Feature Subset Selection and Multi-dimensional Continuous-Time Bayesian Network Classifiers for Streaming Data

DOCTORAL THESIS
Submitted for the degree of Doctor by:

Carlos Villa Blanco
MSc Artificial Intelligence

Madrid, 2023
DOCTORAL THESIS
Submitted for the degree of Doctor by:

Carlos Villa Blanco
MSc Artificial Intelligence

Under the supervision of:
Dr. Concha Bielza (Supervisor)
Dr. Pedro Larrañaga (Co-supervisor)

Madrid, 2023
Title: Feature Subset Selection and Multi-dimensional Continuous-Time Bayesian Network Classifiers for Streaming Data

Author: Carlos Villa Blanco

Doctoral Programme: Artificial Intelligence

Thesis Supervision:

   Dr. Concha Bielza, Full Professor (Universidad Politécnica de Madrid) (Supervisor)

   Dr. Pedro Larrañaga, Full Professor (Universidad Politécnica de Madrid) (Co-supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This thesis has been partially supported by the following institutions: the Spanish Ministry of Science and Innovation through the PID2022-139977NB-I00 and TED2021-13130B-I00 projects, the Spanish Ministry of Science, Innovation and Universities through the PID2019-109247GB-I00 and RTC2019-006871-7 projects, the Spanish Ministry of Economy, Industry and Competitiveness through the TIN2016-79684-P project, the Spanish Centre for the Development of Industrial Technology through the IDI-20180156 - LearnIIoT project in partnership with Etxetar and Aingura IIoT, the BBVA Foundation (2019 call) through the project BAYES-CLIMA-NEURO and the Universidad Politécnica de Madrid through a predoctoral contract for the formation of doctors.
A spark at the right time can ignite a story’s rebirth
To my parents and brothers, pillars of my life
Acknowledgement

En primer lugar, quiero expresar mi más profundo agradecimiento a mis supervisores, Pedro y Concha. Sin su guía y apoyo a lo largo de toda esta investigación, este proyecto no habría sido posible. Estoy inmensamente agradecido por la oportunidad de aprender y crecer bajo su dirección.

Mi agradecimiento a la empresa Etxetar, y en particular a Javier Díaz, por proporcionar datasets reales y conocimiento experto que enriqueció significativamente este trabajo. Asimismo, también quiero agradecer a Miguel Ángel Bermeo el valioso intercambio de ideas en proyectos en común y ser mi anfitrión en San Sebastián.

Quiero hacer una mención especial a mis colegas y amigos del MADLab de la Universidad de Milán-Bicocca. Estoy profundamente agradecido a Fabio Stella por su orientación y valioso feedback que han sido de gran valor para esta tesis. Mi estancia en Italia fue una etapa llena de crecimiento personal y profesional gracias también a la hospitalidad y cariño de Alessandro, Alessio, Alice, Emanuele, Michele y todas las personas que tuve el privilegio de conocer, quienes me hicieron sentir como en casa.

Por supuesto, también quiero expresar mi aprecio a todos mis compañeros del laboratorio, tanto actuales como aquellos con los que coincidí en el pasado: Alaia, Bojan, Dani, David Atienza, David Quesada, Enrique, Esteban, Fernando, Gabriel, Irene, Jorge, José Luis, Kevin, Laura, Carlos Li, Mario, Marta, Rafa, Santi, Sebastiaan, Vicente y Víctor. Ellos han sido una parte esencial de esta etapa de mi vida, que espero mantener para la siguiente y para un futuro mural.

Quiero dedicar un agradecimiento muy especial a Charo, Kiko, Diana, Nico y a la pequeña Emma. Al llegar a Madrid, su amistad y compañía fueron indispensables para adaptarme y sentirme en casa. Su apoyo y cariño han sido fundamentales para mí y, por tanto, para este proyecto.

También quiero destacar a Agata en estos agradecimientos. Juntos hemos enfrentado los desafíos de la investigación predoctoral, y su apoyo y los momentos que hemos compartido en este viaje han sido invaluables para mí.

