket $MB(X)$ consists of the union of the set of parents, children, and parents of children (i.e., spouses) of X [171]. For instance, in Figure 2.2, the Markov blanket of variable X_2 includes its parent X_1, its child X_4, and its spouse X_3; that is, $MB(X_2) = \{X_1, X_4, X_3\}$.

2.4 Bayesian network learning

The learning process of Bayesian networks involves two phases: first, learning the graphical structure, then quantifying this structure via learning the parameters. Nevertheless, sometimes the structure can be provided by domain experts that determine the set of conditional (in)dependence relationships between variables. In this case, only the parameters have to be induced, and they can also be derived from various sources: they can be either obtained by interviewing domain experts to elicit their subjective probabilities, gathered from published statistical studies, or they can be learned directly from data.

When the structure is unknown, especially because domain experts are unwilling or unavailable, or the number of variables is large enough, the most appropriate solution is to resort to structure learning from data. This issue is considered as one of the most challenging tasks in dealing with Bayesian networks, since although having accurate parameters is important, they are completely useless if the structure is of bad quality [47, 79]. Note that, the work presented in this dissertation is focused on BN structure learning from data.

Let $\mathcal{D}$ be a complete data set of N independent and identically distributed (i.i.d.) observations containing a value assignment for each variable $X_1, \ldots, X_n$, i.e., $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$. Each instance $\mathbf{x}^{(l)} = (x_1^{(l)}, \ldots, x_i^{(l)}, \ldots, x_n^{(l)})$, such that $1 \leq i \leq n$ and $1 \leq l \leq N$, where $x_i^{(l)}$ represents the value of X_i in instance $\mathbf{x}^{(l)}$. Table 2.1 shows a classical disposition of $\mathcal{D}$.

Given the data set $\mathcal{D}$, the BN learning problem aims to determine the BN that best fits $\mathcal{D}$. This problem has received and receives until now much attention since its enormous usefulness for building high quality models required for developing better end-user applications.

Notice that, the learning process of BNs depends on whether data set $\mathcal{D}$ is complete or incomplete, that is, whether all values of each variable are available, or there are some missing values or hidden variables. Since this dissertation has more emphasis on learning BNs from complete data, we limit this section to present a brief review on principled approaches for learning BNs from complete data. Several methods for learning BN parameters from incomplete data can be consulted in [65, 95, 106, 179, 192], and additional methods dealing more specifically with inducing the BN graphical structure in the presence of missing values and hidden variables can be found in [20, 71, 79, 80, 202].

Table 2.1: Data matrix of a general BN learning task.

	X_1	X_2	$\dots$	X_i	$\dots$	X_n
1	$x_1^{(1)}$	$x_2^{(1)}$	$\dots$	$x_i^{(1)}$	$\dots$	$x_n^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
l	$x_1^{(l)}$	$x_2^{(l)}$	$\dots$	$x_i^{(l)}$	$\dots$	$x_n^{(l)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
N	$x_1^{(N)}$	$x_2^{(N)}$	$\dots$	$x_i^{(N)}$	$\dots$	$x_n^{(N)}$

2.4.1 Parameter learning

Parameter learning aims to estimate the values of the conditional probability distribution $P(x_i \mid \mathbf{pa}(x_i))$ of each variable X_i given any value of its parent set $\mathbf{Pa}(X_i)$. In this section, we describe two well-known approaches for parameter learning from complete data, namely, the maximum likelihood estimation (MLE) and the maximum a posteriori (MAP) estimation.

2.4.1.1 Maximum likelihood estimation

The maximum likelihood estimation (MLE) is a frequentist approach [74] that assesses the probabilities of variables from data without assuming any prior knowledge. It is based on the frequency of occurrences of variables in the data set, and selects the parameter configuration for a Bayesian network model, $\hat{\mathbf{\Theta}}$, that maximizes the probability of the data set given the model structure $\mathcal{G}$ (a.k.a. likelihood):

$$\hat{\mathbf{\Theta}} = arg \max_{\mathbf{\Theta}} P(\mathcal{D} \mid \mathcal{G}, \mathbf{\Theta}) \tag{2.2}$$

where $P(\mathcal{D} \mid \mathcal{G}, \mathbf{\Theta})$ represents the likelihood function computed as:

$$P(\mathcal{D} \mid \mathcal{G}, \mathbf{\Theta}) = \prod_{l=1}^{N} P(\mathbf{x}^{(l)} \mid \mathcal{G}, \mathbf{\Theta}) \tag{2.3}$$

where $\mathbf{x}^{(l)}$ denotes the value of $\mathbf{X}$ given by the l^{th} instance in $\mathcal{D}$. Then, by maximizing the likelihood function, the maximum likelihood estimators can be computed as follows [145]:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} \tag{2.4}$$

with N_{ijk} being the number of data instances in $\mathcal{D}$ where X_i takes its k^{th} value and $\mathbf{Pa}(X_i)$ takes its j^{th} configuration, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \; \forall \; i, j,$ and k.

2.4.1.2 Maximum a posteriori estimation

Maximum a posteriori (MAP) estimation is based on the Bayesian approach [104, 115]. It selects the parameter configuration for a Bayesian network model, $\hat{\mathbf{\Theta}}$, that maximizes the posterior probability of the parameters given the data set $\mathcal{D}$:

$$\hat{\mathbf{\Theta}} = arg \max_{\mathbf{\Theta}} P(\mathbf{\Theta} \mid \mathbf{D}) \tag{2.5}$$

Three assumptions are generally required for the MAP computation: 1) the data is complete, 2) parameter independence, i.e., $P(\mathbf{\Theta}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\Theta_{ij})$, with $\Theta_{ij} = \sum_{k=1}^{r_i} \theta_{ijk} \; \forall \; i, j,$ and k, and 3) the prior distribution over the parameters is Dirichlet [94, 204] as expressed by the following formula:

$$P(\Theta_{ij}) \sim Dir(\Theta_{ij} \mid \alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i}) = \Gamma(\alpha_{ij}) \prod_{k=1}^{r_i} \frac{\theta_{ijk}^{\alpha_{ijk}-1}}{\Gamma(\alpha_{ijk})} \tag{2.6}$$

where $\Gamma(.)$ is the *gamma function*, $(\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i})$ are the hyperparameters of the Dirichlet distribution, and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk} \; \forall \; i, j,$ and k. Hence, under the three above assumptions, the MAP calculation is given by:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}} \tag{2.7}$$

2.4.2 Structure learning

Two main approaches can be distinguished in the literature when dealing with BN structure learning: *score-based* and *constraint-based*. Score-based approaches [51, 106, 138] use a scoring function that evaluates how well the structure matches the data set and search for the structure that maximizes this function, whereas constraint-based approaches [169, 170, 205] perform statistical tests to determine the conditional (in)dependence relationships among the variables in the given data set.

2.4.2.1 Score-based approaches

Score-based approaches typically require the specification of three components [45]: 1) the *search space*, that defines the states of the search and the set of operators, 2) the *scoring function*, that evaluates the quality of the network with respect to the data, and 3) the *search algorithm*, that aims to find the BN that has the best scoring function.

Search space

The simplest and most used search space is the Bayesian network search space. The states of the search are individual Bayesian networks, represented by DAGs, and the operators are defined to be local changes applied to those DAGs. Basically, for any pair of nodes X and Y, if X and Y are adjacent, the arc connecting them can be either deleted or reversed. If X and Y are not adjacent, an arc can be inserted in either direction. All these operators are subject to the constraint that a cycle cannot be formed.

Other search spaces are also defined in the literature to mainly reduce the complexity of the Bayesian network search space, such as the ones imposing a total ordering among variables [33, 51, 210], restricting the number of parents for each node [58, 83], or operating on the space of equivalence classes of Bayesian networks [10, 45, 160].

Scoring function

Several scoring functions are developed to assess the goodness-of-fit of a particular BN structure and guide the learning process. In what follows, we review some of the most used ones.

- **Bayesian score (BS)**

Roughly speaking, the Bayesian score [51, 106] evaluates the probability of the data $\mathcal{D}$ given the structure $\mathcal{G}$, i.e., the likelihood $P(\mathcal{D} \mid \mathcal{G}, \boldsymbol{\Theta})$. Under the complete data and parameter independence assumptions, and given that the prior probability distribution over the parameters is a Dirichlet distribution, the Bayesian score can be calculated in closed form as follows:

$$BS(\mathcal{B}, \mathcal{D}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2.8}$$

where $\Gamma(.)$ is the *gamma function*, $(\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i})$ are the hyperparameters of the Dirichlet distribution, and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk} \ \forall\ i, j,$ and k.

- **Minimal Description Length (MDL)**

The MDL score [138, 187] is based on the idea that the best model to represent a data set is the one that minimizes the sum of the encoding lengths of the model and the data set given the model, both of which are measured in bits. The MDL score is defined as follows:

$$MDL(\mathcal{B}, \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}\ log \frac{N_{ijk}}{N_{ij}} - \frac{logN}{2} Dim(\mathcal{G}) \tag{2.9}$$

where $\frac{logN}{2} Dim(\mathcal{G})$ is the length needed to encode the model, such that $Dim(\mathcal{G})$ represents the dimension of the BN calculated as the number of parameters needed to specify it, i.e., $Dim(\mathcal{G}) = \sum_{i=1}^{n} q_i(r_i - 1)$, and $\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}\ log \frac{N_{ijk}}{N_{ij}}$ is the description length of the data set given the model.

Many other scoring functions have also been defined in the literature, namely, the Akaike's information criterion (AIC) based on a penalized log-likelihood score [3], the entropy [93, 108], or the mutual information [48].

Search algorithm

It aims to find the BN with the best score. The most commonly used algorithm is the greedy search (GS) algorithm [51] applied to the Bayesian network search space. It starts with a candidate structure, which may be empty or provided by some expert as a starting point. Then, at each iteration, it considers three possible operations: arc insertion, deletion or reversal. Next, the score is computed for every resulting candidate, and the candidate presenting the best score is selected and becomes the current candidate. This search process is iterated until there is no more score improvement.

Another greedy searching method operating instead in the equivalence class space of Bayesian networks is the greedy equivalence search (GES) algorithm [45].

Moreover, the literature includes various works where heuristic searching methods have been applied to learn the structure of Bayesian network models, such as simulated annealing [46], tabu search [28], best-first search [132], genetic algorithms [142], estimation of distribution algorithms [18], and ant colony optimization [59].

2.4.2.2 Constraint-based approaches

The general idea of constraint-based approaches is the use of statistical independence tests in order to learn a directed acyclic graph that represents the set of conditional (in)dependence

relationships captured in the data set.

A prominent example for constraint-based approaches is the Predictive Causation (PC) algorithm [205]. The PC algorithm starts from a complete undirected graph, then performs recursive conditional independence tests for deleting edges. The result is a skeleton in which all edges are still undirected and should be transformed into arcs using edge orientation rules. These rules consist first of detecting the v-structures, then orienting inductively the remaining edges without introducing directed cycles nor new v-structures.

Additional prevalent works on constraint-based approaches can be also consulted in [40, 169, 170, 235].

2.5 Inference in Bayesian networks

Given the learned BN model, one of the most fundamental mechanisms for reasoning under uncertainty is to compute the conditional distributions of one or a few variables. This task is usually referred to as probabilistic inference. In fact, the BN model is able to provide specific information about the probability distributions of variables of interest in the presence of information about the values of some other variables (called evidence).

Through marginalization it is possible to compute conditionals, posteriors, and make predictions. The process of exact probabilistic inference is proved to be NP-Hard [50]. The literature includes plenty of works providing different approaches to perform inference, and the textbooks by Koller and Friedman [129] and Darwiche [54] present an extensive discussion on this topic describing the most popular used exact and approximate inference methods, such as variable elimination, clique trees, loopy belief propagation, and Markov chain Monte Carlo methods. Several interesting references can be found as well in [29, 57, 66, 145, 196, 232].

Chapter 3

Multi-dimensional classification

3.1 Introduction

One of the main reasons for using Bayesian networks is to perform classification. Roughly speaking, in the machine learning community, classification refers to the ability to identify to which category (or class) a new observation belongs, on the basis of a given input data set (also called training data set).

Depending on the nature of the training data set, the classification problem can be categorized into three subtypes, namely supervised, semi-supervised and unsupervised.

In *supervised classification* [17, 68], all observations in the training data set are labeled beforehand, i.e., their category membership is known. We say that the training data set is *fully labeled*, and the supervised classification task consists of learning a model or a classifier from this data set, then using it to classify or predict the category (also known as class or label) of the incoming unseen data observations.

In *semi-supervised classification* [36, 174, 243], only some of the observations in the training data set (usually a small percentage) are labeled beforehand. In this case, the training data set is referred to as *partially labeled* data, and the semi-supervised classification task firstly aims to learn a model from both labeled and unlabeled observations. The rationale behind is that unlabeled data may contribute with valuable information and enhance the quality of the classification. Therefore, the built model using both labeled and unlabeled observations is better than the one built using only the limited amount of labeled data. Next, similar to supervised classification task, the second phase is to use the built model to classify the future unseen data observations.

In *unsupervised classification* [73, 114, 234], also known as clustering problem, all the

observations in the training data set are unlabeled, i.e., there is no knowledge about their category membership. In this case, the objective is to discover the different data categories (also known as groups or clusters) that permit to explain and describe the characteristics of the given *unlabeled* data set.

In addition to the above taxonomy, the classification problem can be also divided into uni-dimensional or multi-dimensional depending on whether the observations are assigned to only one class variable or to multiple class variables at the same time.

In fact, in the traditional and more popular task of *uni-dimensional classification*, each observation in the training data set is associated with a single class variable. An example would be classifying movies at the online internet movie database (IMDb). In this case, a given movie has to be associated with only one category, for instance drama.

However, in many real-world applications, more than one class variable may be required. That is, each observation in the training data set has to be associated with a set of many different class variables at the same time. For example, at IMDb, a movie may be classified simultaneously into three different categories: action, crime and drama. Additional examples may include a patient suffering from multiple diseases, a text document belonging to several topics, a gene associated with multiple functional classes, etc. Intuitively, *multi-dimensional classification* can be viewed as a generalization of the uni-dimensional classification problem where simultaneous prediction of a set of class variables is needed.

Note that, all the contributions in this thesis belong to the supervised multi-dimensional classification domain, except for Chapter 7 that deals with a semi-supervised uni-dimensional classification problem. Therefore, we opt to devote the current chapter mainly to present the supervised multi-dimensional classification task, and include more details about semi-supervised classification later in Chapter 7. Unsupervised classification is out of the scope of this thesis; some interesting works on this topic can be consulted in [73, 76, 99, 114, 234].

Chapter outline

For simplicity's sake, from now on, we merely use the term classification to refer to supervised classification. In Section 3.2, we start by presenting the traditional uni-dimensional classification task, as well as commonly used uni-dimensional classification methods based on Bayesian networks, called uni-dimensional Bayesian network classifiers. In Section 3.3, we define the multi-dimensional classification problem and briefly review the related work on multi-dimensional classification. Then, in Section 3.4, we introduce multi-dimensional Bayesian network classifiers (MBCs) and discuss the specific topics and related works on learning MBCs. Finally, in Section 3.5, we present some performance metrics and their esti-

mation methods used for evaluating the goodness of classifiers.

3.2 Uni-dimensional classification

We first present a formal definition of the uni-dimensional classification task. Let the training data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(N)}, c^{(N)})\}$ be a set of labeled observations or instances, such that each instance $(\mathbf{x}^{(l)}, c^{(l)})$ is characterized by a pair of a vector of m features or predictive variables $\mathbf{x}^{(l)} = (x_1^{(l)}, \ldots, x_i^{(l)}, \ldots, x_m^{(l)})$ and a class variable label $c^{(l)}$, with $1 \leq l \leq N$ and $1 \leq i \leq m$. Table 3.1 shows a disposition of such complete training data $\mathcal{D}$. A classification algorithm aims to learn a classifier from $\mathcal{D}$, which will be then used to assign class values to future instances $\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N+T)}\}$, also known as a testing data set.

Table 3.1: Data matrix of a uni-dimensional supervised classification task.

	X_1	X_2	$\ldots$	X_i	$\ldots$	X_m	C
1	$x_1^{(1)}$	$x_2^{(1)}$	$\ldots$	$x_i^{(1)}$	$\ldots$	$x_m^{(1)}$	$c^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
l	$x_1^{(l)}$	$x_2^{(l)}$	$\ldots$	$x_i^{(l)}$	$\ldots$	$x_m^{(l)}$	$c^{(l)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
N	$x_1^{(N)}$	$x_2^{(N)}$	$\ldots$	$x_i^{(N)}$	$\ldots$	$x_m^{(N)}$	$c^{(N)}$

The uni-dimensional classification problem consists of finding a function f that predicts for each input instance $\mathbf{x}$ a class value c:

$$\begin{aligned} f : \Omega_{X_1} \times \ldots \times \Omega_{X_m} &\longrightarrow \Omega_C = \{1, \ldots, r_C\} \\ \mathbf{x} = (x_1, \ldots, x_m) &\longmapsto c \end{aligned}$$

where $\Omega_C = \{1, \ldots, r_C\}$ denotes the sample space of the class variable C, and Ω_{X_i} denotes the sample space of each feature variable X_i, for all $i = 1, \ldots, m$. In our case, we consider that the class variable as well as all the feature variables are discrete random variables such that their respective cardinalities $|\Omega_C|$ and $|\Omega_{X_i}|$ are greater than 1.

Several methods and algorithms have been developed in the past years to perform uni-dimensional classification including, among many others, classification trees [31], k-nearest neighbor (k-NN) classifiers [52], neural networks [155], support vector machines [223], and logistic regression [110]. However, in this section, we focus on Bayesian network classifiers, which are Bayesian network models used for classification purposes.

Notice that Bayesian networks, also known as general Bayesian networks or unrestricted Bayesian networks, can be readily used to solve the uni-dimensional classification problem. However, as demonstrated by Friedman et al. [81], they usually exhibit a poor performance because they do not take into account the specific characteristics of the classification problem. Therefore, to deal appropriately with this problem, several Bayesian network classifiers have been proposed, mainly imposing restrictions on the general Bayesian network structure. In what follows, we briefly present some of the widely used uni-dimensional Bayesian network classifiers, namely, naive Bayes classifier, selective naive Bayes, tree augmented naive Bayes, and k-dependence Bayesian classifier.

Naive Bayes classifier

The simplest and most known Bayesian network classifier is naive Bayes (NB) [158]. It is based on the Bayes rule and on a strong independence assumption: all the feature variables X_i, $1 \leq i \leq m$, are conditionally independent given the class variable C. An example of the graphical representation of a naive Bayes structure with $m = 4$ feature variables is depicted in Figure 3.1(a). As observed, NB has a simple structure where the class variable C is the root, i.e., $\mathbf{Pa}(C) = \emptyset$, and each feature variable has the class variable C as its unique parent, i.e., $\mathbf{Pa}(X_i) = \{C\},\ \forall\, i, 1 \leq i \leq m$.

Selective naive Bayes

The selective naive Bayes (SNB) [140] is a variant of NB classifier. In its standard version, SNB selects only a subset of feature variables to make predictions based on a wrapper greedy forward search technique. That is, it starts with an empty set of feature variables, selects at each iteration the feature variable providing the best accuracy, and ends when no additional feature variable brings more accuracy improvement. Figure 3.1(b) shows an example of a selective naive Bayes structure, where from $m = 4$ feature variables, only three are selected to make predictions.

In addition, filter versions of SNB have been also proposed in the literature using feature selection criteria which are independent of the classification model such as mutual information, entropy, Euclidean distance, and Kullback-Leibler divergence [19, 113].

Tree augmented naive Bayes

The tree augmented naive Bayes (TAN) [81] is an extension of the NB classifier. It aims to dispense with the NB strong independence assumption by augmenting its structure with

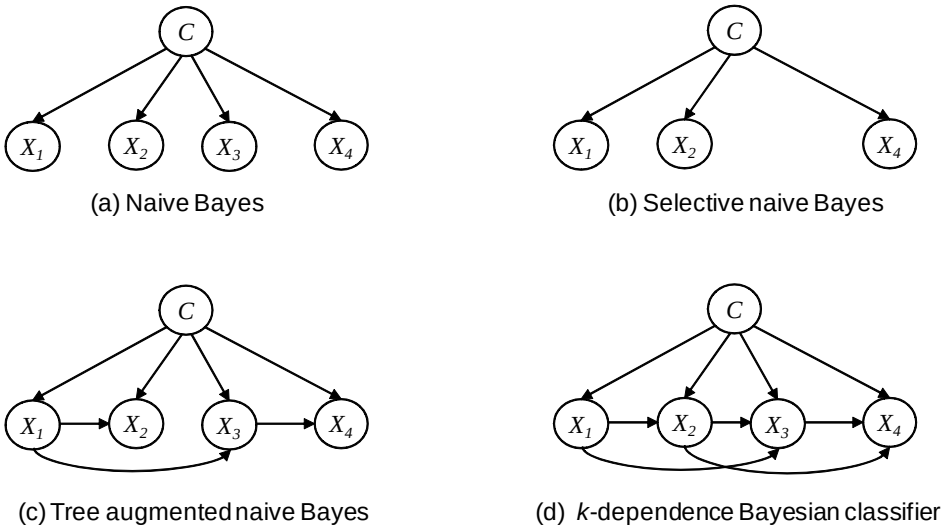


Figure 3.1: Example of structures of uni-dimensional Bayesian network classifiers including a single class variable and four predictive variables.

dependence relationships among the feature variables. In its standard version, these dependence relationships are captured using a maximal weighted spanning tree learned with the well-known Chow and Liu algorithm [48]. Thus, the number of parents of each feature variable is restricted to two: the class variable and an additional feature variable. Figure 3.1(c) shows an example of a TAN structure with $m = 4$ feature variables, where, compared with NB structure, additional arcs are inserted between the feature variables.

A wrapper greedy approach for building TAN structure has been also proposed in [123]. It starts with a naive Bayes, then iteratively adds the arcs which agree with TAN structural restrictions and improve the most the classification accuracy.

k-dependence Bayesian classifier

The k-dependence Bayesian classifier (kDB) [193] is a generalization of both NB and TAN, where k is an input parameter fixing the maximum number of feature parents allowed for each feature variable. In fact, NB can be viewed as a kDB with $k = 0$, whereas TAN corresponds to a kDB with $k = 1$. The inclusion of dependence relationships among feature variables is based on the calculation of mutual information and conditional mutual information statistics. An example of a kDB structure with $m = 4$ feature variables and $k = 2$ is depicted in Figure 3.1(d). As it can be seen, the parent set of each feature variable includes the class parent C and a maximum of two additional feature parents.

For all the presented uni-dimensional Bayesian network classifiers, and under a 0-1 loss function, the classification task consists of determining the maximum a posteriori class value c^* given an evidence about the input vector of feature variables $\mathbf{x} = (x_1, \ldots, x_m)$, that is,

$$\begin{aligned} c^* &= arg\ \max_c\ P(c \mid x_1, \ldots, x_m) \\ &= arg\ \max_c\ P(c) \cdot \prod_{i=1}^{m} P(x_i \mid \mathbf{pa}(x_i)) \\ &= arg\ \max_c\ P(c) \cdot \prod_{i=1}^{m} P(x_i \mid c, \mathbf{pa}_{\mathbf{V}_X}(x_i)) \end{aligned} \tag{3.1}$$

where the parent set $\mathbf{Pa}(X_i)$ of the feature variable X_i includes the class parent C and the feature parents $\mathbf{Pa}_{\mathbf{V}_X}(X_i)$ of X_i in the graphical structure $\mathcal{G}$ of the considered Bayesian network classifier. Obviously, for NB and SNB, $\mathbf{Pa}_{\mathbf{V}_X}(X_i) = \emptyset$; for TAN, $|\mathbf{Pa}_{\mathbf{V}_X}(X_i)| = 1$ except the root of the tree; and for kDB, $|\mathbf{Pa}_{\mathbf{V}_X}(X_i)| \leq k$.

3.3 Multi-dimensional classification

As commented above, the multi-dimensional classification problem is an extension of the classical uni-dimensional classification problem, where we have to deal with multiple output class variables rather than a single output class variable [215, 222]. To formally define the multi-dimensional classification problem, let us first consider the training data set $\mathcal{D}$ of N instances containing a value assignment for each variable $X_1, \ldots, X_m, C_1, \ldots, C_d$, i.e., $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{c}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{c}^{(N)})\}$. In this case, each instance $(\mathbf{x}^{(l)}, \mathbf{c}^{(l)})$ is characterized by a vector of m features or predictive variables $\mathbf{x}^{(l)} = (x_1^{(l)}, \ldots, x_i^{(l)}, \ldots, x_m^{(l)})$ and a vector of d class variables $\mathbf{c}^{(l)} = (c_1^{(l)}, \ldots, c_j^{(l)}, \ldots, c_d^{(l)})$, with $1 \leq l \leq N$, $1 \leq i \leq m$, and $1 \leq j \leq d$. Table 3.2 shows a disposition of such training data $\mathcal{D}$. Therefore, a multi-dimensional classification algorithm aims to learn a multi-dimensional classifier from $\mathcal{D}$, which will be then used to predict the d-dimensional vectors of class values for incoming instances $\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N+T)}\}$.

Formally, the multi-dimensional classification problem consists of finding a function h that predicts for each input instance given by a vector of m features $\mathbf{x} = (x_1, \ldots, x_m)$, a vector of d class values $\mathbf{c} = (c_1, \ldots, c_d)$, that is,

$$\begin{aligned} h : \Omega_{X_1} \times \ldots \times \Omega_{X_m} &\longrightarrow \Omega_{C_1} \times \ldots \times \Omega_{C_d} \\ \mathbf{x} = (x_1, \ldots, x_m) &\longmapsto \mathbf{c} = (c_1, \ldots, c_d) \end{aligned}$$

Table 3.2: Data matrix of a multi-dimensional classification task.

	X_1	X_2	$\dots$	X_i	$\dots$	X_m	C_1	C_2	$\dots$	C_j	$\dots$	C_d
1	$x_1^{(1)}$	$x_2^{(1)}$	$\dots$	$x_i^{(1)}$	$\dots$	$x_m^{(1)}$	$c_1^{(1)}$	$c_2^{(1)}$	$\dots$	$c_j^{(1)}$	$\dots$	$c_d^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
l	$x_1^{(l)}$	$x_2^{(l)}$	$\dots$	$x_i^{(l)}$	$\dots$	$x_m^{(l)}$	$c_1^{(l)}$	$c_2^{(l)}$	$\dots$	$c_j^{(l)}$	$\dots$	$c_d^{(l)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
N	$x_1^{(N)}$	$x_2^{(N)}$	$\dots$	$x_i^{(N)}$	$\dots$	$x_m^{(N)}$	$c_1^{(N)}$	$c_2^{(N)}$	$\dots$	$c_j^{(N)}$	$\dots$	$c_d^{(N)}$

where Ω_{X_i} and Ω_{C_j} denote the sample spaces of each feature variable X_i, for all $i \in \{1, \dots, m\}$, and each class variable C_j, for all $j \in \{1, \dots, d\}$, respectively. Note that, as before, we consider that all class and feature variables are discrete random variables such that $|\Omega_{X_j}|$ and $|\Omega_{C_j}|$ are greater than 1, and the data is complete, i.e., there is no missing values in $\mathcal{D}$.

When $|\Omega_{C_j}| = 2$ for all $j \in \{1, \dots, d\}$, i.e., all class variables are binary, the multi-dimensional classification problem is known as a *multi-label classification* problem [149, 215]. In general, a multi-label classification problem can be easily modeled as a multi-dimensional classification problem where each label corresponds to a binary class variable. However, modeling a multi-dimensional classification problem, that possibly includes non-binary class variables, as a multi-label classification problem may require a transformation over the data set to meet multi-label framework requirements. Note that, since our contributions in this dissertation are general and can be applied to classification problems where class variables are not necessarily binary, we opt to use, unless mentioned otherwise, the term multi-dimensional classification as a more general concept.

Most of the work in the literature deals with the multi-label classification problem. Overviews are given in [60, 215, 217]. Basically, the proposed methods are divided into two main categories: *problem transformation* and *algorithm adaptation*.

Problem transformation methods in multi-label classification

Methods in this category are algorithm independent and proceed by transforming the multi-label classification task into one or more single-label classification tasks.

One of the most widely used transformation method is binary relevance (BR) [98]. It is based on the idea of decomposing the multi-label learning problem into d independent binary classification problems, such that each binary classification problem aims to predict a single label value. Classifier chains method [183] overcomes the BR label independence assumption by transforming the multi-label learning problem into a chain of binary classification problems,

such that the prediction of each binary classifier takes into account the predictions of all binary classifiers preceding it in the chain. Ranking via single-label method [98] transforms the multi-label data set into a single label data set by either ignoring multi-label instances, selecting the least frequent label, selecting the most frequent label, selecting randomly one label, or assigning a weight to each label. Other examples also include pairwise methods such as ranking by pairwise comparison (RPC) [111] which learns $\frac{d(d-1)}{2}$ binary models, one for each pair of labels; and calibrated label ranking [84] which extends RPC by introducing an additional artificial calibration label to the original label set separating the relevant labels from the irrelevant ones.

Some other transformation methods are based on label combinations like label powerset (LP) [30] which transforms each unique subset (distinct label set) of labels that exists in a given multi-label data set into a single label; pruned sets (PS) [182] which improves LP by pruning away infrequent label sets based on a user-defined threshold; ensemble of pruned sets [182] which employs PS in an ensemble framework to further enhance the predictive performance; and random k-label sets [218] which builds an ensemble of LP classifiers, such that each LP classifier is trained using a different small random label subset of the original label set. Additional proposed methods are based on the identification of label dependencies such as correlation-based pruning of stacked binary relevance models [214]; Chi-square tests used to discover and join together pairs of most dependent labels [209]; and hierarchy of multi-label classifiers where each classifier in the hierarchy deals with a much smaller set of labels identified via a balance clustering algorithm [216].

Algorithm adaptation methods in multi-label classification

Algorithm adaptation methods extend specific single-label learning algorithms in order to handle directly the multi-label data sets.

Several adaptation methods have been proposed, including decision trees [224] which extend the popular C4.5 decision tree algorithm to deal with multi-label data based on the definition of multi-label entropy; support vector machines [72] which aim to minimize a ranking loss function while maintaining a large margin; neural networks [239] which adapt the error function of the popular back-propagation algorithm to take multiple labels into account; and k-nearest neighbors [240] which extend the k-nearest neighbor lazy algorithm for multi-label classification and use the maximum a posteriori principle to predict the label set.

Other examples include the combination of logistic regression and k-nearest neighbors coping with multiple labels [41]; generative parametric mixture model applied to multi-label text categorization [221]; and boosting [198] which presents a multi-label adaptation of the

traditional AdaBoost learning algorithm.

Table 3.3 summarizes the presented problem transformation and algorithm adaptation methods in multi-label classification.

Table 3.3: Summary of multi-label classification methods.

Multi-label classification	**Problem transformation methods**	Binary relevance (BR) [98]
		Classifier chains [183]
		Ranking via single-label learning [98]
		Ranking by pairwise comparison [111]
		Calibrated label ranking [84]
		Label powerset (LP) [30]
		Pruned sets (PS) [182]
		Ensemble of pruned sets [182]
		Random k-label sets [218]
		Correlation-based pruning of stacked BR [214]
		Chi-square tests [209]
		Hierarchy of multi-label classifiers [216]
	Algorithm adaptation methods	Decision trees [224]
		Support vector machines [72]
		Neural networks [239]
		k-nearest neighbors [240]
		Logistic regression + k-nearest neighbors [41]
		Generative parametric mixture model [221]
		Boosting [198]

For Bayesian networks, problem transformation methods can either opt for: (a) building a compound class variable that models all possible combinations of the class variables and then applying any existing uni-dimensional Bayesian network classifier (such as NB, SNB, TAN or kDB) following an LP strategy; or (b) learning multiple uni-dimensional Bayesian network classifiers, one for each individual class variable as if all of them were independent following a BR strategy. In the former case, the class variable can easily end up with a huge number of values which may weakness the predictive performance of such a classifier. In the latter case, by building multiple classifiers, the interactions among the various class variables and their simultaneous interactions with feature variables cannot be detected, which does not represent accurately the multi-dimensional classification problem.

Hence, to deal properly with this problem, the so-called multi-dimensional Bayesian networks classifiers (MBCs) have been recently proposed and defined as an *algorithm adaptation*

method that extends the general Bayesian networks to the multi-dimensional classification problem. MBCs as well as works more closely related to our contributions, dealing with learning MBCs from stationary data, are presented in the next section.

3.4 Multi-dimensional Bayesian network classifiers

MBCs extend the uni-dimensional Bayesian network classifiers and provide an accurate modeling of the probabilistic dependence relationships among all variables, the class variables included [222]. As Bayesian networks, the first component of the MBCs is the *graphical structure*. It partitions the set of class and feature variables into three different subgraphs: *class subgraph* representing the dependence relationships between class variables, *bridge subgraph* representing the dependence relationships between class and feature variables, and *feature subgraph* representing the dependence relationships between feature variables. The second component consists of the *parameters* that define the conditional probability distribution of each variable given the set of its parents in the graphical structure.

Definition 3.1. *A* multi-dimensional Bayesian network classifier *(MBC) [222] is a Bayesian network* $\mathcal{B} = (\mathcal{G},\boldsymbol{\Theta})$ *where the structure* $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ *has a restricted topology. The set of* n *vertices* $\mathbf{V}$ *is partitioned into two subsets:* $\mathbf{V}_C = \{C_1, \ldots, C_d\}, d \geq 1$*, of class variables and* $\mathbf{V}_X = \{X_1, \ldots, X_m\}, m \geq 1$*, of feature variables* $(d+m = n)$*. The set of arcs* $\mathbf{A}$ *is partitioned into three subsets* $\mathbf{A}_C$*,* $\mathbf{A}_X$ *and* $\mathbf{A}_{CX}$*, such that:*

- $\mathbf{A}_C \subseteq \mathbf{V}_C \times \mathbf{V}_C$ *is composed of the arcs between the class variables having a subgraph* $\mathcal{G}_\mathcal{C} = (\mathbf{V}_C, \mathbf{A}_C)$ -class subgraph- *of* $\mathcal{G}$ *induced by* $\mathbf{V}_C$*.*
- $\mathbf{A}_X \subseteq \mathbf{V}_X \times \mathbf{V}_X$ *is composed of the arcs between the feature variables having a subgraph* $\mathcal{G}_\mathcal{X} = (\mathbf{V}_X, \mathbf{A}_X)$ -feature subgraph- *of* $\mathcal{G}$ *induced by* $\mathbf{V}_X$*.*
- $\mathbf{A}_{CX} \subseteq \mathbf{V}_C \times \mathbf{V}_X$ *is composed of the arcs from the class variables to the feature variables having a subgraph* $\mathcal{G}_{\mathcal{CX}} = (\mathbf{V}, \mathbf{A}_{CX})$ -bridge subgraph- *of* $\mathcal{G}$ *induced by* $\mathbf{V}$ *[13].*

Note that depending on the graphical structures of the class and feature subgraphs we can differentiate between several families of MBCs. Such families can be denoted as `class subgraph structure-feature subgraph structure` MBCs, where the possible structures of each subgraph are: empty, tree, polytree, or DAG. MBC families used in the literature include `tree-tree` MBC [222], `polytree-polytree` MBC [61], `DAG-empty` MBC [177], and `DAG-DAG` MBC [188]. Throughout this dissertation, we do not consider any restrictions on the

learned MBC structures, i.e., any possible structure type is allowed for either class or feature subgraphs.

Classification with an MBC under a 0-1 loss function is equivalent to solving the most probable explanation (MPE) problem which consists of finding the most likely instantiation of the vector of class variables $\mathbf{c}^* = (c_1^*, \ldots, c_d^*)$ given an evidence about the input vector of feature variables $\mathbf{x} = (x_1, \ldots, x_m)$. More formally, for a given observed evidence $\mathbf{x}$, we have to determine

$$\begin{aligned}
\mathbf{c}^* &= (c_1^*, \ldots, c_d^*) \\
&= arg \max_{c_1,\ldots,c_d} P(c_1, \ldots, c_d \mid x_1, \ldots, x_m) \\
&= arg \max_{c_1,\ldots,c_d} \prod_{j=1}^{d} P(c_j \mid \mathbf{pa}_{\mathbf{V}_C}(c_j)) \cdot \prod_{i=1}^{m} P(x_i \mid \mathbf{pa}_{\mathbf{V}_C}(x_i), \mathbf{pa}_{\mathbf{V}_X}(x_i))
\end{aligned} \tag{3.2}$$

where $\mathbf{Pa}_{\mathbf{V}_C}(C_j)$ denotes the class parents of class variable C_j in the MBC graphical structure $\mathcal{G}$, and $\mathbf{Pa}_{\mathbf{V}_C}(X_i)$, $\mathbf{Pa}_{\mathbf{V}_X}(X_i)$ denote respectively the class parents and feature parents of the feature variable X_i in $\mathcal{G}$. Obviously, we have $\mathbf{Pa}_{\mathbf{V}_X}(C_j) = \emptyset$, $\forall j \in \{1, \ldots, d\}$, since arcs from feature variables to class variables are not allowed.

Example 3.1. *An example of an MBC structure is shown in Figure 3.2.* $\mathbf{V}_C$ *contains* $d = 4$ *classes,* $\mathbf{V}_X$ *includes* $m = 7$ *features, and the structure* $\mathcal{G}$ *is equal to* $\mathcal{G}_\mathcal{C} \cup \mathcal{G}_\mathcal{X} \cup \mathcal{G}_{\mathcal{CX}}$*. We have*

$$\begin{aligned}
\mathbf{c}^* = arg \max_{c_1,\ldots,c_4} & P(c_1, \ldots, c_4 \mid x_1, \ldots, x_7) \\
= arg \max_{c_1,\ldots,c_4} & P(c_1 \mid c_2, c_3) P(c_2) P(c_3) P(c_4) \\
& \cdot P(x_1 \mid c_2, x_4) P(x_2 \mid c_1, c_2) P(x_3 \mid c_4) P(x_4 \mid c_1) \\
& \cdot P(x_5 \mid x_2) P(x_6 \mid c_3, x_3, x_7) P(x_7 \mid c_4)
\end{aligned}$$

In recent years, several methods have been proposed to learn MBCs from data. Basically, this learning problem aims to find an MBC that best fits the available multi-dimensional training data set $\mathcal{D}$, and ensures afterwards an accurate and efficient classification for the future instances. In what follows, we briefly review the existing MBC learning methods.

Van der Gaag and de Waal [222] decompose the learning problem of `tree-tree` MBCs into two separate optimization problems: first learning the class subgraph using Chow and Liu's algorithm [48], then, given a fixed bridge subgraph, learning the feature subgraph using also Chow and Liu's algorithm. The bridge subgraph is greedily selected from an empty graph

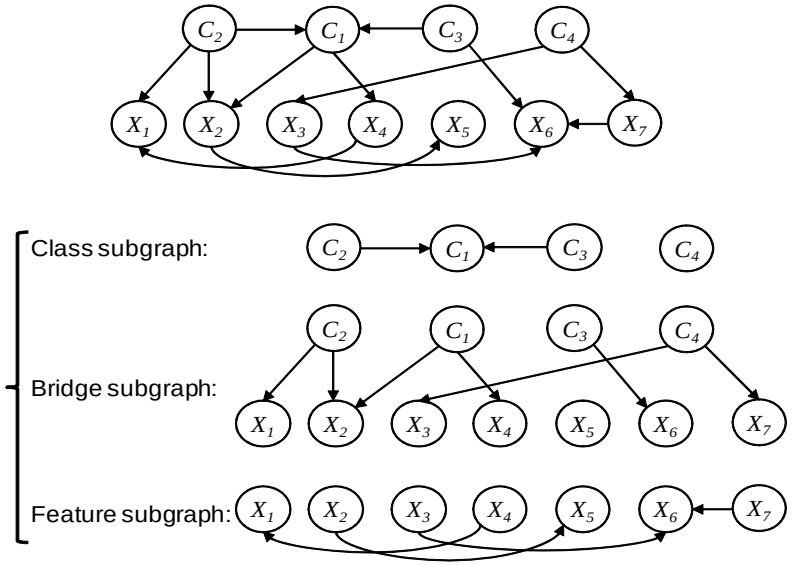


Figure 3.2: Example of an MBC structure with its class, bridge and feature subgraphs.

using a wrapper approach guaranteeing a high classifier accuracy. Later, the same authors [61] present a theoretical approach for learning `polytree-polytree` MBCs where class and feature subgraphs are separately learnt based on Rebane and Pearl's algorithm [184]. Nevertheless, the induction of the bridge subgraph was not specified.

Moreover, Qazi et al. [177] learn `DAG-empty` MBCs where the class subgraph is induced by standard Bayesian network procedures, the bridge subgraph is learnt by adding dependence relationships from each class variable to a subset of selected and non-overlapping features, and the feature subgraph is kept empty.

Rodríguez and Lozano [188] use a multi-objective evolutionary approach to learn `DAG-DAG` MBCs. By using a genetic algorithm, each permitted MBC structure is coded as an individual with three substrings, one per subgraph. Based on different classification measures, joint and marginal, they define the objective functions as k-fold cross-validated estimators of each class classification error. The aim is to find non-dominated structures according to the objective functions.

Bielza et al. [13] propose different learning algorithms, namely, pure filter (guided by any filter algorithm based on a fixed ordering among the variables), pure wrapper (guided by the classification accuracy) and a hybrid algorithm (a combination of pure filter and pure wrapper). None of these three algorithms places any constraints on the subgraph structures of the generated MBCs.

More recently, Zaragoza et al. [237] propose a two-step method to also learn `polytree-polytree` MBCs. First, they build class and feature subgraphs using Chow and Liu's algorithm [48] and generate an initial bridge subgraph based on mutual information. Then, in a second step, they refine the bridge subgraph by adding more arcs to improve the MBC accuracy. Later, Zaragoza et al. [238] present a two-step method for learning Bayesian chain classifiers for multi-dimensional classification. In the first step, a tree-based Bayesian network that represents the dependency relations between the class variables is learned. In the second step, several chain classifiers are built using selective naive Bayes models, such that the order of the class variables in the chain is consistent with the class tree subgraph. At the end, the results of the different generated orders are combined in a final ensemble model.

3.5 Classifier evaluation

Once the learning phase ends, the obtained classifiers should be evaluated (or validated). This evaluation process is in fact crucial in order to assess the behavior of a classifier when applied to unseen data set, and thereby assess the learning method used for the induction of that classifier. This process also allows the comparison of various classifiers on a same data set and the analysis of whether one classifier is superior to another or not.

To carry out this evaluation, two parts should be considered, namely, the performance evaluation metrics and the used methodology for estimating the classifier performance.

3.5.1 Performance evaluation metrics

A commonly and widely used performance evaluation metric is the classification accuracy, which is defined as the probability of correctly classifying a new instance $\mathbf{x}$. The accuracy computation depends on the type of the classification problem, i.e., uni-dimensional or multi-dimensional.

3.5.1.1 Uni-dimensional classification

In this case, we have only one class variable C and the accuracy estimation is expressed as:

$$Acc = \frac{1}{N} \sum_{l=1}^{N} \delta(\hat{c}_l, c_l) \tag{3.3}$$

where N is the size of the testing data set, $\hat{c}_l$ is the class value predicted by the uni-dimensional

classifier for instance l, and c_l is its corresponding true value. $\delta(\hat{c}_l, c_l) = 1$ if the predicted and true class values are equal, i.e., $\hat{c}_l = c_l$, and $\delta(\hat{c}_l, c_l) = 0$ otherwise.

3.5.1.2 Multi-dimensional classification

In this case, we have a set of class variables $\mathbf{C} = (C_1, \ldots, C_d)$ and the accuracy estimation metrics, that extend the uni-dimensional one, can be computed in two different ways [13], namely:

- The *mean accuracy* over the d class variables. It is defined as a class-based measure where the accuracy is calculated separately for each class variable, then averaged across all the class variables:

$$Acc_m = \frac{1}{d} \sum_{j=1}^{d} \frac{1}{N} \sum_{l=1}^{N} \delta(\hat{c}_{lj}, c_{lj}) \tag{3.4}$$

  where N is the size of the testing data set, $\hat{c}_{lj}$ denotes the C_j class value predicted by the multi-dimensional classifier for sample l, and c_{lj} denotes its corresponding true value. $\delta(\hat{c}_{lj}, c_{lj}) = 1$ if the predicted and true class values are equal, i.e., $\hat{c}_{lj} = c_{lj}$, and $\delta(\hat{c}_{lj}, c_{lj}) = 0$ otherwise.

- The *global accuracy* over the d-dimensional class variable (also known as *exact match* [181]). It is considered as an instance-based measure where the accuracy is calculated separately for each instance in the testing data set, then averaged across all the instances:

$$Acc_g = \frac{1}{N} \sum_{l=1}^{N} \delta(\hat{\mathbf{c}}_l, \mathbf{c}_l) \tag{3.5}$$

  In this more strict case, the (d-dimensional) vector of predicted classes $\hat{\mathbf{c}}_l$ is compared to the vector of true classes $\mathbf{c}_l$, so that we have $\delta(\hat{\mathbf{c}}_l, \mathbf{c}_l) = 1$ if both vectors are equal in all their components, i.e., $\hat{\mathbf{c}}_l = \mathbf{c}_l$, and $\delta(\hat{\mathbf{c}}_l, \mathbf{c}_l) = 0$ otherwise.

Moreover, if all the class variables are binary, i.e., we have a multi-label classification problem, the so-called subset accuracy has been proposed by Ghamrawi and McCallum [96] as a trade-off between the mean accuracy (which tends to be overly lenient) and the global accuracy (which tends to be overly strict):

- The *subset accuracy*: This is an instance-based measure that alleviate the very strict global accuracy measure by taking into account the partial correctness of the predicted class values. The subset accuracy is computed as

$$Acc_{subset} = \frac{1}{N}\sum_{l=1}^{N}\frac{|\hat{\mathbf{c}}_l \cap \mathbf{c}_l|}{|\hat{\mathbf{c}}_l \cup \mathbf{c}_l|} \tag{3.6}$$

Note that, in this special case of multi-label classification, plenty of performance metrics have also been defined, such as precision, recall, micro and macro averaged F-measures, log-loss, as well as ranking-based metrics like one-error and ranking loss. The reader can find more details about the multi-label evaluation metrics in [180].

3.5.2 Performance metric estimation methods

Several methodologies have been proposed to estimate the performance metrics of a classifier. We present here four commonly used ones:

- *Hold-out estimation* [143]: In this case, the available data set is split into two disjoint sets: training (typically 2/3) and testing (the remaining 1/3). The training data set is then used to learn the classifier, while the testing data set is used to evaluate it and estimate its performance.

- *K-fold cross-validation* [206]: In this estimation method, the data set is split into K mutually exclusive folds. These folds are typically balanced, i.e., they include approximately the same number of instances in each fold. The training and testing processes are then repeated K times, such that, at each iteration, $K-1$ folds are used to learn the classifier and the remaining fold is used to evaluate it. The final estimated classifier performance is then calculated by averaging over the K iterations usually providing the mean and the standard deviation. The number of folds depends on the size of the available data set. Nevertheless, the commonly used values are $K = 5$ and $K = 10$.

- *Leave-one-out cross-validation* [137]: This estimation method can be seen as a particular case of K-fold cross-validation where the number of folds is equal to the number of instances, i.e., $K = N$. Therefore, in this case, the learning process is repeated N times, such that, at each iteration, the classifier is learned using $N-1$ instances, and evaluated on a single instance. At the end, the averaged value over the N iterations and its standard deviation are reported.

- *Bootstrap* [69]: Given a data set of size N, this estimation method is based on creating a new data set by sampling uniformly with replacement N instances from the original data set. The obtained new data set is called a *bootstrap sample*. Since the sampling process is performed with replacement, the probability of any given instance not being selected after N samples is equal to $(1 - \frac{1}{N})^N \approx e^{-1} \approx 0.368$, whereas its probability of being selected is approximately 0.632. That is why, this estimation method is commonly known in the literature as the *0.632 bootstrap*. The bootstrap sample is then used for training and the non selected instances are used for testing, i.e., the training and testing data sets will contain on average 63.2% and 36.8% of the instances, respectively. The accuracy is estimated by combining accuracies measured on both testing and training data sets as: $Acc = 0.632 \times AccTesting + 0.368 \times AccTraining$. This process is generally repeated over B bootstrap samples, B being a parameter of the method, and the final accuracy estimate is calculated by averaging over all B samples.

Chapter 4

Data streams

4.1 Introduction

Throughout Chapter 3 we have presented multi-dimensional Bayesian network classifiers as well as the state-of-the-art approaches for learning those classifiers from stationary data. In this stationary environment, the whole stored training data set is used to induce a classifier that will never be revised and will be valid to classify all the new incoming instances over time. The rationale behind this is that the distribution underlying the data is assumed to be stable and does not change over time.

However, this assumption is often violated and the data distribution is more likely to change, especially when data is collected over an extended period of time. Nowadays, with the rapid growth of information technology, this non-stationary setting is more noticeable, and generally,"infinite"flows of records are generated daily from a wide range of real-world applications, such as network monitoring, social networks, information filtering, fraud and intrusion detection, telecommunications data management, etc. These flows are named *data streams*.

Contrary to finite stationary data sets, data streams are characterized by their concept-drifting aspect, which means that the learned concepts and/or the underlying data distribution are not stable and may change over time. Moreover, data streams pose many challenges to computing systems due to limited memory resources (i.e., the stream could not be fully stored in memory), and time (i.e., the stream should be continuously processed and the learned classification model should be ready at any time to be used for prediction).

In recent years, the field of mining evolving concept-drifting data streams has received an increasing attention and a plethora of approaches have been developed and deployed in

several applications [1, 2, 85, 86, 87]. All proposed approaches have a main objective consisting of coping with concept drift and maintaining the classification model up-to-date along the continuous flows of data. They are usually composed of a detection method to monitor the concept drift and an adaptation method used for updating the classification model over time.

Similar to the stationary classification problem, the streaming classification problem can be categorized into supervised, semi-supervised, or unsupervised depending on the availability of labeled instances, as well as into uni-dimensional or multi-dimensional depending on the number of class variables, one or more, respectively.

This chapter is restricted to the presentation of a brief overview of the state-of-the-art approaches dealing with *supervised uni-dimensional* streaming classification. Semi-supervised uni-dimensional and supervised multi-dimensional streaming classification problems will be discussed later in Chapter 7 and Chapter 8, respectively.

Chapter outline

Section 4.2 defines the concept drift problem and different types of drift. Next, Section 4.3 overviews the most prevalent change detection methods used for monitoring and signalling concept drifts. Finally, Section 4.4 presents a brief survey on state-of-the-art adaptation methods addressing different issues on how to update the classification model over time.

4.2 Concept drift

In dynamic environments, the characteristic properties of data streams are often not stable but change over time. This is known as the concept drift problem [231]. According to Tsymbal [219], there are two possible types of concept drift: *real concept drift*, defined as a change of the target concept that the classification model is trying to predict, and *virtual concept drift*, defined as a change of the underlying data distribution.

From a probabilistic point of view, and given a uni-dimensional supervised classification setting, concept drift can be defined as the change in the joint probability distribution $P(\mathbf{x}, c)$, where c is the class label of a feature vector $\mathbf{x}$. $P(\mathbf{x}, c)$ is the product of the class posterior distribution $P(c \mid \mathbf{x})$ and the feature distribution $P(\mathbf{x})$. According to Gao et al. [92], there are three possible sources of concept drifts:

- *Conditional change*: In this case, a change occurs in $P(c \mid \mathbf{x})$, that is, the class labels change given the feature vectors. For instance, a conditional change may occur in an information filtering domain consisting of classifying a stream of documents, denoted

by $\mathbf{x}$, as relevant or irrelevant, denoted by c, if the relevance of some documents changes over time, that is, their class labels change from relevant to irrelevant or vice versa. With respect to Tsymbal's concept drift categorization, a conditional change corresponds to a *real* concept drift.

- *Feature change*: In this case, a change occurs in $P(\mathbf{x})$. Intuitively, some previously infrequent feature vectors may become more frequent or vice versa. For instance, the relative frequency of some documents in information filtering domain changes over time regardless of their relevance, which may remain constant over a long period of time. With respect to Tsymbal's concept drift categorization, a feature change represents a *virtual* concept drift.

- *Dual change*: In this case, changes occur in both $P(\mathbf{x})$ and $P(c \mid \mathbf{x})$. According to the information filtering example, changes in both the relative frequency and the relevance of some documents are observed, i.e., a virtual and a real concept drift both occur together.

A similar categorization is proposed by Zhang et al. [241]. It is also based on the decomposition of $P(\mathbf{x}, c)$ into two parts, as $P(\mathbf{x}, c) = P(c \mid \mathbf{x}) \cdot P(\mathbf{x})$. In fact, they define *rigorous concept drifting* for changes in both $P(c \mid \mathbf{x})$ and $P(\mathbf{x})$, and *loose concept drifting* for changes in $P(\mathbf{x})$ only.

Moreover, depending on the rate (also known as the extent or the speed) of change, concept drift can be also categorized into either *abrupt* or *gradual*. An abrupt concept drift occurs at a specific time point by suddenly switching from one concept to another. On the contrary, in a gradual concept drift, a new concept is slowly introduced over an extended time period. Usually, the detection of abrupt drifts is easier and requires few instances to signal the change. Gradual changes, however, are more difficult to detect, since at least during the initial phases of change, the perturbations in the data stream can be seen as noise by the drift detection algorithm, and therefore, more instances are often required to distinguish gradual change from noise [88].

An additional important categorization is based on whether the concept drift is *local* or *global*. A concept drift is said to be local when it only occurs in some regions of the instance space (sub-spaces), and besides, when the type and the rate of changes also depend on specific locations in the instance space [220]. In contrast, a global concept drift refers to a change occurring in the whole instance space. As pointed out by Tsymbal [220], most gradual concept drifts may be considered as local if most regions of the data remain stable, while most

abrupt concept drifts are not local, unless substantial sub-areas remain stable between the two changing concepts.

Concept drifts may be characterized as *reoccurring concepts* if previously seen concepts reappear (generally at irregular time intervals) over time. Seasonal changes present an example of reoccurring concepts that reappear periodically and regularly (each season) [89].

Concept drifts may be also characterized as *novelties* when some new class variables, some new feature variables, or some of their respective states appear or disappear over time.

A recent categorization for concept drift was also proposed by Minku et al. [156] characterizing concept drifts according to different additional criteria, namely, *severity* (severe if no instance maintains its target class in the new concept, or intersected otherwise), *frequency* (periodic or non-periodic) and *predictability* (predictable or random).

Figure 4.1 summarizes the different presented concept drift categorizations. It is important to note that these types of drift are not exhaustive and the categorizations discussed here are not mutually exclusive. In fact, to describe a concept drift, several categorizations may be considered at the same time. For instance, a given concept drift may be simultaneously characterized as real, abrupt, and reoccurring.

Finally, in spite of the above categorizations, some existing approaches dealing with the concept drift problem do not distinguish between the different types of drifts. In fact, these methods consider that in all cases and in the presence of any kind of concept drift, the current model needs to be updated [219].

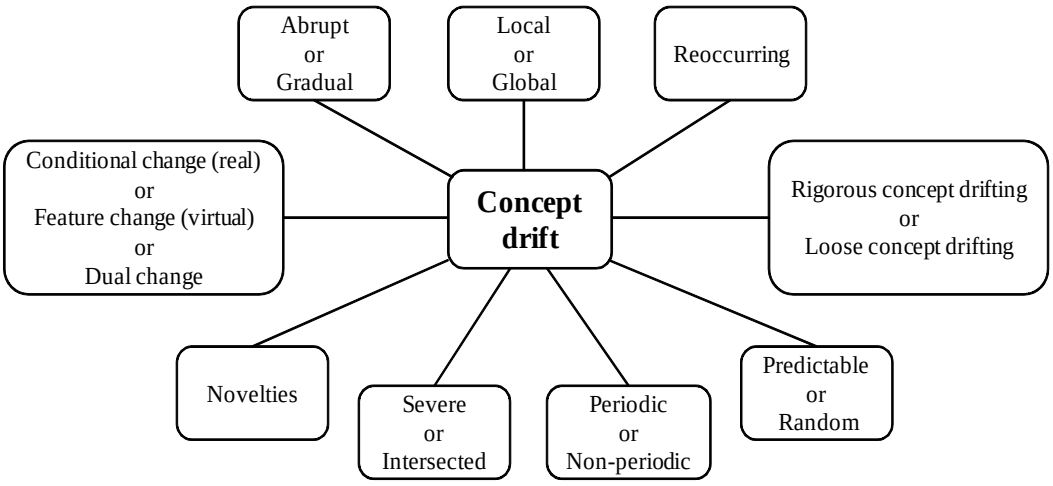


Figure 4.1: Concept drift categorizations.

4.3 Change detection methods

Detecting concept drift is a challenging task due to the diversity and instability of changes occurring over time. Moreover, another difficult issue in handling change in data streams is differentiating between a true concept drift and noise [219, 231]. Some detection methods may overreact to noise, erroneously interpreting it as change, while others may be highly robust to noise, adjusting to change too slowly. Hence, an effective change detection method should be robust to noise and resilient to false alarms, i.e., it should signal a concept drift only when it occurs and always when it occurs. Moreover, it is important for a detection method to be able to not only detect the change, but also to provide more useful information about its type and/or rate.

In recent years, various detection methods have been developed. In general, they either monitor the evolution of performance indicators over time or compare the distributions on two different data samples belonging to different time intervals (called time windows or merely windows). In what follows, we briefly review the main existing change detection methods.

4.3.1 Monitoring the evolution of performance indicators

In general, the classification accuracy represents one of the most used performance indicators for monitoring the concept drift. Roughly speaking, it is assumed that whenever there is a significant drop in the predictive accuracy, a drift should be signaled.

A relevant work in this category is the FLORA family of algorithms proposed by Widmer and Kubat [231]. To detect concept drifts, the accuracy and the coverage of the current learner are monitored over time and compared to user-defined thresholds, and the window size is adjusted accordingly (i.e., decreased when a concept drift is suspected, or increased otherwise). Klinkenberg and Renz [127] monitor the values of three performance indicators over time: accuracy, recall and precision, defining a window adjustment heuristic for text categorization problems.

Lanquillon [141] employs statistical quality control to detect concept drifts in the context of an adaptive information filtering system. Two different control charts are used to monitor the drift: a variant of the Shewhart control chart and the cumulative sum control chart. A control chart is defined as a plot of a certain characteristic value taken sequentially in time. By means of the control charts, Lanquillon monitors three performance indicators over time, namely, the sample error rate (a weighted average of the error rates measured on recent windows), the expected error rate (measuring the average uncertainty of a set of classification decisions), and the virtual rejects (corresponding to the fraction of classification decisions

which should be rejected due to uncertainty). A similar approach is also proposed by Castillo [35] where a Shewhart P-Chart is used to monitor the classification error rate over time.

Furthermore, the drift detection method (DDM) proposed by Gama et al. [90] is based on the idea of monitoring the online error rate of a learning algorithm looking for significant deviations. In accordance with the probability approximately correct (PAC) learning model [159], it is assumed that, as time advances and more instances are processed, the error rate of the learning algorithm will decrease if the target concept is stationary. Nevertheless, any significant increase of the error rate suggests that the concept is changing. DDM performs well but has difficulties when the changes are gradual and very slow.

Baena-García et al. [12] develop then the early drift detection method (EDDM) in order to improve DDM and ensure the detection in presence of slow gradual concept drift. The basic idea is to monitor the distance between two consecutive classification error rates instead of monitoring only the online error rate.

Nishida and Yamauchi [163] propose a detection method (STEPD) that uses a statistical test of equal proportions to track various types of concept drift. The key idea is to consider two accuracies: the recent one and the overall one. They assume that the accuracy of the algorithm on the recent instances will be equal to the overall accuracy from the beginning of the learning process if the target concept is stationary, whereas a significant decrease of the recent accuracy informs that there is a concept drift.

Moreover, more recently, Ross et al. [190] propose a new method called ECDD (EWMA for concept drift detection) that uses an exponentially weighted moving average (EWMA) chart to monitor the error rate of a streaming classifier. When a concept drift is detected, the current classifier is relearned using the incoming recent data. ECDD is evaluated using two different base classifiers: the linear discriminant analysis classifier and k-nearest neighbors.

4.3.2 Comparing the distributions on two windows

An alternative approach for detecting concept drift is to compare the distributions on two different windows: a reference window usually containing the past information of old instances and a current window containing the most recent data instances. The windows can have either equal or different sizes. In general, it is assumed that as long as the distribution of old instances is similar to the distribution of recent ones, no concept drift occurred. On the other hand, a distribution difference indicates a concept drift.

An example of this approach is proposed by Kifer et al. [124]. Their change detection algorithm uses non-parametric statistical tests based on Chernoff bound to compare the data distribution in some reference window to the data distribution in a current window, and decide

about the presence of a concept drift. Both windows contain a fixed number of successive data points. The current window slides forward with each incoming data point, and the reference window is only updated when a change is detected.

Vorburger and Bernstein [226] present an entropy-based metric to measure the distribution inequality between two sliding windows including respectively older and more recent instances. If the distributions are equal, the entropy measure results in a value of 1, and if they are absolutely different the measure will result in a value of 0. The entropy measure is continuously monitored over time, and a concept drift is signaled when the entropy measure decreases below a given fixed user-defined threshold.

Additional examples include the change detection methods proposed by Dasu et al. [55] and Sebastião and Gama [200]. Both use the Kullback-Leibler (KL) divergence to measure the distance between the probability distributions of two different windows (old and recent) to detect possible changes. They empirically prove the efficiency of Kullback-Leibler divergence in detecting changes and its resilience to false alarms when there are no drifts in the data.

Finally, a prominent change detector is also presented by Bifet and Gavaldà [15]. Their adaptive sliding window method (ADWIN) automatically grows the window when no change is apparent, and shrinks it when data changes. More precisely, ADWIN keeps a sliding window W with the most recently received instances and compares the distribution on two sub-windows of W by means of a statistical test. Whenever the two subwindows exhibit distinct enough averages, a change is signaled and the older portion of the window is dropped.

Table 4.1 summarizes the different presented change detection methods.

Table 4.1: Summary of change detection methods.

Change detection methods	**Monitoring the evolution of performance indicators**	Accuracy + coverage [231] Accuracy + recall + precision [127] Control charts [35, 141] Online error rate [90] Distance between error rates [12] Recent and overall accuracy [163] EWMA chart [190]
	Comparing the distributions on two windows	Statistical tests [124] Entropy-based method [226] KL divergence-based methods [55, 200] ADWIN [15]

4.4 Model adaptation methods

Different model adaptation methods have been proposed to handle concept-drifting data streams and maintain the classification model up-to-date. As pointed out in [88], model adaptation methods can be classified into two main groups: *blind methods* and *informed methods*. This taxonomy depends on whether a change detection method is employed or not. In fact, *blind adaptation methods* (also known as evolving methods) achieve adaptation without the use of a change detection method. That is, they continuously repeat, at regular time intervals, specific actions to update the classification model and keep it consistent with the new incoming data. On the contrary, *informed adaptation methods* are always used in conjunction with a change detection method and they only update the classification model after a change is signaled.

Clearly, informed adaptation methods present a more efficient way in coping with concept drifts since they avoid the uncontrolled updating of the current model, and may also provide meaningful information about the generating data (when and where the change occurs) as well as the type of change and its rate.

4.4.1 Blind adaptation methods

Basically, three major groups of blind adaptation methods can be distinguished, namely, methods based on windows of fixed size, methods continuously weighting the instances in the stream, and methods managing an evolving ensemble of classifiers.

Windows of fixed size

Methods using windows of fixed size [134, 195, 231] are based on the simple idea that the classification model is learned, at each step, only from a fixed number of the most recent instances. These windows are similar to first in, first out (FIFO) data structures [87]. Whenever a new instance (or set of instances) is received and inserted into the window, another old instance (or set of old instances) is discarded, such that the window keeps always a fixed size. In this case, the main difficulty is the choice of an appropriate window size that trades off fast adaptation in phases with concept drifts against good generalization in stable phases without concept drifts. In fact, a small window can ensure a fast adaptability in phases with drifts, but in more stable phases, it can affect the classifier performance. However, a large window would produce good and stable classification results in stable phases, but usually cannot react quickly to concept drifts and this may delay the classifier adaptation [88].

Instance weighting

In this case, old instances are not discarded but they remain always stored in memory, with a continuous decrease of their importance over time. Roughly speaking, instance weighting methods (also known as gradual aging or gradual forgetting methods) proceed by assigning lower weights to old instances according to their age and/or their relevance or competence with respect to the current concept. The basic objective here is to focus more on recent instances incorporating the new concepts, and be able in this way to keep the classification model up-to-date. Examples of these methods can be found in [126, 133, 241].

Ensemble methods

The general technique applied by the evolving ensemble methods consists of dividing the data stream into sequential blocks of fixed size, then using each of these blocks to learn an individual classifier. Afterwards, the ensemble is continuously refined by adding a new classifier, removing the oldest or the weakest classifier, or through increasing or decreasing the classifier weights using some criteria which are usually based on their current performance. Typically, the global prediction of the ensemble is obtained through either the combination of all classifier predictions using a form of voting or the selection of the best one among them.

Ensemble methods have received a lot of attention during the last years. Several works have been proposed investigating different ways of managing and updating the ensemble over time and discussing various combination and selection rules. Examples include the streaming ensemble algorithm [207], the accuracy weighted ensembles [227], the dynamic weighted majority algorithm [130], the incremental classification ensemble algorithm [236], and the ensemble integration techniques, namely, dynamic selection, dynamic voting, and dynamic voting with selection [220].

4.4.2 Informed adaptation methods

Contrary to blind adaptation methods, informed adaptation methods make use of a detection method to firstly check if a drift has occurred, then update, if necessary, the classification model. In general, informed adaptation methods are either based on adaptive windows or ensembles used jointly with a detection method, called adaptive ensembles.

Adaptive windows

The main idea behind these methods is to adjust dynamically the window size over time according to the potential changes in the data stream. An adaptive window method is nec-

essarily equipped with a detection method, that determines the change point at which the concept drift occurs. Therefore, the window size is only adjusted when a concept drift is detected. As a general and commonly used rule, if a concept drift is detected, the window size is reduced to discard out-of-date instances; otherwise, the window size is increased to include more recent instances. Afterwards, the classification model is updated accordingly with the resulting window. Prevalent methods based on adaptive windows can be found in [15, 90, 136, 163, 231].

Adaptive ensembles

Adaptive ensembles are basically introduced to improve the blind ensemble methods that keep updating continuously the ensemble without detecting if a drift has occurred or not. Specifically, adaptive ensembles incorporate a detection method at an ensemble level allowing a more efficient adaptation process. For instance, an online bagging method [165] is extended with the ADWIN detection algorithm in [16], a meta-learner composed of a set of models and equipped with a drift detection method is also proposed in [89], and a data stream classification framework that dynamically updates an ensemble of incremental classifiers is presented in [122]. Moreover, a new ensemble learning method called diversity for dealing with drifts (DDD) is recently presented in [157] ensuring a study on the diversity and the behavior of ensemble learning methods under different types of concept drifts.

Finally, note that the list of all discussed model adaptation methods, summarized in Figure 4.2, is not meant to be exhaustive, as we just focused on prevalent methods pertaining to each group.

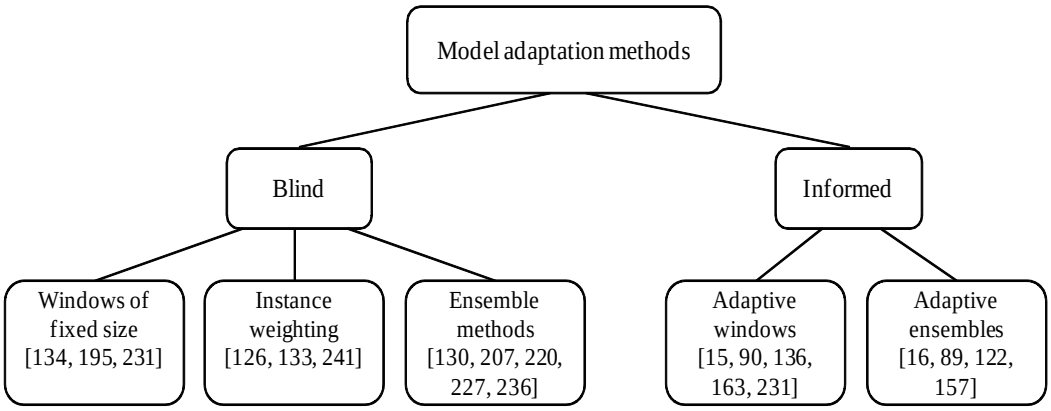


Figure 4.2: Summary of model adaptation methods.

Part III

Multi-dimensional Bayesian network classifiers for stationary data

Chapter 5

Learning CB-decomposable MBCs

5.1 Introduction

As commented in Chapter 3, multi-dimensional classification using MBCs involves the computation of the most probable explanation (MPE), which consists of finding the most likely instantiation of the class variables given an evidence about the feature variables. MPE is NP-hard and is considered as one of the most challenging problems when dealing with MBCs.

In this chapter, in order to alleviate the MPE computational burden, we consider the family of class-bridge decomposable MBCs (CB-decomposable MBCs) introduced by Bielza et al. [13]. In fact, by decomposing class and bridge subgraphs of an MBC graphical structure into w maximal connected components, the maximization problem for MPE computation can be transformed into w maximization problems operating in lower dimensional spaces. Moreover, using CB-decomposable MBCs may provide more insight about the domain and better interpretability of learned structures than large and complex MBCs which have no explicit representation for domain decomposability.

However, CB-decomposable MBCs merits have been only discussed and proved theoretically in [13], and no learning approach nor an experimental study have been presented to empirically demonstrate the usefulness of this new family of MBCs.

For these reasons, and in order to tackle the above shortcomings, we propose through this chapter a novel algorithm for learning CB-decomposable MBCs based on a wrapper greedy forward selection approach. Broadly speaking, in a first phase our algorithm, named `CB-MBC`, learns a CB-decomposable MBC with a number of maximal connected components equal to the number of class variables. This is carried out by learning a selective naive Bayes [140] for each class variable C, then, removing their possible common children to have an initial bridge

subgraph and an initial corresponding CB-decomposable MBC. In a second phase, a feature subgraph defining dependence relationships between the set of feature variables is learned. Finally, in a third phase, while the number of maximal connected components is greater than one and there is an accuracy improvement, the algorithm iteratively and sequentially merges together the components, then updates the bridge and feature subgraphs. By learning CB-decomposable MBCs, it has been proven that the computations of MPE are alleviated comparing to general MBCs. This work appears in the published paper [21].

Chapter outline

Section 5.2 defines CB-decomposable MBCs. Then, Section 5.3 describes our proposed algorithm for learning CB-decomposable MBCs from stationary data, and Section 5.4 covers the experimental study and discusses the obtained results. Finally, Section 5.5 rounds off the chapter with some conclusions.

5.2 Class-bridge decomposable MBCs

Definition 5.1. *A class-bridge decomposable multi-dimensional Bayesian network classifier (CB-decomposable MBC) is an MBC* $\mathcal{B} = (\mathcal{G},\boldsymbol{\Theta})$ *where the class subgraph* $\mathcal{G}_{\mathcal{C}}$ *and bridge subgraph* $\mathcal{G}_{\mathcal{CX}}$ *are decomposed into* w *maximal connected components, such that*

1. $\mathcal{G}_C \cup \mathcal{G}_{CX} = \bigcup_{j=1}^{w}(\mathcal{G}_{C_j} \cup \mathcal{G}_{(CX)_j})$, *where* $\mathcal{G}_{C_j} \cup \mathcal{G}_{(CX)_j}$, *with* $j = 1,\ldots,w$, *are its* w maximal connected components, *and*

2. $\mathbf{Ch}(\mathbf{V}_{C_j}) \cap \mathbf{Ch}(\mathbf{V}_{C_k}) = \emptyset$, *with* $j,k = 1,\ldots,w$ *and* $j \neq v$, *where* $\mathbf{Ch}(\mathbf{V}_{C_j})$ *denotes the children of all the variables in* $\mathbf{V}_{C_j}$, *the subset of class variables in* $\mathcal{G}_{C_j}$ (non-shared children property).

Bielza et al. [13] proved that the MPE computation can be alleviated thanks to MBC class-bridge decomposability. In fact, maximizing over the set of all class variables amounts to maximizing over each class variable subset of the identified maximal connected components, i.e., maximizing over lower dimensional subspaces than originally.

Theorem 5.1. *Given a CB-decomposable MBC where* $\mathcal{I}_j = \prod_{C\in\mathbf{V}_{C_j}} \Omega_C$ *represents the sample space associated with* $\mathbf{V}_{C_j}$, *then*

$$\max_{c_1,\ldots,c_d} p(c_1,\ldots,c_d \mid x_1,\ldots,x_m)$$
$$\propto \prod_{j=1}^{w} \max_{\boldsymbol{c}^{\downarrow \mathbf{V}_{C_j}} \in \mathcal{I}_j} \left(\prod_{C \in \mathbf{V}_{C_j}} P\big(c \mid \boldsymbol{pa}_{\mathbf{V}_C}(c)\big) \cdot \prod_{X \in \mathbf{Ch}(\mathbf{V}_{C_j})} P\big(x \mid \boldsymbol{pa}_{\mathbf{V}_C}(x), \boldsymbol{pa}_{\mathbf{V}_X}(x)\big) \right) \tag{5.1}$$

where $\boldsymbol{c}^{\downarrow \mathbf{V}_{C_j}}$ represents the projection of vector $\boldsymbol{c}$ to the coordinates found in $\mathbf{V}_{C_j}$. $\boldsymbol{Pa}_{\mathbf{V}_C}(X)$ and $\boldsymbol{Pa}_{\mathbf{V}_X}(X)$ denote, respectively, the class parents and feature parents of X in $\mathcal{G}$. $\boldsymbol{Pa}_{\mathbf{V}_C}(C)$ denotes the class parents of C in $\mathcal{G}$, and obviously, for any class variable C, $\boldsymbol{Pa}_{\mathbf{V}_X}(C) = \emptyset$.

Given $\mathbf{x}$, each expression to be maximized in Equation (5.1) will be denoted as $\phi_j^{\mathbf{x}}(\mathbf{c}^{\downarrow \mathbf{V}_{C_j}})$, $j = 1, \ldots, w$, i.e.,

$$\phi_j^{\mathbf{x}}(\mathbf{c}^{\downarrow \mathbf{V}_{C_j}}) = \prod_{C \in \mathbf{V}_{C_j}} P\big(c \mid \mathbf{pa}_{\mathbf{V}_C}(c)\big) \cdot \prod_{X \in \mathbf{Ch}(\mathbf{V}_{C_j})} P\big(x \mid \mathbf{pa}_{\mathbf{V}_C}(x), \mathbf{pa}_{\mathbf{V}_X}(x)\big)$$

It holds that $\phi_j^{\mathbf{x}}(\mathbf{c}^{\downarrow \mathbf{V}_{C_j}}) \propto P\big(\mathbf{C}^{\downarrow \mathbf{V}_{C_j}} = \mathbf{c}^{\downarrow \mathbf{V}_{C_j}} \mid \mathbf{x}\big)$.

Example 5.1. *Let us consider the MBC shown in Figure 5.1(a). It is a CB-decomposable MBC with $w = 3$. Its three maximal connected components are depicted in Figure 5.1(b):*

- *the first one is $\mathcal{G}_{C_1} \cup \mathcal{G}_{(CX)_1}$ with $\mathbf{V}_{C_1} = \{C_1, C_2\}$ and $\mathbf{Ch}(\mathbf{V}_{C_1}) = \{X_1, X_2\}$,*
- *the second is $\mathcal{G}_{C_2} \cup \mathcal{G}_{(CX)_2}$ with $\mathbf{V}_{C_2} = \{C_3\}$ and $\mathbf{Ch}(\mathbf{V}_{C_2}) = \{X_3, X_4, X_6\}$, and*
- *the third is $\mathcal{G}_{C_3} \cup \mathcal{G}_{(CX)_3}$ with $\mathbf{V}_{C_3} = \{C_4\}$ and $\mathbf{Ch}(\mathbf{V}_{C_3}) = \{X_5\}$.*

Note that $\mathbf{Ch}(\mathbf{V}_{C_1}) \cap \mathbf{Ch}(\mathbf{V}_{C_2}) = \mathbf{Ch}(\mathbf{V}_{C_1}) \cap \mathbf{Ch}(\mathbf{V}_{C_3}) = \mathbf{Ch}(\mathbf{V}_{C_2}) \cap \mathbf{Ch}(\mathbf{V}_{C_3}) = \emptyset$, as required. As a maximization problem we get:

$$\begin{aligned}
\max_{c_1,\ldots,c_4} P(C_1 = c_1, \ldots, C_4 = c_4 \mid \boldsymbol{x}) &= \max_{c_1,c_2} P(c_1)P(c_2 \mid c_1)P(x_1 \mid c_1, x_2)P(x_2 \mid c_1, c_2) \\
&\quad \cdot \max_{c_3} P(c_3)P(x_3 \mid c_3)P(x_4 \mid c_3)P(x_6 \mid c_3, x_3) \\
&\quad \cdot \max_{c_3} P(c_4)P(x_5 \mid c_4, x_1, x_6) \\
&= \max_{c_1,c_2} \phi_1^{\boldsymbol{x}}(c_1, c_2) \cdot \max_{c_3} \phi_2^{\boldsymbol{x}}(c_3) \cdot \max_{c_4} \phi_3^{\boldsymbol{x}}(c_4)
\end{aligned}$$

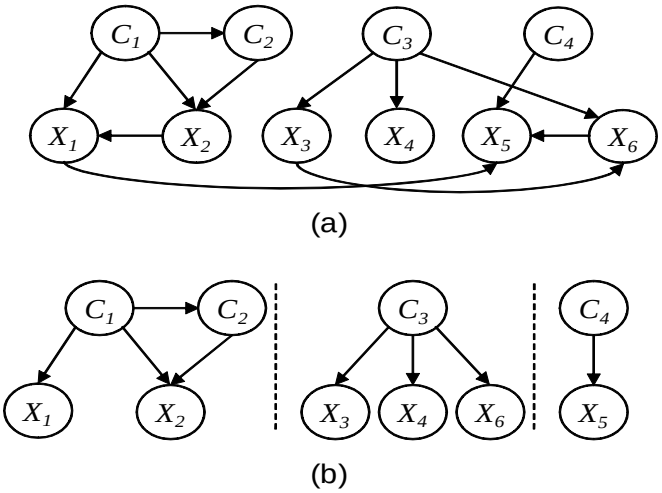


Figure 5.1: Example of (a) a class-bridge decomposable MBC and (b) its three maximal connected components.

5.3 CB-MBC Algorithm

In this section, we introduce a new algorithm for learning CB-decomposable MBCs from stationary data based on a wrapper greedy forward selection approach.

Our learning algorithm consists of three main phases, outlined by Algorithm 5.1, and detailed in what follows. Note that, in the different phases, the classifier accuracy is denoted Acc, which is equal to Acc_m (see Equation (3.4)) or Acc_g (see Equation (3.5)) depending on using the mean or the global accuracy, respectively.

5.3.1 Phase I: Learn bridge subgraph

Starting from an empty graphical structure, the first step in this phase is learning a selective naive Bayes (SNB) [140] for each class variable C_j, $j = 1, \ldots, d$ (step 3). The initial state of each selective naive Bayes is an empty set of features. Using a wrapper greedy search technique, the algorithm iteratively associates with each class the feature providing the best accuracy, and ends when no feature improves more the accuracy.

The d resulting selective naive Bayes models represent $w = d$ maximal connected components that may have common children. Thus, the next step is to check the non-shared children property in order to induce an initial CB-decomposable MBC. This is accomplished by removing, if necessary, all common children, based on two criteria, namely, the feature insertion rank and the accuracy (steps 4 to 17).

Let $rank_i^j$ denotes the insertion rank of feature X_i in the selective naive Bayes SNB_j for C_j, and $rank_i^k$ denotes the insertion rank of feature X_i in the selective naive Bayes SNB_k for C_k. $rank_i^j < rank_i^k$ means that X_i is firstly selected by SNB_j. Hence, in this case, X_i will be kept in SNB_j and removed from SNB_k. Otherwise, and in case that $rank_i^j = rank_i^k$, we proceed to compare the accuracies Acc^j and Acc^k, denoting respectively the accuracy of SNB_j and SNB_k when X_i was included in, then keep X_i in the SNB presenting the best accuracy and remove it from the other.

The result of this first phase is a simple CB-decomposable MBC, denoted $\textit{CB-MBC}_w^b$, where only the bridge subgraph is defined and the class and feature subgraphs are still empty (step 18).

5.3.2 Phase II: Learn feature subgraph

This phase consists of learning the feature subgraph by introducing the dependence relationships between the feature variables. Since it may be impractical to consider all possible arc additions between the feature variables, especially if the number of features m is large, we will fix a parameter T as a maximum number of iterations (steps 20 to 25).

In each iteration, an arc is selected at random between a pair of feature variables. If there is an accuracy improvement, the arc is added to $\mathcal{G}_X$, otherwise it is discarded and will not be considered in subsequent iterations. This phase ends when T is reached, and the induced MBC is denoted as $\textit{CB-MBC}_w^{bf}$ (step 26).

Note that, thanks to MBC decomposability, the classification accuracy associated with the arc addition in each iteration can be evaluated in a straightforward and local way. In fact, after adding an arc from X_i to X_k, only the term corresponding to variable X_k changes, that is, only the MPE computation of the maximal connected component to which X_k pertains, changes and needs to be reevaluated. The MPE computation over all remaining maximal connected components remains unchanged, which considerably reduces the computational burden.

5.3.3 Phase III: Merge maximal connected components

Taking as input the CB-decomposable MBC found in the previous phase, having w maximal connected components and a corresponding accuracy denoted Acc^w, the third phase consists of learning the class subgraph, which leads to merging the maximal connected components of the current CB-decomposable MBC, then updating the bridge and feature subgraphs.

As a first step, all possible arc additions between the class variables pertaining to different maximal connected components are evaluated (steps 30 to 32). If there is an accuracy

improvement, i.e., $Acc^{w-1} > Acc^{w}$, the subgraph $\mathcal{G}_C$ is updated by adding the arc improving the accuracy the most, and w is reduced to $w-1$ maximal connected components (steps 33 to 38).

Subsequently, a bridge update step is performed inside the new induced maximal connected component. Dependence relationships that may be added from class to feature variables of the corresponding component are greedily evaluated, and only when there is an accuracy improvement, the best one is added (steps 39 to 41).

Note that, once again, the MBC decomposability plays a key role in alleviating the complexity of MPE computation since each possible arc addition between class variables or from class to feature variables is evaluated locally. Moreover, for bridge updating step, the MPE is only recomputed for the new merged component, which alleviates more the complexity.

The last step in this phase consists of updating the feature subgraph by inserting, one by one, additional arcs between feature variables while this improves the accuracy (steps 42 to 44).

This phase iterates over these three steps, and terminates when no more component merging can improve the accuracy or until the condition $w = 1$ is reached. Finally, a CB-decomposable MBC denoted as $\textit{CB-MBC}_w^{bfc}$ is returned at step 49.

5.4 Experimental study

In order to evaluate our `CB-MBC` learning algorithm, we firstly perform experiments with a synthetic data set. We randomly generate an MBC, containing 6 class and 10 feature binary variables, decomposed into 3 maximal connected components. Then, we randomly sample a data set of size 1000 using the probabilistic logic sampling method [107]. We apply our algorithm denoted as `CB-MBC`, and other four MBC learning algorithms, namely, `Tree-Tree` [222], `Polytree-Polytree` [61], `Pure Filter` [13] and `Pure Wrapper` [13], all starting from an empty structure.

We consider both the mean (see Equation (3.4)) and the global accuracy (see Equation (3.5)) to learn and then evaluate the performance of the classifiers. Furthermore, in order to test the ability of the classifiers to recover the initial MBC structure, we compare each learned structure (LS) to the initial one (IS) using the following structural evaluation metrics:

- M1: percentage of arcs in LS that are present in IS, i.e., percentage of correctly-found arcs.
- M2: percentage of arcs in LS that are absent in IS, i.e., percentage of superfluous arcs.

- M3: percentage of arcs in IS that are oriented in an opposite direction in LS, i.e., percentage of badly-oriented arcs.
- M4: percentage of arcs in IS that are absent in LS, i.e., percentage of missing arcs.

Five-fold cross-validation experiments are run for each MBC learning method. Table 5.1 shows the average results over these runs.

Table 5.1: Experimental results over the synthetic data set.

Mean accuracy					
Method	Acc_m	M1	M2	M3	M4
`CB-MBC`	0.7182	26.66	37.55	11.88	61.44
`Tree-Tree`	0.7129	30.55	49.44	5.55	63.89
`Polytree-Polytree`	0.6330	30.10	15.10	6.21	63.66
`Pure Filter`	0.5351	7.55	14.66	8.88	83.55
`Pure Wrapper`	0.7098	22.22	42.88	17.77	60.00
Global accuracy					
Method	Acc_g	M1	M2	M3	M4
`CB-MBC`	0.2877	11.55	14.66	1.33	87.10
`Tree-Tree`	0.2838	25.00	53.88	8.33	66.66
`Polytree-Polytree`	0.2845	28.99	15.10	5.44	65.55
`Pure Filter`	0.2160	7.99	15.55	7.54	84.44
`Pure Wrapper`	0.2800	16.00	48.44	14.22	69.77

Our algorithm outperforms the state-of-the-art algorithms in terms of mean and global accuracy. For structural evaluation, `Tree-Tree` and `Polytree-Polytree` present the best percentages of correctly-found arcs (M1) while `Pure Filter` has the lowest one. `Tree-Tree` and `Pure Wrapper` induce the highest percentages of superfluous arcs (M2), and `Pure Wrapper` also induces a high percentage of badly-oriented arcs (M3) comparing to the rest of the algorithms. Moreover, we may observe that, with global accuracy, the learned structures are sparser, leading to more important percentages of missing arcs (M4) for all MBC learning algorithms.

As additional experiments, we consider the real data set Emotions [212]. It is about a multi-dimensional classification of music into emotions. It contains 72 music features for 593 songs categorized into one or more out of 6 classes of emotions: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-aggressive.