Por último, la razón por la que he podido llegar a realizar esta tesis doctoral se la debo principalmente a mi familia. A mis tíos, Ana, Javi y Mari, Geli y Tino, Geli y Olegario, y mis primos, Ángela, Dani, Jandro, Jose y Pedro. Muy especialmente, quiero agradecer a mis
padres, Carlos y Clara, a mis hermanos, Cristian y David, y a mis abuelos, Ángela, Inés, José Antonio y Pedro, que siempre me han apoyado, guiado y creído en mí, incluso cuando yo no lo hacía. Este trabajo es, en cada página, tanto un reflejo de mi esfuerzo como del amor y apoyo que me han dado. Es tanto mío como suyo.
Abstract

The growing role of machine learning in decision-making stresses the need for transparent and interpretable models. Probabilistic graphical models (PGMs) and feature subset selection (FSS) help meet this demand. PGMs provide a graphical representation that captures complex probabilistic dependencies within data, aiding in comprehending underlying processes and decisions. Meanwhile, FSS simplifies models by focusing on fewer yet relevant variables, thereby improving their interpretability and, potentially, performance. This dissertation explores the application of these techniques to streaming data. Such a setting is common in real-world domains, either because data is received dynamically, requiring solutions to be continuously updated, or because it is crucial to model the temporal dimension.

The first contribution focuses on the multi-dimensional classification of multivariate categorical time series, where sequences of time-ordered data are classified into multiple classes. Despite the benefits of PGMs, they have received no attention to solve this challenging task. Thus, a multi-dimensional continuous-time Bayesian network classifier (Multi-CTBNC) is proposed. This novel PGM extends continuous-time Bayesian networks (CTBNs) to solve the previous task while modeling the probabilistic dependencies of class variables. Experiments yield promising results compared to independent classifiers, and its potential is demonstrated in an Industry 4.0 application.

Structure learning of CTBNs has traditionally relied on score-based algorithms. Yet, a recent constraint-based method has shown advantages in certain settings. Hence, given the inherent strengths of distinct algorithms, diversifying learning strategies is essential to address different data-driven tasks effectively. The second contribution is an in-depth study on Multi-CTBNC structure learning to improve the model versatility, leading to the introduction of the MB-CTPC constraint-based algorithm. This algorithm leverages the model topology to focus on conditional independence tests relevant to the classification task. Furthermore, a pioneering hybrid algorithm is explored, which aims to combine the advantages of score- and constraint-based methodologies. Experiments compare several structure learning algorithms, both on synthetic data and on real-world data from longitudinal socio-economic studies.

In real-world applications, the ever-increasing volume and dynamic nature of data can lead to new instances or variables being continuously generated. FSS proves useful in this context, where high feature dimensionality is common, and variables may be, or become, irrelevant or redundant to learning tasks. The third contribution is a comprehensive review of FSS algorithms for data and feature streams, commonly referred to as incremental or online algorithms, which update selected features as data (or features) evolve. The review discusses
several algorithms, describing in detail their functioning and some shortcomings. A variety of learning problems are covered, including supervised and unsupervised approaches, multi-task and ensemble learning, and rough set-based methodologies, among others.

The final contribution applies Multi-CTBNCs in data and feature stream scenarios, building upon previous contributions. Two frameworks tailored to these environments are presented. For data streams, the focus is on local concept drift detection and model updates to improve the classification accuracy of static models while reducing the learning times of a global update. An online extension of the MB-CTPC is introduced to locally update Multi-CTBNCs. In the case of feature streams, the framework integrates an online FSS algorithm, adapted for categorical time series, to enhance the learning of Multi-CTBNCs. These two methodologies aim to optimize model performance and computational efficiency when learning from dynamic data or in constrained offline settings.
Resumen

El auge del aprendizaje automático en la toma de decisiones resalta la necesidad de modelos transparentes e interpretables. Los modelos gráficos probabilísticos (PGMs) y la selección de variables (FSS) atienden estas demandas. Los PGMs ofrecen una representación gráfica que captura complejas dependencias probabilísticas en los datos, aclarando procesos subyacentes y decisiones. La FSS, en cambio, simplifica los modelos al centrarse en variables relevantes, mejorando su interpretabilidad y, posiblemente, precisión. Esta disertación aborda estas técnicas para datos en streaming. Este escenario es común en dominios reales, bien porque los datos se reciben dinámicamente, requiriendo actualizar constantemente las soluciones, o bien porque es transcendental modelar la dimensión temporal.