As previously, the accuracies of the considered learning algorithms are computed using five-fold cross-validation. The results are summarized in Table 5.2. Note that, with this real data set, we do not have an initial MBC structure, so the structural evaluation is omitted in

this set of experiments.

Table 5.2: Experimental results over Emotions data set.

Method	Acc_m	Acc_g
`CB-MBC`	0.8326	0.3639
`Tree-Tree`	0.8135	0.2977
`Polytree-Polytree`	0.8052	0.3422
`Pure Filter`	0.6733	0.2690
`Pure Wrapper`	0.8293	0.3650

From Table 5.2, we may conclude that our algorithm performs well. In fact, with the mean accuracy, `CB-MBC` presents the best accuracy, while with the global accuracy, `Pure Wrapper` slightly outperforms `CB-MBC`.

Finally, in Figure 5.2, we plot the computation learning time of the various learning algorithms, using the mean and global accuracy, for both synthetic and Emotions data sets.

Clearly, the algorithms using a filter approach require less computation than those using a wrapper approach. Moreover, the computation time of our algorithm is lower than the other wrapper approaches, mainly over the synthetic data set, which is basically due to the MBC CB-decomposability and the alleviation of MPE computation. Note also that the computation time with global accuracy is lower, since it is more difficult to improve the learned models, so that the algorithm ends in earlier iterations.

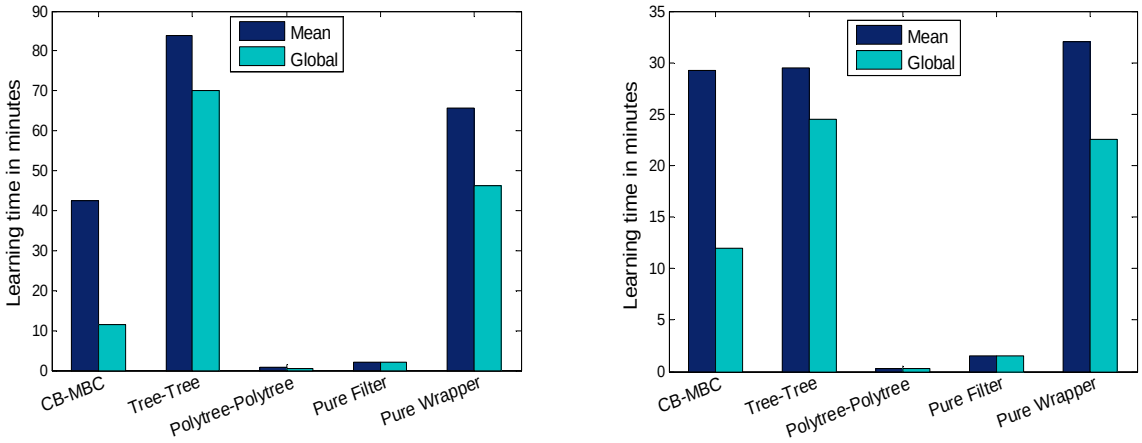


Figure 5.2: Computation learning times over (a) synthetic and (b) Emotions data sets.

5.5 Conclusion

A novel algorithm for learning CB-decomposable MBCs from data based on a wrapper forward selection approach has been presented in this chapter. Indeed, CB-decomposability allows the alleviation of MPE computations. Experimental results with both synthetic and real-world data sets show that our algorithm performs well and requires less computation time than the state-of-the-art wrapper learning algorithms.

Various possible improvements of algorithm `CB-MBC` could be investigated and carried out. For instance, we may intend to test the alternation between forward and backward wrapper selection techniques, instead of only relying on a forward selection approach. Furthermore, it would be interesting to study the use of a filter approach mainly for the feature subgraph learning step in order to avoid the random arc additions between features.

Algorithm 5.1: `CB-MBC` algortihm

1. Initialization: $\mathcal{G}_C = \emptyset$; $\mathcal{G}_{CX} = \emptyset$; $\mathcal{G}_X = \emptyset$; $w = d$.
2. **[Phase I: Learn bridge subgraph]**
3. Learn selective naive Bayes SNB_j, $j = 1, \ldots, w$.
4. **for** Each SNB_j, SNB_k having a common feature X_i **do**
5. **if** $rank_i^j < rank_i^k$ **then**
6. Remove X_i from SNB_k
7. **else**
8. **if** $rank_i^j = rank_i^k$ **then**
9. **if** $Acc^j > Acc^k$ **then**
10. Remove X_i from SNB_k
11. **else**
12. Remove X_i from SNB_j
13. **end if**
14. **end if**
15. Remove X_i from SNB_j
16. **end if**
17. **end for**
18. Obtain $\mathcal{G}_{CX} = \bigcup_{j=1}^{w}(\text{SNB}_j)$, that is, $\textit{CB-MBC}_w^b$
19. **[Phase II: Learn feature subgraph]**
20. **for** $TrialNumber = 1 : T$ **do**
21. Add randomly one arc to $\mathcal{G}_X$
22. **if** No accuracy improvement **then**
23. Discard the arc and do not consider it in subsequent iterations
24. **end if**
25. **end for**
26. Obtain $\textit{CB-MBC}_w^{bf}$
27. **[Phase III: Merge maximal connected components]**
28. $\textit{CB-MBC}_w^{bfc} \longleftarrow \textit{CB-MBC}_w^{bf}$; Stop = False.
29. **while** $w > 1$ and *not Stop* **do**
30. **for** Each C_j, C_k pertaining to two different maximal connected components **do**
31. Evaluate the arc insertion from C_j to C_k
32. **end for**
33. Select the arc with the best accuracy Acc^{w-1}
34. **if** $Acc^{w-1} > Acc^w$ **then**
35. Update $\mathcal{G}_C$
36. $Acc^w = Acc^{w-1}$
37. $\textit{CB-MBC}_w^{bfc} \longleftarrow \textit{CB-MBC}_{w-1}^{bfc}$
38. $w = w - 1$
39. **while** Accuracy improvement **do**
40. Update $\mathcal{G}_{CX}$: add an arc from a class to a feature of the new merged component
41. **end while**
42. **while** Accuracy improvement **do**
43. Update $\mathcal{G}_X$: add an arc between feature variables
44. **end while**
45. **else**
46. Stop = True
47. **end if**
48. **end while**
49. **return** $\textit{CB-MBC}_w^{bfc}$

Chapter 6

Learning MBCs using Markov blankets

6.1 Introduction

The main motivation for this contribution is to tackle the shortcomings of our previous `CB-MBC` learning method presented in Chapter 5, mainly its computational cost, by taking advantage of the merits of a filter constraint-based approach. This should considerably lighten the computational burden, especially when the data set includes a large number of class and feature variables, while guaranteeing good performance.

Additionally, applying our previous learning method may not always lead to an accurate MBC structure, since arcs between features are selected at random in the feature subgraph learning steps. In fact, in each feature subgraph learning step, `CB-MBC` iteratively selects a random arc between features, then adds it if it improves the accuracy. This means that, in each iteration, no exhaustive search is performed to add the arc that improves the most the accuracy. Such exhaustive search is avoided since, as pointed out in Section 5.3, it may be impractical and very time-consuming. In other words, these random arcs added to the feature subgraph, though they improve in each iteration the accuracy, they may not lead to the MBC structure with the best accuracy. Therefore, this may affect the overall quality of the learned MBC structure and consequently lead to misinterpretations.

To deal with this issue, and motivated by the fact that the classification is unaffected by parts of the structure that lie outside the Markov blankets of the class variables, we introduce in this chapter a novel MBC learning algorithm based on Markov blankets.

In recent years, several specialized Markov blanket learning methods have been proposed

in the literature, such as GS, TPDA, IAMB and its variants, MMHC, MMMB and HITON (see [4, 5] and their references for reviews). In this chapter, we only consider and adapt the HITON algorithm [4, 5, 7] extended to the context of multi-dimensional Bayesian network classifiers. In fact, the HITON algorithm was empirically proven to outperform most of the state-of-the-art Markov blanket discovery algorithms in terms of both classification performance and feature set parsimony [4].

Our approach, named Markov blanket MBC (`MB-MBC`), firstly consists of determining the Markov blanket around each class variable using the HITON algorithm and then specifying the directionality over all three MBC subgraphs. `MB-MBC` is applied to two different and important real-world multi-dimensional problems. The first one deals with predicting human immunodeficiency virus type 1 (HIV-1) reverse transcriptase and protease inhibitors given an input set of corresponding resistance mutations that an HIV patient carries, and the second consists of predicting the European Quality of Life-5 Dimensions (EQ-5D) from the 39-item Parkinson's Disease Questionnaire (PDQ-39). Both applications will be discussed and presented in more details throughout this chapter. This work appears in the published papers [22] and [24], and the submitted paper [23].

Chapter outline

The remainder of this chapter is organized as follows. The `MB-MBC` learning approach is described in Section 6.2. Then, the first case study in the data sets of HIV-1 reverse transcriptase and protease inhibitor prediction is presented in Section 6.3, while the second case study in the prediction of EQ-5D from PDQ-39 is introduced in Section 6.4. Finally, a summary and some conclusions of the chapter can be found in Section 6.5.

6.2 `MB-MBC` Algorithm

In this section, we describe our new `MB-MBC` algorithm for learning MBCs from stationary data based on Markov blanket discovery. As far as we know, this is the first MBC learning algorithm relying exclusively on a constraint-based approach.

The idea of `MB-MBC` learning algorithm is simple and consists of applying the HITON algorithm to each class variable and then specifying directionality over the MBC subgraphs. HITON identifies the Markov blanket of each class variable in a two-phase scheme, HITON-MB and HITON-PC, outlined respectively in Algorithms 6.1 and 6.2.

Step 1 of HITON-MB identifies the parents and children of each class variable C_i, denoted $PC(C_i)$, by calling the HITON-PC algorithm. Then, it determines the parents-children set

for every member T of $PC(C_i)$ (steps 2 to 4). The Markov blanket set $MB(C_i)$ is initialized with $PC(C_i)$ (step 5) and the set $\mathbf{S}$ includes potential spouses of C_i (step 6). From steps 7 to 14, HITON-MB loops over all members of $\mathbf{S}$ to identify correct spouses of C_i. $MB(C_i)$ is finally returned in step 15.

Algorithm 6.1: HITON-MB(C_i)

1. $PC(C_i) \leftarrow$ HITON-PC(C_i)
2. **for** every variable $T \in PC(C_i)$ **do**
3. $\quad PC(T) \leftarrow$ HITON-PC(T)
4. **end for**
5. $MB(C_i) \leftarrow PC(C_i)$
6. $\mathbf{S} \leftarrow \{\bigcup_{T \in PC(C_i)} PC(T)\} \setminus \{PC(C_i) \cup C_i\}$
7. **for** every variable $X \in \mathbf{S}$ **do**
8. $\quad$ Retrieve a subset $\mathbf{Z}$ s.t. $I(X, C_i \mid \mathbf{Z})$
9. $\quad$ **for** every variable $T \in PC(C_i)$ s.t. $X \in PC(T)$ **do**
10. $\quad\quad$ **if** $\neg I(X, C_i \mid \mathbf{Z} \cup \{T\})$ **then**
11. $\quad\quad\quad$ Insert X into $MB(C_i)$
12. $\quad\quad$ **end if**
13. $\quad$ **end for**
14. **end for**
15. **return** $MB(C_i)$

HITON-PC starts with an empty set of candidates $PC(T)$, ranks the variables X in OPEN by priority of inclusion according to $I(X,T)$ and discards variables having $I(X,T) = 0$. Then, for every new variable inserted into $PC(T)$, it checks if there is any variable inside $PC(T)$ that is independent of T given some subset $\mathbf{Z}$. In this case, this variable will be removed from $PC(T)$ (steps 6 to 11). These steps are iterated until there are no more variables in OPEN. Finally, $PC(T)$ is filtered using the symmetry criterion (steps 13 to 17). In fact, for every $X \in PC(T)$, the symmetrical relation holds iff $T \in PC(X)$. Otherwise, i.e., if $T \notin PC(X)$, X will be removed from $PC(T)$. At the end of this step, we obtain $PC(T)$.

Note that the complexity of both algorithms could be controlled using a parameter max_{CS} restricting the maximum number of elements in the conditioning sets $\mathbf{Z}$ [4]. In our experiments with HIV-1 reverse transcriptase and protease inhibitor data sets, we use the G^2 statistical test to evaluate the conditional independencies between variables with a threshold significance level of $\alpha = 0.05$, and we consider different values of $max_{CS} = 1, 2, 3, 4, 5$.

Unlike the HITON algorithm that only determines the Markov blanket of a single target variable for solving the variable selection problem, our algorithm considers many target vari-

Algorithm 6.2: HITON-PC(T)

1. $PC(T) \leftarrow \emptyset$
2. OPEN $\leftarrow \mathbf{U} \setminus \{T \cup PC(T)\}$
3. Sort the variables X in OPEN in descending order according to $I(X, T)$
4. Remove from OPEN variables X having $I(X, T) = 0$
5. **repeat**
6. Insert at end of $PC(T)$ the first variable in OPEN and remove it from OPEN
7. **for** every variable $X \in PC(T)$ **do**
8. **if** $\exists\ \mathbf{Z} \subseteq PC(T) \setminus \{X\}$, s.t. $I(X, T \mid \mathbf{Z})$ **then**
9. Remove X from $PC(T)$.
10. **end if**
11. **end for**
12. **until** OPEN $= \emptyset$
13. **for** every variable $X \in PC(T)$ **do**
14. **if** $T \notin PC(X)$ **then**
15. Remove X from $PC(T)$
16. **end if**
17. **end for**
18. **return** $PC(T)$.

ables, then induces the MBC graphical structure. Given the MBC definition, direct parents of any class variable C_i, $i = 1, ..., d$, can only be among the remaining class variables, whereas direct children or spouses of C_i can include either class or feature variables. We can then easily deduce the different MBC subgraphs based on the results of the HITON algorithm:

- *Class subgraph*: we firstly insert an edge between each class variable C_i and any class variable belonging to its corresponding parents-children set $PC(C_i)$. Then, we direct all these edges using the PC algorithm's edge orientation rules [205].

- *Bridge subgraph*: this is built by inserting an arc from each class variable C_i to every feature variable belonging to $PC(C_i)$.

- *Feature subgraph*: for every feature X in the set $MB(C_i) \setminus PC(C_i)$, i.e., for every spouse X, we insert an arc from X to the corresponding common child given by PC(X) $\cap$ PC(C_i).

Example 6.1. *Let us assume that we apply HITON algorithm to a data set coming out of the MBC structure of Figure 6.1. By the end of HITON-PC and HITON-MB algorithms, we identify, respectively, the parents-children and the Markov blanket sets of each class variable:*

- $PC(C_1) = \{C_2, C_3, X_2, X_4\}$; $MB(C_1) = PC(C_1)$
- $PC(C_2) = \{C_1, X_1, X_2\}$; $MB(C_2) = \{C_1, C_3, X_1, X_2, X_4\}$
- $PC(C_3) = \{C_1, X_6\}$; $MB(C_3) = \{C_1, C_2, X_6, X_3, X_7\}$
- $PC(C_4) = \{X_3, X_7\}$; $MB(C_4) = PC(C_4)$

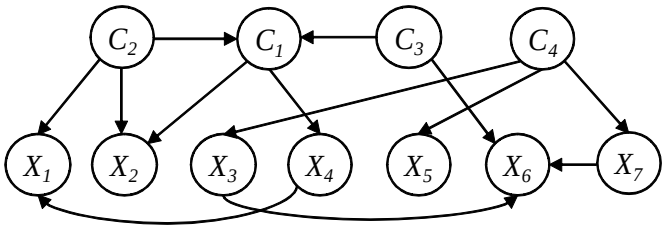


Figure 6.1: Example of an MBC structure.

Next, we specify the three MBC subgraphs as follows:

- *Class subgraph: edges are inserted between the class variables C_1, C_2 and C_3. Then, using the PC algorithm's edge orientation rules, these edges are directed from C_2 and C_3 to C_1.*
- *Bridge subgraph: arcs are inserted from C_1 to X_2 and X_4; from C_2 to X_1 and X_2; from C_3 to X_6; and from C_4 to X_3, X_5 and X_7.*
- *Feature subgraph: given that $MB(C_2) \setminus PC(C_2) = \{X_4\}$, an arc is inserted from spouse X_4 to the common child X_1 determined by $PC(X_4) \cap PC(C_2) = \{X_1\}$. Similarly, given that $MB(C_3) \setminus PC(C_3) = \{X_3, X_7\}$ and $PC(X_3) \cap PC(C_3) = PC(X_7) \cap PC(C_3) = \{X_6\}$, arcs are inserted from X_3 and X_7 to X_6.*

6.3 Case study in predicting HIV-1 RTIs and PIs

Commonly used therapies for human immunodeficiency virus type 1 (HIV-1) consists of combinations or cocktails of antiretroviral drugs, that should be repeatedly administered for each patient to treat the HIV infection. Typically, these drugs may belong to one or more different drug groups that target different stages of the viral HIV-1 life cycle.

We applied `MB-MBC` algorithm to the problem of predicting HIV-1 reverse transcriptase inhibitors (RTIs) and protease inhibitors (PIs) given an input set of corresponding resistance

mutations that an HIV-1 patient carries. The objective here is not to build only an MBC with a high predictive power but also to discover the resistance pathways of each HIV-1 drug by means of the learned MBC graphical structure.

We analyzed both reverse transcriptase and protease data sets obtained from the online Stanford HIV-1 database [185]. In the reverse transcriptase data set (respectively protease data set), the class variables are ten reverse transcriptase inhibitors (respectively eight protease inhibitors) and the feature variables are 38 predefined mutations [119] associated with resistance to reverse transcriptase inhibitors (respectively 74 predefined mutations [119] associated with resistance to protease inhibitors).

The section continues in the two next subsections with presenting more details about the used HIV-1 data sets. Then, in Subsection 6.3.3, the experimental design and the obtained results are shown and discussed for reverse transcriptase and protease inhibitors, respectively.

6.3.1 Reverse transcriptase inhibitors (RTIs)

RTIs consist of two groups of antiretroviral drugs preventing HIV-1 replication, namely nucleoside and nucleotide reverse transcriptase inhibitors (NRTIs) and non-nucleoside reverse transcriptase inhibitors (NNRTIs). NRTIs inhibit reverse transcription by being incorporated into the newly synthesized viral deoxyribonucleic acid (DNA) and preventing its further elongation [8]. We study seven drugs in this group: Abacavir (ABC), Didanosine (DDI), Emtricitabine (FTC), Lamivudine (3TC), Stavudine (D4T), Tenofovir (TDF), and Zidovudine (AZT). NNRTIs inhibit reverse transcriptase directly by binding to the enzyme, restricting its mobility and making it unable to function [8]. We consider three drugs in this group: Efavirenz (EFV), Nevirapine (NVP), and Delavirdine (DLV).

We studied a data set obtained from the Stanford HIV-1 reverse transcriptase database [185] containing treatment histories from 2855 patients that received either NRTIs, NNRTIs or both. These treatment histories were collected from previously published studies and all belonged to subtype B. There may be one or multiple isolates for the same patient. Each isolate corresponds to a sample in the data set, including a list of resistance mutations and a combination of RTIs administered to a patient at a specified time point during his or her course of RTI treatment. Only samples where no drug was administered were discarded. Accordingly, the final data set contained a total of 4884 samples. Note that the number of RTIs in the administered combinations varies from 1 to 8 drugs, such that the highest number of samples comprise 5 RTIs (1852 samples) and 6 RTIs (1600 samples). There are only 698 samples including 4 RTIs, 483 samples including 7 RTIs, 157 samples including 3 RTIs, 56 samples including 8 RTIs, and finally, we have only 17 and 25 samples, respectively for 1 and

2 RTIs.

Additionally, we considered a total of 38 mutations associated with resistance to RTIs and defined in the latest International AIDS Society-USA resistance mutation list [119]. There are no common resistance mutations between the two RTI groups; in fact, 22 mutations are associated with NNRTs and 16 mutations are associated with NNRTIs.

6.3.2 Protease inhibitors (PIs)

PIs represent the third group of antiretroviral drugs. They bind to the protein cleavage site, and therefore prevent the enzyme from releasing the individual core proteins and virus particles from subsequently maturing as infections [225]. We considered eight PI drugs: Atazanavir (ATV), Darunavir (DRV), Fosamprenavir (FPV), Indinavir (IDV), Lopinavir (LPV), Nelfinavir (NFV), Saquinavir (SQV) and Tipranavir (TPV), and we analyzed a data set obtained from the Stanford HIV-1 protease database [185] containing antiretroviral PI treatment histories from 1255 patients. As with the RTI data set, the treatment histories were collected from previously published studies. There may be one or more patient instances in the data set such that each instance includes a list of resistance mutations and a set of administered PIs drugs. Only samples where no drug was administered were discarded.

The final data set contained a total of 4341 samples belonging mainly to subtype B (subtype B: 92%, subtype C: 3%, and other subtypes (A, D, F, G, H, J, K, CRF01_AE, CRF02_AG): 5%). Note also that the number of PI combinations is not evenly represented; in fact, there are 3256 samples including only 1 PI, 862 samples including 2 PIs, 213 samples including 3 PIs and only 10 samples containing 4 PIs.

Using the International AIDS Society-USA resistance mutation list [119], we also considered a set of established PI resistance mutations. The total number of mutations in the protease gene associated with resistance to PIs is 74, where 23 are classified as major and the remainder as minor mutations. *Major mutations* are defined as mutations selected first in the presence of the drug or mutations substantially reducing drug susceptibility, whereas *minor mutations* generally emerge later than major mutations and, by themselves, do not have a substantial effect [119].

In both the RTI and PI data sets, drug combinations (respectively resistance mutations) were represented using binary vectors such that every value indicates either the presence, 1, or absence, 0, of an individual drug (respectively an individual resistance mutation) in the corresponding sample of each data set. Using two multi-dimensional Bayesian network classifiers learned separately from each data set, we were able to predict the antiretroviral combination

of RTI and PI therapies given sets of corresponding input resistance mutations. Thanks to the graphical structure of the learned MBCs, we were also able to investigate dependencies among classes (i.e., RTI or PI drugs), features (i.e., RTI or PI resistance mutations) and between classes and features (i.e., interactions between RTI or PI drugs and their respective resistance mutations).

6.3.3 Experimental design

We compare `MB-MBC` algorithm with what is defined as a `independent classifiers` method (called binary relevance in the literature on multi-label classification) where each classifier with one class variable is learned independently using the same HITON algorithm [4, 5]. In what follows, we denote `independent classifiers` method as `IndepMBs`. Additionally, we compare `MB-MBC` with five other MBC learning algorithms, namely, `Tree-Tree` [222], `Polytree-Polytree` [61], `Pure Filter` [13], `Pure Wrapper` [13], and class-bridge decomposable MBC (`CB-MBC`) presented in Chapter 5.

As non Bayesian network-based approaches, we consider for comparison three different methods: multi-label k-nearest neighbors (`ML-`k`NN`) [240], back propagation for multi-label learning (`BP-MLL`) [239], and multinomial logistic regression (`MNL`) [101]. `ML-`k`NN` extends the k-nearest neighbors lazy algorithm to a multi-label version and uses the maximum a posteriori principle to predict the label set; `BP-MLL` is derived from the popular back propagation algorithm by modifying its error function with a new function that takes into account the characteristics of multi-label learning; and `MNL` uses the multinomial logistic regression on an input set of feature variables, and returns the class value with the highest posterior probability. Similar to `IndepMBs`, `MNL` is applied independently to each class variable, and the results are then concatenated to obtain the predicted class vector.

All methods were run in Matlab R2010b on a laptop 2.2 GHz with 6 GB RAM using Windows operating system. The HITON algorithm was run using Causal Explorer Toolkit [6] provided as compiled Matlab functions, and G^2 statistical test was used to evaluate the conditional independencies between variables with a significance level $\alpha = 0.05$. `ML-`k`NN` and `BP-MLL` were run using the Matlab packages from `http://lamda.nju.edu.cn/datacode/MLkNN.htm` and `http://lamda.nju.edu.cn/datacode/BPMLL.htm`, respectively. For the `ML-`k`NN` algorithm, the number of clusters was to set to 4 for both RTI and PI data sets, and for `BP-MLL` the number of training epochs was set to 20, and the number of hidden neurons was set to 7 for the RTI data set and 14 for the PI data set. For the remaining MBC learning approaches and `CB-MBC`, Matlab implementations from [13] and [21] were used, respectively.

Five 10-fold cross-validation experiments were run for each learning algorithm for both

RTI and PI data sets. Bayesian network-based methods all start from an empty structure. For `MB-MBC` and `IndepMBs` methods, the five 10-fold cross-validation experiments were run with five different conditioning set size values, i.e., with $max_{CS} = 1, 2, 3, 4, 5$. As metrics, we consider both the mean (see Equation (3.4)) and the global accuracy (see Equation (3.5)) to evaluate the predictive performance of the learned classifiers.

6.3.4 RTIs analysis results

Table 6.1 shows the prediction results for the RTI data set with mean values and standard deviations for each metric and each learning method. The best result of each metric is written in bold. `ML-`k`NN` presents the best mean accuracy (73%), whereas `MB-MBC` outperforms the remaining approaches in the global accuracy with 11%. Note that the best results for the `MB-MBC` algorithm are obtained with $max_{CS} = 1$, and as max_{CS} grows, the overall mean and global accuracies decrease. We performed a multiple comparison of all algorithm performances using the Friedman test followed by the Tukey-Kramer post-hoc test with a significance level of $\alpha = 0.05$. For the mean accuracy, it turns out that (1) `ML-`k`NN` and `MNL` are significantly better than `MB-MBC` with $max_{CS} = 4$, `MB-MBC` with $max_{CS} = 5$, `Tree-Tree`, and `Polytree-Polytree`; (2) `IndepMBs` with $max_{CS} = 2$ are significantly better than `MB-MBC` with $max_{CS} = 4$, `MB-MBC` with $max_{CS} = 5$, and `Tree-Tree`; and (3) `IndepMBs` with $max_{CS} = 1$ is only significantly better than `Tree-Tree`. For the global accuracy, it turns out that `MB-MBC` with $max_{CS} = 1$ is significantly better than `IndepMBs` with $max_{CS} = 5$, `Tree-Tree`, `Polytree-Polytree`, `Pure Filter`, and `Pure Wrapper`. For all remaining algorithms, the differences in classification performance are not statistically significant.

Using the learned graphical structure of the most accurate MBC, shown in Figure 6.2, we identified and analyzed the different interactions in the RTI data set between drugs belonging to both the NRTI and NNRTI groups and established resistance mutations.

Firstly, the class subgraph (red arcs) in the RTI network shows associations between the following NRTI drugs: AZT, ABC, 3TC, TDF and DDI, which may reveal the extent of cross-resistance between each related pair of these drugs. The NRTI drug D4T has a unique association with the NRTI drug AZT, and two associations with the NNRTI drugs EFV and NVP. Note that these identified dependence relationships are partially consistent with the previous work by Deforche et al. [63] that proved the existence of direct influences between the NRTI drugs AZT, 3TC, ABC, DDI, and D4T, as well as direct influences between the NNRTI drugs EFV and NVP and D4T. Deforche et al. also used Bayesian networks to discover interactions between drugs and resistance mutations within and between NRTIs and NNRTIs; however, contrary to our approach, they do not deal with the HIV treatment

Table 6.1: Estimated accuracies (mean ± std. dev.) over RTI data set.

Method		Mean accuracy	Global accuracy
MB-MBC	$max_{CS} = 1$	0.7108 ± 0.0221	$\mathbf{0.1151 \pm 0.0466}$
	$max_{CS} = 2$	0.7062 ± 0.0191	0.0881 ± 0.0403
	$max_{CS} = 3$	0.7019 ± 0.0153	0.0780 ± 0.0363
	$max_{CS} = 4$	0.6995 ± 0.0145	0.0701 ± 0.0336
	$max_{CS} = 5$	0.6978 ± 0.0106	0.0646 ± 0.0241
IndepMBs	$max_{CS} = 1$	0.7331 ± 0.0178	0.0561 ± 0.0199
	$max_{CS} = 2$	0.7344 ± 0.0143	0.0484 ± 0.0142
	$max_{CS} = 3$	0.7314 ± 0.0141	0.0398 ± 0.0101
	$max_{CS} = 4$	0.7316 ± 0.0141	0.0380 ± 0.0098
	$max_{CS} = 5$	0.7315 ± 0.0141	0.0377 ± 0.0099
Tree-Tree		0.6968 ± 0.0163	0.0364 ± 0.0101
Polytree-Polytree		0.6999 ± 0.0147	0.0299 ± 0.0062
Pure Filter		0.7074 ± 0.0063	0.0240 ± 0.0066
Pure Wrapper		0.7095 ± 0.0040	0.0291 ± 0.0008
CB-MBC		0.7261 ± 0.0113	0.0382 ± 0.0105
ML-kNN		$\mathbf{0.7373 \pm 0.0180}$	0.0729 ± 0.0259
BP-MLL		0.7189 ± 0.0095	0.0428 ± 0.0165
MNL		0.7365 ± 0.0159	0.0595 ± 0.0203

prediction problem and they learned just two Bayesian networks separately for two NNRTI drugs, namely EFV and NVP, to investigate resistance pathways [63].

The unique NRTI drug that has no dependency relationships with the other drugs is FTC. This can be attributed either to the fact that there is not enough data on this drug since it appears in only 205 samples, or to their being no influence between FTC and all other drugs in this data set.

In the class subgraph, we also find that no dependency relationships are detected within the NNRTI group, that is, there is no dependency relationships between the three NNRTI drugs: NVP, EFV and DLV. In fact, NVP only has associations with two NRTI drugs D4T and DDI, EFV has a unique association with the NRTI drug D4T, whereas DLV has no associations with any other drugs. The same finding as for FTC may also apply to DLV. However, the second hypothesis, i.e., absence of influence between DLV and all other drugs, is more likely since there is a greater frequency of appearance of DLV in the data set (1990 samples) than FTC.

Secondly, the bridge subgraph (blue arcs) reveals dependency relationships between RTI drugs and resistance mutations. For the first group of NRTIs, we find that each NRTI drug,

except TDF, was directly related to at least one of its established resistance mutations, which confirms the current knowledge on the role of these mutations in relation to corresponding NRTI drugs [119]. For instance, ABC was associated with mutations L74V and Y115F; D4T was associated with mutations M41L and D67N; FTC was associated with mutations K70R and T215F/Y; and DDI was associated with mutation L74V. Additionally, AZT and 3TC were directly connected to mutations K70R and 184V, respectively. This result was also confirmed in recent work by Theys et al. [211], where the K70R and 184V mutations were identified as two major resistance mutations to the combination of AZT and 3TC.

In general, two basic types of NRTI-resistance mechanism are known for HIV-1. The first resistance mechanism is *exclusion* and involves enhanced discrimination at the time the NRTI is incorporated. One example is the M184-V/I mutation that reduces the incorporation of 3TC and FTC by steric hindrance. The second mechanism is *excision* and involves the selective removal of NRTI from the end of the viral DNA after it has been incorporated by RT. This is, for instance, the excision mechanism involved in AZT resistance and is achieved through the accumulation of a specific set of mutations including M41L, D67N, K70R, L210W, T215F, and K219E/Q. Interestingly, the same set of mutations is also selected in viruses from patients under D4T therapy and are commonly designated as TAMs (i.e., thymidine analogue resistance mutations) [197]. Cross-resistance due to the presence of TAMs affects all NRTI drugs to some extent [119, 229].

Dependency relationships identified for NNRTI group are also consistent with current knowledge [63, 119] as EFV and DLV were directly associated with the established resistance mutation K103N, and NVP was associated with G190A.

The bridge subgraph indicated that all NRTI drugs were directly associated with some NNRTI resistance mutations, such as K103N (associated with AZT, FTC, DDI and TDF), Y181C (associated with AZT and FTC), P225H (associated with D4T and 3TC), V108I, V179D and G190S (associated respectively with ABC, AZT and FTC). Similarly, all NNRTI drugs were directly associated with some NRTI resistance mutations, such as M41L, D67N, T69D and F77L (associated with EFV), K70R and M184V (associated with NVP), and L74V (associated with DLV). This certainly reveals the extent of inter-group interactions between NRTIs and NNRTIs.

Finally, the feature subgraph (green arcs) allowed us to identify interactions between NRTI and NNRTI resistance mutations. The mutations with the greatest number of dependence relationships were mutations D67N (4 connections: M41L, T69D, A98G, K219Q), T69D (4 connections: M41L, D67N, A98G, T215F), K70R (4 connections: L74V, M184V, T215F, K219Q) and K103N (4 connections: V179D, M184V, Y188H, G190G). Notice that there are

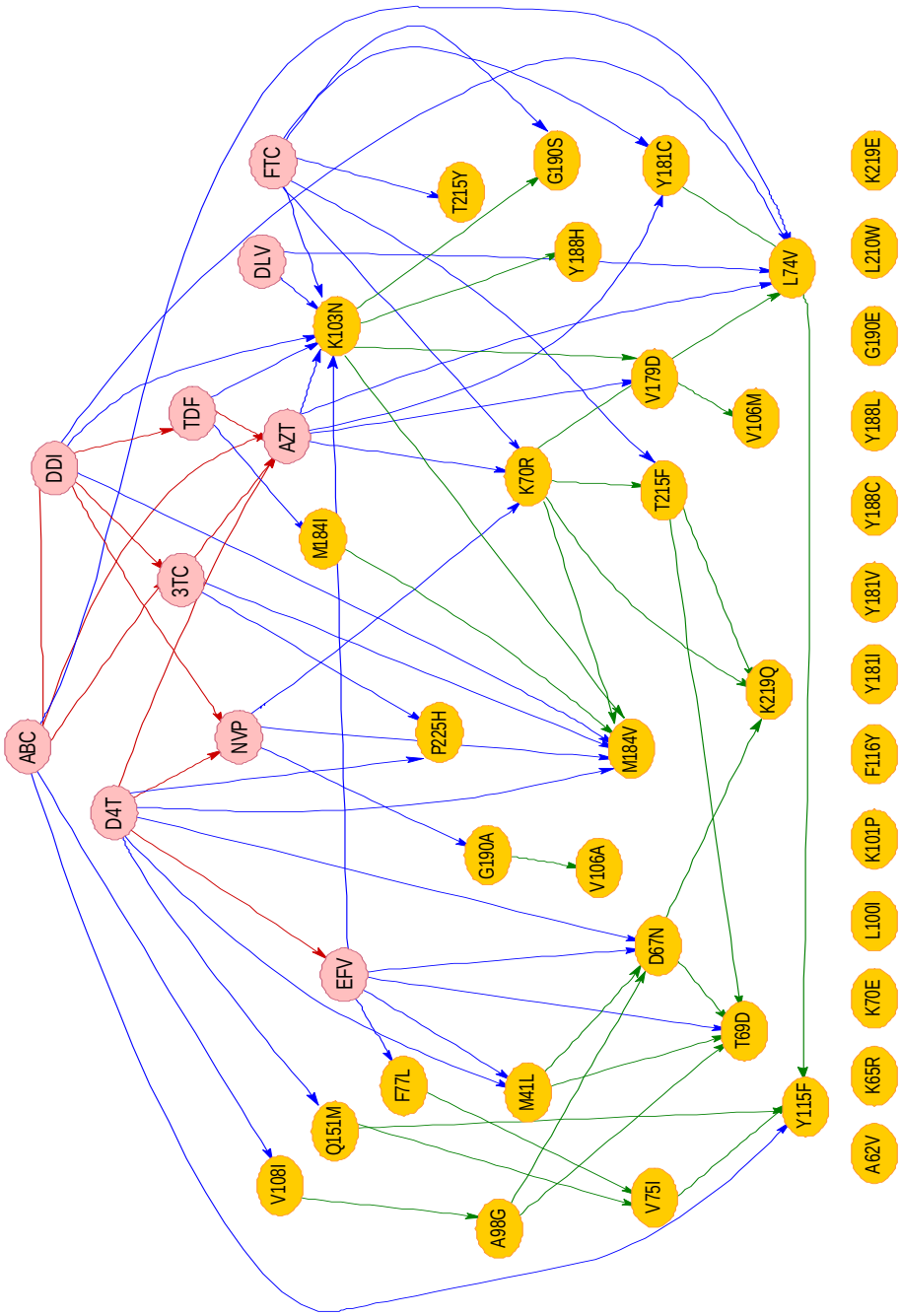


Figure 6.2: MBC graphical structure learnt by `MB-MBC` for the RTI data set.

13 resistance mutations (at the bottom) that do not interact with drugs or features. A possible explanation is the lack of instances of such mutations and/or their low interactions with the other variables in the data set.

For structural comparison we have considered only two additional MBC graphical structures, because the discussion of all dependence relationships recovered by all remaining MBC learning algorithms is complicated, requires at least one more page for each of the six MBC networks, and may also become confusing and long to read. The two selected structures are those induced by the MBC learning algorithms presenting the second best global accuracies (`MB-MBC` was the best for both RTI and PI data sets). Our selection is based on the global accuracy because it is the metric that reflects more appropriately the main objective of the multi-dimensional classification, namely, predicting simultaneously all the class values. According to Table 6.1, it turns out that the second best MBC learning algorithms are `IndepMBs` with $max_{CS} = 1$ and `CB-MBC`.

We have depicted in Figures 6.3 to 6.12 the ten Markov blanket-based Bayesian network classifiers, learnt independently by `IndepMBs` for each RTI class variable. Being based on the same HITON algorithm, `IndepMBs` discovers similar dependence relationships between RTI drugs and resistance mutations, as `MB-MBC` does, augmented with additional dependence relationships.

For instance, in the graphical structure learnt by `MB-MBC` (Figure 6.2), ABC is associated with mutations L74V, V108I, and Y115F; however, in Figure 6.3, ABC is associated with mutations K70R, L74V, K103N, V108I, Y115F, F116Y, T215Y. This structural difference is expected and is related to the independent learning strategy of `IndepMBs`. In fact, the additional mutations detected by `IndepMBs` are relevant to ABC class variable, but they become weakly relevant or irrelevant when the remaining RTI class variables are considered during the `MB-MBC` learning process. In the graphical structure learnt by `MB-MBC`, K103N, for example, becomes weakly relevant to ABC as it does not pertain anymore to the Markov blanket of ABC, but there is an undirected path from K103N to ABC; however, F116Y becomes irrelevant to ABC because no undirected path exists between F116Y and ABC.

The same is observed for the bridge subgraphs learnt by `IndepMBs` for the remaining RTI class variables, as well as the corresponding feature subgraphs. For instance, in both MBC graphical structures learnt by `IndepMBs` (Figure 6.3) and `MB-MBC` (Figure 6.2), the following associations between the pair of mutations V75I/Y115F, Q151M/ Y155F, M184V/ K103N, and M184V/K70R are identified. However, several other associations are added to the `IndepMBs` graphical structure that do not exist in the `MB-MBC` graphical structure. As previously explained, this is mainly due to the non-consideration of the remaining RTI class

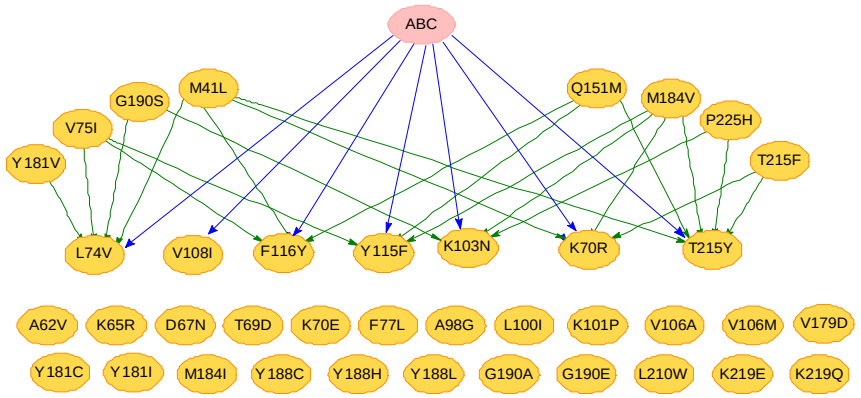


Figure 6.3: MBC graphical structure learnt by `IndepMBs` for the RTI class ABC.

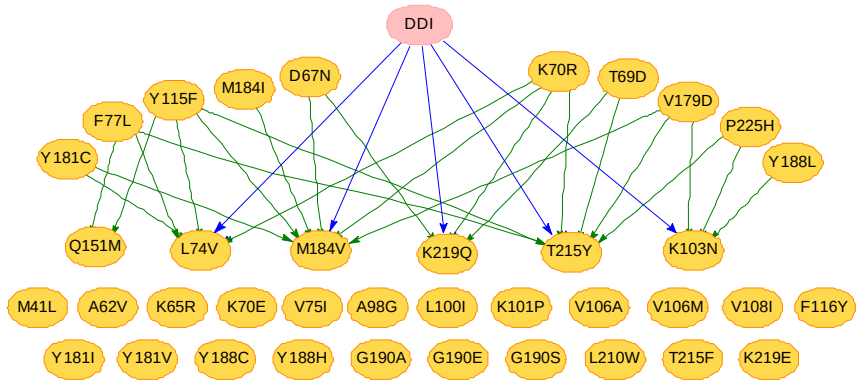


Figure 6.4: MBC graphical structure learnt by `IndepMBs` for the RTI class DDI.

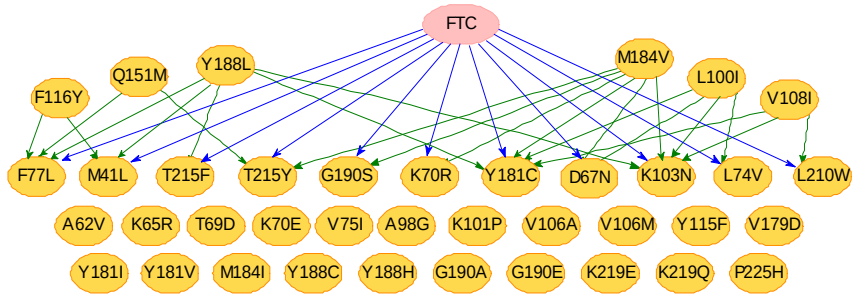


Figure 6.5: MBC graphical structure learnt by `IndepMBs` for the RTI class FTC.

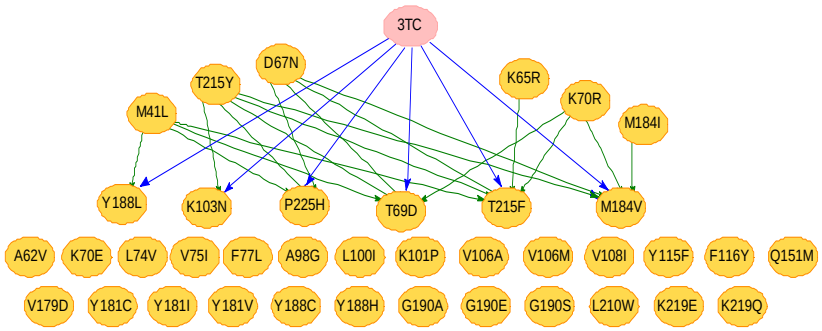


Figure 6.6: MBC graphical structure learnt by `IndepMBs` for the RTI class 3TC.