La primera contribución trata la clasificación multidimensional de series temporales categóricas multivariantes, donde secuencias de datos ordenados en el tiempo son clasificadas en múltiples clases. Pese a los beneficios de los PGMs, estos no han sido explorados en este ámbito. Surge así el clasificador Bayesiano multidimensional en tiempo continuo (Multi-CTBNC), extendiendo las redes Bayesians en tiempo continuo (CTBNs) para abordar esta tarea considerando las dependencias entre variables clase. Esta propuesta arroja resultados prometedores frente a clasificadores independientes y su potencial se evidencia en una aplicación desarrollada dentro de la Industria 4.0.

El aprendizaje de estructuras de CTBNs ha usado tradicionalmente algoritmos basados en puntuaciones. No obstante, un enfoque reciente basado en restricciones ha destacado en ciertos escenarios. Dado que cada algoritmo tiene sus ventajas, diversificar estrategias de aprendizaje es clave para afrontar eficazmente diferentes problemas. A fin de mejorar la versatilidad del Multi-CTBNC, la segunda contribución profundiza en su aprendizaje estructural. Esto ha resultado en el MB-CTPC, un algoritmo basado en restricciones que aprovecha la topología del modelo para centrarse en tests relevantes para la clasificación. Además, se ha explorado una nueva solución híbrida para integrar las ventajas de las metodologías basadas en puntuaciones y restricciones. Diversos algoritmos se han comparado experimentalmente, tanto en datos sintéticos como en datos reales de estudios socioeconómicos longitudinales.

En escenarios reales, el volumen creciente y la naturaleza dinámica de los datos pueden implicar la generación continua de nuevas instancias o variables. La FSS es útil en este contexto, donde es común una alta dimensionalidad, y las variables pueden ser, o volverse, irrelevantes o redundantes. La tercera contribución es una revisión exhaustiva de algoritmos de FSS incrementales, u online, para flujos de datos y de variables, algoritmos que actualizan las variables seleccionadas conforme los datos (o variables) evolucionan. Múltiples propuestas
son analizadas, detallando su funcionamiento y algunas de sus limitaciones. Además, se abordan distintos problemas de aprendizaje, incluyendo enfoques supervisados y no supervisados, aprendizaje multitarea y por conjuntos, o metodologías basadas en conjuntos aproximados.

La última contribución aplica Multi-CTBNCs en escenarios con flujos de datos o variables, presentando dos métodos para ello. Para flujos de datos, el método se centra en la detección local de cambios de concepto y la actualización del modelo, a fin de mejorar las clasificaciones de un modelo estático y los tiempos de aprendizaje respecto a una actualización global. Una extensión online del MB-CTPC es introducida para actualizar localmente los Multi-CTBNCs. Para flujos de variables, se integra un algoritmo online de FSS, adaptado para series temporales categóricas, con el fin de optimizar el aprendizaje de los Multi-CTBNCs. Ambas metodologías buscan mejorar el rendimiento del modelo y la eficiencia computacional al aprender de datos dinámicos o en entornos offline con restricciones.
# Table of Contents

Acknowledgement ................................................................. v
Abstract ........................................................................ vii
Resumen ........................................................................... ix
List of Figures .................................................................... xiv
List of Tables ....................................................................... xvi
Abbreviations and acronyms ................................................ xix
Notation ............................................................................. xxi

## I INTRODUCTION

1 Introduction ........................................................................ 3
   1.1 Hypotheses and objectives ........................................... 5
   1.2 Document organisation ................................................ 6