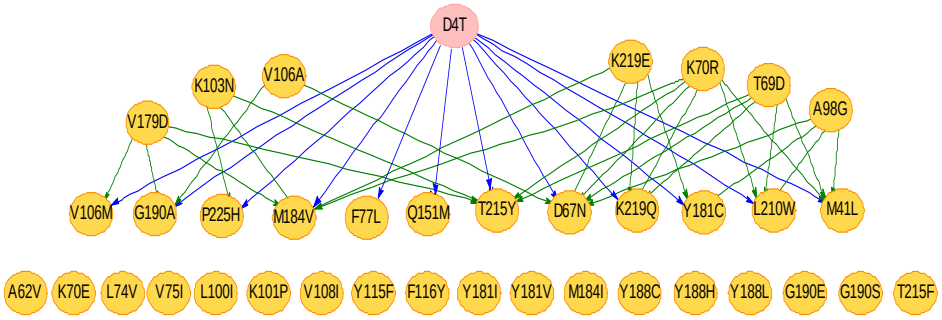


Figure 6.7: MBC graphical structure learnt by `IndepMBs` for the RTI class D4T.

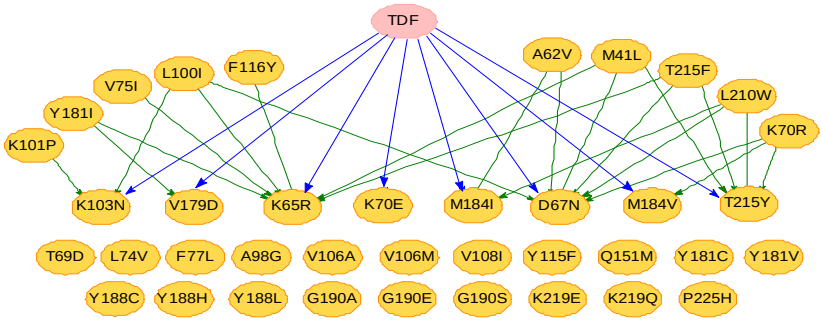


Figure 6.8: MBC graphical structure learnt by `IndepMBs` for the RTI class TDF.

variables when building the Markov blanket of each RTI class variable.

Note also that, as it can be observed in Figures 6.3 to 6.12, the main drawback of `IndepMBs`

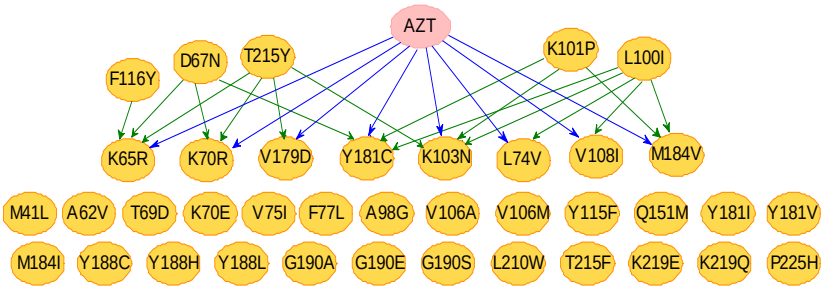


Figure 6.9: MBC graphical structure learnt by `IndepMBs` for the RTI class AZT.

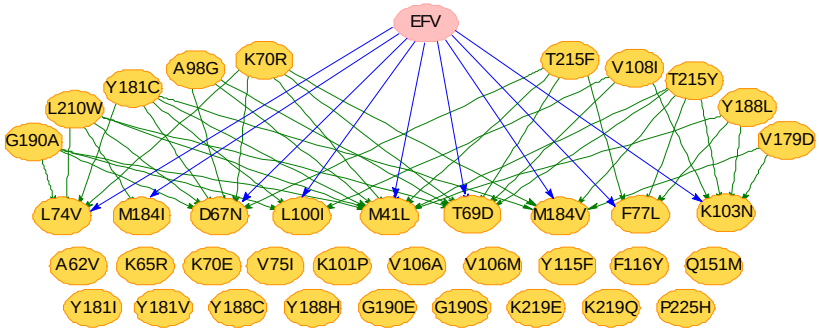


Figure 6.10: MBC graphical structure learnt by `IndepMBs` for the RTI class EFV.

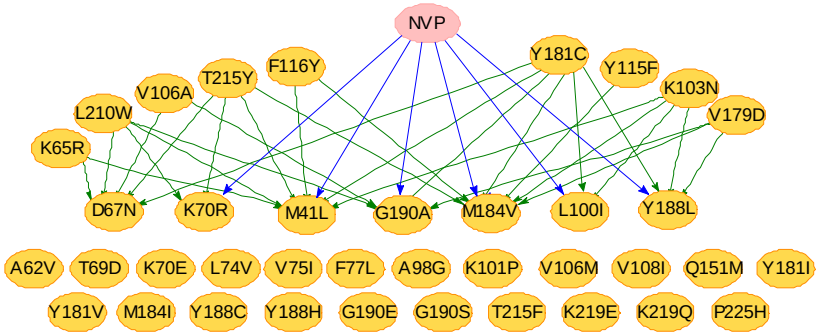


Figure 6.11: MBC graphical structure learnt by `IndepMBs` for the RTI class NVP.

approach is its inability to detect the dependence relationships between the different RTI drugs and their simultaneous interactions with resistance mutations.

Moreover, we have depicted in Figure 6.13, the MBC graphical structure learnt by `CB-MBC`

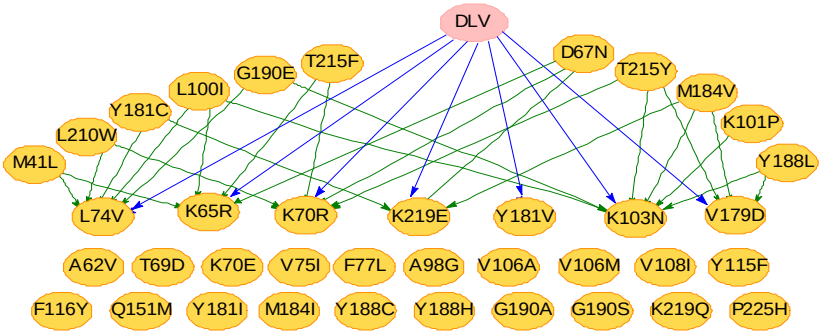


Figure 6.12: MBC graphical structure learnt by `IndepMBs` for the RTI class DLV.

for RTI data set (which allows dependencies between class variables). For the class subgraph, `CB-MBC` only detects a direct dependence relationship among the RTI class variables ABC and DDI (red arc), which has been also detected in the MBC graphical structure learnt by `MB-MBC`. All the remaining RTI class variables are kept independent. For the bridge subgraph (blue arcs), and similar to `MB-MBC`, the following dependence relationships are detected: ABC is associated with L74V and Y115F; DDI is associated with L74V, K103N, and M184V; 3TC is associated with P225H; AZT is associated with K70R and Y188C; and D4T is associated with M41L and Q151M. Additional dependence relationships are also discovered in the bridge subgraph learnt by `CB-MBC` between the RTI class variables ABC, DDI, AZT, D4T and other resistance mutations; however, contrary to `MB-MBC` graphical structure, no mutations are associated with the RTI class variables FTC, EFV, NVP, and DLV. These variables are kept independent and do not interact with the rest of variables. For the feature subgraph (green arcs), the only dependence relationships are detected between the pair of mutations K70R/G190E, Y181V/K219E, M184V/P225H and M184V/F77L, and none of them have been detected in the `MB-MBC` graphical structure.

Overall, the graphical structure learnt by `CB-MBC` is less dense than the one learnt by `MB-MBC` and this is mainly due to the `CB-MBC` learning strategy which is based on a wrapper approach. In fact, in `CB-MBC`, arcs are inserted only if there is an improvement in the global accuracy. So that, after the first phase of `CB-MBC` consisting of building independently a selective naive Bayes for each class variable, it becomes more difficult to improve the global accuracy. In other words, this means that after the first phase of `CB-MBC`, it becomes more difficult to merge the maximal connected components (i.e., adding arcs between the class variables) and update the bridge and feature subgraphs. As in our case, this generally may lead to more independent class variables and less dense bridge and feature subgraphs.

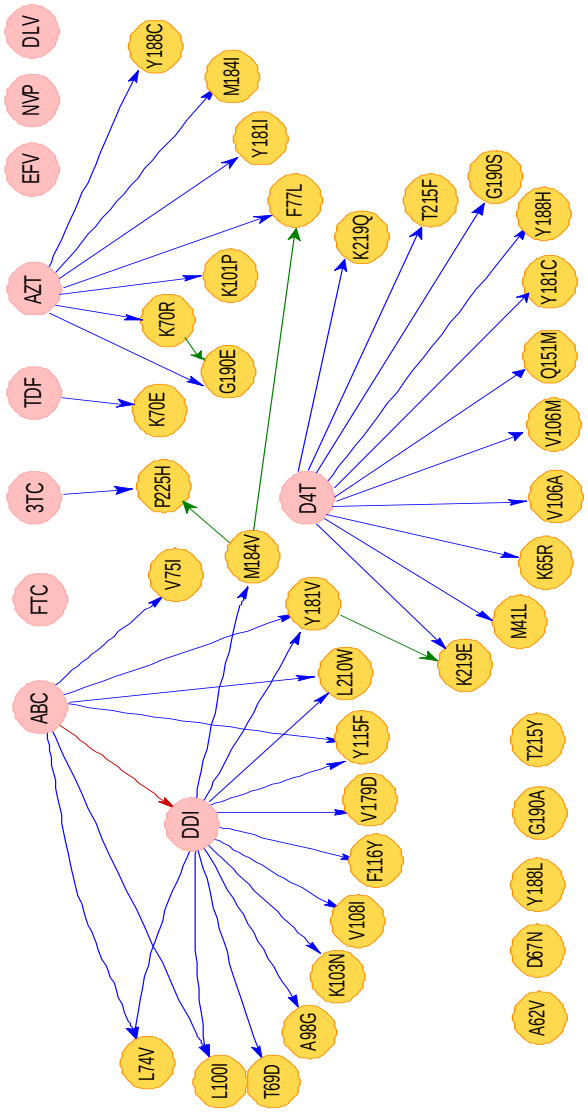


Figure 6.13: MBC graphical structure learnt by `CB-MBC` for the RTI data set.

6.3.5 PIs analysis results

Table 6.2 presents the prediction results for the PI data set with mean values and standard deviations for each metric and each learning method. The best result of each metric is written in bold. `MNL` presents the best mean accuracy (86%), whereas `MB-MBC` induces the best global accuracy (31%). As with the RTI data set, we ran a multiple comparison of all

algorithm performances using the Friedman test followed by the Tukey-Kramer post-hoc test with a significance level of $\alpha = 0.05$. For the mean accuracy, it turns out that (1) `MNL` is significantly better than `MB-MBC` with $max_{CS} = 3$, `MB-MBC` with $max_{CS} = 4$, `MB-MBC` with $max_{CS} = 5$, `Tree-Tree`, `Polytree-Polytree` and `BP-MLL`; (2) `ML-`k`NN` and `IndepMBs` with $max_{CS} = 5$ are significantly better than `MB-MBC` with $max_{CS} = 4$, `MB-MBC` $max_{CS} = 5$, `Tree-Tree` and `BP-MLL`; and (3) `IndepMBs` with $max_{CS} = 3$ and `IndepMBs` with $max_{CS} = 4$ are only significantly better than `BP-MLL`. For the global accuracy, it turns out that (1) `MB-MBC` with $max_{CS} = 1$, `MB-MBC` $max_{CS} = 2$, and `MB-MBC` with $max_{CS} = 3$ are significantly better than `Polytree-Polytree`, `Pure Filter`, `Pure Wrapper`, and `BP-MLL`; and (2) `MB-MBC` with $max_{CS} = 4$ and `MB-MBC` with $max_{CS} = 5$ are only significantly better than `Pure Filter` and `Pure Wrapper`. For all remaining algorithms, the differences in classification performance are not statistically significant.

Note finally that the performance results are better over the PI than the RTI data set; this can be explained by the fact that the number of classes (8 in PI and 10 in RTI) and the number of possible class combinations (256 in PI and 1024 in RTI) are lower in PI than in the RTI data set.

Table 6.2: Estimated accuracies (mean ± std. dev.) over PI data set.

Method		Mean accuracy	Global accuracy
`MB-MBC`	$max_{CS} = 1$	0.8476 ± 0.0072	$\mathbf{0.3187 \pm 0.0304}$
	$max_{CS} = 2$	0.8463 ± 0.0086	0.3123 ± 0.0412
	$max_{CS} = 3$	0.8449 ± 0.0070	0.3035 ± 0.0298
	$max_{CS} = 4$	0.8408 ± 0.0072	0.2886 ± 0.0329
	$max_{CS} = 5$	0.8407 ± 0.0069	0.2904 ± 0.0333
`IndepMBs`	$max_{CS} = 1$	0.8518 ± 0.0051	0.2726 ± 0.0256
	$max_{CS} = 2$	0.8563 ± 0.0054	0.2231 ± 0.0134
	$max_{CS} = 3$	0.8594 ± 0.0036	0.2010 ± 0.0047
	$max_{CS} = 4$	0.8593 ± 0.0038	0.1896 ± 0.0099
	$max_{CS} = 5$	0.8610 ± 0.0021	0.1912 ± 0.0060
`Tree-Tree`		0.8399 ± 0.0019	0.1931 ± 0.0256
`Polytree-Polytree`		0.8432 ± 0.0050	0.1509 ± 0.0220
`Pure Filter`		0.8527 ± 0.0030	0.0966 ± 0.0157
`Pure Wrapper`		0.8530 ± 0.0006	0.0998 ± 0.0038
`CB-MBC`		0.8552 ± 0.0037	0.2224 ± 0.0232
`ML-`k`NN`		0.8612 ± 0.0069	0.2279 ± 0.0364
`BP-MLL`		0.8080 ± 0.0283	0.1473 ± 0.0328
`MNL`		$\mathbf{0.8682 \pm 0.0051}$	0.2551 ± 0.0174

In addition, we examined the graphical structure of the most accurate learned MBC, shown in Figure 6.14, in order to evaluate the usefulness of the proposed learning algorithm for identifying the different interactions between drugs and mutations in the HIV-1 protease data set.

Firstly, the learned network, specifically the class subgraph (red arcs), shows dependency relationships between the following drugs IDV, ATV, NFV, LPV and SQV, which may reveal the extent of cross-resistance between each related pair of these drugs. Notice that, for IDV, which has associations with LPV, ATV and NFV, Rhee et al. [186] recently proved in their PI cross-resistance study that IDV and LPV are among the most strongly correlated PIs. In fact, these two drugs had a correlation coefficient value equal to 0.57 [186]. Similarly, based on their study, IDV and ATV, ATV and NFV as well as NFV and IDV had high correlation coefficients. Nevertheless, correlation coefficients between LPV and both drugs NFV and SQV were lower, equal to 0.14 and 0.05, respectively. This goes to confirm then that the dependency relationships among the above PI drugs identified in the network are consistent with Rhee et al.'s study [186].

However, our results were less conclusive for other drugs (FPV, DRV and TPV) since we did not find any associations between these three drugs and between these and the other drugs. A possible explanation is the lack of available data, as there were fewer than 30 samples for each of these drugs. This may be due to the fact that DRV and TPV, considered as new-generation PI drugs, have different profiles to the other PI drugs, and hence they are mainly used in rescue therapies for PI-experienced patients displaying failure on previous PI drugs [139, 150]. On this ground, we would require a larger and more diverse data set for future analyses in order to investigate possible interactions between these drugs and the other network variables.

Concerning relationships between PI drugs and mutations, visualized by the bridge subgraph (blue arcs), let us first discuss the two possible types of mutations, major and minor, and then how their associations with PI drugs have been previously interpreted in the literature in the Bayesian network context. As Defroche et al. [62, 64] found, a major mutation actually plays a key role in drug resistance, and thus should have an unconditional dependency on the drug. This is indicated in the network graphical structure by the presence of an arc between the major mutation and the drug.

In contrast, a minor mutation further increases drug resistance mostly only in the presence of major mutations. Thus, it is expected to be conditionally independent of the drug but dependent on other major resistance mutations. This is indicated in the network by the presence of an arc between major or minor mutations instead of an arc between the minor

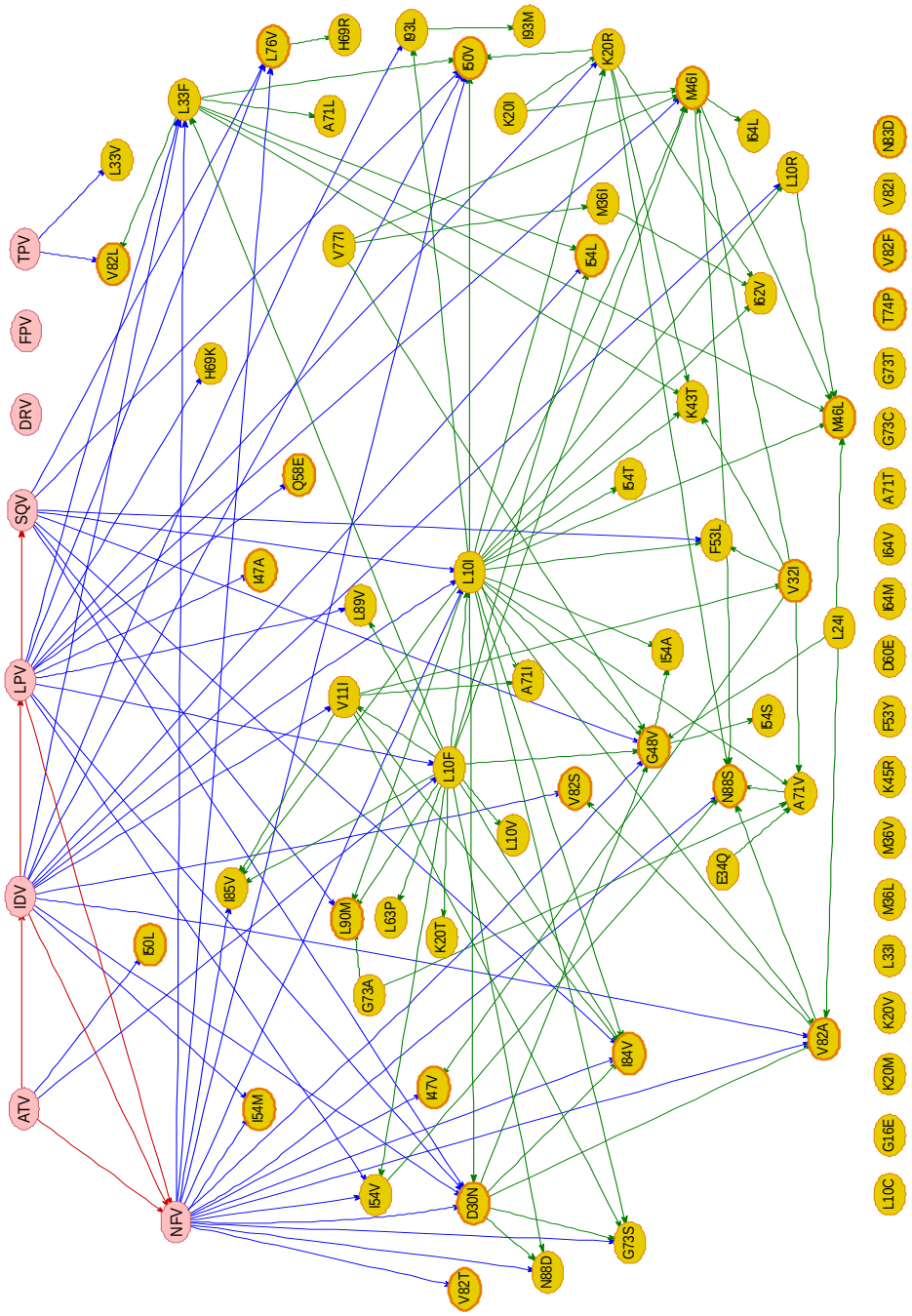


Figure 6.14: MBC graphical structure learnt by `MB-MBC` for the PI data set.

mutation and the drug node. Even so, as claimed by Defroche et al. [64], a minor mutation may still be connected to the drug.

Notice that the conditional independencies revealed in our bridge subgraph in Figure 6.14 are largely consistent with the above definitions, since most of the major mutations are directly connected to one or more drug nodes. For instance, on the left, D30N (which is defined in [119] as a major mutation of NFV) was not only associated with NFV but also with IDV, LPV and SQV, again attesting to the extent of cross-resistance between these drugs. Similarly, on the right, L76V (which is defined in [119] as a major mutation of LPV) was directly associated with LPV, SQV and NFV. At the center bottom of the network, G48V (major mutation of SQV [119]) was directly associated with SQV and NFV. L90M (another major mutation of SQV [119]) was also directly associated with SQV. I47A, I50L, V82A, V82L, defined in [119] as major mutations of LPV, ATV, IDV and TPV, respectively, were directly associated with the right drugs in the MBC graphical structure.

A good number of minor mutations were also directly connected to drug nodes. L10I and L33F seem to be the main minor mutations: they have the highest number of connections (3) with PI drugs, followed by the minor mutations L10F and I54V. L10I was associated with IDV, NFV and SQV; L33F with LPV, IDV and NFV; L10F with ATV and IDV, and I54V with LPV and NFV. Additionally, consistently with the latest knowledge reported in [119], more minor mutations, namely V82A/T, I84V, N88D/S, were associated directly with NFV. Also in agreement with [119], the minor mutation K20R was associated with LPV, and the minor mutation I84V was associated with SQV.

From the feature subgraph (green arcs) of the learned MBC, we were able to identify interactions among different protease mutations. The mutations with the greatest number of dependency relationships were L10I (21 connections: L10F, L10R, K20R, D30N, M46L, M46I, K43T, G48V, I50V, F53L , I54A, I54T, I62V, A71I, A71V, G73S, V82A, I84V, I85V, L90M, I93L), L10F (15 connections: L10I, L10V, V11I, K20T, L33F, M46I, G48V, I54L, I54V, L63P, I84V, I85V, N88D, L89V, L90M), M46I (8 connections: L10F, L10I, K20I, V32I, M46L, I64L, V77I, N88S), and 7 connections for L33F(L10F, K43T, M46L, I50V, I54L, A71L, V82L) and G48V (L10F, L10I, L24I, D30N, I54A, I54S, V77I).

Note that, only three of the 19 mutations that present no interactions with other drugs or features (at the bottom), are major ones, namely T74P, V82F and N83D. As with RTIs, a possible explanation is the low number of occurrences of these mutations and/or their low interactions with the other variables in the data set.

Similar to RTI data sets, we have considered only two additional MBC graphical structures for structural comparison. Our selection is also based on the global accuracy because it

is the metric that reflects more appropriately the main objective of the multi-dimensional classification, namely, predicting simultaneously all the class values. According to Table 6.2, it turns out that the second best MBC learning algorithms are `IndepMBs` with $max_{CS} = 1$ and `CB-MBC`.

We have depicted in Figures 6.15 to 6.21 the graphical structures learnt independently by `IndepMBs` for respectively, the PI class variables ATV, DRV, IDV, LPV, NFV, SQV, and TPV. No graphical structure is available for the FPV class variable because no associations are detected between FPV and the remaining variables. Note also that, in all the figures, the features that do not present any interactions with the corresponding PI class variable or the rest of features are omitted (because the PI data set contains a larger number of features, equal to 74, comparing to the previous RTI data set containing only 38 feature variables).

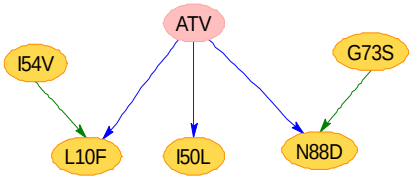


Figure 6.15: MBC graphical structure learnt by `IndepMBs` for the PI class ATV.

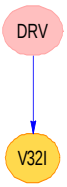


Figure 6.16: MBC graphical structure learnt by `IndepMBs` for the PI class DRV.

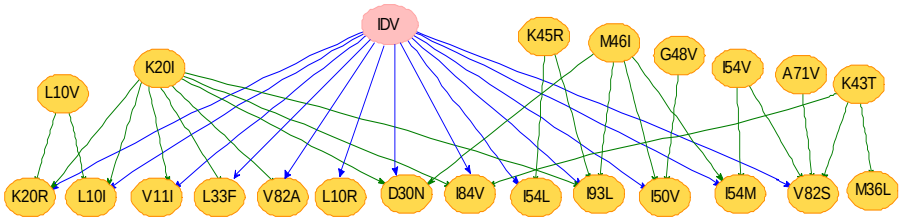


Figure 6.17: MBC graphical structure learnt by `IndepMBs` for the PI class IDV.

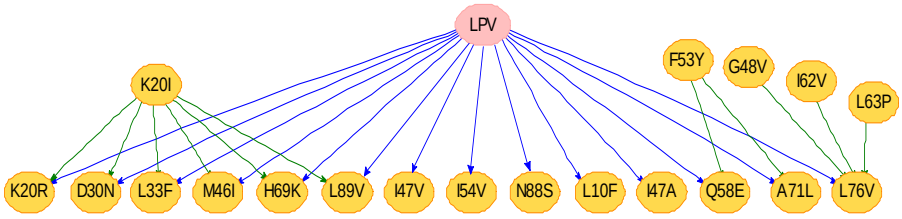


Figure 6.18: MBC graphical structure learnt by `IndepMBs` for the PI class LPV.

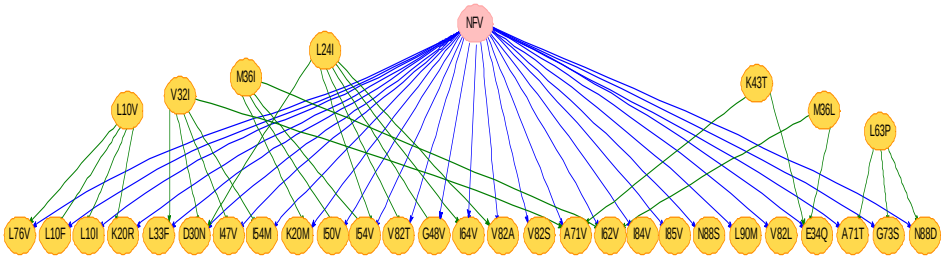


Figure 6.19: MBC graphical structure learnt by `IndepMBs` for the PI class NFV.

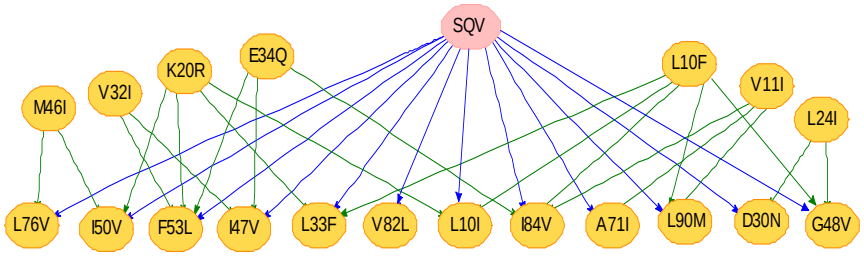


Figure 6.20: MBC graphical structure learnt by `IndepMBs` for the PI class SQV.

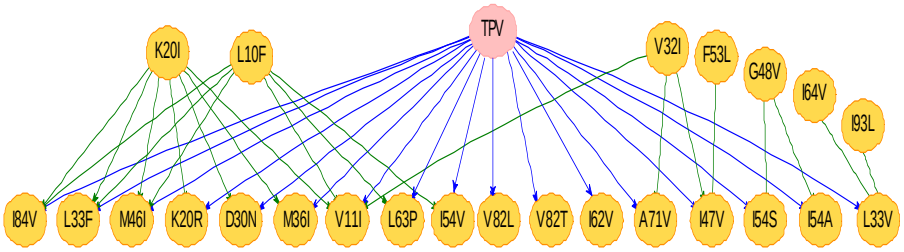


Figure 6.21: MBC graphical structure learnt by `IndepMBs` for the PI class TPV.

`IndepMBs` graphical structures in Figures 6.15 to 6.21 are compared with the MBC graphical structure learnt by `MB-MBC` (Figure 6.14), and similar conclusions to the ones discussed with RTI data set are drawn. Basically, `IndepMBs` discovers similar dependence relationships among each PI drug and the resistance mutations, as `MB-MBC` does, augmented with additional dependence relationships. For instance, in Figure 6.15, ATV is associated with L10F, I50L (which are also detected in the `MB-MBC` graphical structure), and with N88D (which becomes weakly relevant to ATV in the `MB-MBC` graphical structure).

Similar conclusions are also drawn for the bridge subgraphs, and the inability of `IndepMBs` to detect the dependence relationships between the different PI drugs and their simultaneous interactions with the major and minor resistance mutations.

Finally, we have depicted in Figure 6.22 the MBC graphical structure learnt by `CB-MBC` for PI data set. For the class subgraph, `CB-MBC` only detects a dependence relationship among the PI class variables IDV and NFV (red arc), which has been also detected in the MBC graphical structure learnt by `MB-MBC` (Figure 6.14). All remaining RTI class variables are kept independent.

For the bridge subgraph (blue arcs), and similar to `MB-MBC`, the following dependence relationships are detected: ATV is associated with I50L; IDV is associated with L10I, L10R, V11I, I54L and V82F; LPV is associated with I47A and L76V; NFV is associated with D30N, I84V and N88S; and SQV is associated with G48V, I50V and F53L. Additional dependence relationships are also discovered in the bridge subgraph, learnt by `CB-MBC`, between the PI class variables IDV, LPV, NFV, SQV and other resistance mutations.

For the feature subgraph (green arcs), the only dependence relationships are detected between the pair of mutations L10I/V82F, V11I/K20V, I50L/K45R, I54A/K45R, K20R/K45R, V82A/K45R and T74P/I50V, and none of them have been detected in the `MB-MBC` graphical structure. Note also that, in `MB-MBC` graphical structure, only 3 out of the 19 mutations that do not present any interactions with other drugs or features are major ones; however, in `CB-MBC` graphical structure, there are 34 mutations that do not present any interactions with the rest of variables, where 9 of them are major, namely, M46L/M, I54V, Q58E, V82I/L/T, N83D and L90M.

Similar to the RTI data set, we can conclude that `CB-MBC` produces a less dense MBC graphical structure than `MB-MBC`, since it is based on a wrapper approach.

In summary, for both RTI and PI data sets, the identified dependence relationships were proved to be consistent with the current knowledge, and were also verified by the medical doctor Carlos Toro, who is a specialist in the HIV problem. However, for the variables (inhibitors

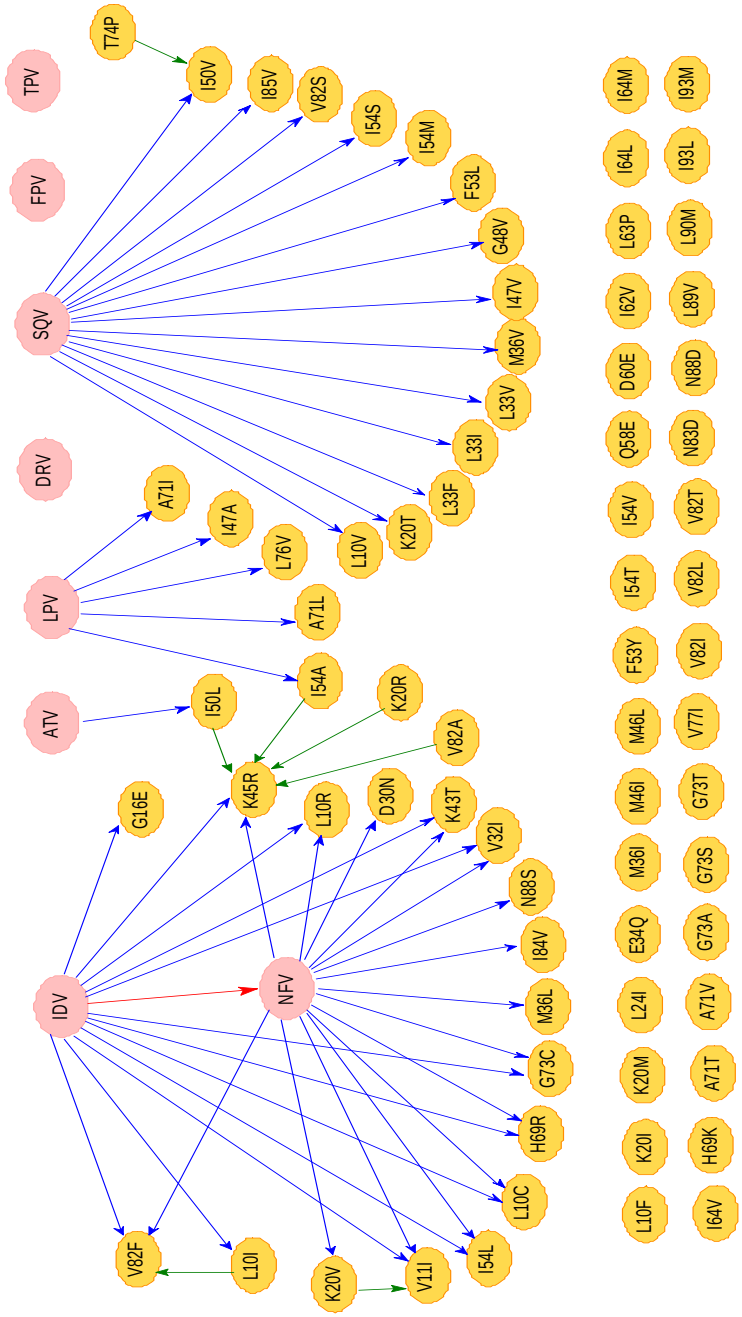


Figure 6.22: MBC graphical structure learnt by `CB-MBC` for PI data set.

or resistance mutations) that do not present any interactions with the rest of variables, larger and more diverse RTI and PI data sets need to be considered, and additional analyses need

to be performed to thoroughly prove the current findings.

Finally, the computation times consumed by each algorithm on RTI and PI data sets are plotted in Figures 6.23 and 6.24, respectively. As observed, ML-kNN is always the fastest, followed by MNL and the filter approaches, namely, Pure Filter and Polytree-Polytree. For RTI data set, IndepMBs is also quite efficient and requires less computation time than MB-MBC. However, for PI data set, including a larger number of variables (82 variables instead of 48 variables in RTI data set), IndepMBs with $max_{CS} = 1$ takes a similar computation time than MB-MBC with $max_{CS} = 1$, but more computation times than MB-MBC for $max_{CS} = \{2, 3, 4, 5\}$. Note here that, for both MB-MBC and IndepMBs, the consumed computation times increase as max_{CS} grows, because the number of executed statistical independence tests increases as max_{CS} grows. This is mainly noticed over PI data set that contains a larger number of variables. Moreover, as pointed out by Aliferis et al. [5], as max_{CS} grows, the overall power decreases. This is also verified in our case, especially for the global accuracy values in Tables 6.1 and 6.2, that drop for both MB-MBC and IndepMBs as max_{CS} grows. Therefore, using smaller max_{CS} avoids excessive computations while producing better predictive performance.

In addition, as shown in Figures 6.23 and 6.24, BP-MLL consumes more computation times than ML-kNN, MNL and the filter approaches mainly due to its complex error function which needs to be optimized through an iterative learning process [239]. CB-MBC, Tree-Tree and Pure Wrapper take always the longest computation times comparing to the rest of the methods since they are all based on wrapper approaches that involve time-consuming MPE computations.

Tree-Tree is the slowest over RTI data set, whereas CB-MBC is the slowest over PI data set; and this can be explained by the different learning strategies behind both algorithms. In fact, the learning process of the bridge subgraph for Tree-Tree algorithm requires, in each iteration, an evaluation of the global accuracy of each possible MBC candidate using MPE computations. These computations mainly depend on the number of class variables and their value combinations; so that, as the number of class variables grows, Tree-Tree running time increases. In our case, PI data set contains 8 binary class variables, i.e., 256 class value combinations, however, RTI data set contains 10 binary class variables, i.e., 1024 class value combinations, which increases considerably Tree-Tree computation time over the RTI data set. The same observation applies as well for Pure Wrapper that iteratively evaluates and selects the MBC candidates using MPE computations.

CB-MBC is based on a different learning wrapper strategy. It first learns an initial bridge subgraph by building a selective naive Bayes for each class variable, defines the feature sub-

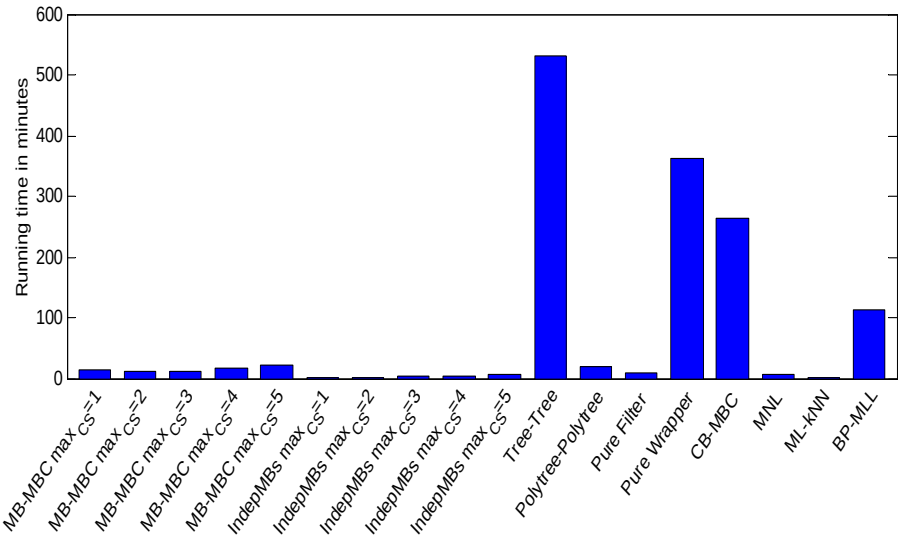


Figure 6.23: Computation learning times over RTI data set.

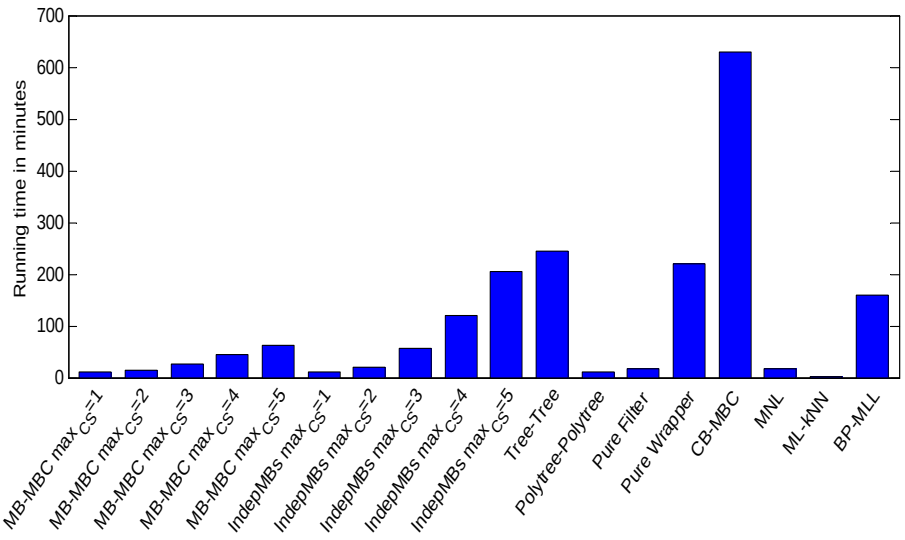


Figure 6.24: Computation learning times over PI data set.

graph by randomly selecting arcs between features and adding them if they improve the accuracy, then iteratively updates MBC subgraphs as long as the number of maximal connected components is greater than one and there is an accuracy improvement. The first step in `CB-MBC` depends on the number of class and feature variables, and may require the longest

computation time during CB-MBC learning process. In fact, for each class variable, it iterates over all feature variables to select, in each iteration, the feature that improves the accuracy the most. This, indeed, explains the highest computation time consumed by CB-MBC over PI data set (including 82 variables) compared to the one consumed over RTI data set (including only 48 variables).

Note finally that, being defined as a filter constraint-based approach, MB-MBC requires less computational time than all existing wrapper algorithms, since its learning process is based on statistical independence tests instead of accuracy metrics. Generally speaking, MNL, ML-kNN and the filter approaches require shorter computation times, whereas the wrapper approaches always take longer.

6.4 Case study in predicting EQ-5D from PDQ-39

Parkinson's Disease (PD) is a neurodegenerative disorder characterized by motor manifestations (bradykinesia, rest tremor and balance impairment) and non-motor symptoms (depression, psychosis and sleep disturbance) [37, 148]. In PD, the symptoms, the complications, and the subsequent disability are progressive over time and cause an increasing deterioration of the patients' quality of life [121, 151, 153, 203].

Health-related quality of life (HRQoL) is a patient-reported outcome reflecting the impact of the disease on the physical, mental, functional, and social aspects of life which are important for the individual. There is no a universal definition for HRQoL, but in pragmatic terms it may be considered as the perception and evaluation, by patients themselves, of the impact caused on their lives by the disease and its consequences [151].

HRQoL measures can be categorized into generic and specific. *Generic measures* are usable in general populations and in any disorder as they compile information about the most relevant health domains. The European Quality of Life-5 Dimensions (EQ-5D) is considered a valid generic instrument and is recommended for evaluation of HRQoL in PD [42, 102, 152, 199]. EQ-5D contains five items: `mobility`, `self-care`, `usual activities`, `pain/discomfort`, and `anxiety/depression`, each has three options of response: no problems, some problems and severe problems. Hence, the number of all possible EQ-5D item value combinations is 243. Each possible combination corresponds to a *health state*, which can be then quantified using a *utility score*, a.k.a. *utility index*. Based on the UK scoring system [103], this corresponding *utility score* may range from -0.594 (i.e., worse health state where all EQ-5D items report severe problems) to 1 (i.e., best health state where all EQ-5D items report no problems) [32, 147].

On the contrary, *specific measures* are usable only in the population for which they were designed and cover the most important areas of interest in that setting. The 39-item Parkinson's Disease Questionnaires are *specific* HRQoL instruments widely used in PD and they are also recommended for use in this disorder [152]. It contains 39 questions, represented in Table 6.3, each scoring on a five-point scale: never, occasionally, sometimes, often and always [116, 117, 172, 173].

In clinical studies, PDQ-39 could be used in detriment of EQ-5D due to the excessive burden for the respondents to assess two questionnaires simultaneously, or the lack of resources and time. This also may be due to the lack of the clinical interest in generic measures, and the relative difficulty for the calculation of EQ-5D utility index and the interpretation of its outcomes. Nevertheless, PDQ-39 can not be directly applied in cost-effectiveness analyses which require generic measures and quantitative utility scores, such as EQ-5D. To deal with this problem, a commonly used solution is the prediction of EQ-5D from disease-specific measures. For instance, several studies have been proposed to map the EQ-5D utility score from the Health Surveys SF-12 [78, 146, 147, 208] and SF-36 [191]. Moreover, in a more related work, Cheung et al. [43] developed several functions for generating the EQ-5D utility index from PDQ-8, the short version of PDQ-39.

Notice that, most of these studies were mainly based on ordinary least squares (`OLS`) or censored least absolute deviation (`CLAD`) regression methods. Nevertheless, Le and Doctor [147] recently discussed certain limitations of these regression methods (such as predictive values that are outside the range of the EQ-5D utility scores and ceiling/floor effects, i.e., when predictive values can be better/worse than the best/lowest score in the range of EQ-5D scores) and proposed a probabilistic mapping of Health Surveys SF-12 into EQ-5D using Bayesian networks. Specifically, Le and Doctor proved that Bayesian networks consistently outperformed the commonly used regression methods and pointed out the merits of the Bayesian network graphical component, which may be useful for researchers in further investigating the correlational relationships of health dimensions among and/or between generic preference-based measures and specific health-profile measures.

In this study, we apply `MB-MBC` to predict EQ-5D from PDQ-39. Contrary to Le et and Doctor's method [147] that learns an independent Bayesian network for each EQ-5D item, our approach builds a single MBC identifying interactions among all variables involved in EQ-5D. In fact, taking into account the dependence relationships among EQ-5D items is crucial here for both better prediction performance and graphical structure interpretation.

In what follows, we firstly introduce the used Parkinson's data set, then present the experimental study and the results for predicting EQ-5D from PDQ-39.

Table 6.3: The Parkinson's disease questionnaire PDQ-39 items.

Mobility	
pdq1	Had difficulty doing the leisure activities you would like to do
pdq2	Had difficulty after your home e.g. DIY, housework, cooking
pdq3	Had difficulty carrying bags of shopping
pdq4	Had problems walking half a mile
pdq5	Had problems walking 100 yards
pdq6	Had problems getting around the house as easily as you would like
pdq7	Had problems getting around in public
pdq8	Needed someone else to accompany you when you went out
pdq9	Felt frightened or worried about falling over in public
pdq10	Been confined to the house more than you would like
Activities of daily living	
pdq11	Had difficulty washing yourself
pdq12	Had difficulty dressing yourself
pdq13	Had problems doing up buttons or shoe laces
pdq14	Had problems writing clearly
pdq15	Had difficulty cutting up your food
pdq16	Had difficulty holding a drink without spilling it
Emotional well-being	
pdq17	Felt depressed
pdq18	Felt isolated and lonely
pdq19	Felt weepy or tearful
pdq20	Felt angry or bitter
pdq21	Felt anxious
pdq22	Felt worried about your future
Stigma	
pdq23	Felt you had to conceal you Parkinson's from people
pdq24	Avoided situations which involve eating or drinking in public
pdq25	Felt embarrassed in public due to having Parkinson's disease
pdq26	Felt worried by other people's reaction to you
Social support	
pdq27	Had problems with your close personal relationships
pdq28	Lacked support in the ways you need from your spouse or partner
pdq29	Lacked support in the ways you need from your family or close friends
Cognitions	
pdq30	Unexpectedly fallen asleep during the day
pdq31	Had problems with your concentration, e.g. when reading or watching TV
pdq32	Felt your memory was bad
pdq33	Had distressing dreams or hallucinations
Communication	
pdq34	Had difficulty with your speech
pdq35	Felt unable to communicate with people properly
pdq36	Felt ignored by people
Bodily discomfort	
pdq37	Had painful muscle cramps or spasms
pdq38	Had aches and pains in your joints or body
pdq39	Felt unpleasantly hot or cold

6.4.1 Parkinson's disease data set

The used Parkinson's disease data set was obtained from an international multipurpose database collected by the National Center of Epidemiology, Carlos III Institute of Health, Madrid. Patients with diagnosis of Parkinson's disease by neurologists with expertise in movement disorders, and according to internationally recognized diagnostic criteria [148], were followed up in movement disorder clinics. Patients in all stages of PD (Hoehn and Yahr 1 to 5) were included.

In total, the analyzed data set contains $N = 488$ patients, where 59.43% are male and 40.57% are female, and the average age for all patients is 65 years old (minimum = 30, maximum = 89). For each patient, we have information about the PDQ-39 items, represented in Table 6.3, with values ranging from 0 (never) to 4 (always); and the corresponding EQ-5D with values ranging from 1 (no problems) to 3 (severe problems).

Table 6.4 presents the frequencies (%) of the different EQ-5D item values in the data set. It can be clearly observed that the number of patients with severe problems are limited for all class variables as in most real cohorts, especially for `mobility` where only 7 instances are available.

Table 6.4: EQ-5D items distribution in PD data set including 488 patients.

Item	No problems	Some problems	Severe problems
`Mobility`	192(39.34%)	289(59.22%)	7(1.43%)
`Self-care`	255(52.25%)	201(41.19%)	32(6.56%)
`Usual activities`	177(36.27%)	263(53.89%)	48(9.84%)
`Pain/discomfort`	175(35.86%)	247(50.61%)	66(13.52%)
`Anxiety/depression`	233(47.75%)	216(44.26%)	39(7.99%)

The objective is to simultaneously predict the 5 class values of EQ-5D from PDQ-39 using `MB-MBC` algorithm. Given the EQ-5D values, to complement them, the corresponding utility score utility index could also be induced using the UK general scoring system [103].

For instance, let's assume that we obtain an EQ-5D equal to $\mathbf{c} = (1, 1, 2, 2, 3)$ indicating that the considered patient has no problems with `mobility` and `self-care`; some problems with `usual activities` and `pain/discomfort`; and severe problems with `anxiety/depression`. Based on UK scoring system [103], EQ-5D utility index is 1 - 0.081 - 0.036 - 0.123 - 0.236 - 0.269 = 0.255.

6.4.2 Experimental design

We applied `MB-MBC` on the Parkinson's disease data set and we compared it as previously against `CB-MBC` and `IndepMBs` algorithms. We also considered, as additional Bayesian network-based approach, the `IndepPC-BNs` algorithm recently proposed in [147] to predict EQ-5D from Health Surveys SF-12. Moreover, we compared `MB-MBC` against three additional state-of-the-art approaches, namely, `MNL`, `OLS` and `CLAD`, the last two being based on the utility score. In what follows, we briefly present more details for each of these new considered approaches:

- Independent PC Bayesian networks (`IndepPC-BNs`): PC algorithm [205] is a constraint-based approach for learning Bayesian networks from data. It starts with a fully connected DAG, then sequentially removes edges between nodes based on statistical independence tests. Similar to Le and Doctor [147] that recently applied this approach to predict EQ-5D utility index from Health Surveys SF-12, we used the PC algorithm to learn independently a Bayesian network for each class variable in EQ-5D.

- Multinomial logistic regression (`MNL`) [100, 147]: Using the multinomial logistic regression on an input set of feature variables, this approach returns the estimated posterior probabilities of each class value; then the class value with the highest probability is selected. Similar to `IndepMBs` and `IndepPC-BNs`, `MNL` is applied independently to each class variable, and the results are aggregated to obtain the predicted class vector.

- Ordinary least squares (`OLS`): is one of the mostly used methods for mapping specific HRQoL instruments such as Health Surveys SF-12 and PDQ-8 into a generic utility index [43, 78, 147]. In the `OLS` model, the EQ-5D utility index is directly regressed on the PDQ-39 items. In other words, `OLS` does not provide the 5 estimated class values of EQ-5D, but only returns the estimated EQ-5D utility index.

- Censored least absolute deviation (`CLAD`) [176]: is a generalization of the least absolute deviations method. Similar to `OLS`, `CLAD` is widely used to convert specific HRQoL instruments into a generic utility index [43, 147, 208], and it only estimates EQ-5D utility index without predicting the 5 class values of EQ-5D.

All methods were run in Matlab R2010b on a laptop 2.2 GHz with 6 GB RAM using Windows operating system. The HITON and PC algorithms were run using Causal Explorer Toolkit [6] provided as compiled Matlab functions. G^2 statistical test was used to evaluate the conditional independencies between variables with a significance level $\alpha = 0.01$, and experiments are performed only with the conditioning set size value $max_{CS} = 1$. Figure 6.25 summarizes the approaches used for predicting EQ-5D from PDQ-39.

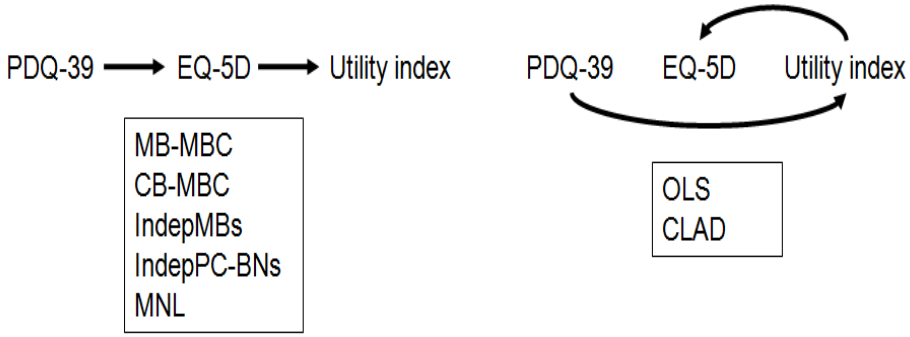


Figure 6.25: Approached used for predicting EQ-5D from PDQ-39.

Note finally that we applied `MB-MBC` and `IndepMBs` with a restriction of the Markov blanket set of each class variable $MB(C_i)$ to the set of its parents-children $PC(C_i)$. This restriction was introduced based upon the theoretical discussion introduced by Aliferis et al. in [5] and the empirical observation that including more spouses leads to a less accurate MBC classifier. In fact, Aliferis et al. [5] discussed in Section 4.6 five plausible scenarios explaining the better performance of substituting the parents-children set in place of the Markov blanket set. The third scenario applies in our case, where the spouses have connecting paths to the class variables that cannot be blocked due to the small sample size, i.e., the conditional independencies between the spouses and the class variables could not be established due to the small number of instances in the PD data set (including only 488 instances).

As performance evaluation metrics, we considered the mean (see Equation (3.4)) and the global accuracy (see Equation (3.5)). In addition, we used the following metrics, commonly used in comparison with `MNL`, `OLS` and `CLAD` methods [147]:

- The mean squared error (MSE) between the true and predicted EQ-5D utility scores.
- The mean absolute error (MAE) between the true and predicted EQ-5D utility scores.
- The square of the Pearson product-moment correlation (R^2) between the true and predicted EQ-5D utility scores.
- The absolute difference (AbsDiff) between the true and predicted EQ-5D utility mean scores, i.e. the absolute difference is computed between the mean of all true EQ-5D utility scores and the mean of all predicted EQ-5D utility scores.

6.4.3 Experimental results

Table 6.5 shows the classification performance results for the 5-fold cross-validation experiments performed on PD data set with mean values and standard deviations for each metric and each method. Recall that `OLS` and `CLAD` only return the utility index; thus, in order to compute the mean and global accuracies for `OLS` and `CLAD`, we proceeded by retrieving the EQ-5D class values as follows: first, we look for the utility index from the UK scoring list [103] closest to the one returned by `OLS` and `CLAD`, then we determine the EQ-5D vector corresponding to that index.

In Table 6.5, `MB-MBC` presents the best mean accuracy (71%), whereas surprisingly `IndepMBs` outperforms in the global accuracy the remaining approaches with 20%. We ran a multiple comparison test of all method performance using the Friedman test followed by the Tuckey-Karmer post-hoc test with a significance level equal to 0.05. It turns out that, for both mean and global accuracy, `MB-MBC` and `IndepMBs` results are only significantly better than `OLS` and `CLAD` methods.

Table 6.5: Estimated accuracies (mean ± std. dev.) over PD data set.

Method	Mean accuracy	Global accuracy
`MB-MBC`	$\mathbf{0.7119 \pm 0.0338}$	0.2030 ± 0.0718
`CB-MBC`	0.6807 ± 0.0285	0.1865 ± 0.0429
`IndepMBs`	0.7009 ± 0.0427	$\mathbf{0.2051 \pm 0.0835}$
`IndepPC-BNs`	0.6587 ± 0.0636	0.1867 ± 0.0937
`MNL`	0.6926 ± 0.0430	0.1802 ± 0.0713
`OLS`	0.4201 ± 0.0252	0.0123 ± 0.0046
`CLAD`	0.4254 ± 0.0488	0.0143 ± 0.0171

In addition, Table 6.6 presents results for MSE, MAE, R^2 and AbsDiff metrics. The best result for each metric is written in bold. Once again, `MB-MBC` outperforms other predictive approaches in terms of MSE and MAE. `IndepMBs` presents the best R^2 and `MNL` produces the best AbsDiff.

Note that, both `OLS` and `CLAD` methods performed poorly for all the performance metrics in Tables 6.5 and 6.6. As pointed out by Le and Doctor [147], this may be due to certain limitations of these regression methods such as predictive values that are outside the domain of the preference-based target, ceiling/floor effects, and assignment to health states that are not defined in the UK scoring list. Previous studies testing `OLS` and `CLAD` for predicting the EQ-5D utility index from the Health Surveys SF-12 [147, 208], or from PDQ-8, the short version of PDQ-39 [43], proved that `OLS` and `CLAD` methods induce very similar results with a

possible better performance of the simple `OLS` over the more theoretically justifiable `CLAD`. In our case, `OLS` method resulted in a slightly better MSE, MAE and R^2 than `CLAD`, but for the absolute difference between the true and the predicted EQ-5D mean scores, `CLAD` performed better.

`MNL` performed quite well compared to `OLS` and `CLAD` as well as compared to the Bayesian network-based approaches. For instance, it had better results for mean accuracy, MSE, MAE, and R^2 than `IndepPC-BNs` and `CB-MBC`; it also resulted in the best AbsDiff compared to all the remaining approaches. However, `MNL` presented a lower global accuracy compared to all Bayesian network-based approaches. This can be explained by the fact that taking into account the probabilistic dependence relationships among class and feature variables ensures a better predictive performance, and in this context, `MB-MBC` and `IndepMBs` performed better than `CB-MBC` and `IndepPC-BNs` through determining the Markov blanket around each class variable.

Table 6.6: MSE, MAE, R^2 and AbsDiff (mean $\pm$ std. dev.) over PD data set.

Method	MSE	MAE	R^2	AbsDiff
`MB-MBC`	$\mathbf{0.0650 \pm 0.0156}$	$\mathbf{0.1737 \pm 0.0316}$	0.5996 ± 0.0683	0.0659 ± 0.0373
`CB-MBC`	0.0905 ± 0.0167	0.1973 ± 0.0298	0.4094 ± 0.0860	0.0790 ± 0.0541
`IndepMBs`	0.0699 ± 0.0188	0.1784 ± 0.0328	$\mathbf{0.6026 \pm 0.0653}$	0.0737 ± 0.0315
`IndepPC-BNs`	0.0909 ± 0.0909	0.2026 ± 0.0391	0.4602 ± 0.1379	0.0863 ± 0.0670
`MNL`	0.0759 ± 0.0152	0.1922 ± 0.0284	0.4935 ± 0.0961	$\mathbf{0.0503 \pm 0.0331}$
`OLS`	0.1832 ± 0.0373	0.3502 ± 0.0422	0.0186 ± 0.0177	0.0943 ± 0.0388
`CLAD`	0.1962 ± 0.0360	0.3583 ± 0.0395	0.0165 ± 0.0155	0.0779 ± 0.0278

Moreover, contrary to `MNL`, `OLS` and `CLAD`, Bayesian network-based approaches present also the merit of representing the relationships between all variables through their graphical structure component. In our study, in order to investigate the dependence relationships among EQ-5D and PDQ-39 variables, we first examine in Figure 6.26 the graphical structure of the MBC network learnt by the `MB-MBC` algorithm from the full PD data set, then compare it to the graphical structures learnt by `CB-MBC`, `IndepMBs` and `IndepPC-BNs`. The medical significance of the obtained graphical structures was verified by the medical doctor Pablo Martínez-Martín, who is a neurologist specialist in Parkinson's disease.

Firstly, the class subgraph in Figure 6.26 (red arcs) shows associations between the three class variables `mobility`, `self-care` and `usual activities` which may reveal the strong relevance between these classes. `Pain/discomfort` is not directly related to any other class variable, but its Markov blanket includes the class `usual activities` which proves as well

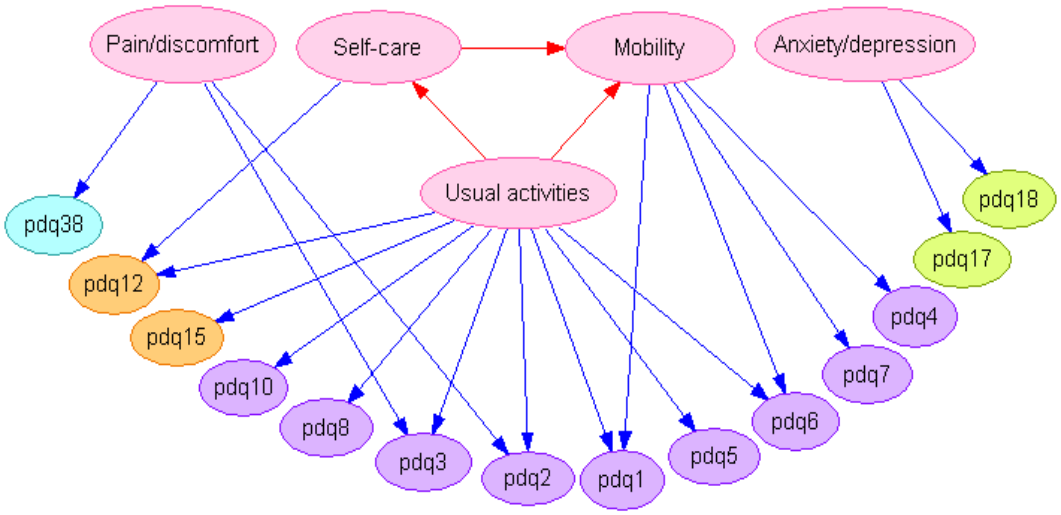


Figure 6.26: MBC graphical structure learnt by `MB-MBC` for Parkinson's disease data set.

the strong relevance between both classes. `Anxiety/depression` has no direct connections with the remaining classes. This can be explained by the fact that `anxiety/depression` is more related to emotional problems rather than physical health problems (i.e., `mobility`, `self-care`, `usual activities` and `pain/discomfort`).

Secondly, the bridge subgraph (blue arcs) reveals direct dependence relationships between EQ-5D classes and PDQ-39 features. Table 6.3 lists the PDQ-39 features grouped into 8 domains: Mobility, activities of daily living, emotional well-being, stigma, social support, cognitions, communication and bodily discomfort. Each domain is depicted in Figure 6.26 with a different color. We have the following probabilistic dependence relationships from EQ-5D to PDQ-39:

- `Mobility` is directly associated with pdq1, pdq4, pdq6, and pdq7.
- `Self-care` is directly associated with pdq12.
- `Usual activities` is directly associated with pdq1, pdq2, pdq3, pdq5, pdq6, pdq8, pdq10, pdq12 and pdq15.
- `Pain/discomfort` is directly associated with pdq2, pdq3, pdq38.
- `Anxiety/depression` is directly associated with pdq17, pdq18.

Note that the detected associations are very appropriate and clearly related to mobility, self-care and usual activities. The selected pdq variables associated with these three class variables exclusively pertain to *mobility* and *activities of daily living* domains, which cover the main information about PD. In fact, PD is a neurodegenerative disorder characterized by motor manifestations (bradykinesia, rest tremor and balance impairment) and non-motor symptoms (depression, psychosis and sleep disturbance) [148]; and for the huge majority of PD patients, from earliest to most advanced stages, the common perceived health problems are reflected as limitations for *mobility* and *activities of daily living*, whereas the most prevalent non-motor symptoms are associated with the impact on patients' health status perception (pain and depression, for example).

For `pain/discomfort`, the associations are well explained, as a whole, from the point of view of bone and joint disorders (pdq38), mainly. There are other types of pain in PD, more difficult to associate with these findings. Perhaps, pain due to dystonia in off state can also be related, in some way, with those selected PDQ-39 items. Distinction between the types of pain that may be present in PD is not easy.

For `anxiety/depression`, it seems that depression is quite well represented by the detected pdq items, but not the anxiety. In fact, there are PDQ-39 items about anxiety (pdq21 about feeling anxious, pdq22 about feeling worried about the future), but they do not appear in association with the EQ-5D class variable `anxiety/depression`. This can be explained by the well-known close relationship between depression and anxiety. A useful representation of this connection is made evident with the Hospital Anxiety and Depression scale, a measure furnishing scores for rating anxiety and depression separately but also usable as a unique score of emotional distress [77, 189].

Taking the previous arguments into account, the findings in this study have sense from a clinical point of view. Moreover, EQ-5D is more restricted in content than the PDQ-39, explaining why several components of the PDQ-39, in addition to the most immediately relatable (for example, pain), can converge on a domain of the EQ-5D. Therefore, we may conclude that the combination of the selected variables in the network properly represents the relationships between the generic (EQ-5D) and specific (PDQ-39) instruments, and covers both motor and non-motor symptoms of PD.

The feature subgraph is empty due to the restriction of the Markov blanket set of each class variable $MB(C_i)$ to the set of its parents children $PC(C_i)$, that is, no feature spouses are allowed and thereby the parents of each feature variable can be only among class variables. Finally, notice that several features are also not present in Figure 6.26 since no associations were detected between them and the EQ-5D class variables. These features are considered

irrelevant and this may be due to the lack of instances of these features and/or their low interactions with the other variables in the analyzed data set.

For structural comparison, we depict in Figures 6.27, 6.28 and 6.29, the graphical structures learnt by `CB-MBC`, `IndepMBs` and `IndepPC-BNs` from the full PD data set, respectively. As shown in Figure 6.27, `CB-MBC` fails to detect any direct dependence relationship among the EQ-5D class variables, i.e., the class subgraph is empty. For the bridge subgraph, and similar to `MB-MBC`, the following dependence relationships are detected: `mobility` is associated with pdq4 and pdq7, `self-care` with pdq12, `usual activities` with pdq2, `pain/discomfort` with pdq3 and pdq38, and `anxiety/depression` with pdq17. Moreover, additional arcs are discovered in the bridge subgraph between the EQ-5D class variables and pdq9, pdq21, pdq23, pdq25, pdq28, pdq30, pdq32 and pdq39. Regarding the feature subgraph (green arcs), only three arcs were added between pdq11 and pdq23, pdq12 and pdq28, and pdq21 and pdq27.

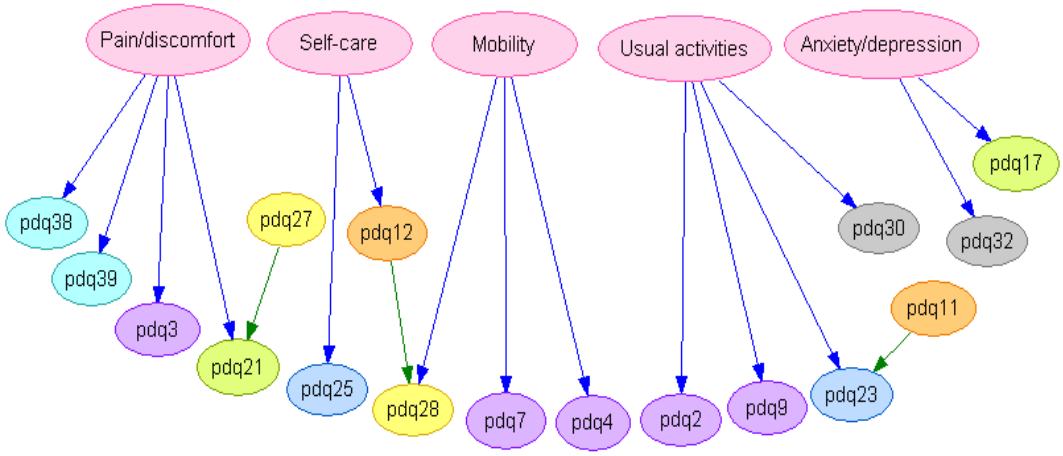


Figure 6.27: MBC graphical structure learnt by `CB-MBC` for Parkinson's disease data set.

Figure 6.28 shows the five Markov blanket-based Bayesian network classifiers learnt independently for each EQ-5D class variable by `IndepMBs`. Being based on the same HITON algorithm [4, 5], `IndepMBs` discovered similar dependence relationships between the EQ-5D and the pdq items, as `MB-MBC` does. However, as it can be observed, the main drawback of `IndepMBs` is its inability to detect the dependence relationships between the different EQ-5D class variables and their simultaneous interactions with the pdq items.

Additionally, Figure 6.29 presents the graphical structure of the Bayesian network learnt by `IndepPC-BNs` for the `mobility` class variable. Many dependence relationships are added between the pdq items, and as determined by `MB-MBC` and `Indep-MBs`, `mobility` is only di-

rectly associated with pdq1, pdq4, pdq6 and pdq7. Similar conclusions are obtained for the Bayesian network graphical structures learnt by `IndepPC-BNs` for the class variables `self-care`, `usual activities`, `pain/discomfort`, and `anxiety/depression` (details and graphs not shown). As in `IndepMBs`, the main drawback of `IndepPC-BNs` is that each network is learnt independently for each class variable, and thus interactions between class variables could not be detected.

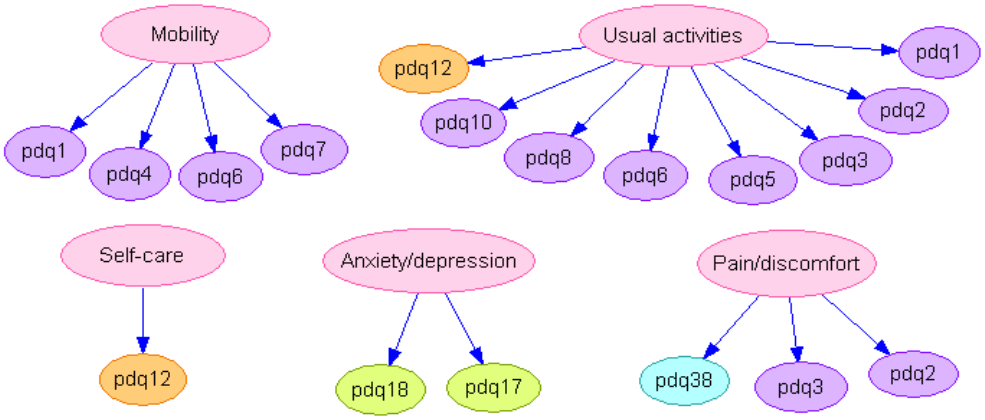


Figure 6.28: Graphical structures learnt by `IndepMBs` for Parkinson's disease data set.

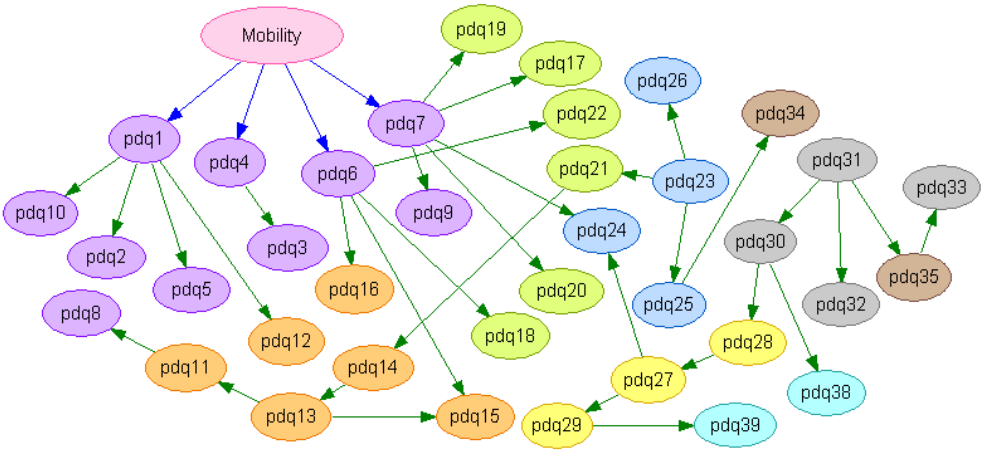


Figure 6.29: Bayesian network graphical structure learnt by `IndepPC-BNs` for EQ-5D `mobility` class and PDQ-39 variables.

Finally, the computation times consumed by each method on PD data set are reported

in Table 6.7 measured in seconds. `OLS` is the fastest. `MB-MBC` is quite efficient and requires less time than `IndepMBs` and `IndepPC-BNs`. `IndepPC-BNs` may take more time since it builds a full Bayesian network for each class variable whereas `MB-MBC` and `IndepMBs` only locally determine the Markov blanket around each class variable. `CB-MBC` takes always the longest computation time since it is based on a wrapper approach. Thus, we may conclude that regression and filter approaches require less computation, whereas the wrapper approach requires more computation.

Table 6.7: Computation learning times over PD data set.

Method	Time (seconds)
`MB-MBC`	13.38 ± 0.72
`CB-MBC`	757.51 ± 47.91
`IndepMBs`	34.98 ± 1.57
`IndepPC-BNs`	23.40 ± 0.38
`MNL`	10.56 ± 4.06
`OLS`	$\mathbf{0.72 \pm 0.08}$
`CLAD`	0.75 ± 0.21

6.5 Conclusion

In this chapter, we have presented a novel MBC learning approach using Markov blankets jointly with its application to two important real-world problems dealing with predicting the HIV-1 reverse transcriptase and protease inhibitors, and predicting the European Quality of Life-5 Dimensions EQ-5D from the Parkinson PDQ-39 items.

Experimental results on both considered case studies were promising compared with state-of-the-art learning algorithms. As a constraint-based approach, `MB-MBC` ensured, through the induced MBC graphical structures, an accurate identification of the probabilistic dependence relationships among class and feature variables. The analysis of learned MBC graphical structures allowed us to gain insight into both known and novel interactions among RTI and PI drugs and their corresponding resistance mutations for the HIV-1 application, as well as among EQ-5D and PDQ-39 for the Parkinson's disease application. Moreover, thanks to this filter and a local approach to MBC learning, we can lighten the computational burden of MBC learning process of wrapper algorithms and provide accurate MBC structures.

Note, however, that our approach has two main limitations. First, it cannot deal directly with continuous variables and requires a discretization pre-processing step before its application to continuous data. So, in the future, it will be an interesting issue to extend `MB-MBC`

to allow the combination of both discrete and continuous predictor variables. Second, our approach cannot handle instances with missing values. To overcome this limitation, we have to perform missing values imputation before using `MB-MBC`, or adapt for example the more sophisticated EM algorithm [65] to enable parameter estimations or structure learning from incomplete data [80].

Finally, since the class distributions in both analyzed data sets were imbalanced, another line for future research is to extend our approach to deal with the challenging task of imbalanced multi-dimensional data sets. This in fact might improve the learning and the classification performance results.

Part IV

Mining uni- and multi-dimensional evolving streaming data

Chapter 7

Mining uni-dimensional data streams with partially labeled instances

7.1 Introduction

As commented in Chapter 4, the field of mining uni-dimensional concept-drifting data streams has received an increasing attention and has been intensively researched in recent years. Several approaches have been proposed and applied to a wide range of real-world applications [1, 86]. However, most of these approaches are based on supervised classification algorithms assuming the availability of labeled data for accurate learning. In general, they continuously monitor classification performance and detect a concept drift if there is a significant fall over time. Unfortunately, the assumption of entirely labeled data streams availability is often violated in real-world problems, as labels may be scarce and not readily available.

For instance, for the malware detection problem, only a few true labels (i.e., malware or goodware) may be available immediately after the classification process, and therefore we may have to wait for a quite long time until all the instances are labeled. This leads a traditional stream classification algorithm to choose between updating the classifier with just a few labeled data, which usually results in a poor classifier, or waiting longer to get all labeled data. This can also affect the quality of the classifier since most of the data will be outdated.

Semi-supervised learning methods have proved to be useful in such cases since they combine both labeled and unlabeled data to enhance the performance of classification algorithms

[243]. However, they mainly assume that data is generated according to some stationary distribution, which is not true when learning from evolving data streams, where changes may occur over time.

In this chapter, we propose a new approach (`CPL-DS`) for classifying partially labeled uni-dimensional data streams. Our aim is to take advantage of unlabeled data to detect possible concept drifts and, if necessary, update the classifier over time even if only a few labeled data are available. To this end, inspired by earlier work by Dasu et al. [55], we use the Kullback-Leibler (KL) divergence [135] to measure distribution differences between data stream batches. Then, based on a bootstrapping method [70], we determine whether or not the KL measures are statistically significant, i.e., whether or not a drift occurs. However, our approach differs from Dasu et al.'s work on three key points. First, we do not only detect whether or not a drift occurs, but we further distinguish and monitor three possible kinds of drift: feature, conditional or dual drift (see Section 4.2). Second, we do not assume that the available data streams are entirely labeled. Indeed, we detect possible drifts using both labeled and unlabeled instances. Moreover, we propose a general approach for learning from all these instances. In fact, when any of the three possible kinds of drift is detected, a new classifier is learned using the expectation-maximization (EM) algorithm [65]. EM has been widely used in semi-supervised learning where it has been found to improve classification accuracy, especially when there is a small number of labeled data [162]. Otherwise, i.e., when no drift is detected, the current classifier is left unchanged.

Our approach is then informed since it only adapts the classifier when a concept drift is detected. It is also general so that it can be applied with different classification learning algorithms. In this chapter, we consider two classifiers, namely naive Bayes and logistic regression. We perform experiments on a synthetic and a real data set using different percentages of labeled instances. Moreover, we evaluate our approach using a real-world malware detection data set, where we deal with the additional problem of imbalanced data streams and make use of two recently proposed approaches for mining skewed data streams, namely clustering-sampling [228] and SERA [39]. The results show that our approach performs well even using limited amounts of labeled data.
This chapter is based on the published papers [25, 26].

Chapter outline

The remainder of this chapter is organized as follows. Section 7.2 briefly reviews existing approaches for learning from uni-dimensional data streams containing partially labeled instances. Section 7.3 introduces our new `CPL-DS` approach. Sections 7.4 and 7.5 present the

used data sets and the experimental results, respectively. Finally, Section 7.6 rounds the chapter off with some conclusions.

7.2 Related work

Semi-supervised classification is useful when only a limited amount of labeled data is available. A semi-supervised learning literature surveys are presented in [38, 174, 243] where several semi-supervised classification methods are discussed, such as self-training, co-training, graph-based methods, and expectation-maximization algorithm. However, they mainly assume that data is generated according to some stationary distribution, which is not true when learning from evolving data streams, where changes may occur over time.

Moreover, as commented above, most of previously proposed works for mining data streams assume that true labels are entirely available in data streams. To the best of our knowledge, only two adaptive semi-supervised learning methods have been proposed to address the problem of scarceness of labeled instances in concept-drifting data streams.

The first, proposed by Klinkenberg [125], is based on transductive support vector machines and it maintains two separate adaptive windows on labeled and unlabeled data in order to monitor, respectively, the probabilities $P(c \mid \mathbf{x})$ captured by labeled data and $P(\mathbf{x})$ underlying both labeled and unlabeled data. This was justified by the fact that $P(c \mid \mathbf{x})$ and $P(\mathbf{x})$ may change at different rates. However, although theoretically well-founded, this method has never been evaluated.

The second work was recently proposed by Masud et al. [154]. It is based on an ensemble approach where each model in the ensemble is built as micro-clusters using a semi-supervised clustering technique. In fact, the learning step of each model starts by choosing k_c points from the labeled data of class C to initialize k_c centroids. Then, the EM algorithm is applied by iterating the following two steps until convergence: The E-step assigns each unlabeled data point $\mathbf{x}$ to a cluster such that its contribution to a cluster-impurity function is minimized, and the M-step recomputes each cluster centroid by averaging all the points in that cluster. Finally, a summary of the statistics of the instances belonging to each built cluster is saved as a micro-cluster. These micro-clusters serve as a classification model.

To cope with stream evolution, Masud et al. [154] keep an ensemble of L models. Whenever a new model is built from a new data chunk, they update the ensemble by choosing the best L models from $L + 1$ models (previous L models and the new model), based on their individual accuracies on the labeled instances of the new data chunk. Besides, they refine the existing models in the ensemble whenever a new class of data evolves in the stream.

7.3 CPL-DS: Classifying partially labeled uni-dimensional data streams

In this section, we will first present a background on EM algorithm, then introduce the two considered classifiers, namely naive Bayes and logistic regression, learnt from both labeled and unlabeled instances. Afterwards, we will present the drift detection method.

7.3.1 Background on EM algorithm

Let $\mathcal{D}$ denote the uni-dimensional data stream that arrives over time in batches. Let $\mathcal{D}^s$ denotes the batch at step s. $\mathcal{D}^s$ consists of the union of two disjoint subsets $\mathcal{D}^s_{ul}$ and $\mathcal{D}^s_{fl}$. D^s_{ul} denotes a set of N^s_{ul} unlabeled instances ($\mathbf{x}$), whereas $\mathcal{D}^s_{fl}$ denotes a set of N^s_{fl} fully labeled instances $(\mathbf{x}, c)$, s.t. $\mathbf{x}$ represents an n-dimensional feature vector $(x_1, ..., x_n)$ and $c \in \Omega_C = \{c_1, c_2, ..., c_{|C|}\}$ represents the corresponding class value for labeled instances. $N^s = N^s_{ul} + N^s_{fl}$ denotes the total size of $\mathcal{D}^s$.

Learning a classifier from the $\mathcal{D}^s$ data corresponds to maximizing the likelihood of $\mathcal{D}^s$ given the parameters $\mathbf{\Theta}^s$. Assuming that instances are independent, this likelihood is the product of all (labeled and unlabeled) instance probabilities expressed as follows [162]:

$$\begin{aligned} P(\mathcal{D}^s \mid \mathbf{\Theta}^s) = & \prod_{l=1}^{N^s_{fl}} P(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s)\ P(\mathbf{x}^{(l)} \mid \mathbf{\Theta}^s) \\ & \cdot \prod_{l=1}^{N^s_{ul}} \sum_{j=1}^{|C|} P(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s)\ P(\mathbf{x}^{(l)} \mid \mathbf{\Theta}^s) \end{aligned} \tag{7.1}$$

where the first term is derived from fully labeled instances, and the second one is based on unlabeled data where the sum expresses the fact that the unknown class value can be any of the existing values. Then, considering $log\ P(\mathcal{D}^s \mid \mathbf{\Theta}^s) = LL(\mathcal{D}^s \mid \mathbf{\Theta}^s)$, we have:

$$\begin{aligned} LL(\mathcal{D}^s \mid \mathbf{\Theta}^s) = & \sum_{l=1}^{N^s_{fl}} log\ (P(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s)\ P(\mathbf{x}^{(l)} \mid \mathbf{\Theta}^s)) \\ & + \sum_{l=1}^{N^s_{ul}} log \sum_{j=1}^{|C|} P(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s)\ P(\mathbf{x}^{(l)} \mid \mathbf{\Theta}^s) \end{aligned} \tag{7.2}$$

Notice that this equation contains a log of sums for the unlabeled data, which makes a

maximization by partial derivatives with respect to $\mathbf{\Theta}^s$ analytically intractable.

Consider that we can have access to the class labels of all the instances, represented using a matrix of binary indicator variables $\mathbf{z}$, where rows correspond to different instances and columns to different classes, so that an entry is $z_{lj} = 1$ iff $c_j^{(l)}$ is the class of the feature vector $\mathbf{x}^{(l)}$, and $z_{lj} = 0$ otherwise. Thus, Equation (7.2) can be rewritten as follows without a log of sums, because only one term inside the sum would be non-zero:

$$LL(\mathcal{D}^s \mid \mathbf{\Theta}^s; \mathbf{z}) = \sum_{l=1}^{N^s} \sum_{j=1}^{|C|} z_{lj} \; log\Big(P(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s) \; P(\mathbf{x}^{(l)} \mid \mathbf{\Theta}^s)\Big) \tag{7.3}$$

We use the EM algorithm [65] to find the maximum $\hat{\mathbf{\Theta}}^s$ of Equation (7.3). Let $\hat{\mathbf{z}}_t$ and $\hat{\mathbf{\Theta}}_t^s$ denote the estimates for $\mathbf{z}$ and $\mathbf{\Theta}^s$ at iteration t. EM starts with an initial estimate of classifier parameters $\hat{\mathbf{\Theta}}_1^s$ from only the labeled data in $\mathcal{D}_{fl}^s$. Then, it iterates over the E- and M-steps:

- The E-step uses the current classifier parameters to probabilistically assign labels to the unlabeled instances in $\mathcal{D}_{ul}^s$. Formally, it computes the expected value of

$$\hat{\mathbf{z}}_{t+1} = E[\mathbf{z} \mid \mathcal{D}^s; \hat{\mathbf{\Theta}}_t^s] \tag{7.4}$$

  Clearly, for labeled data, z_{lj} is easily determined since classes are already known. For unlabeled data, z_{lj} should be estimated as follows:

$$E[z_{lj} \mid \mathcal{D}^s; \hat{\mathbf{\Theta}}_t^s] = \begin{cases} 1 & \text{if } c_j^{(l)} = arg \underset{c}{max} \; P(c \mid \mathbf{x}^{(l)}; \hat{\mathbf{\Theta}}_t^s) \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

- The M-step re-estimates the classifier for all the data in $\mathcal{D}^s$, i.e., using all instances (the originally labeled and the newly labeled by the E-step). In fact, this step consists of computing new parameters $\hat{\mathbf{\Theta}}_{t+1}^s$ using the current expected value of $\hat{\mathbf{z}}$. Formally, we have

$$\hat{\mathbf{\Theta}}_{t+1}^s = arg \underset{\mathbf{\Theta}^s}{max} \; LL(\mathcal{D}^s \mid \mathbf{\Theta}^s; \hat{\mathbf{z}}_{t+1}) \tag{7.6}$$

These two steps are iterated until convergence as proved by Dempster et al. [65]. At convergence, EM finds $\hat{\mathbf{\Theta}}^s$ that locally maximizes the log-likelihood with respect to both labeled and unlabeled data.

7.3.2 Used classifiers

Naive Bayes (NB)

Naive Bayes [158] is a generative classifier that optimizes the joint log-likelihood of the data. Based on the assumption that the features are all conditionally independent of one another given the class variable C, parameters $\mathbf{\Theta}^s$ denote the probability table of C, i.e., $P(C)$, as well as the conditional probability tables of each feature X_i given C, i.e., $P(X_i \mid C)$, with $i \in \{1, \ldots, n\}$.

To classify a given instance, the posterior probability of each possible class value c_j is computed, and then, the most probable class c^* is selected. More formally,

$$c^* = arg \max_{c_j} P(c_j) \prod_{i=1}^{n} P(x_i \mid c_j) \tag{7.7}$$

Logistic regression (LR)

Logistic regression [110] is a discriminative classifier that maximizes the conditional log-likelihood instead of the log-likelihood. Hence, in this case, instead of (7.3), EM algorithm maximizes the following formula:

$$LL(\mathcal{D}^s \mid \mathbf{\Theta}^s; \mathbf{z}) = \sum_{l=1}^{N^s} \sum_{j=1}^{|C|} z_{lj} \; logP(c_j^{(l)} \mid \mathbf{x}^{(l)}; \mathbf{\Theta}^s) \tag{7.8}$$

where parameters $\mathbf{\Theta}^s$ are represented by the vector $(\mathbf{\Theta}^s_{j0}, \mathbf{\Theta}^s_{j1}, .., \mathbf{\Theta}^s_{jn})^T$ for $j = 1, ..., |C|$.

To classify a given instance, the posterior probability of each possible class value c_j is computed as follows:

$$P(C = c_j \mid \mathbf{x}; \mathbf{\Theta}^s) = \begin{cases} \frac{\exp(\mathbf{\Theta}^s_{j0} + \sum_{i=1}^{n} \mathbf{\Theta}^s_{ji} x_i)}{1 + \sum_{p=1}^{|C|-1} \exp(\mathbf{\Theta}^s_{p0} + \sum_{i=1}^{n} \mathbf{\Theta}^s_{pi} x_i)} & \forall j < |C| \\ \\ \frac{1}{1 + \sum_{p=1}^{|C|-1} \exp(\mathbf{\Theta}^s_{p0} + \sum_{i=1}^{n} \mathbf{\Theta}^s_{pi} x_i)} & \text{for } j = |C| \end{cases} \tag{7.9}$$

Then, the c_j value with the maximum probability is assigned as a label.

7.3.3 Detecting a concept drift

Given a new batch of uni-dimensional data stream $\mathcal{D}^{s+1}$, the objective is to detect changes whenever they occur and adapt the current classifier if necessary. In general, it is assumed

that as long as the joint probability distribution of $\mathcal{D}^{s+1}$ is similar to the distribution of $\mathcal{D}^{s}$, no concept drift occurs. Otherwise, a concept drift should be indicated.

In order to detect possible changes, we use the KL divergence [135], also known as the relative entropy, to measure differences between the empirical distributions of $\mathcal{D}^{s+1}$ and $\mathcal{D}^{s}$. Note that the KL divergence has two fundamental properties, namely, non-negativity, being 0 iff the two compared distributions are the same, and asymmetry. Moreover, a higher KL value indicates a higher dissimilarity between distributions, and so, a pronounced drift.

First, in order to monitor the conditional change, we proceed to measure the KL divergence kl_{cc} between the class posterior distributions of $\mathcal{D}^{s+1}$ and $\mathcal{D}^{s}$ using only their corresponding labeled instances. kl_{cc} is computed as a sum of KL divergences, each of which measuring the divergence between the conditional distributions of the class given feature instantiation, expressed as follows:

$$\begin{aligned} kl_{cc} &= \sum_{\mathbf{x}} KL\Big(\hat{P}_{\mathcal{D}^{s+1}_{fl}}(C \mid \mathbf{x}) || \hat{P}_{\mathcal{D}^{s}_{fl}}(C \mid \mathbf{x})\Big) \\ &= \sum_{\mathbf{x}} \sum_{j=1}^{|C|} \hat{P}_{\mathcal{D}^{s+1}_{fl}}(c_j \mid \mathbf{x}) \; log_2 \; \frac{\hat{P}_{\mathcal{D}^{s+1}_{fl}}(c_j \mid \mathbf{x})}{\hat{P}_{\mathcal{D}^{s}_{fl}}(c_j \mid \mathbf{x})} \end{aligned} \tag{7.10}$$

In addition, to monitor the feature change, we measure the KL divergence kl_{fc} between the feature distributions of $\mathcal{D}^{s+1}$ and $\mathcal{D}^{s}$ using all the labeled and unlabeled instances except the class variable:

$$kl_{fc} = KL\Big(\hat{P}_{\mathcal{D}^{s+1}}(\mathbf{x}) || \hat{P}_{\mathcal{D}^{s}}(\mathbf{x})\Big) = \sum_{\mathbf{x}} \hat{P}_{\mathcal{D}^{s+1}}(\mathbf{x}) \; log_2 \; \frac{\hat{P}_{\mathcal{D}^{s+1}}(\mathbf{x})}{\hat{P}_{\mathcal{D}^{s}}(\mathbf{x})} \tag{7.11}$$

In order to determine whether or not the computed KL measures are statistically significant, we use the bootstrapping method [70] following previous work reported in [55]. Intuitively, this method allows us to determine, by repeated sampling with replacement from the data, whether or not a specific measurement on the data is significant.

Specifically, to decide whether or not the resulting kl_{cc} value is significant, we consider the null hypothesis

$$H_{0_{cc}} : P_{\mathcal{D}^{s+1}_{fl}}(C \mid \mathbf{X}) = P_{\mathcal{D}^{s}_{fl}}(C \mid \mathbf{X}),$$

denoting that no conditional change has occurred. So, our objective is to determine the

probability of observing the value kl_{cc} if $H_{0_{cc}}$ is true.

To this end, given the empirical distribution $\hat{P}_{\mathcal{D}^s_{fl}}(C \mid \mathbf{x})$, we sample k data sets denoted S_b, $b = 1, ..., k$, each of size $2N^s_{fl}$. Then, we consider the first N^s_{fl} instances S_{b1} as coming from the distribution $\hat{P}_{\mathcal{D}^s_{fl}}(C \mid \mathbf{x})$, and the remaining N^s_{fl} instances $S_{b2} = S_b \setminus S_{b1}$ as coming from the other distribution $\hat{P}_{\mathcal{D}^{s+1}_{l}}(C \mid \mathbf{x})$; and we compute the bootstrap estimates $\hat{kl}_{ccb} = \sum_{\mathbf{x}} KL\big(\hat{P}_{S_{b2}}(C \mid \mathbf{x}) || \hat{P}_{S_{b1}}(C \mid \mathbf{x})\big)$ between each two samples S_{b2} and S_{b1}, $b = 1, ..., k$. The obtained estimates form an empirical distribution from which we construct a critical region $[\hat{kl}^{\alpha}_{cc}, \infty)$, where $\hat{kl}^{\alpha}_{cc}$ represents the $(1-\alpha)$-percentile of the bootstrap estimates, and α is a desired significance level.

Finally, if we observe that kl_{cc} falls into the critical region, i.e., $kl_{cc} > \hat{kl}^{\alpha}_{cc}$, we conclude that it is statistically significant and invalidates $H_{0_{cc}}$. In other words, we conclude that a conditional change is detected.

Similarly, in order to decide whether or not the resulting kl_{fc} value is significant, we consider the null hypothesis

$$H_{0_{fc}} : P_{\mathcal{D}^{s+1}}(\mathbf{x}) = P_{\mathcal{D}^s}(\mathbf{x}),$$

and apply the same process to determine the critical region $[\hat{kl}^{\alpha}_{fc}; \infty)$ and decide about a feature change. Note that, if either a feature or conditional change is detected, we proceed to learn a new classifier. Otherwise, the current classifier is left unchanged.

To recapitulate, Algorithm 7.1 outlines the whole proposed `CPL-DS` approach. First, KL divergence and the bootstrapping method are used to monitor possible conditional and feature changes (steps 3 to 6). If any change is detected, a new classifier is learned using the expectation maximization algorithm (step 8): an initial estimate of classifier parameters $\hat{\boldsymbol{\Theta}}^{s+1}_1$ is induced using only the labeled instances of the new data set $\mathcal{D}^{s+1}$ (step 9), then EM iterates over the E- and M-steps until convergence (steps 10 to 13). In case that no change occurred, the classifier is left unchanged (step 16).

7.4 Experimental design

We tested our approach on the following synthetic and real data sets.

7.4.1 Rotating hyperplane data set

The rotating hyperplane data set is considered as a benchmark synthetic data set and has been widely used to simulate the concept drift problem [92, 112, 220, 227]. In fact, this

Algorithm 7.1: `CPL-DS` algorithm

1. **Input** : $\mathcal{D}^s, \mathbf{\Theta}^s, \mathcal{D}^{s+1}, k, \alpha$
2. **Output** : $\mathbf{\Theta}^{s+1}$
3. Compute kl_{cc}
4. Compute the bootstrap estimates $\hat{kl}_{ccb}$, $b = 1, ..., k$, and critical region $[\hat{kl}^{\alpha}_{cc}, \infty)$
5. Compute kl_{fc}
6. Compute the bootstrap estimates $\hat{kl}_{fcb}$, $b = 1, ..., k$, and critical region $[\hat{kl}^{\alpha}_{fc}, \infty)$
7. **if** $kl_{cc} > \hat{kl}^{\alpha}_{cc}$ or $kl_{fc} > \hat{kl}^{\alpha}_{fc}$ **then**
8. A change is detected, learn a new classifier from $\mathcal{D}^{s+1}$
9. $\hat{\mathbf{\Theta}}^{s+1}_1 \leftarrow$ initial parameters induced only from labeled data $\mathcal{D}^{s+1}_{fl}$
10. **while** *no convergence* **do**
11. E-step: compute the expected labels for all unlabeled instances using (7.4)
12. M-step: update classifier parameters using (7.6) obtaining $\hat{\mathbf{\Theta}}^{s+1}$
13. **end while**
14. $\mathbf{\Theta}^{s+1} \longleftarrow \hat{\mathbf{\Theta}}^{s+1}$
15. **else**
16. No change is detected: $\mathbf{\Theta}^{s+1} \longleftarrow \mathbf{\Theta}^{s}$
17. **end if**
18. **return** $\mathbf{\Theta}^{s+1}$

synthetic data set allows us to carry out experiments with different types of drift, as well as different percentages of labeled data and, hence, to investigate the performance of our approach under controlled conditions.

A hyperplane in an n-dimensional space is denoted by $\sum_{i=1}^{n} w_i x_i = w_0$, where $\mathbf{w} = (w_1, ..., w_n)^T$ is the weight vector. Instances for which $\sum_{i=1}^{n} w_i x_i \geq w_0$ are labeled positive, and instances for which $\sum_{i=1}^{n} w_i x_i < w_0$ are labeled negative. Weights w_i are initialized by random values in the range of $[0, 1]$, and w_0 values are determined so that $w_0 = \frac{1}{2} \sum_{i=1}^{n} w_i$.

We generated x_i from a Gaussian distribution with mean μ_i and variance σ_i^2. The feature change is simulated by changing the mean, i.e., μ_i is changed to $\mu_i s_i(1+t)$, and the conditional change is simulated by the change of weights w_i to $w_i s_i(1+t)$. Parameter $t \in [0, 1]$ represents the magnitude of the changes, and parameter $s_i \in \{-1, 1\}$ specifies the direction of the changes which could be reversed with a probability of 0.1. We generated a data stream of 10 dimensions ($n = 10$) with 80000 instances, using different magnitudes of change t respectively set to 0.1, 0.2, 0.5, 1 for each 20000 instances. Then, we split the whole data stream into sets of blocks of size 2000, and from each block we considered equal training and testing subsets of size 1000, such that every training set is followed by a testing set.

7.4.2 Mushroom data set

The mushroom data set, from the UCI repository [11], is regarded as having virtual concept drift (i.e., feature changes) but no real concept drift (i.e., conditional changes) [130]. The mushroom data set contains 22 variables and 8124 instances. We split it into 6 blocks, and used 1000 instances from each block for training and 354 instances for testing.

7.4.3 Malware detection data set

The malware detection data set represents the important problem of continuously classifying received files into malware (e.g. viruses, spyware, trojans, phishing, spam, etc.) or goodware to ensure that users are protected against malicious code. This data set has been provided by Panda Security company and consists of 40000 records. It contains 5398 features and a single binary class variable taking either the malware or goodware value. Due to the confidentiality of the data, we omit the detailed description of the features.

Contrary to experiments with the previous data, we do not know whether or not changes occur in this real data set; and if so, we do not know when and which kind of changes occur. Moreover, we do not fix the percentage of labeled data in each block. Instead, we use all the available labeled data, the number may vary from one data block to another.

We also deal with two additional issues to process this malware detection data set. The first is *feature selection*, which aims to select a small subset of relevant features in order to avoid features dependency and redundancy and enhance classifier performance. In this work, we use the conditional mutual information maximization criterion (CMIM) [75]. It iteratively picks features that maximize their mutual information with the class variable, conditionally upon the response of the already picked features. In this way, CMIM ensures weak dependency and no redundancy as it does not select a feature similar to any that have already been picked even if it is individually powerful.

In our case, feature selection is applied each time we learn a new classifier, i.e., each time we detect changes. Hence, a new and more informative subset of features is selected given new incoming data. In fact, some old selected features may be removed and new different features may be selected. This, consequently, allows us to build more efficient classifiers.

The second issue is *imbalanced data* since the number of malware instances is much higher than goodware instances. This leads to an important problem since the learned classifier may be biased towards the malware class, and therefore its predictive accuracy may be very poor on the goodware class. We apply two recent approaches to balance the class distribution:

- The clustering-sampling approach proposed by Wang et al. [228] makes use of the k-

means clustering algorithm to select negative instances for representing the negative class (i.e., malware class in our case). Firstly, the number of clusters nc is set to the size of positive instances (i.e., goodware instances). Then, the negative instances are clustered into nc clusters and the centroid of each cluster is used as as negative instance for representing the negative class.

- The selectively recursive approach (SERA) proposed by Chen and He [39] makes use of the previous data blocks knowledge to balance the current data block. In fact, it consistently collects the positive instances from the previous data blocks. Then, it applies the Mahalanobis distance to measure the similarity between each instance and the current positive instances, and includes a subset of the most similar previous positive instances of a size proportional to the size of the current negative set only. This is justified by the fact that only the previous positive instances not including the drifting concepts are actually helpful for the learning process.

The malware detection data set is divided into sets of blocks of size 4000, and from each block, the first 2000 instances are used for training while the remaining instances are used for testing. For feature selection, we select 50 of the 5398 features.

To summarize, the details of the three considered data sets are given in Table 7.1. Note finally that, for bootstrap parameters, we use the significance level $\alpha = 0.05$ and samples number $k = 500$ in all experiments. Our choice is based on Dasu et al.'s work [55] where they prove that the number of samples does not significantly affect the quality of the results and suggest that $k = 500$ is a reasonable number of samples. They also point out that lower α values make the null hypothesis harder to reject, leading to a lower change detectability. According to our experiments, $\alpha = 0.05$ works well and is considered as an appropriate value.

Table 7.1: Data set descriptions.

Data set	Number of features	Total number of instances	Number of blocks	Number of instances in a block
Rotating hyperplane	10	80000	40	2000
Mushroom	22	8124	6	1354
Malware detection	50	40000	10	4000

7.5 Experimental results

7.5.1 Rotating hyperplane data set analysis results

Table 7.2 represents the results for the drift detection proposal. The first column represents the block numbers of the training sets. For instance, $1-2$ denotes that the current data is the training set of the first block, while the new data corresponds to the training set of the second block. Then, in columns 2 and 3, we show the kl_{fc} and $\hat{kl}^{\alpha}_{fc}$ values. These values are the same for all experiments irrespective of the different percentages of labeled data, since they only use the feature values. Finally, columns 4 to 9 report kl_{cc} and $\hat{kl}^{\alpha}_{cc}$ respectively, for 2%, 5% and 10% of labeled data.

As expected, a feature change is only detected between blocks 10 and 11 where the magnitude of change t goes from 0.1 to 0.2, blocks 20 and 21 where t goes from 0.2 to 0.5, and blocks 30 and 31 where the t goes from 0.5 to 1. The larger the modification of t values, the higher the kl_{fc} values are, showing a more significant drift in the feature distributions between the data blocks.

The same applies to the conditional distributions monitored by kl_{cc} values for both 5% and 10% of labeled data, where higher kl_{cc} values are obtained for higher t values. However, in the case of 2% of labeled data, no conditional changes are detected. This can be explained by the fact that the true conditional distribution cannot be accurately approximated with very few labeled instances. In their experiments studying the effect of window size on the performance of the change detection scheme, Dasu et al. [55] come to the same conclusion, i.e., a larger window size gives better approximation of the true underlying distribution and results in a better detection of changes.

Furthermore, Figure 7.1 presents accuracy curves for NB and LR. For each curve, the X-axis represents the block number, and the Y-axis represents the classification accuracy. Obviously, the performance of both NB and LR is much better when higher percentages of labeled data are considered. Note also that in this data set LR always outperforms NB, which is mainly due to the small percentages of labeled data. In fact, as also pointed out in [9], the presence of only few labeled data may lead to poor estimates of the generative approach.

7.5.2 Mushroom data set analysis results

According to the results in Table 7.3, feature changes are only detected between blocks 1 and 2, and blocks 4 and 5. However, no conditional changes are detected for any of the percentages of labeled data as expected. This proves that our detection method is resilient to false alarms.

Table 7.2: Drift detection results for rotating hyperplane data set.

	Feature change		Conditional change					
			2% labeled		5% labeled		10% labeled	
blocks	kl_{fc}	$\hat{kl}^{\alpha}_{fc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$
$1-2$	0.0962	0.1386	0.4807	3.1025	0.1168	0.5257	0.0222	0.4989
$2-3$	0.1353	0.1801	0.8112	3.9845	0.0862	1.7670	0.0514	0.6499
$3-4$	0.1214	0.1364	0.0637	5.7233	0.3874	0.1958	0.1013	0.7847
$4-5$	0.1245	0.1381	0.0158	6.1474	0.4248	1.4025	0.2139	0.5257
$5-6$	0.1069	0.1404	0.3166	2.6613	0.2985	1.4885	0.3100	0.7380
$6-7$	0.1008	0.1378	1.4039	6.9068	0.7706	1.3782	0.5962	0.8997
$7-8$	0.1013	0.1381	1.2359	5.6246	0.6927	1.3755	0.1665	0.3234
$8-9$	0.1304	0.1388	0.9124	6.2563	0.5369	1.6424	0.1990	0.2339
$9-10$	0.1017	0.1398	1.2339	4.2875	0.2925	1.3847	0.2180	0.6060
$10-11$	**0.1434**	**0.1405**	3.2875	6.4493	**1.3862**	**1.2369**	**0.9812**	**0.6499**
$11-12$	0.1249	0.1402	1.2026	6.2875	0.2534	1.7670	0.1355	0.5428
$12-13$	0.1154	0.1390	1.9967	7.3604	0.4209	1.5021	0.1544	0.3575
$13-14$	0.0882	0.1398	1.8104	8.2256	1.0638	1.7860	0.1419	0.7648
$14-15$	0.0956	0.1378	2.4464	6.4493	0.5212	1.3369	0.2552	0.6499
$15-16$	0.1344	0.1369	2.3008	6.3567	0.8237	1.6660	0.1580	0.6694
$16-17$	0.1373	0.1866	0.7418	6.2875	1.6932	1.8060	0.1862	0.3013
$17-18$	0.1374	0.1444	1.0548	5.1297	0.7915	1.9310	0.1273	0.6499
$18-19$	0.1316	0.1400	2.3245	6.5493	1.3169	1.8142	0.1355	0.5005
$19-20$	0.0932	0.1392	1.4075	6.3649	0.5472	1.3725	0.3919	0.7842
$20-21$	**0.1544**	**0.1408**	4.1283	6.4512	**1.3821**	**1.2364**	**1.0118**	**0.8162**
$21-22$	0.1378	0.1538	0.0637	6.4502	0.4669	1.7670	0.0456	0.5257
$22-23$	0.1141	0.1387	0.9158	5.8614	0.3925	1.6927	0.1153	0.2978
$23-24$	0.1296	0.1366	1.4257	6.5915	0.4248	1.2849	0.1030	0.1375
$24-25$	0.0851	0.1370	1.1079	6.1807	0.3472	0.5288	0.0168	0.2311
$25-26$	0.0849	0.1382	1.0042	6.4346	0.2812	1.2745	0.0953	0.2110
$26-27$	0.0653	0.1342	1.4721	6.3684	0.5004	1.3369	0.3651	0.5147
$27-28$	0.0880	0.1373	1.2339	6.6503	0.5257	1.1575	0.2120	0.6499
$28-29$	0.1139	0.1434	1.9099	6.5288	0.6927	1.7670	0.3275	0.5005
$29-30$	0.1383	0.1521	1.7233	6.8027	0.0818	1.7495	0.1094	0.4257
$30-31$	**0.1767**	**0.1466**	4.7233	6.4346	**1.9310**	**1.3660**	**1.3369**	**0.7648**
$31-32$	0.1011	0.1373	1.4792	6.2875	0.1613	0.2534	0.0375	0.3409
$32-33$	0.1332	0.1394	0.6748	6.7841	0.4838	1.7897	0.1978	0.4885
$33-34$	0.1360	0.1565	0.9099	5.1964	0.6927	1.0546	0.3248	0.7648
$34-35$	0.1171	0.1395	0.8475	3.5168	0.4354	0.4999	0.1279	0.5694
$35-36$	0.0951	0.1376	1.2339	6.4593	0.9211	1.3369	0.1947	0.6348
$36-37$	0.0957	0.1395	0.9213	4.8143	0.3234	1.7495	0.1456	0.5257
$37-38$	0.1056	0.1367	0.6014	6.3684	0.2648	0.6364	0.1898	0.2339
$38-39$	0.1264	0.1625	0.9078	6.3125	0.1763	0.4376	0.1504	0.4342
$39-40$	0.1378	0.1367	0.3478	5.6177	0.4517	1.6849	0.3973	0.5389

Moreover, according to Figure 7.2, using more labeled data improves the predictive accuracies of both NB and LR. Nevertheless, the improvement is negligible for LR from 5% to

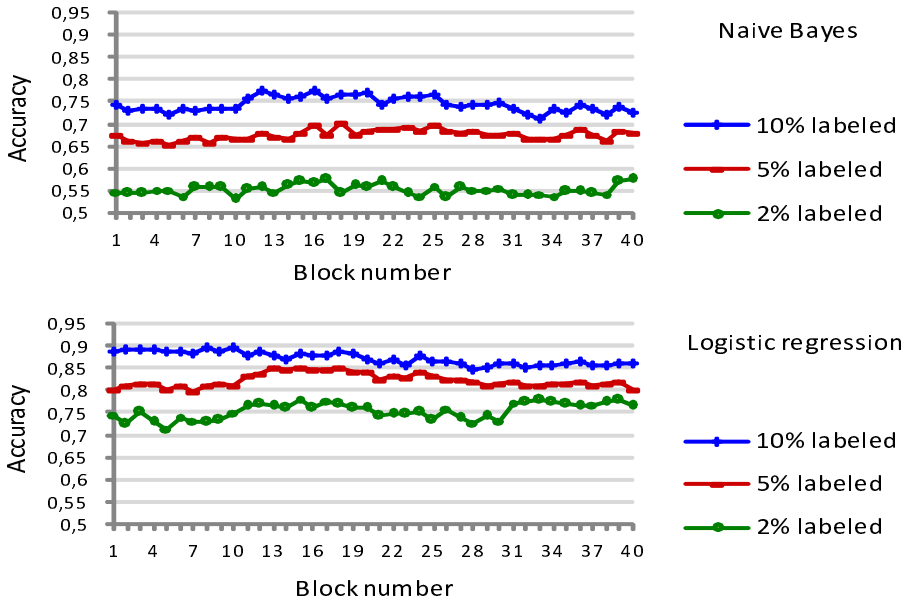


Figure 7.1: Classification results for rotating hyperplane data set.

Table 7.3: Drift detection results for mushroom data set.

	Feature change		Conditional change					
			2% labeled		5% labeled		10% labeled	
blocks	kl_{fc}	$\hat{kl}^{\alpha}_{fc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$
$1-2$	**0.2251**	**0.0450**	0.0003	0.1430	0.0001	0.0035	0.0079	0.0176
$2-3$	0.0365	0.0755	0.0019	0.0156	0.0005	0.0030	0.0003	0.0009
$3-4$	0.1184	0.1536	0.0031	0.0049	0.0054	0.0057	0.0002	0.0181
$4-5$	**1.2973**	**0.2373**	0.0043	0.1347	0.0002	0.0063	0.0013	0.0030
$5-6$	0.0007	0.0814	0.0028	0.0105	0.0018	0.0092	0.0001	0.0053

10% of labeled data and the corresponding curves are almost superimposed. Notice also that LR always outperforms NB and has a more stable behavior especially when more labeled data are used.

7.5.3 Malware detection data set analysis results

Table 7.4 presents drift detection results for the malware detection data set. The first column reports as previously the block numbers, while the second column represents the percentage of labeled instances in each considered block. Then, columns 3 to 8 show respectively kl_{fc}, $\hat{kl}^{\alpha}_{fc}$, kl_{cc} and $\hat{kl}^{\alpha}_{cc}$ values. We observe that feature and conditional changes occur together

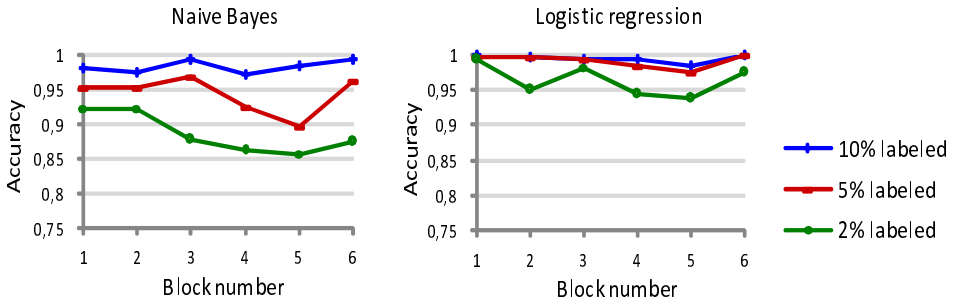


Figure 7.2: Classification results for mushroom data set.

and are detected between blocks 3 and 4, and again between blocks 8 and 9.

Table 7.4: Drift detection results for malware detection data set.

blocks	% labeled instances	kl_{fc}	$\hat{kl}^{\alpha}_{fc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$
$1-2$	$70.00-80.45$	0.1304	0.2049	0.0797	0.9180
$2-3$	$80.45-69.30$	0.1057	0.1553	0.1609	0.2062
$3-4$	$69.30-67.10$	**0.3892**	**0.1408**	**0.4330**	**0.2717**
$4-5$	$67.10-78.35$	0.1272	0.1939	0.2975	0.3473
$5-6$	$78.35-71.25$	0.1095	0.3717	0.0190	0.1652
$6-7$	$71.25-74.85$	0.0665	0.2760	0.5373	0.6614
$7-8$	$74.85-76.05$	0.0967	0.1434	0.5028	0.5484
$8-9$	$76.05-83.80$	**1.0006**	**0.2445**	**0.7558**	**0.6033**
$9-10$	$83.80-78.85$	0.1221	0.9668	0.3390	0.4493

To evaluate classifier performance, we previously used only the overall classification accuracy. However, when dealing with imbalanced data sets, this metric is often insufficient, as it does not distinguish between the number of correctly classified instances of different classes.

Using balancing methods mainly aims to improve the classifier performance over the positive class, i.e., reduce the number of false positives. In order to appropriately monitor the behavior of NB and LR classifiers on the positive class in this case, then, we also calculate the precision, recall, F1 and G-mean metrics based on the confusion matrix analysis.

The results of the NB and LR classifiers for the first balancing approach SERA are described in Table 7.5. We observe that, in most cases, LR accuracies are slightly higher than for NB.

Furthermore, both NB and LR provide high precision values for all testing blocks, where all values are greater than 95%, and yield good results in terms of F1 and G-mean values, which is indicative of a good performance predicting the positive instances.

Table 7.5: Classification results for malware detection data set using SERA balancing approach.

blocks	Algo.	Accuracy	Recall	Precision	F1	G-mean
1	NB	0.7063	0.6392	0.9792	0.7735	0.7795
	LR	0.7211	0.6483	0.9942	0.7848	0.7996
2	NB	0.7995	0.7701	0.9613	0.8551	0.8312
	LR	0.7212	0.6696	0.9537	0.7868	0.7729
3	NB	0.7540	0.7354	0.9838	0.8417	0.8148
	LR	0.7951	0.7788	0.9883	0.8711	0.8491
4	NB	0.7970	0.7375	0.9656	0.8363	0.8316
	LR	0.8328	0.7846	0.9722	0.8684	0.8620
5	NB	0.7668	0.7330	0.9818	0.8393	0.8270
	LR	0.7787	0.7409	0.9904	0.8477	0.8455
6	NB	0.7462	0.7289	0.9781	0.8353	0.7995
	LR	0.8051	0.7906	0.9858	0.8775	0.8503
7	NB	0.7488	0.7019	0.9890	0.8211	0.8227
	LR	0.7576	0.7111	0.9911	0.8281	0.8308
8	NB	0.8385	0.8334	0.9993	0.9089	0.9058
	LR	0.8006	0.7964	0.9966	0.8853	0.8568
9	NB	0.7625	0.7551	0.9746	0.8509	0.7907
	LR	0.7339	0.7182	0.9799	0.8289	0.7909
10	NB	0.8044	0.7878	1.0000	0.8813	0.8876
	LR	0.8011	0.7884	0.9947	0.8797	0.8658

For the clustering-sampling balancing approach, as shown in Table 7.6, LR outperforms NB, except on the last two testing sets, where NB shows better accuracies, as well as better recall and F1 values.

Finally, note that the results of the two applied balancing approaches are comparable, with a slightly better performance of the clustering-sampling approach in terms of overall accuracy, recall and F1 metrics. In most cases, though, SERA provides slightly better precision with both NB and LR classifiers.

7.6 Conclusion

We have addressed through this chapter a more realistic and important problem in uni-dimensional data stream mining that most existing research has failed to deal with, assuming data streams to be entirely labeled.

In our proposed `CPL-DS` approach, both labeled and unlabeled instances are used to not only assert the presence or absence of drift, but also to efficiently determine which kind of drift has occurred –feature, conditional or dual– by means of Kullback-Leibler divergence

Table 7.6: Classification results for malware detection data set using clustering-sampling balancing approach.

blocks	Algo.	Accuracy	Recall	Precision	F1	G-mean
1	NB	0.6992	0.6279	0.9823	0.7661	0.7759
	LR	0.7087	0.6347	0.9906	0.7737	0.7879
2	NB	0.7546	0.7226	0.9447	0.8188	0.7884
	LR	0.8455	0.8344	0.9592	0.8925	0.8581
3	NB	0.7793	0.7696	0.9774	0.8611	0.8122
	LR	0.8381	0.8314	0.9840	0.9013	0.8609
4	NB	0.7508	0.7042	0.9231	0.7989	0.7787
	LR	0.8078	0.7745	0.9418	0.8500	0.8287
5	NB	0.7936	0.7645	0.9834	0.8602	0.8462
	LR	0.8590	0.8389	0.9899	0.9082	0.8964
6	NB	0.8014	0.7949	0.9756	0.8760	0.8221
	LR	0.8753	0.7837	0.9832	0.9252	0.8807
7	NB	0.8403	0.8240	0.9781	0.8945	0.8684
	LR	0.9359	0.9304	0.9909	0.9597	0.9456
8	NB	0.8795	0.8966	0.9769	0.9350	0.5918
	LR	0.9195	0.9238	0.9924	0.9569	0.8580
9	NB	0.8457	0.8735	0.9506	0.9104	0.7253
	LR	0.8264	0.8435	0.9582	0.8972	0.7559
10	NB	0.9038	0.9142	0.9802	0.9460	0.8453
	LR	0.8648	0.8623	0.9897	0.9217	0.8782

and a bootstrapping method. Then, if required, the classifier is updated using the EM algorithm. Experimental results carried out on both synthetic and real-world data sets, with naive Bayes and logistic regression classifiers, showed that our approach is effective for detecting different kinds of changes from partially labeled data, as well as having a good classification performance.

In the future, it would be interesting to investigate and compare the performance of other classifiers with our results. Furthermore, note that in this work we assume that labeled and unlabeled data come from the same distribution. This usually leads to a better classification accuracy. An interesting future line of research would be to consider the scenario where labeled and unlabeled data possibly come from different distributions, inspect the impact of unlabeled data, and study the possibility of refining the change detection proposal.

Chapter 8

Mining multi-dimensional data streams using MBCs

8.1 Introduction

Most of the work within the field of mining data streams has focused on *uni-dimensional data streams* where only a single output class variable is available. As reviewed in Chapter 4, a plethora of approaches have been proposed to deal with this problem, and several issues have been intensively addressed including, among many others, concept drift detection and model adaptation [1, 86, 87].

Nevertheless, the recent problem of mining *multi-dimensional data streams* has received less attention, and only few works have been proposed [131, 178, 181, 233]. Contrary to uni-dimensional data streams, multiple output class variables are available in multi-dimensional data streams, i.e., each input instance in the stream has to be simultaneously associated with more than one class variable.

In this chapter, we present new adaptive informed methods for mining multi-dimensional data streams based on multi-dimensional Bayesian network classifiers (MBCs). The so-called `Globally Adaptive-MB-MBC` (`GA-MB-MBC`) and `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) extend our previous `MB-MBC` algorithm, presented in Chapter 6, to cope with concept-drifting evolving data streams. Basically, both `GA-MB-MBC` and `LA-MB-MBC` monitor the concept drift over time using the Page-Hinkley test. Then, if a drift is detected, `LA-MB-MBC` adapts the current MBC network locally around each changed node, whereas `GA-MB-MBC` learns a new MBC network from scratch.

This work can be also considered as an extension of our `CPL-DS` method presented in

Chapter 7. CPL-DS is uni-dimensional, i.e., it can be only applied with a single class variable, and its main objective is to take advantage of the unlabeled instances for both monitoring concept drifts and updating the current classification model. Basically, its drift detection method is based on using the Kullback-Leibler divergence to measure distribution differences between old and recent data stream batches. Then, based on a bootstrapping method, it determines whether the Kullback-Leibler measures are statistically significant or not, to decide accordingly whether a drift has occurred or not.

The CPL-DS change detection method can be easily extended to multi-dimensional setting; however, its main shortcoming is that it does not permit a local detection of the drift, and thus, if any drift is detected, a new MBC classifier should be learned from scratch from the recent data. To overcome this shortcoming, we opt in this work to propose two different detection methods from two different points of view: global and local, and study their behaviors under different settings. Similar to CPL-DS, a concept drift signaled by the global detection method GA-MB-MBC leads to learn the MBC network from scratch. Nevertheless, a concept drift signaled by the local adaptive method LA-MB-MBC only updates the local structures of the detected changed nodes.

Note finally that, with CPL-DS we differentiated between feature, conditional or dual drift, and this was basically motivated by the presence of unlabeled instances that only incorporate feature drifts. However, in this chapter, since all the available instances are assumed to be fully labeled, we do not distinguish between different types of drifts. We rather focus on the local and global aspects of the concept drifts.
This chapter is based on the submitted paper [27].

Chapter outline

Section 8.2 defines the multi-dimensional concept drift problem, and Section 8.3 briefly reviews the related work on mining multi-dimensional data streams. Next, Section 8.4 introduces the proposed change detection method as well as the local and global MBC adaptation methods. Sections 8.5 and 8.6 cover the experimental study presenting the used data, the evaluation metrics, and a discussion on the obtained results. Finally, Section 8.7 rounds the chapter off with some conclusions.

8.2 Multi-dimensional concept drift

The same categorization of uni-dimensional concept drift into feature, conditional and dual changes (presented in Section 4.2) can be applied in the context of multi-dimensional data

streams. In fact, the feature change involving only a change in $P(\mathbf{x})$ is exactly the same; whereas, for the conditional change, we have now a vector of d class variables $\mathbf{C} = (C_1, \ldots, C_d)$ instead of a single class variable C, i.e., the conditional change may occur in the distribution $P(\mathbf{c} \mid \mathbf{x})$. Moreover, as previously, the change is called dual when both feature and conditional changes occur together. As we commented above, in this work, we do not take this categorization into account and we consider that the presence of any kind of change requires the adaption of the current MBC network.

Furthermore, similar to the uni-dimensional concept drift, the multi-dimensional concept drift can be categorized into *abrupt* or *gradual* depending on the rate of change, and into *local* or *global* depending on whether it occurs in some regions of the instance space or in the whole instance space, respectively.

Consequently, the main differences between the uni-dimensional and the multi-dimensional concept drifts consist mainly in the changes that may occur in the distribution and the dependence relationships between the class variables, as well as the distribution and the dependence relationships between each class variable and the set of feature variables.

Besides these categorizations, and in the context of multi-label streaming classification, Read et al. [181] discuss that concept drift may also involve a change in the *label cardinality*, that is, a change in the average number of labels associated with each instance computed as $LCard = \frac{1}{N}\sum_{l=1}^{N}\sum_{j=1}^{d} c_j^{(l)}$ with $c_j^{(l)} \in \{0, 1\}$, where N denotes the total number of instances and d the number of labels (or binary class variables).

In addition, Xioufis et al. [233] consider that a multi-label data stream contains separate multiple targets (concepts) and each concept is likely to exhibit independently its own drift pattern. This assumption allows to track the drift of each concept separately using binary relevance and pre-defined uni-dimensional detection approaches; however, its main drawback is its inability to deal with the correlations that concepts may have with each other and which may drift over time.

8.3 Related work

In this section, we briefly review the existing related works. They have been all developed under the streaming multi-label classification setting, and they can be considered as extension of stationary multi-label methods (see Section 3.3) to concept-drifting data streams.

Qu et al. [178] propose an ensemble of improved binary relevance (`MBR`) taking into account the dependency among labels. The basic idea is to add each classified label vector as a new feature participating in the classification of the other related labels. Moreover, in order to

cope with concept drifts, Qu et al. use a dynamic classifier ensemble jointly with a weighted majority voting strategy. No drift detection method is employed in `MBR`. In fact, the ensemble keeps a fixed number K of base classifiers, and is updated continuously over time by adding new classifiers, trained on the recent data blocks, and discarding the oldest ones. Naive Bayes, C4.5 decision tree algorithm, and support vector machines (SVM) are used as different base classifiers to test the `MBR` method.

Xioufis et al. [233] tackle a special problem when dealing with multi-label data streams, namely *class imbalance*, i.e., the skewness in the distribution of positive and negative instances for all or some labels. In fact, each label in the stream may have more negative than positive instances, and some labels may have much more positive instances than others. To deal with this problem, the authors propose a multiple windows classifier (`MWC`) that maintains two windows of fixed size for each label: one for positive instances and one for negative ones. The size N_p of the positive windows is a parameter of the approach and the size N_n of the negative windows is determined using the formula $N_n = \frac{N_p}{r}$, where r is another parameter of the approach, called distribution ratio. r has the role of balancing the distribution of positive and negative instances in the union of the two windows. The authors assume an independent concept drift for each label, and use a binary relevance method with k-nearest neighbors (kNN) as base classifier. No drift detection method is employed in `MWC`. Positive and negative windows of each label are updated continuously over time by including new incoming instances and removing older ones.

Moreover, Kong and Yu [131] propose also an ensemble-based method for multi-label stream classification. The idea is to use an ensemble of multiple random decision trees [242] where tree nodes are built by means of random selected testing variables and cutting values. The so-called streaming multi-label random trees (`SMART`) does not include a change detection method. In fact, to handle concept drifts in the stream, the authors simply use a fading function on each tree node to gradually reduce the influence of historical data over time. The fading function consists of assigning to each old instance with time stamp t_i a weight $W(t) = 2^{-(t-t_i)/\lambda}$, where t is the current time, and λ is a parameter of the approach, called fading factor, indicating the speed of the fading effects. The higher the value of λ, the slower the weight of each instance will decay.

Finally, Read et al. [181] present a framework for generating synthetic multi-label data streams along with a novel multi-label streaming classification ensemble method based on Hoeffding trees. Their method, named $\texttt{EaHT}_{PS}$, extends the single-label incremental Hoeffding tree (HT) classifier [67] by using a multi-label definition of entropy and by training multi-label pruned sets (PS) at each leaf node of the tree. To handle concept drifts in the stream,

they use the ADWIN Bagging method [16] which consists of an online bagging method [165] extended with ADWIN algorithm as a change detector. When a concept drift is detected, the worst performing classifier of the ensemble of classifiers is replaced with a new classifier.

In addition, Read et al. introduce `BRa`, `EaBR`, `EaPS`, `HTa` methods, that extend respectively binary relevance (`BR`) [98], ensembles of BR (`EBR`) [183], ensembles of PS (`EPS`) [182], and multi-label Hoeffding trees (`HT`) [49, 181] methods by including the ADWIN algorithm (denoted by the letter `a`) to detect concept drifts.

Table 8.1 summarizes the existing streaming multi-label algorithms in the literature.

Table 8.1: Summary of streaming multi-label classification methods.

Reference	**Method**	**Base classifier**	**Adaptation strategy**
Qu et al. [178]	Ensemble of improved binary relevance (`MBR`)	Naive Bayes, C4.5, SVM	Evolving ensemble. No detection
Xioufis et al. [233]	Multiple windows classifier (`MWC`)	kNN	Two windows of fixed size for each label. No detection
Kong and Yu [131]	Streaming multi-label random trees (`SMART`)	Random tree	Fading function. No detection
Read et al. [181]	Ensemble of Multi-label Hoeffding trees with PS at the leaves ($\texttt{EaHT}_{PS}$), as well as `BRa`, `EaBR`, `EaPS`, and `HTa` methods	Hoeffding tree	Evolving ensemble. Detection using the ADWIN algorithm

Contrary to existing approaches, which are all based on a multi-label setting, requiring all the class variables to be binary, our proposed adaptive methods have no constraints on the cardinalities of the class variables.

Moreover, as commented above, the existing streaming multi-label approaches either do not present any drift detection method (for instance, `MBR` [178], `MWC` [233] and `SMART` [131] approaches) or they use a drift detection method and keep updating an ensemble of classifiers over time by replacing the worst performing classifier with a new one when a drift is detected (such as the ensemble of multi-label Hoeffding trees $\texttt{EaHT}_{PS}$ [181] using ADWIN algorithm as a change detector). In both cases, the concept drift cannot be detected locally, and the adaptation process is basically based on ensemble updating. In our case, we only use a single model (i.e., MBC) and our proposed drift detection method can be either performed locally or globally: it is based on monitoring either the average local log-likelihood of each node or the average global log-likelihood of the whole MBC network using the Page-Hinkley test.

Being based on MBCs, our adaptive methods also present the merit of explicitly modeling

the dependence relationships among all variables through the graphical structure component.

8.4 `Adaptive-MB-MBC` methods

Before providing more details about the proposed approach, let us introduce the following notations. Let $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^s, \ldots\}$ denote a multi-dimensional data stream that arrives over time in batches, such that $\mathcal{D}^s = \{(\mathbf{x}^{(1)}, \mathbf{c}^{(1)}), \ldots, (\mathbf{x}^{(N^s)}, \mathbf{c}^{(N^s)})\}$ denotes the multi-dimensional batch stream received at step s, and containing N^s instances. For each instance in the stream, the input vector $\mathbf{x} = (x_1, \ldots, x_m)$ of m feature values is associated with an output vector $\mathbf{c} = (c_1, \ldots, c_d)$ of d class values. For the sake of simplicity, and regardless of being class or feature variable, we denote by V_i each variable in the MBC, $i = 1, \ldots, n$, such that n represents the total number of variables, i.e., $n = d + m$. Given an MBC learned from $\mathcal{D}^s$, denoted MBC^s, and a new coming batch stream $\mathcal{D}^{s+1}$, the adaptive learning problem consists first of detecting possible concept drifts, then, if required, updating the current MBC^s to best fit the new distribution of $\mathcal{D}^{s+1}$.

In what follows, we start by presenting the proposed drift detection method in Section 8.4.1, next we introduce the MBC adaptation methods in Section 8.4.2.

8.4.1 Drift detection method

The objective here is to continuously process the batches of data streams and detect the concept drift when it occurs. Our proposed detection method is based on the average log-likelihood score and the Page-Hinkley test, and can be applied either globally, i.e., to the whole MBC network, or locally, i.e., to each variable in the MBC network.

8.4.1.1 The average log-likelihood score

The likelihood measures the probability of a data set $\mathcal{D}^s$ given the current multi-dimensional Bayesian network classifier. For convenience in the calculations, the logarithm of the likelihood $LL(D^s \mid \boldsymbol{\theta}^s)$, simply denoted here LL^s, is usually used:

$$\begin{aligned} LL^s &= log\ P(D^s \mid \boldsymbol{\theta}^s) \\ &= log \prod_{l=1}^{N^s} \prod_{i=1}^{n} P(v_i^{(l)} \mid \mathbf{pa}(v_i)^{(l)}, \boldsymbol{\theta}^s) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} log(\theta_{ijk}^s)^{N_{ijk}^s} \end{aligned} \tag{8.1}$$

where $v_i^{(l)}, \mathbf{pa}(v_i)^{(l)}$ are respectively the values of variable V_i and its parent set $\mathbf{Pa}(V_i)$ in the l^{th} instance in $\mathcal{D}^s$. r_i denotes the number of possible states of V_i, and q_i denotes the number of possible configurations that the parent set $\mathbf{Pa}(V_i)$ can take. N_{ijk}^s is the number of instances in $\mathcal{D}^s$ where variable V_i takes its k^{th} value and $\mathbf{Pa}(V_i)$ takes its j^{th} configuration.

In this work, we use the average log-likelihood score per instance, which is equal to the original log-likelihood score LL^s divided by the total number of instances N^s. This in fact will allow us to compare the likelihood of an MBC network based on different batch streams that may present different numbers of instances. Hence, using the maximum likelihood estimation for the parameters $\hat{\theta}^s$, *the average global log-likelihood* can be expressed as follows:

$$\begin{aligned} \overline{LL}^s &= \frac{1}{N^s} \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}^s log \frac{N_{ijk}^s}{N_{ij}^s} \\ &= \sum_{i=1}^{n} \frac{1}{N^s} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}^s log \frac{N_{ijk}^s}{N_{ij}^s} \\ &= \sum_{i=1}^{n} ll_i^s \end{aligned} \tag{8.2}$$

where $\overline{ll}_i^s = \frac{1}{N^s} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}^s log \frac{N_{ijk}^s}{N_{ij}^s}$ is *the average local log-likelihood* of each variable V_i in the MBC network, and $N_{ij}^s = \sum_{k=1}^{r_i} N_{ijk}^s$ for every i, j and k.

8.4.1.2 Change point detection

In order to detect the change point, i.e., the point at which the concept drift occurs, we make use of the Page-Hinkley (PH) test [109, 166]. The PH test is a sequential analysis technique commonly used for change detection in signal processing, and is considered to be one of the most appropriated algorithms to detect concept drifts in data streams [201].

In this work, we apply the PH test in order to determine whether a sequence of average log-likelihood values can be attributed to a single statistical law (null hypothesis); or it demonstrates a change in the statistical law underlying these values (change point). Let $\overline{LL}^1, ..., \overline{LL}^s$ denote the global average log-likelihood values computed with Equation (8.2) using the first batch stream $\mathcal{D}^1$ till the last received one $\mathcal{D}^s$, respectively. To test the above hypothesis, the PH test considers first a cumulative variable CUM^s, defined as the *cumulated difference* between the obtained average log-likelihood values and their mean till the current moment (i.e., the last batch $\mathcal{D}^s$):

$$CUM^s = \sum_{t=1}^{s} (\overline{LL}^t - mean_{\overline{LL}^t} - \delta) \tag{8.3}$$

where $mean_{\overline{LL}^t} = \frac{1}{t}\sum_{h=1}^{t} \overline{LL}^h$ denotes the mean of $\overline{LL}^1, ..., \overline{LL}^s$ values, and δ is a *tolerance parameter* corresponding to the magnitude of changes which are allowed. The maximum value MAX^s of variable CUM^t for $t = 1, ..., s$, is then computed:

$$MAX^s = max\Big\{CUM^t, t = 1, ..., s\Big\} \tag{8.4}$$

Next, the PH value is computed as the difference between MAX^s and CUM^s:

$$PH^s = MAX^s - CUM^s \tag{8.5}$$

When this difference is greater than a given threshold λ (i.e., $PH^s > \lambda$), the null hypothesis is rejected and the PH test alarms a change, otherwise, no change is signaled. Specifically, depending on the result of this test, two states can be distinguished:

- If $PH^s \leq \lambda$ then there is no concept drift: the distribution of the average log-likelihood values is stable. The new batch $\mathcal{D}^s$ is deemed to come from the same distribution as the previous instances. In this case, the MBC network is left unchanged.
- If $PH^s > \lambda$ then a concept drift occurred: the distribution of the average log-likelihood values is drifting. The new batch $\mathcal{D}^s$ is deemed to come from a different distribution than the previous instances, and in this case, the MBC network should be adapted.

Note that, the threshold λ is a parameter allowing to control the rate of false alarms. In general, small λ values may increase the number of false alarms, whereas higher λ values may lead to a fewer false alarms but may rise at the same time the risk of missing some concept drifts.

Similarly, the PH test can also be applied to the local average log-likelihood values $\overline{ll}_i^s$ of each variable V_i in the network, instead of the global average log-likelihood $\overline{LL}^s$ of the whole MBC network. In this case, each local PH test value, denoted PH_i^s, allows us to check if a drift occurs or not at each considered variable V_i. This in fact will locally specify where (i.e., for which set of variables) the concept drift occurs. Afterwards, the challenge is to locally update the MBC structure, i.e., update only the parts that are in conflict with the new data stream $\mathcal{D}^s$ without re-learning the whole MBC from scratch.

8.4.2 MBC adaptation

As discussed above, if a new batch stream comes from the same distribution as the previous one, then no change should be detected by the PH test and the current MBC structure is maintained. Otherwise, we conclude that the new batch stream has a different distribution, and the current MBC network may be updated in two different possible ways depending respectively on whether the drift has been detected globally or locally:

- Global MBC adaptation: In this case, if a change is detected, then the whole MBC^s network is learned from scratch.
- Local MBC adaptation: In this case, if a change is detected, then only the changed parts of the current MBC^s network are locally adapted.

8.4.2.1 Global MBC Adaptation

The proposed method for global MBC adaptation is outlined in Algorithm 8.1. `Globally Adaptive-MB-MBC` takes as input the current network MBC^s, the new data stream D^{s+1}, and the PH test parameters δ and λ. It starts by computing the average global log-likelihood $\overline{LL}^{s+1}$ (step 2), and the PH test value PH^{s+1} (step 3). Next, if PH^{s+1} is higher than λ, then a new network MBC^{s+1} is learned from D^{s+1} using the `MB-MBC` algorithm [24] (step 5). Otherwise, i.e., $PH^s \leq \lambda$, the MBC network is kept unchanged (step 7).

Algorithm 8.1: `Globally Adaptive-MB-MBC`

1. *Input*: Current MBC^s, new multi-dimensional data stream D^{s+1}, δ, λ
2. Compute the global average log-likelihood $\overline{LL}^{s+1}$
3. Compute PH^{s+1}
4. **if** $PH^{s+1} > \lambda$ **then**
5. Learn a new network MBC^{s+1} from D^{s+1} using the `MB-MBC` algorithm.
6. **else**
7. $MBC^{s+1} \leftarrow MBC^s$, i.e., no change is detected
8. **end if**
9. **return** MBC^{s+1}

8.4.2.2 Local MBC Adaptation

Contrary to the global adaptation, the objective in this case is to locally update the MBC network over time, so that if a concept drift occurs, only the changed parts in the current

MBC are re-learned from the new data stream and not the whole network. This in fact presents two main challenges: First, how to locally detect the changes, and second how to update the current MBC.

To deal with these challenges, we propose the `Locally Adaptive-MB-MBC` method, outlined by Algorithm 8.2. Given the current network MBC^s, the new data stream D^{s+1}, and the PH test parameters δ and λ, the local change detection firstly computes the average log-likelihood $\overline{ll}_i^{s+1}$ of each variable V_i using the new data stream D^{s+1} (step 4), then computes the corresponding value PH_i^{s+1} (step 5). Next, if this PH_i^{s+1} value is higher than λ, then variable V_i is added to the set of changed nodes (steps 6 to 8). Subsequently, whenever the resulting set of $ChangedNodes$ is not empty, i.e., a drift is detected, then the $UpdateMBC$ function, outlined by Algorithm 3, is invoked to locally update the current MBC^s network (step 11); otherwise, we conclude that no drift is detected and the MBC network is kept unchanged (step 13).

Algorithm 8.2: `Locally Adaptive-MB-MBC`

1. *Input*: Current MBC^s, new multi-dimensional data stream D^{s+1}, δ, λ
2. $ChangedNodes = \emptyset$
3. **for** every variable V_i **do**
4. Compute the average local log-likelihood $\overline{ll}_i^{s+1}$
5. Compute the local PH test PH_i^{s+1}
6. **if** $PH_i^{s+1} > \lambda$ **then**
7. $ChangedNodes \leftarrow ChangedNodes \cup \{V_i\}$
8. **end if**
9. **end for**
10. **if** $ChangedNodes \neq \emptyset$ **then**
11. $MBC^{s+1} \leftarrow UpdateMBC(ChangedNodes, MBC^s, D^{s+1}, PC^s, MB^s)$
12. **else**
13. $MBC^{s+1} \leftarrow MBC^s$, i.e., no drift is detected
14. **end if**
15. **return** MBC^{s+1}

Before introducing the $UpdateMBC$ algorithm, note that since the local log-likelihood computes the probability of each variable V_i given the set of its parents in the MBC structure, then a detected change for a variable V_i informs that the set of parents of the variable V_i has changed due to either the removal of some existing parents or the inclusion of new parents:

- The removal of an existing parent means that this parent was strongly relevant to V_i given D^s, and becomes either weakly relevant or irrelevant to V_i given D^{s+1}. In other

words, this parent was a member of the parent set, or more broadly a member of the parents-children set of V_i, but with respect to D^{s+1}, it does not pertain to the parents-children set of V_i.

- The inclusion of a new parent means that this parent was either weakly relevant or irrelevant to V_i given D^s, and becomes strongly relevant to V_i given D^{s+1}. In other words, this parent was not a member of the parents-children set of V_i, but with respect to D^{s+1}, it should be added as a new member of the parents-children set of V_i.

Recall that, variables are defined to be *strongly relevant* if they contain information about V_i not found in all other remaining variables. That is, the strongly relevant variables are the members of the Markov blanket of V_i, and thereby, all the members in the parents-children set of V_i are also strongly relevant to V_i. On the other hand, variables are said to be *weakly relevant* if they are informative but redundant, i.e., they consist of all the variables with an undirected path to V_i which are not themselves members of the Markov blanket nor the parents-children set of V_i. Finally, variables are defined as *irrelevant* if they are not informative, and in this case, they consist of variables with no undirected path to V_i [4, 128].

Therefore, the intuition behind $UpdateMBC$ algorithm, outlined by Algorithm 8.3, is basically to firstly learn with D^{s+1} the new parents-children set of each changed node using the HITON-PC algorithm [4, 5], determine the sets of its old and new adjacent nodes, and then locally update the MBC structure.

In fact, $UpdateMBC$ takes as input the set of changed nodes, the current network MBC^s, the new data stream D^{s+1}, the parents-children sets of all variables PC^s, and the Markov blanket sets of all class variables MB^s. For each variable V_i in the set of changed nodes, $UpdateMBC$ firstly learns from D^{s+1} the new parents-children set of V_i $PC(V_i)^{s+1}$ using HITON-PC algorithm (step 3). Then, it determines the set of its old adjacent nodes, i.e., $\left\{PC(V_i)^s \backslash PC(V_i)^{s+1}\right\}$ (step 4). The variables included in this set are variables that pertained to $PC(V_i)^s$ but do not pertain anymore to $PC(V_i)^{s+1}$, which means that they represent the set of variables that were strongly relevant to V_i and have become either weakly relevant or irrelevant to V_i. In this case, for each variable $OldAdj$ belonging to this set, the arc between it and V_i is removed from MBC^{s+1} (step 5), then, the parents-children and Markov blanket sets are updated accordingly. Specifically, the following rules are performed:

- Remove V_i from the parents-children set of $OldAdj$ (step 6): since the arc between V_i and $OldAdj$ was removed, V_i does not pertain anymore to the parents-children set of $OldAdj$.

Algorithm 8.3: $UpdateMBC(ChangedNodes, MBC^s, D^{s+1}, PC^s, MB^s)$

1. *Initialization:* $MBC^{s+1} \leftarrow MBC^s$; $PC^{s+1} \leftarrow PC^s$; $MB^{s+1} \leftarrow MB^s$
2. **for** every variable $V_i \in ChangedNodes$ **do**
3. Learn $PC(V_i)^{s+1} \leftarrow$ HITON-PC(V_i)
   %Determine the set of the old adjacent nodes of the changed node V_i
4. **for** every variable $OldAdj \in \left\{PC(V_i)^s \setminus PC(V_i)^{s+1}\right\}$ **do**
5. Remove the arc between $OldAdj$ and V_i from MBC^{s+1}
6. $PC(OldAdj)^{s+1} \leftarrow PC(OldAdj)^{s+1} \setminus \{V_i\}$
7. **if** $OldAdj \in \mathbf{C}$ **then**
8. $MB(OldAdj)^{s+1} \leftarrow MB(OldAdj)^{s+1} \setminus \left\{V_i \cup \{\mathbf{Pa}(V_i)^{s+1} \setminus PC(OldAdj)^{s+1}\}\right\}$
9. **end if**
10. **if** $V_i \in \mathbf{C}$ **then**
11. $MB(V_i)^{s+1} \leftarrow MB(V_i)^{s+1} \setminus \left\{OldAdj \cup \{\mathbf{Pa}(OldAdj)^{s+1} \setminus PC(V_i)^{s+1}\}\right\}$
12. **end if**
13. **for** every class $H \in \left\{\mathbf{Pa}(V_i)^{s+1} \setminus PC(OldAdj)^{s+1}\right\}$ **do**
14. $MB(H)^{s+1} \leftarrow MB(H)^{s+1} \setminus \{OldAdj\}$
15. **end for**
16. **end for**
    %Determine the set of the old adjacent nodes of the changed node V_i
17. **for** every variable $NewAdj \in \left\{PC(V_i)^{s+1} \setminus PC(V_i)^s\right\}$ **do**
18. Insert an arc from $NewAdj$ to V_i in MBC^{s+1}
19. $PC(NewAdj)^{s+1} \leftarrow PC(NewAdj)^{s+1} \cup \{V_i\}$
20. **if** $NewAdj \in \mathbf{C}$ **then**
21. $MB(NewAdj)^{s+1} \leftarrow MB(NewAdj)^{s+1} \cup \{V_i \cup \mathbf{Pa}(V_i)^{s+1}\}$
22. **end if**
23. **if** $V_i \in \mathbf{C}$ **then**
24. $MB(V_i)^{s+1} \leftarrow MB(V_i)^{s+1} \cup \{NewAdj \cup \mathbf{Pa}(NewAdj)^{s+1}\}$
25. **end if**
26. **for** every class $H \in \left\{\mathbf{Pa}(V_i)^{s+1} \setminus \{NewAdj \cup PC(NewAdj)^{s+1}\}\right\}$ **do**
27. $MB(H)^{s+1} \leftarrow MB(H)^{s+1} \cup \{NewAdj\}$
28. **end for**
29. **end for**
30. **end for**
31. Lean from D^{s+1} new CPTs for nodes that have got a new parent set in MBC^{s+1}
32. **return** MBC^{s+1}; PC^{s+1}; MB^{s+1}

- If the old adjacent node $OldAdj$ is a class variable, then update its Markov blanket $MB(OldAdj)^{s+1}$ by removing from it the changed node V_i and its parents that do not belong to the parents-children set $PC(OldAdj)^{s+1}$ of $OldAdj$ (steps 7 to 9).

- If the changed node V_i is a class variable, then update its Markov blanket $MB(V_i)^{s+1}$ by removing from it the old adjacent node *OldAdj* and its parents that do not belong to the parents-children set of V_i, $PC(V_i)^{s+1}$ (steps 10 to 12).

- Update the Markov blanket of each class variable that belongs to the parent set of V_i, without being a parent nor a child of *OldAdj*, by removing from it the old adjacent node *OldAdj* (steps 13 to 15).

Subsequently, $UpdateMBC$ determines the set of the new adjacent nodes of the changed node V_i, denoted as $\left\{PC(V_i)^{s+1} \setminus PC(V_i)^{s}\right\}$ (step 17). The variables included in this set are variables that belong to $PC(V_i)^{s+1}$ but they were not previously in $PC(V_i)^{s}$, which means that they represent the set of variables that were weakly relevant or irrelevant to V_i and become strongly relevant to V_i. Hence, new dependence relationships should be inserted between those variables and V_i verifying at each insertion that no direct cycles are introduced. In this case, a new arc is inserted from each new adjacent node $NewAdj$ to V_i (step 18), then the parents-children and Markov blanket sets are updated accordingly. The following rules are performed:

- Add V_i to the parents-children set of $NewAdj$ (step 19): since an arc was inserted between V_i and $NewAdj$, V_i becomes a member of the parents-children set of $NewAdj$.

- If the new adjacent node $NewAdj$ is a class variable, then update its Markov blanket $MB(NewAdj)^{s+1}$ by adding to it the changed node V_i as well as its parent set $\mathbf{Pa}(V_i)$ (steps 20 to 22).

- If the changed node V_i is a class, then update its Markov blanket $MB(V_i)^{s+1}$ by adding to it $NewAdj$ and its parent set $\mathbf{Pa}(NewAdj)$ (steps 23 to 25).

- Update the Markov blanket of each class variable that belongs to the parent set of V_i, without being a parent nor a child $NewAdj$, by adding to it the new adjacent node $NewAdj$ (steps 26 to 28).

Finally, new conditional probability tables (CPTs) are learnt from D^{s+1} for all the nodes that have got a new parent set in MBC^{s+1} (step 31), and then the updated MBC network MBC^{s+1}, the sets PC^{s+1} and MB^{s+1} are returned in step 32.

Note here that, all variables that belong to both $PC(V_i)^{s}$ and $PC(V_i)^{s+1}$ of a changed node V_i do not trigger any kind of change. In fact, these variables were strongly relevant to V_i and still strongly relevant to V_i, so that the dependence relationships between them and

V_i remain the same. Moreover, the order of processing the changed nodes does not affect the final result, that is, independently of the order, the updated MBC network MBC^{s+1} and the sets PC^{s+1} and MB^{s+1} will be the same by the end of the $UpdateMBC$ algorithm. This is guaranteed because the identification of the old and new adjacent nodes is performed independently for each changed node, and thereby, it is not affected by the order nor by the results of other nodes. The updating process of PC and MB sets is also ensured via simple operations such as removing or adding variables, and hence, the order of variable removal or addition will not affect the final sets.

Example 8.1. *To illustrate the* `Locally Adaptive-MB-MBC` *algorithm, let us consider in Figure 8.1 an example of an MBC^s structure learnt from a batch stream D^s, and assume that we receive afterwards a new batch stream D^{s+1} generated from the MBC^{s+1} structure shown in Figure 8.2. Given both MBC^s and D^{s+1}, the* `Locally Adaptive-MB-MBC` *algorithm starts by computing the average log-likelihood and the PH test for each variable in MBC^s. Then, a change should be signaled for variables C_1, C_4, X_2, and X_5, i.e., $ChangedNodes = \{C_1, C_4, X_2, X_5\}$. Then, the MBC network should be locally updated via the $UpdateMBC$ algorithm.*

The $UpdateMBC$ algorithm updates the local structure around each changed node, then updates accordingly the parents-children and Markov blanket sets. Note that $UpdateMBC$ takes as input the current network MBC^s, the set of $ChangedNodes$, the new data stream D^{s+1}, as well as the current parents-children sets of all the variables PC^s, and the current Markov blankets sets of all the class variables MB^s represented in Table 8.2.

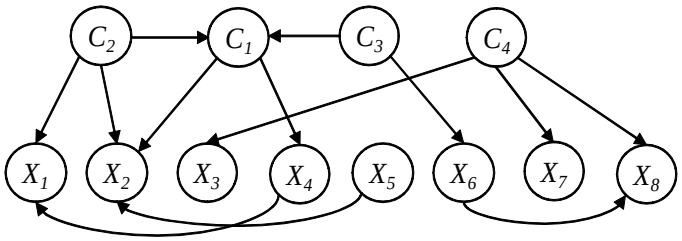


Figure 8.1: Example of an initial MBC structure.

In what follows, we present a trace of $UpdateMBC$ algorithm for each variable in the $ChangedNodes$ set:

- *The changed node C_1 (see Figure 8.3): The first step is to determine the new parents-children set of C_1 given D^{s+1} and using the HITON-PC algorithm. We assume that*

Table 8.2: PC^s and MB^s sets for the MBC structure shown in Figure 8.1.

PC^s	MB^s
$PC(C_1)^s = \{C_2, C_3, X_2, X_4\}$	$MB(C_1)^s = \{C_2, C_3, X_2, X_4, X_5\}$
$PC(C_2)^s = \{C_1, X_1, X_2\}$	$MB(C_2)^s = \{C_1, C_3, X_1, X_2, X_4, X_5\}$
$PC(C_3)^s = \{C_1, X_6\}$	$MB(C_3)^s = \{C_1, C_2, X_6\}$
$PC(C_4)^s = \{X_3, X_7, X_8\}$	$MB(C_4)^s = \{X_3, X_7, X_8, X_6\}$
$PC(X_1)^s = \{C_2, X_4\}$	
$PC(X_2)^s = \{C_1, C_2, X_5\}$	
$PC(X_3)^s = \{C_4\}$	
$PC(X_4)^s = \{C_1, X_1\}$	
$PC(X_5)^s = \{X_2\}$	
$PC(X_6)^s = \{C_3, X_8\}$	
$PC(X_7)^s = \{C_4\}$	
$PC(X_8)^s = \{C_4, X_6\}$	

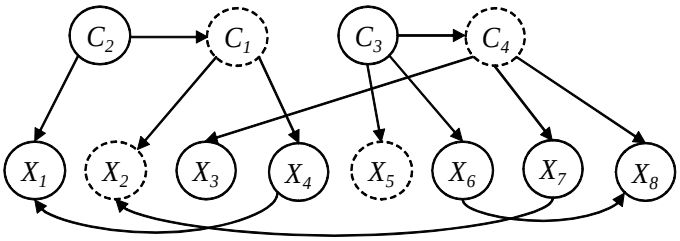


Figure 8.2: Example of an MBC structure including structural changes in comparison with the initial MBC structure in Figure 8.1. Nodes C_1, C_4, X_2, and X_5, represented in dashed line, are characterized as changed nodes.

HITON-PC detects the new parents-children set of C_1 *correctly, so we should have* $PC(C_1)^{s+1} = \{C_2, X_2, X_4\}$. *Next step is to determine the set of old and new adjacent nodes for* C_1. *For the old adjacent nodes we have* $PC(C_1)^s \setminus PC(C_1)^{s+1} = \{C_3\}$. *Thus, we remove the arc between* C_1 *and* C_3, *update* $PC(C_3)^{s+1} = PC(C_3)^{s+1} \setminus \{C_1\} = \{X_6\}$, *and update* $MB(C_3)^{s+1} = MB(C_3)^{s+1} \setminus \big\{C_1 \cup \{\mathbf{Pa}(C_1)^{s+1} \setminus PC(C_3)^{s+1}\}\big\}$. *Here we have* $\mathbf{Pa}(C_1)^{s+1} = \{C_2\}$ *and since* $\{C_2\}$ *does not pertain to* $PC(C_3)^{s+1}$, *then it should be removed from the Markov blanket of* C_3, *that is* $MB(C_3)^{s+1} = MB(C_3)^{s+1} \setminus \{C_1, C_2\} = \{X_6\}$. *Moreover, we update* $MB(C_1)^{s+1} = MB(C_1)^{s+1} \setminus \big\{C_3 \cup \{\mathbf{Pa}(C_3)^{s+1} \setminus PC(C_1)^{s+1}\}\big\} = \{C_2, X_2, X_4, X_5\}$. *Finally, we update the Markov blanket set of each class parent of* C_1. *In our case, we have only* C_2 *as parent of* C_1, *which does not pertain to* $PC(C_3)$, *thus* C_3 *should be removed from the Markov blanket of* C_2, *that is* $MB(C_2)^{s+1} = MB(C_2)^{s+1} \setminus \{C_3\} = \{C_1, X_1, X_2, X_4, X_5\}$.

For the new adjacent nodes, we have $PC(C_1)^{s+1} \setminus PC(C_1)^s = \emptyset$. *Thus, no new dependence relationships must be added for* C_1.

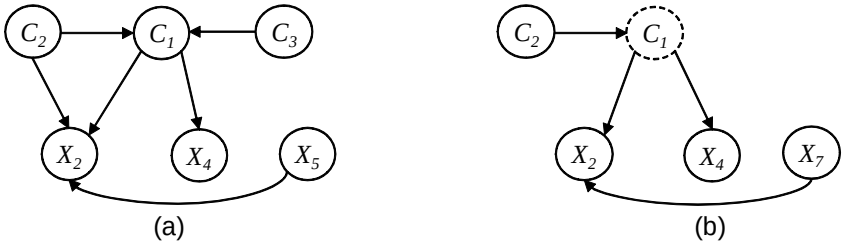


Figure 8.3: Markov blanket of node C_1 (a) before and (b) after change.

- *The changed node* C_4 *(see Figure 8.4): The first step is to determine the new parents-children set of* C_4 *given* D^{s+1} *and using the HITON-PC algorithm. As previously, we assume that HITON-PC detects the new parents-children set of* C_4 *correctly, so we should have* $PC(C_4)^{s+1} = \{C_3, X_3, X_7, X_8\}$. *Next, we determine the set of old adjacent nodes, which in our case is empty, i.e,* $PC(C_4)^s \setminus PC(C_4)^{s+1} = \emptyset$, *and the set of new adjacent nodes which is equal to* $PC(C_4)^{s+1} \setminus PC(C_4)^s = \{C_3\}$. *Consequently, we insert an arc from* C_3 *to* C_4, *we update* $PC(C_3)^{s+1} = PC(C_3)^{s+1} \cup \{C_4\} = \{C_4, X_6\}$, *and* $MB(C_3)^{s+1} = MB(C_3)^{s+1} \cup \{C_4 \cup \mathbf{Pa}(C_4)^{s+1}\} = \{C_4, X_6\}$. *Similarly, update* $MB(C_4)^{s+1} = MB(C_4)^{s+1} \cup \{C_3 \cup \mathbf{Pa}(C_3)^{s+1}\} = \{C_3, X_3, X_7, X_8, X_6\}$. C_4 *has no more parents except* C_3, *so steps 26-28 in the* $UpdateMBC$ *algorithm are not applied in this case.*

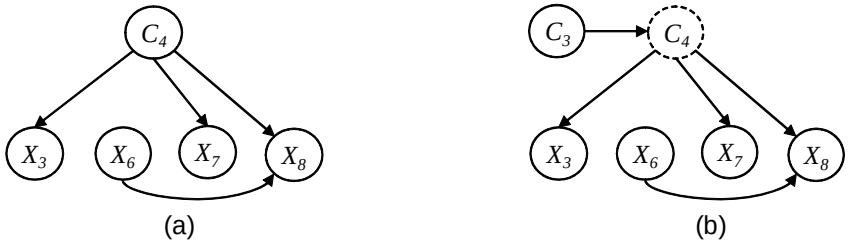


Figure 8.4: Markov blanket of node C_4 (a) before and (b) after change.

- *The changed node* X_2 *(see Figure 8.5): As previously, the first step is to determine the new parents-children set of* X_2 *given* D^{s+1} *and using the HITON-PC algorithm. Assuming that HITON-PC detects the new parents-children set of* X_2 *correctly, we should have* $PC(X_2)^{s+1} = \{C_1, X_7\}$. *Next, given that* $PC(X_2)^s = \{C_1, C_2, X_5\}$, *the set of old adjacent nodes is determined as* $PC(X_2)^s \setminus PC(X_2)^{s+1} = \{C_2, X_5\}$.

For the first old adjacent variable C_2, we remove the arc between C_2 and X_2, we update $PC(C_2)^{s+1} = PC(C_2)^{s+1} \setminus \{X_2\} = \{C_1, X_1\}$, and we update $MB(C_2)^{s+1} = MB(C_2)^{s+1} \setminus \{X_2 \cup \{\mathbf{Pa}(X_2)^{s+1} \setminus PC(C_2)^{s+1}\}\}$. Here X_2 has two parents namely C_1 and X_5 (in fact X_5 is not removed yet from the set of parents of X_2 because we start by processing the old adjacent variable C_2), and since C_1 pertains to $PC(C_2)^{s+1}$, the only variables to be removed from $MB(C_2)^{s+1}$ are then X_2 and X_5, i.e., $MB(C_2)^{s+1} = \{C_1, X_1, X_4\}$.

For the second old adjacent variable X_5, we remove the arc between X_5 and X_2, we update $PC(X_5)^{s+1} = PC(X_5)^{s+1} \setminus \{X_2\} = \emptyset$, then update the Markov blanket set for every class variable of X_2 that does not pertain to $PC(X_5)^{s+1}$. In our case, X_2 has only C_1 as a class parent (because both C_2 and X_5 have been already removed), so its Markov blanket is modified as follows $MB(C_1)^{s+1} = MB(C_1)^{s+1} \setminus \{X_5\} = \{C_2, X_2, X_4\}$.

For the new adjacent nodes, we have $PC(X_2)^{s+1} \setminus PC(X_2)^{s} = \{X_7\}$. Thus, we insert an arc from X_7 to X_2, update $PC(X_7)^{s+1} = PC(X_7)^{s+1} \cup \{X_2\} = \{C_4, X_2\}$, then update the Markov blanket set for every class variable of X_2 that does not pertain to $PC(X_7)^{s+1}$. In our case, X_2 has only C_1 as a class parent, which is different from X_7 and not pertaining to $PC(X_7)$, so its Markov blanket is modified as follows $MB(C_1)^{s+1} = MB(C_1)^{s+1} \cup \{X_7\} = \{C_2, X_2, X_4, X_7\}$.

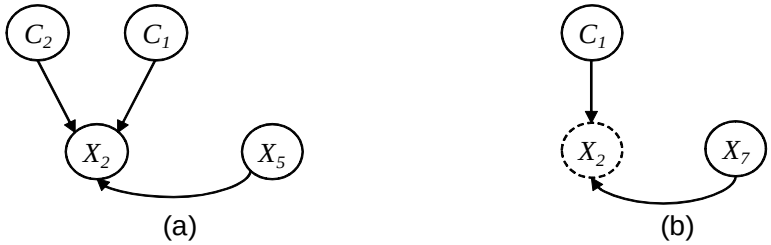


Figure 8.5: Parents-children set of node X_2 (a) before and (b) after change.

- *The changed node X_5 (see Figure 8.6): The first step is to determine the new parents-children set of X_5 given D^{s+1} and using the HITON-PC algorithm. Assuming that HITON-PC detects the new parents-children set of X_5 correctly, we obtain $PC(X_5)^{s+1} = \{C_3\}$. Then, given that $PC(X_5)^{s} = \{X_2\}$, we determine first the set of old adjacent nodes $PC(X_5)^{s} \setminus PC(X_5)^{s+1} = \{X_2\}$. Since the changed variable X_2 has been processed before the changed node X_5, we can see that the arc between these two variables has been already removed during the previous phase. Moreover, X_5 has been already removed from $PC(X_2)^{s+1}$, so there is no change for $PC(X_2)^{s+1} = \{C_1, X_7\}$. X_5 at this step has no*

class parents, so steps 13-15 in UpdateMBC algorithm are not applied in this case.

For the new adjacent nodes, we have $PC(X_5)^{s+1} \setminus PC(X_5)^s = \{C_3\}$. *Thus, we insert an arc from* C_3 *to* X_5, *update* $PC(C_3)^{s+1} = PC(C_3)^{s+1} \cup \{X_5\} = \{C_4, X_5, X_6\}$, *and update its Markov blanket set* $MB(C_3)^{s+1} = MB(C_3)^{s+1} \cup \{X_5\} = \{C_4, X_5, X_6\}$. X_5 *is not a class variable and has no more class parents except* C_3, *so no more changes have to be considered.*

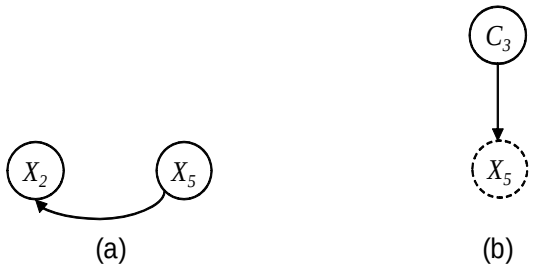


Figure 8.6: Parents-children set of node X_5 (a) before and (b) after change.

Note finally that, the changes performed on the local structure of each changed node lead as well to the changes of the PC and MB sets of some adjacent nodes such as, in our case, those of the variables C_2, C_3 *and* X_7. *However, some other variables do not present any change and their PC sets are kept the same, namely,* X_1, X_3, X_4, X_6, *and* X_8. *In addition, the order of processing the changed variables affects the order of the execution of some operations, however it does not affect the final result.*

8.5 Experimental design

8.5.1 Data sets

- Synthetic multi-dimensional data streams: We randomly generated a sequence of five MBC networks, such that the first MBC network is randomly defined on a set of $d = 6$ class variables and $m = 14$ feature variables. Then, each subsequent MBC network is obtained by randomly changing the dependence relationships around a percentage p of nodes with respect to the preceding MBC network in the sequence. Depending on parameter p, we set three different configurations to test different rates of concept drift:

  - Configuration 1: no concept drift ($p = 0\%$). In this case, the same MBC network is used to sample the total number of instances in the sequence. This aims to

generate a stationary data stream and allows us to verify the resilience of the proposed algorithm to false alarms.

- Configuration 2: gradual concept drift ($p = 20\%$). The percentage of changed nodes between each consecutive MBC networks is equal to $p = 20\%$. For each selected changed node, its parent set is modified by removing the existing parents and randomly adding new ones. For the parameters, new CPTs are randomly generated for the set of changed nodes presenting new parent sets, whereas the CPTs of the non-changed are kept the same as the preceding MBC.
- Configuration 3: abrupt concept drift ($p = 50\%$). Similar to configuration 2, but we fixed the percentage of changed nodes between each consecutive MBC networks to $p = 50\%$.

Afterwards, for each configuration, 10000 instances are randomly sampled from each MBC network in the sequence, using the probabilistic logic sampling method [107], then concatenated to form a data stream of 50000 instances.

- SynT-drift data stream provided by Read et al. [181]: In order to compare our approach against existing multi-label stream classification methods, namely, `BRa`, `EaBR`, $\texttt{EaHT}_{PS}$, `EaPS`, `HTa`, `MBR`, and `MWC`, we test our `Locally` and `Globally Adaptive-MB-MBC` methods on SynT-drift.

  SynT-drift is a multi-label synthetic data stream including 1000000 instances with $d = 8$ binary class variables and $m = 30$ binary feature variables. It is sampled using the random tree generator proposed by Domingos and Hulten [67], that constructs a decision tree by choosing attributes at random to split, and assigning a random class label to each leaf. Once the tree is built, new examples are generated by assigning uniformly random values to attributes which then determine the class label via the tree.

  Read et al. [181] included three concept drifts in SynT-drift of varying type, magnitude and extent. In the first drift, they changed only 10% of label dependencies. In the second drift, the underlying concept changes and more labels are associated on average with each instance (i.e., the label cardinality $LCard$ changes from 1.8 to 3.0), and in the third drift, 20% of label dependencies change.

8.5.2 Evaluation metrics

The synthetic data streams are processed by windows of size 1000 instances, and the *prequential* setting [56, 91] is used to evaluate the predictive performance of the MBC network on

each window. In this setting, each incoming window is used for testing the MBC network before it is used for training, in such a way that the MBC network is always tested on instances that have not been seen before.

As evaluation metrics, we considered the mean (see Equation (3.4)) and the global accuracy (see Equation (3.5)) to assess the predictive performance of the learned classifiers (higher is better). In addition, we used the following two metrics, to mainly evaluate the quality of the learned MBC networks:

- **Kullback-Leibler Divergence (KLDiv)** [135]: It measures the divergence between the learned MBC networks and the original ones. The lower the KLDiv values, the better the quality of learning algorithm.

- **Structural Hamming Distance (SHD)** [213]: It compares the structure of the learned and the original MBC networks, and is defined as the number of operations required to make two completed partially DAGs (CPDAGs) match. The operations are add or delete an undirected edge, and add, delete, or reverse the orientation of an edge. Each of these operations is penalized with the same strength by increasing the SHD by 1. In our case, since all learned and original MBCs are DAGs, we build first the CPDAGs of both learned and original MBC DAGs using the DAG-to-CPDAG algorithm [45], then we compute the SHD metric. The lower the resulting SHD value is, the better the algorithm performed.

- **Running time**: It reports the cumulative learning plus testing times in seconds.

Note that for experiments on SynT-drift data stream, the KLDiv and SHD evaluation are omitted since we do not have an original MBC network for this data. Moreover, as reported in [181], we compute the subset accuracy (see Equation (3.6)) instead of the mean accuracy.

8.6 Experimental results

For the first set of experiments, performed using 20 variables (6 class variables and 14 feature variables), we sampled randomly five different data streams for each configuration (i.e., for each p=0%, p=20% and p=50%) and we applied both `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) and `Globally Adaptive-MB-MBC` (`GA-MB-MBC`) using four different values of λ, namely $\lambda = 1, 2.5, 5, 10$. This allows us to study the sensitivity of both algorithms with respect to the input parameter λ.

Tables 8.3, 8.4 and 8.5 show the estimated performance results as mean values and standard deviations for each metric and each method over the five randomly generated data streams. The best result for each metric is written in bold. In Table 8.3, presenting the results with $p = 0\%$ (i.e., stationary data streams), we can first notice the very low sensitivity of both algorithms with respect to λ values. In fact, even if the best result for the mean accuracy is obtained with `LA-MB-MBC` with $\lambda = 5$, and the best result for the global accuracy is obtained with `LA-MB-MBC` with $\lambda = 10$, both algorithms `LA-MB-MBC` and `GA-MB-MBC` present similar predictive performance for the remaining λ values. Moreover, `LA-MB-MBC` and `GA-MB-MBC` show similar results for KLDiv, SHD and running time with, generally, a slightly better performance for `LA-MB-MBC`.

Table 8.3: Experimental results (mean ± std. dev.) over synthetic data with $p = 0\%$.

	`LA-MB-MBC`	`GA-MB-MBC`
$\lambda = 1$		
Mean accuracy	0.739 ± 0.075	0.737 ± 0.078
Global accuracy	0.253 ± 0.158	0.255 ± 0.156
SHD	34.492 ± 5.796	33.700 ± 4.568
KLDiv	0.826 ± 0.156	0.926 ± 0.321
Running time	906.750 ± 98.484	908.154 ± 122.792
$\lambda = 2.5$		
Mean accuracy	0.739 ± 0.075	0.737 ± 0.077
Global accuracy	0.255 ± 0.154	0.252 ± 0.152
SHD	33.084 ± 5.056	32.564 ± 5.633
KLDiv	0.827 ± 0.170	0.921 ± 0.367
Running time	886.900 ± 82.758	887.384 ± 105.939
$\lambda = 5$		
Mean accuracy	$\mathbf{0.740 \pm 0.075}$	0.737 ± 0.075
Global accuracy	0.255 ± 0.155	0.250 ± 0.150
SHD	33.156 ± 6.922	35.120 ± 7.108
KLDiv	0.839 ± 0.186	0.856 ± 0.156
Running time	861.600 ± 68.628	864.948 ± 92.517
$\lambda = 10$		
Mean accuracy	0.739 ± 0.074	0.739 ± 0.074
Global accuracy	$\mathbf{0.256 \pm 0.156}$	0.251 ± 0.151
SHD	$\mathbf{32.056 \pm 7.508}$	32.960 ± 7.327
KLDiv	0.843 ± 0.176	$\mathbf{0.813 \pm 0.184}$
Running time	$\mathbf{846.212 \pm 84.929}$	886.285 ± 92.033

In Table 8.4, presenting the experimental results with a drift rate $p = 20\%$, `LA-MB-MBC` is

performing the best with $\lambda = 1$ for the mean accuracy, global accuracy and KLDiv. However, the best SHD result is obtained with `LA-MB-MBC` with $\lambda = 2.5$. In addition, contrary to results in Table 8.3 ($p = 0\%$), we can observe that under a higher drift rate ($p = 20\%$), both algorithms become more sensitive to the value of λ. For both `LA-MB-MBC` and `GA-MB-MBC` algorithms, the best accuracies are obtained with $\lambda = 1$, and as long as λ increases, the mean and global accuracies decrease whereas the SHD and KLDiv values increase. In fact, using higher λ values, some concept drifts cannot be detected and consequently the model cannot be updated correctly; which may affect its performance over time. In this case, we can see that `GA-MB-MBC` is more sensitive since missing the detection of a drift affects the whole MBC network.

Table 8.4: Experimental results (mean ± std. dev.) over synthetic data with $p = 20\%$.

	`LA-MB-MBC`	`GA-MB-MBC`
$\lambda = 1$		
Mean accuracy	$\mathbf{0.701 \pm 0.038}$	0.695 ± 0.046
Global accuracy	$\mathbf{0.181 \pm 0.072}$	0.173 ± 0.075
SHD	32.592 ± 7.728	32.152 ± 5.249
KLDiv	$\mathbf{0.868 \pm 0.142}$	1.222 ± 0.361
Running time	956.983 ± 88.816	926.456 ± 81.923
$\lambda = 2.5$		
Mean accuracy	0.695 ± 0.040	0.682 ± 0.042
Global accuracy	0.177 ± 0.069	0.157 ± 0.074
SHD	$\mathbf{31.540 \pm 8.708}$	33.516 ± 5.793
KLDiv	0.959 ± 0.196	1.498 ± 0.490
Running time	$\mathbf{882.608 \pm 71.498}$	931.117 ± 130.663
$\lambda = 5$		
Mean accuracy	0.689 ± 0.040	0.686 ± 0.050
Global accuracy	0.166 ± 0.067	0.167 ± 0.076
SHD	33.916 ± 8.403	33.208 ± 5.187
KLDiv	1.076 ± 0.178	1.363 ± 0.384
Running time	899.575 ± 80.939	964.248 ± 96.363
$\lambda = 10$		
Mean accuracy	0.688 ± 0.045	0.668 ± 0.056
Global accuracy	0.166 ± 0.070	0.147 ± 0.080
SHD	35.576 ± 7.539	34.840 ± 5.030
KLDiv	1.141 ± 0.180	1.825 ± 0.579
Running time	894.720 ± 80.057	915.479 ± 95.438

Table 8.5 shows the experimental results with a drift rate $p = 50\%$. Similar to results

in Table 8.4, we may conclude that for both algorithms, the best accuracies were obtained with $\lambda = 1$, and as long as λ increases, all the performance measures get worse. `GA-MB-MBC` outperforms `LA-MB-MBC` in mean and global accuracies with $\lambda = 1$, however, for all remaining λ values, `LA-MB-MBC` presents better predictive performance. The better performance of `GA-MB-MBC` compared to `LA-MB-MBC` can be explained by the fact that having 50% of drift affects a larger instance space (i.e., it can be viewed as a global change), and consequently it might be better to re-build all the MBC network rather than updating it locally.

Table 8.5: Experimental results (mean ± std. dev.) over synthetic data with $p = 50\%$.

	`LA-MB-MBC`	`GA-MB-MBC`
$\lambda = 1$		
Mean accuracy	0.723 ± 0.037	$\mathbf{0.724 \pm 0.047}$
Global accuracy	0.190 ± 0.056	$\mathbf{0.197 \pm 0.071}$
SHD	33.760 ± 5.586	$\mathbf{28.460 \pm 2.539}$
KLDiv	$\mathbf{0.822 \pm 0.260}$	0.945 ± 0.259
Running time	985.971 ± 66.058	895.399 ± 44.161
$\lambda = 2.5$		
Mean accuracy	0.712 ± 0.043	0.709 ± 0.057
Global accuracy	0.184 ± 0.060	0.178 ± 0.078
SHD	35.040 ± 3.581	30.504 ± 3.846
KLDiv	0.971 ± 0.281	1.286 ± 0.495
Running time	996.916 ± 87.938	892.666 ± 39.651
$\lambda = 5$		
Mean accuracy	0.709 ± 0.046	0.699 ± 0.042
Global accuracy	0.174 ± 0.061	0.160 ± 0.049
SHD	34.380 ± 2.675	31.068 ± 3.911
KLDiv	0.966 ± 0.196	1.622 ± 0.567
Running time	973.825 ± 70.961	$\mathbf{865.943 \pm 35.643}$
$\lambda = 10$		
Mean accuracy	0.704 ± 0.046	0.688 ± 0.037
Global accuracy	0.164 ± 0.064	0.146 ± 0.039
SHD	38.068 ± 2.297	31.83 ± 4.083
KLDiv	1.202 ± 0.354	1.776 ± 0.819
Learning time	947.876 ± 75.133	884.797 ± 48.339

In addition, we plot in Figures 8.7, 8.8 and 8.9 the mean and global accuracy curves for `LA-MB-MBC` and `GA-MB-MBC` algorithms, with $\lambda = 1$ and with p equal to 0%, 20% and 50%, respectively. For each curve, the X-axis represents the block number, and the Y-axis represents the classification accuracy.

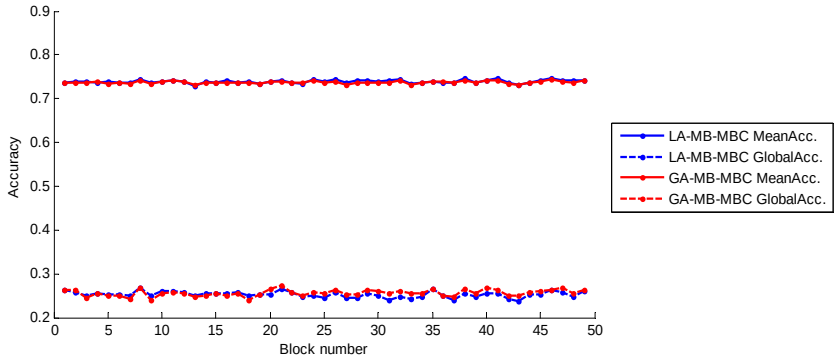


Figure 8.7: Classification results with the drift rate $p = 0\%$ and $\lambda = 1$.

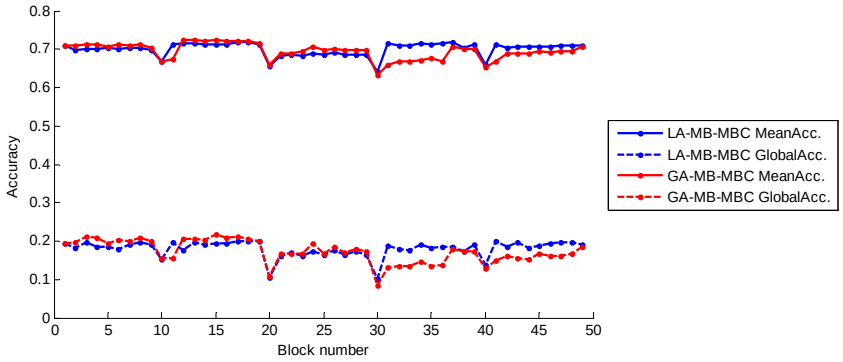


Figure 8.8: Classification results with the drift rate $p = 20\%$ and $\lambda = 1$.

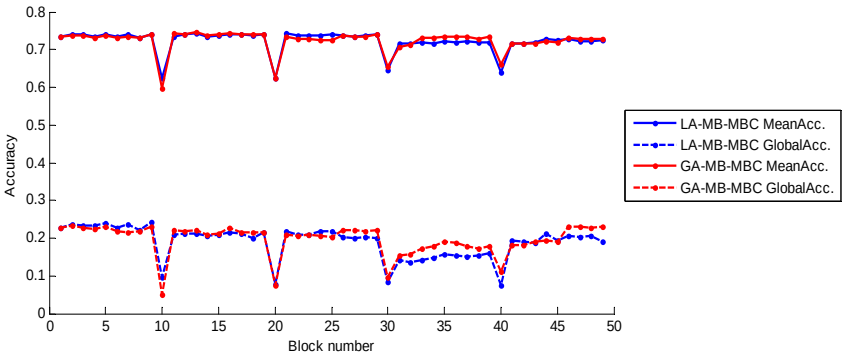


Figure 8.9: Classification results with the drift rate $p = 50\%$ and $\lambda = 1$.

In Figure 8.7, we can observe that LA-MB-MBC and GA-MB-MBC curves are almost superposed showing the similar performance of both algorithms, as well as their resilience to false alarms.

In Figures 8.8 and 8.9, we can first notice that both algorithms perform well in detecting the change at blocks 10, 20, 30 and 40. With $p = 20\%$ (Figure 8.8) the change is more gradual, whereas, in Figure 8.9 with $p = 50\%$, the change is abrupt and more important fluctuations in predictive performance are present. We can also see that in Figure 8.8, between blocks 30 and 50, LA-MB-MBC clearly outperforms GA-MB-MBC in updating the MBC network and recuperating more quickly its performance. Nevertheless, with higher drift rate, i.e., $p = 50\%$, GA-MB-MBC presents a slightly better performance than LA-MB-MBC.

In the second set of experiments with SynT-drift data streams, we compare LA-MB-MBC and GA-MB-MBC algorithms against seven multi-label classification methods. Five of them, i.e., BRa, EaBR, EaHT$_{PS}$, EaPS and HTa were proposed by Read et al. [181], whereas MBR and MWC were proposed respectively in [178] and [233]. Similarly, as Read et al. [181], we divide the stream into 20 windows and we report the average of subset and global accuracies across the data windows, as well as the cumulative running time in seconds. Note that both LA-MB-MBC and GA-MB-MBC are performed using $\lambda = 1$. The obtained results are reported in Table 8.6.

Table 8.6: Experimental results over SynT-drift data.

	Subset accuracy	Global accuracy	Running time
BRa	0.196	0.018	62
EaBR	0.195	0.015	375
EaHT$_{PS}$	**0.221**	0.026	34
EaPS	0.184	0.030	628
HTa	0.164	**0.046**	**14**
MBR	0.199	0.020	678
MWC	0.159	0.014	1869
LA-MB-MBC	0.173	0.040	3714
GA-MB-MBC	0.198	0.038	3097

For the subset accuracy, GA-MB-MBC performs better than LA-MB-MBC, and also better than any other method except MBR and EaHT$_{PS}$. For the global accuracy, LA-MB-MBC performs better than all remaining methods except HTa. Although not the best, LA-MB-MBC and GA-MB-MBC both present a good performance especially because SynT-drift is generated based on tree models, and as expected methods based on Hoeffding trees (i.e., EaHT$_{PS}$ and HTa) provide the best accuracy results. Nevertheless, the main shortcoming of our adaptive algorithms is the running time which is slower than all remaining methods, and this is mainly due to the testing part that involves the computation of the most probable explanation.

8.7 Conclusion

In this chapter, we have presented two new methods for mining multi-dimensional data streams, namely, `GA-MB-MBC` and `LA-MB-MBC`. Basically, `GA-MB-MBC` uses the Page-Hinkley test to monitor the average global log-likelihood over time and detect the concept drift, then, whenever a concept drift is detected, it learns a new MBC from scratch. On the other hand, `LA-MB-MBC` proceeds locally at the level of each node in the MBC network, that is, it monitors the average local log-likelihood of each node over time, then, whenever a concept drift is detected, it learns a new local structure for each changed node.

Experimental results on synthetic data streams including different rates of change were promising. Specifically, `GA-MB-MBC` and `LA-MB-MBC` are shown to be resilient to false alarms, and also efficient in detecting the change points and adapting the MBC networks. Moreover, both methods show similar predictive performance and exhibit competitive accuracy results when compared with existing multi-label classification methods.

In the future, we intend to carry out a more extensive experimental study using additional synthetic and real data streams. Furthermore, it will be interesting to investigate the use of different exact or approximate inference methods in order to alleviate the computational burden when calculating the most probable explanation.

Part V

Conclusions

Chapter 9

Conclusions and future work

This concluding chapter is organized in three sections: first, Section 9.1 summarizes the main contributions and conclusions provided in this dissertation; then, Section 9.2 includes the publications and submissions produced during this research; and finally, Section 9.3 discusses open issues and main lines of future work.

9.1 Summary of contributions

So far, we have introduced through this dissertation several contributions consisting of novel classification methods with applications to different real-world problems.

In particular, in the first part, we tackled the problem of learning multi-dimensional Bayesian network classifiers from stationary data and we proposed two novel algorithms, namely, `CB-MBC` and `MB-MBC`, in Chapter 5 and Chapter 6, respectively. `CB-MBC` is based on a wrapper greedy forward selection approach optimizing at each step the accuracy of the classifier, while `MB-MBC` is a filter approach based on the identification of the Markov blanket around each class variable.

We applied `CB-MBC` and `MB-MBC` to two real-world multi-dimensional problems: 1) the prediction of the human immunodeficiency virus type 1 reverse transcriptase and protease inhibitors, and 2) the prediction of the European Quality of Life-5 Dimensions from 39-item Parkinson's Disease Questionnaire. Experimental results were promising: `CB-MBC` and `MB-MBC` outperformed the state-of-the-art multi-dimensional classification methods and allowed us to gain insight into both known and novel interactions among the studied variables.

In the second part of the dissertation, we dealt with streaming classification problems that present more challenges in detecting concept drifts and adapting the classification model over

time, and we proposed through Chapters 7 and 8 new different approaches.

Specifically, in Chapter 7, the semi-supervised `CPL-DS` approach addressed the problem of classifying partially labeled uni-dimensional data streams. `CPL-DS` handles three kinds of drift (feature, conditional or dual drift) using Kullback-Leibler divergence and a bootstrap method, and re-learns from scratch the classifier, when a drift is detected, using the expectation-maximization algorithm. `CPL-DS` was tested with synthetic data streams and applied to the real-world problem of malware detection. Experimental results showed the good performance of `CPL-DS` in terms of classification accuracy as well as the detection of different kinds of drift.

In Chapter 8, `Globally Adaptive-MB-MBC` (`GA-MB-MBC`) and `Locally Adaptive-MB-MBC` (`LA-MB-MBC`) algorithms were introduced to deal with mining multi-dimensional concept-drifting data streams. Both methods handle concept drifts over time using the average log-likelihood score and the Page-Hinkley test. When a drift is detected, `LA-MB-MBC` adapts the current multi-dimensional Bayesian network classifier locally around each changed node, whereas `GA-MB-MBC` learns a new multi-dimensional Bayesian network classifier from scratch. Experimental results with synthetic multi-dimensional data streams were encouraging and proved the merits of both proposed adaptive methods.

9.2 List of publications

The work presented through this dissertation has produced the following list of publications and submissions:

A. Refereed journals

- H. Borchani, P. Larrañaga, and C. Bielza. Classifying evolving data streams with partially labeled data. *Intelligent Data Analysis*, 15(5):655–670, 2011. Impact factor (2011): 0.448.

- H. Borchani, C. Bielza, P. Martínez-Martín, and P. Larrañaga. Markov blanket-based approach for learning multi-dimensional Bayesian network classifiers: An application to predict the European Quality of Life-5 Dimensions (EQ-5D) from the 39-item Parkinson's Disease Questionnaire (PDQ-39). *Journal of Biomedical Informatics* 45:1175–1184, 2012. Impact factor (2011): 1.792.

- H. Borchani, C. Bielza, P. Larrañaga, and C. Toro. Learning multi-dimensional Bayesian network classifiers using Markov blankets: A case study in the prediction of HIV-1

reverse transcriptase and protease inhibitors. *Artificial Intelligence in Medicine* (in press) doi: 10.1016/j.artmed.2012.12.005, 2013. Impact factor (2011): 1.345.

- H. Borchani, P. Larrañaga, J. Gama, and C. Bielza. Mining multi-dimensional concept-drifting data streams using Bayesian network classifiers. *Data Mining and Knowledge Discovery*, 2012, Submitted. Impact factor (2011): 1.545.

B. Conference and workshop communications

- H. Borchani, P. Larrañaga, and C. Bielza. Mining concept-drifting data streams containing labeled and unlabeled instances. In *Proceedings of the Twenty-third International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, pages 531–540, 2010.

- H. Borchani, C. Bielza, and P. Larrañaga. Learning CB-decomposable multi-dimensional Bayesian network classifiers. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pages 25–32, 2010.

- H. Borchani, C. Bielza, and P. Larrañaga. Learning multi-dimensional Bayesian network classifiers using Markov blankets: A case study in the prediction of HIV protease inhibitors. In *AIME'11 Workshop on Probabilistic Problem Solving in Biomedicine*, pages 29–40, 2011.

9.3 Future work

This section summarizes and emphasizes the most relevant future lines and open issues that have been already enumerated through the specific conclusion section of each chapter.

Firstly, since our contributions were all based on discrete variables, they cannot be applied directly to continuous variables and necessarily require a discretization pre-processing step to convert each continuous variable in the data into a discrete variable. Hence, an interesting direction for future work is to generalize our methods in order to allow the combination of both discrete and continuous variables.

Secondly, for both stationary and streaming multi-dimensional methods, we assumed that there are no missing values in the data. This assumption is not always verified especially when dealing with real-world applications, where class assignments are usually incomplete [164]. In this case, as another line for future research, our multi-dimensional methods might be extended either with the use of missing values imputation methods or with the adaptation of

the expectation-maximization algorithm to enable parameter estimations or structure learning from incomplete data.

Furthermore, for the proposed multi-dimensional methods, it would be also interesting to investigate the use of different exact or approximate inference methods in order to alleviate the computational burden when calculating the most probable explanation.

Finally, an important and crucial problem related to our work is learning from imbalanced data. With `CPL-DS` approach, we used two existing approaches, namely, clustering-sampling and SERA to deal with uni-dimensional imbalanced data streams. In the future, we intend to carry out more investigation on this topic especially with respect to the streaming multi-dimensional setting.

Bibliography

[1] C.C. Aggarwal. *Data Streams: Models and Algorithms*. Springer, 2007.

[2] C.C. Aggarwal. A segment-based framework for modeling and mining data streams. *Knowledge and Information Systems*, 30(1):1–29, 2012.

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

[4] C.F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X.D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification. Part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11:171–234, 2010.

[5] C.F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X.D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification. Part II: Analysis and extensions. *Journal of Machine Learning Research*, 11:235–284, 2010.

[6] C.F. Aliferis, I. Tsamardinos, and A. Statnikov. Causal explorer: A probabilistic network learning toolkit for discovery, 2010. *Available at:* `discover.mc.vanderbilt.edu/discover/public/causal_explorer/`.

[7] C.F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON: A novel Markov blanket algorithm for optimal variable selection. In *American Medical Informatics Association, Annual Symposium Proceedings*, pages 21–25, 2003.

[8] A. Altmann, N. Beerenwinkel, T. Sing, I. Savenkov, M. Däumer, R. Kaiser, S.Y. Rhee, W.J. Fessel, R. Shafer, and T. Lengauer. Improved prediction of response to antiretroviral combination therapy using the genetic barrier to drug resistance. *Antiviral Therapy*, 12(2):169–178, 2007.

[9] M.R. Amini and P. Gallinari. Semi-supervised logistic regression. In *Proceedings of the Fifteenth European Conference on Artificial Intelligence*, pages 390–394, 2002.

[10] S.A. Andersson, D. Madigan, and M.D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25(2):505–541, 1997.

[11] A. Asuncion and D.J. Newman. UCI Machine Learning Repository, University of California, Irvine. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[12] M. Baena-García, J. del Campo-Avila, R. Fidalgo, A. Bifet, and R. Gavaldà. Early drift detection method. In *Proceedings of the Fourth ECML/PKDD International Workshop on Knowledge Discovery from Data Streams*, pages 77–86, 2006.

[13] C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

[14] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*. IOS Press, 2010.

[15] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 29–40, 2007.

[16] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148, 2009.

[17] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] R. Blanco, I. Inza, and P. Larrañaga. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18(2):205–220, 2003.

[19] R. Blanco, I. Inza, M. Merino, J. Quiroga, and P. Larrañaga. Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS. *Journal of Biomedical Informatics*, 38(5):376–388, 2005.

[20] H. Borchani, N. Ben Amor, and F. Khalfallah. Learning and evaluating Bayesian network equivalence classes from incomplete data. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(2):253–278, 2008.

[21] H. Borchani, C. Bielza, and P. Larrañaga. Learning CB-decomposable multi-dimensional Bayesian network classifiers. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pages 25–32, 2010.

[22] H. Borchani, C. Bielza, and P. Larrañaga. Learning multi-dimensional Bayesian network classifiers using Markov blankets: A case study in the prediction of HIV protease inhibitors. In *AIME'11 Workshop on Probabilistic Problem Solving in Biomedicine*, pages 29–40, 2011.

[23] H. Borchani, C. Bielza, P. Larrañaga, and C. Toro. Learning multi-dimensional Bayesian network classifiers using Markov blankets: A case study in the prediction of HIV-1 reverse transcriptase and protease inhibitors. *Artificial Intelligence in Medicine*, 2012, Submitted.

[24] H. Borchani, C. Bielza, P. Martínez-Martín, and P. Larrañaga. Markov blanket-based approach for learning multi-dimensional Bayesian network classifiers: An application to predict the European Quality of Life-5 Dimensions (EQ-5D) from the 39-item Parkinson's Disease Questionnaire (PDQ-39). *Journal of Biomedical Informatics*, 45:1175–1184, 2012.

[25] H. Borchani, P. Larrañaga, and C. Bielza. Mining concept-drifting data streams containing labeled and unlabeled instances. In *Proceedings of the Twenty-third International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 531–540, 2010.

[26] H. Borchani, P. Larrañaga, and C. Bielza. Classifying evolving data streams with partially labeled data. *Intelligent Data Analysis*, 15(5):655–670, 2011.

[27] H. Borchani, P. Larrañaga, J. Gama, and C. Bielza. Mining multi-dimensional concept-drifting data streams using Bayesian network classifiers. *Data Mining and Knowledge Discovery*, 2012, Submitted.

[28] R.R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, University of Utrecht, Utrecht, The Netherlands, 1995.

[29] R.R. Bouckaert, E. Castillo, and J.M. Gutiérrez. A modified simulation scheme for inference in Bayesian networks. *International Journal of Approximate Reasoning*, 14:55–80, 1996.

[30] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[31] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[32] R. Brooks, R. Rabin, and F. Charro. *The Measurement and Valuation of Health Status Using EQ-5D: A European Perspective*. Dordrecht, The Netherlands, Kluwer Academic, 2003.

[33] W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[34] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, 1997.

[35] G. Castillo. *Adaptive Learning Algorithms for Bayesian Network Classifiers*. PhD thesis, University of Aveiro, Portugal, 2006.

[36] O. Chapelle, B. Schökopf, and A. Zien. *Semi-supervised Learning*. MIT Press, 2006.

[37] K.R. Chaudhuri, D. Healy, and A.H. Schapira. The non-motor symptoms of Parkinson's disease: Diagnosis and management. *The Lancet Neurology*, 5(3):235–245, 2006.

[38] N.V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366, 2005.

[39] S. Chen and H. He. SERA: Selectively recursive approach towards nonstationary imbalanced stream data mining. In *Proceedings of the International Joint Conference of Neural Networks*, pages 522–529, 2009.

[40] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.

[41] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[42] K. Cheung, M. Oemar, M. Oppe, and R. Rabin. EQ-5D user guide. Basic information on how to use the EQ-5D, version 2.0. *EuroQol Group*, Available from: `www.euroqol.org`.

[43] Y.B. Cheung, L.C. Tan, P.N. Lau, W.L. Au, and N. Luo. Mapping the eight-item Parkinson's Disease Questionnaire (PDQ-8) to the EQ-5D utility index. *Quality of Life Research*, 17(9):1173–1181, 2008.

[44] J. Chevrolat, J. Golmard, S. Ammar, R. Jouventc, and J.F. Boisvieuxa. Modeling behavior syndromes using Bayesian networks. *Artificial Intelligence in Medicine*, 14(3):259–277, 1998.

[45] D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[46] D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

[47] 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.

[48] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[49] A. Clare and R.D. King. Knowledge discovery in multi-label phenotype data. In *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53, 2001.

[50] G.F. Cooper. Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[51] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(3):309–347, 1992.

[52] T.M. Cover and P.E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[53] R. Daly, Q. Shen, and S. Aitken. Learning Bayesian networks: Approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157, 2011.

[54] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, USA, 2009.

[55] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of the Thirty-eighth Symposium on the Interface of Statistics, Computing Science, and Applications*, pages 1–24, 2006.

[56] A.P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society - A*, 147(2):278–292, 1984.

[57] A.P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[58] L.M. de Campos and J.G. Castellano. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45(2):233–254, 2007.

[59] L.M. de Campos, J.M. Fernández-Luna, J.A. Gámez, and J.M. Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(21):291–311, 2002.

[60] A. de Carvalho and A.A. Freitas. A tutorial on multi-label classification techniques. *Foundations of Computational Intelligence, Studies in Computational Intelligence*, 5:177–195, 2009.

[61] P.R. de Waal and L.C. van der Gaag. Inference and learning in multi-dimensional Bayesian network classifiers. In *Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty, Lecture Notes in Artificial Intelligence, Vol. 4724, Springer*, pages 501–511, 2007.

[62] K. Deforche, R. Camacho, Z. Grossman, T. Silander, M.A. Soares, and et al. Bayesian network analysis of resistance pathways against HIV-1 protease inhibitors. *Infection, Genetics and Evolution*, 7:382–390, 2007.

[63] K. Deforche, R. Camacho, Z. Grossman, M.A. Soares, K. Van Laethem, and et al. Bayesian network analyses of resistance pathways against efavirenz and nevirapine. *AIDS*, 22:2107–2115, 2008.

[64] K. Deforche, T. Silander, R. Camacho, Z. Grossman, M.A. Soares, and et al. Analysis of HIV-1 pol sequences using Bayesian networks: Implications for drug resistance. *Bioinformatics*, 22(24):2975–2979, 2006.

[65] 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*, 39:1–38, 1977.

[66] D.B. Díez. Local conditioning in Bayesian networks. *Artificial Intelligence*, 87:1–20, 1996.

[67] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[68] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[69] B. Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78:316–331, 1983.

[70] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–75, 1986.

[71] G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

[72] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems*, 14:681–687, 2002.

[73] B. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics, 2011.

[74] R.A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A*, 222:309–368, 1922.

[75] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[76] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(4):768–780, 1965.

[77] M.J. Forjaz, C. Rodriguez-Blazquez, P. Martínez-Martín, and et al. Rasch analysis of the hospital anxiety and depression scale in Parkinson's disease. *Movement Disorders*, 24(4):526–532, 2009.

[78] P. Franks, D.I. Lubetkin, M.R. Gold, D.J. Tancredi, and H. Jia. Mapping the SF-12 to the EuroQoL EQ-5D index in a national US sample. *Medical Decision Making*, 24(3):247–254, 2004.

[79] N. Friedman. Learning Belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 125–133, 1997.

[80] N. Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, 1998.

[81] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[82] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[83] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The sparse candidate algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[84] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, 2008.

[85] M.M. Gaber. Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):79–85, 2012.

[86] J. Gama. *Knowledge Discovery from Data Streams*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, 2010.

[87] J. Gama. A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[88] J. Gama and G. Castillo. Learning with local drift detection. In *Proceedings of the Second International Conference on Advanced Data Mining and Applications*, pages 42–55, 2006.

[89] J. Gama and P. Kosina. Tracking recurring concepts with meta-learners. In *Proceedings of the Fourteenth Portuguese Conference on Artificial Intelligence*, pages 423–434, 2009.

[90] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Proceedings of the Seventeenth Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[91] J. Gama, R. Sebastião, and P.P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–338, 2009.

[92] J. Gao, B. Ding, W. Fan, J. Han, and P.S. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6):37–49, 2008.

[93] D. Geiger. An entropy-based learning algorithm of Bayesian conditional trees. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 92–97, 1992.

[94] D. Geiger and D. Heckerman. A characterization of Dirichlet distributions through local and global independence. *Annals of Statistics*, 25:1344–1368, 1997.

[95] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[96] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the Fourteenth ACM International Conference on Information and Knowledge Management*, pages 195–200, 2005.

[97] C.N. Glymour and G.F. Cooper. *Computation, Causation, and Discovery.* American Association for Artificial Intelligence Press, Menlo Park, CA, 1999.

[98] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Proceedings of the Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30, 2004.

[99] A. Gordon. A review of hierarchical classification. *Journal of the Royal Statistical*, 150:119–137, 1987.

[100] A. Gray, O. Rivero-Arias, and P. Clarke. Estimating the association between SF-12 responses and EQ-5D utility values by response mapping. *Medical Decision Making*, 26(1):18–29, 2006.

[101] W.H. Greene. *Econometric Analysis.* London, Prentice Hall, 1997.

[102] EuroQol Group. EuroQol -a new facility for the measurement of health- related quality of life. The EuroQol Group. *Health Policy*, 16(3):199–208, 1990.

[103] EuroQuol Group. A1 tariff based on UK survey. *EuroQuol EQ-5D user guide*, 1990.

[104] D. Heckerman. A tutorial on learning with Bayesian networks. *Innovations in Bayesian Networks*, 156:33–82, 2008.

[105] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. Technical report, MSR-TR-94-07, Microsoft Research, Redmond, Washington, 1994.

[106] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[107] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of the Fourth Conference on the Uncertainty in Artificial Intelligence*, pages 149–163, 1988.

[108] E. Herskovits and G. Cooper. Kutató: An entropy-driven system for construction of probabilistic expert systems from database. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62, 1990.

[109] D. Hinkley. Inference about the change-point from cumulative sum tests. *Biometrika*, 58(3):509–523, 1971.

[110] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 2nd edition, New York, 2000.

[111] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.

[112] G. Hulten, L. Spencer, and P. Domingos. Mining time changing data streams. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 97–106, 2001.

[113] I. Inza, P. Larrañaga, R. Blanco, and A.J. Cerrolaza. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, 31(2):91–103, 2004.

[114] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[115] H. Jeffreys. *Theory of Probability*. Oxford University Press, 1939.

[116] C. Jenkinson, R. Fitzpatrick, J. Norquist, L. Findley, and K. Hughes. Cross-cultural evaluation of the Parkinson's Disease Questionnaire: Tests of data quality, score reliability, response rate, and scaling assumptions in the United States, Canada, Japan, Italy, and Spain. *Journal of Clinical Epidemiology*, 56(9):843–847, 2003.

[117] C. Jenkinson, R. Fitzpatrick, V. Petoa, R. Greenhallc, and N. Hymanc. The PDQ-8: Development and validation of a short-form Parkinson's disease questionnaire. *Psychology and Health*, 12(6):805–814, 1997.

[118] F.V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, University College, London, 1996.

[119] V.A. Johnson, F. Brun-Vezinet, B. Clotet, H.F. Gunthard, D.R. Kuritzkes, and et al. Update of the drug resistance mutations in HIV-1: December 2010. *International AIDS Society-USA, Topics in HIV Medicine*, 18(5):156–163, 2010.

[120] M.I. Jordan. *Learning in Graphical Models*. The MIT Press, 1998.

[121] K. Karlsen, E. Tandberg, D. Aarsland, and J. Larsen. Health-related quality of life in Parkinson's disease: A prospective longitudinal study. *Journal of Neurology, Neurosurgery and Psychiatry*, 69(5):584–589, 2000.

[122] I. Katakis, G. Tsoumakas, and I. Vlahavas. Tracking recurring contexts using ensemble classifiers: An application to email filtering. *Knowledge and Information Systems*, 22(3):371–391, 2010.

[123] E.J. Keogh and M. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and non distribution-based approaches. In *Proceedings of the Seventeenth International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

[124] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 180–191, 2004.

[125] R. Klinkenberg. Using labeled and unlabeled data to learn drifting concepts. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 16–24, 2001.

[126] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[127] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning in the presence of concept drifts. In *Workshop Notes of the ICML/AAAI-98 Learning for Text Categorization*, pages 33–40, 1998.

[128] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[129] D. Koller and N. Friedman. *Probabilistic Graphical Models Principles and Techniques*. MIT Press, Cambridge, MA, 2009.

[130] J.Z. Kolter and M.A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.

[131] X. Kong and P.S. Yu. An ensemble-based approach to fast classification of multi-label data streams. In *Proceedings of the Seventh International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 95–104, 2011.

[132] R. Korf. Linear-space best-first search. *Artificial Intelligence*, 62:41–78, 1993.

[133] I. Koychev. Gradual forgetting for adaptation to concept drift. In *Proceedings of the Fourteenth European Conference on Artificial Intelligence, Workshop on Current Issues in Spatio-Temporal Reasoning*, pages 101–106, 2000.

[134] M. Kubat and G. Widmer. Adapting to drift in continuous domains. In *Proceedings of the Eighth European Conference on Machine Learning*, pages 307–310, 1995.

[135] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[136] L.I. Kuncheva and I. Žliobaitė. On the window size for classification in changing environments. *Intelligent Data Analysis*, 13(6):861–872, 2009.

[137] P. Lachenbruch and R. Michey. Estimation error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[138] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[139] S. Lambert-Niclot, P. Flandre, A. Canestri, G. Peytavin, C. Blanc, R. Agher, and et al. Factors associated with the selection of mutations conferring resistance to protease inhibitors (PIs) in PI-experienced patients displaying treatment failure on darunavir. *Antimicrobial Agents and Chemotherapy*, 52(2):491–496, 2008.

[140] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, 1994.

[141] C. Lanquillon. *Enhancing Text Classification to Improve Information Filtering*. PhD thesis, University of Madgdeburg, Germany, 2001.

[142] P. Larrañaga, M. Poza, Y. Yurramendi, R.H. Murga, and C.M.H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.

[143] S.C. Larson. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22:45–55, 1931.

[144] S.L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[145] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[146] W.F. Lawrence and J.A. Fleishman. Predicting EuroQoL EQ-5D preference scores from the SF-12 Health Survey in a nationally representative sample. *Medical Decision Making*, 24(2):160–169, 2004.

[147] Q.A. Le and J.N. Doctor. Probabilistic mapping of descriptive health status responses onto health state utilities using Bayesian networks: An empirical analysis converting SF-12 into EQ-5D utility index in a national US sample. *Medical Care*, 49(5):451–460, 2011.

[148] A.J. Lees, J. Hardy, and T. Revesz. Parkinson's disease. *Lancet*, 373(9680):2055–2066, 2009.

[149] G. Madjarov, D. Kocevb, D. Gjorgjevikja, and S. Džeroskib. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):705–727, 2012.

[150] A.G. Marcelin, B. Masquelier, D. Descamps, J. Izopet, C. Charpentier, C. Alloui, M. Bouvier-Alias, and et al. Tipranavir-Ritonavir genotypic resistance score in protease inhibitor-experienced patients. *Antimicrobial Agents and Chemotherapy*, 52(9):3237–3243, 2008.

[151] P. Martínez-Martín. An introduction to the concept of "quality of life in Parkinson's disease". *Journal of Neurology*, 245(Suppl 1):2–5, 1998.

[152] P. Martínez-Martín, M. Jeukens-Visser, K.E. Lyons, C. Rodriguez-Blazquez, C. Selai, A. Siderowf, and et al. Health-related quality of life scales in Parkinson's disease: Critique and recommendations. *Movement Disorders*, 26(13):2371–2380, 2011.

[153] P. Martínez-Martín, C. Rodriguez-Blazquez, M.M. Kurtis, K. Ray Chaudhuri, and on Behalf of the NMSS Validation Group. The impact of non-motor symptoms on health-related quality of life of patients with Parkinson's disease. *Movement Disorders*, 26(3):399–406, 2011.

[154] M.M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, pages 929–934, 2008.

[155] W.S. McCulloch and W.H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[156] L. Minku, A. White, and X. Yao. The impact of diversity on on-line ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[157] L. Minku and X. Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012.

[158] M. Minsky. Steps towards artificial intelligence. *Computers and Thought*, pages 406–450, 1961.

[159] T. Mitchell. *Machine Learning*. McGraw Hill, New York, USA, 1997.

[160] P. Munteanu and M. Bendou. The EQ framework for learning equivalence classes of Bayesian networks. In *Proceedings of the First IEEE International Conference on Data Mining*, pages 417–424, 2002.

[161] R.E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[162] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134, 2000.

[163] K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In *Proceedings of the Tenth International Conference of Discovery Science*, pages 264–269, 2007.

[164] J. Ortigosa-Hernández, J.D. Rodríguez, L. Alzate, M. Lucania, I. Inza, and J.A. Lozano. Approaching sentiment analysis by using semi-supervised learning of multi-dimensional classifiers. *Neurocomputing*, 92:98–115, 2012.

[165] N.C. Oza and S. Russell. Online bagging and boosting. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112, 2001.

[166] E. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[167] J.M. Peña, J.A. Lozano, and P. Larrañaga. Unsupervised learning of Bayesian networks via estimation of distribution algorithms: An application to gene expression data clustering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(Supplement 1):63–82, 2004.

[168] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[169] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, Cambridge, MA, 2000.

[170] J. Pearl and T. Verma. A theory of inferred causation. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, pages 441–452, 1991.

[171] J. Pearl and T.S. Verma. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

[172] V. Peto, C. Jenkinson, and R. Fitzpatrick. PDQ-39: A review of the development, validation and application of a Parkinson's disease quality of life questionnaire and its associated measures. *Journal of Neurology*, 245(Supplement 1):S10–S14, 1998.

[173] V. Peto, C. Jenkinson, R. Fitzpatrick, and R. Greenhall. The development and validation of a short measure of functioning and well being for individuals with Parkinson's disease. *Quality of Life Research*, 4(3):241–248, 1995.

[174] N. Pise and P. Kulkarni. A survey of semi-supervised learning methods. In *Proceedings of the International Conference on Computational Intelligence and Security*, pages 30–34, 2008.

[175] O. Pourret, P. Naim, and B. Marcot. *Bayesian Networks: A Practical Guide to Applications*. John Wiley and Sons, 2008.

[176] J.L. Powell. Least absolute deviations estimation for the censored regression model. *Journal of Econometrics*, 25:303–325, 1984.

[177] M. Qazi, G. Fung, S. Krishnan, R. Rosales, H. Steck, R.B. Rao, D. Poldermans, and D. Chandrasekaran. Automated heart wall motion abnormality detection from ultrasound images using Bayesian networks. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 519–525, 2007.

[178] W. Qu, Y. Zhang, J. Zhu, and Q. Qiu. Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In *Proceedings of the First Asian Conference on Machine Learning*, pages 308–321, 2009.

[179] M. Ramoni and P. Sebastiani. Learning conditional probabilities from incomplete data: An experimental comparaison. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 260–265, 1999.

[180] J. Read. *Scalable Multi-label Classification*. PhD thesis, The University of Waikato, Hamilton, New Zealand, 2010.

[181] J. Read, A. Bifet, G. Holmes, and B. Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1):243–272, 2012.

[182] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, pages 995–1000, 2008.

[183] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

[184] G. Rebane and J. Pearl. The recovery of causal polytrees from statistical data. In *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*, pages 222–228, 1989.

[185] S.Y. Rhee, M.J. Gonzales, R. Kantor, J. Betts, J. Ravela, and R.W. Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Research*, 31(1):298–303, 2003.

[186] S.Y. Rhee, J. Taylor, W.J. Fessel, D. Kaufman, W. Towner, and et al. HIV-1 protease mutations and protease inhibitor cross-resistance. *Antimicrobial Agents and Chemotherapy*, 54(10):4253–4261, 2010.

[187] J. Rissanen. Modeling by shortest data description. *Automatics*, 14:465–471, 1978.

[188] J.D. Rodríguez and J.A. Lozano. Multi-objective learning of multi-dimensional Bayesian classifiers. In *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, pages 501–506, 2008.

[189] C. Rodriguez-Blazquez, B. Frades-Payo, M.J. Forjaz, J. de Pedro-Cuesta, and et al. Psychometric attributes of the hospital anxiety and depression scale in Parkinson's disease. *Movement Disorders*, 24(4):519–525, 2009.

[190] G.J. Ross, N.M. Adams, D.K. Tasoulis, and D.J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198, 2012.

[191] D. Rowen, J. Brazier, and J. Roberts. Mapping SF-36 onto the EQ-5D index: How reliable is the relationship? *Health and Quality of Life Outcomes*, 31:7–27, 2009.

[192] D. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.

[193] M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

[194] G.C. Sakellaropoulos and G.C. Nikiforidis. Development of a Bayesian network in the prognosis of head injuries using graphical model selection techniques. *Methods of Information in Medicine*, 38(1):37–42, 1999.

[195] M. Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1-5):133–155, 1997.

[196] A. Salmerón, A. Cano, and S. Moral. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413, 2000.

[197] S.G. Sarafianos, B. Marchand, K. Das, D.M. Himmel, M.A. Parniak, S.H. Hughes, and E. Arnold. Structure and function of HIV-1 reverse transcriptase: Molecular mechanisms of polymerization and inhibition. *Journal of Molecular Biology*, 385(3):693–713, 2009.

[198] R.E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

[199] A. Schrag, C. Selai, M. Jahanshahi, and N.P. Quinn. The EQ-5D -a generic quality of life measure- is a useful instrument to measure quality of life in patients with Parkinson's disease. *Journal of Neurology, Neurosurgery and Psychiatry*, 69(1):67–73, 2000.

[200] R. Sebastião and J. Gama. Change detection in learning histograms from data streams. In *Proceedings of the Thirteenth Portuguese Conference on Artificial Intelligence*, pages 112–123, 2007.

[201] R. Sebastião and J. Gama. A study on change detection methods. In *Proceedings of the Fourteenth Portuguese Conference on Artificial Intelligence*, pages 353–364, 2009.

[202] M. Singh. Learning Bayesian networks from incomplete data. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 27–31, 1997.

[203] S.E. Soh, M.E. Morris, and J.L. McGinley. Determinants of health-related quality of life in Parkinson's disease: A systematic review. *Parkinsonism and Related Disordors*, 17(1):1–9, 2011.

[204] D.J. Spiegelhalter and S.L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

[205] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, Cambridge, MA, 2000.

[206] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B*, 36:111–147, 1974.

[207] W.N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

[208] P.W. Sullivan and V. Ghushchyan. Mapping the EQ-5D index from the SF-12: US general population preferences in a nationally representative sample. *Medical Decision Making*, 26(4):401–409, 2006.

[209] L. Tenenboim, L. Rokach, and B. Shapira. Identification of label dependencies for multi-label classification. In *Proceedings of the Second International Workshop on Learning from Multi-Label Data*, pages 53–60, 2010.

[210] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-first Conference of Uncertainty in Artificial Intelligence*, pages 584–590, 2005.

[211] K. Theys, K. Deforche, P. Libin, R. Camacho, K. Van Laethem, and A.M. Vandamme. Resistance pathways of human immunodeficiency virus type 1 against the combination of zidovudine and lamivudine. *Journal of General Virology*, 91(8):1898–1908, 2010.

[212] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multi-label classification of music into emotions. In *Proceedings of the Ninth International Society for Music Information Retrieval Conference*, pages 325–330, 2008.

[213] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[214] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of the First International Workshop on Learning from Multi-Label Data*, pages 101–116, 2009.

[215] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[216] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, pages 30–44, 2008.

[217] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook, Springer*, pages 667–685, 2010.

[218] G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

[219] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical report, TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, 2004.

[220] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[221] N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. *Advances in Neural Information Processing Systems*, 15:721–728, 2002.

[222] L.C. van der Gaag and P.R. de Waal. Muti-dimensional Bayesian network classifiers. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 107–114, 2006.

[223] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, USA, 1995.

[224] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[225] M. von Kleist, S. Menz, and W. Huisinga. Drug-class specific impact of antivirals on the reproductive capacity of HIV. *PLoS Computational Biology*, 6(3):doi:10.1371/journal.pcbi.1000720, 2010.

[226] P. Vorburg and A. Bernstein. Entropy-based concept shift detection. In *Proceedings of the Sixth International Conference on Data Mining*, pages 1113–1118, 2006.

[227] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235, 2003.

[228] Y. Wang, Y. Zhang, and Y. Wang. Mining data streams with skewed distribution by static classifier ensemble. In *Proceedings of the Twenty-second International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 65–71, 2009.

[229] J.M. Whitcomb, N.T. Parkin, C. Chappey, N.S. Hellmann, and C.J. Petropoulos. Broad nucleoside reverse-transcriptase inhibitor cross-resistance in human immunodeficiency virus type 1 clinical isolates. *The Journal of Infectious Diseases*, 188(7):992–1000, 2003.

[230] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley Series in Probability and Mathematical Statistics, 1991.

[231] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[232] J. Winn and C.M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

[233] E.S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas. Dealing with concept drift and class imbalance in multi-label stream classification. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence*, pages 1583–1588, 2011.

[234] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[235] R. Yehezkel and B. Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.

[236] S. Yue, M. Guojun, L. Xu, and L. Chunnian. Mining concept drifts from data streams based on multi-classifiers. In *Proceedings of the Twenty-first International Conference on Advanced Information Networking and Applications Workshops*, pages 253–263, 2007.

[237] J.H. Zaragoza, L.E. Sucar, and E.F. Morales. A two-step method to learn multidimensional Bayesian network classifiers based on mutual information measures. In *Proceedings of the Twenty-Fourth International FLAIRS Conference*, pages 644–649, 2011.

[238] J.H. Zaragoza, L.E. Sucar, E.F. Morales, P. Larrañaga, and C. Bielza. Bayesian chain classifiers for multidimensional classification. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence*, pages 2192–2197, 2011.

[239] M.L. Zhang and Z.H. Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[240] M.L. Zhang and Z.H. Zhou. ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[241] P. Zhang, X. Zhu, and Y. Shi. Categorizing and mining concept drifting data streams. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 812–820, 2008.

[242] X. Zhang, Q. Yuan, S. Zhao, W. Fan, W. Zheng, and Z. Wang. Multi-label classification without the multi-label cost. In *Proceedings of the Tenth SIAM International Conference on Data Mining*, pages 778–789, 2010.

[243] X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2008.

[244] I. Zliobaitė. Learning under concept drift: An overview. Technical report, Faculty of Mathematics and Informatics Vilnius University, 2009.