## II BACKGROUND

2 Bayesian networks .............................................................. 11
   2.1 Introduction ................................................................... 11
   2.2 Bayesian networks ....................................................... 12
   2.3 Continuous-time Bayesian networks .............................. 16
   2.4 Learning Bayesian networks from data .......................... 19
      2.4.1 Parameter learning .................................................. 20
         2.4.1.1 Bayesian networks ............................................ 20
         2.4.1.2 Continuous-time Bayesian networks .................. 22
      2.4.2 Structure learning ................................................... 24
         2.4.2.1 Score-based structure learning ......................... 24
         2.4.2.2 Constraint-based structure learning .................. 28
         2.4.2.3 Hybrid structure learning ............................... 31
3 Classification algorithms
  3.1 Introduction ................................................. 33
  3.2 Supervised classification .................................. 34
  3.3 Bayesian network classifiers .............................. 35
  3.4 Multi-dimensional classification .......................... 39
  3.5 Continuous-time Bayesian network classifiers .......... 43
  3.6 Performance measures ..................................... 46

4 Feature subset selection and dynamic data
  4.1 Introduction .................................................. 49
  4.2 Feature subset selection .................................... 50
    4.2.1 Relevant and redundant features ...................... 54
  4.3 Data streams ................................................ 57
    4.3.1 Concept drift ......................................... 57
  4.4 Feature streams ............................................ 60
  4.5 Rough set theory .......................................... 61

III CONTRIBUTIONS

5 Multi-dimensional continuous-time Bayesian network classifiers
  5.1 Introduction .................................................. 67
  5.2 Multi-dimensional continuous-time Bayesian network classifier .. 69
  5.3 Learning Multi-CTBNCs from data ........................ 71
    5.3.1 Parameter learning .................................... 71
    5.3.2 Structure learning .................................... 73
      5.3.2.1 Log-likelihood ................................... 74
      5.3.2.2 Conditional log-likelihood ......................... 75
      5.3.2.3 Bayesian Dirichlet equivalent score ............... 76
      5.3.2.4 Structure constraints ............................. 78
  5.4 Classification ............................................. 79
  5.5 Experiments ................................................. 80
    5.5.1 Datasets ............................................... 81
      5.5.1.1 Synthetic datasets ................................ 81
      5.5.1.2 Energy dataset .................................... 82
    5.5.2 Results ................................................. 83
      5.5.2.1 Synthetic datasets ................................ 83
      5.5.2.2 Energy dataset .................................... 86
  5.6 Conclusions and future work ............................ 87
6 Constraint-based and hybrid structure learning of multi-dimensional continuous-time Bayesian network classifiers

6.1 Introduction ................................................................. 91
6.2 Constraint-based structure learning of CTBNs .................................. 93
6.3 Markov blanket-based continuous-time PC algorithm ............................... 94
6.4 Hybrid structure learning algorithm ............................................. 95
6.5 Experiments ....................................................................... 97
   6.5.1 Experimental results on synthetic data ...................................... 98
      6.5.1.1 Hill climbing and tabu search ......................................... 100
      6.5.1.2 CTPC and hill climbing ................................................ 100
      6.5.1.3 MB-CTPC and CTPC .................................................... 101
      6.5.1.4 Hybrid \(|S_{V_iV_j}| = 0| \) and hybrid \(|S_{V_iV_j}| = 1| \) .................. 104
      6.5.1.5 Hybrid \(|S_{V_iV_j}| = 0| \) vs. hill climbing [BIC] and CTPC ....... 104
      6.5.1.6 Robustness to noisy data .............................................. 105
6.5.2 Experimental results on a real-world dataset .................................... 106
6.6 Conclusions and future work ..................................................... 110

7 Feature subset selection for data and feature streams: A review 113

7.1 Introduction ..................................................................... 113
7.2 Classification of incremental FSS approaches .................................... 117
7.3 FSS on data streams .......................................................... 118
   7.3.1 Supervised learning ......................................................... 119
      7.3.1.1 Rough set-based FSS .................................................. 125
      7.3.1.2 FSS for rough set extensions ....................................... 131
      7.3.1.3 Distributed FSS ......................................................... 132
   7.3.2 Unsupervised learning ..................................................... 132
7.4 FSS on feature streams .......................................................... 135
   7.4.1 Individual FSS ............................................................... 136
      7.4.1.1 Supervised one-dimensional learning ............................... 136
      7.4.1.2 Supervised multi-label learning ................................... 144
      7.4.1.3 Unsupervised learning ................................................. 146
   7.4.2 Group FSS ................................................................ 149
      7.4.2.1 Supervised one-dimensional learning ............................... 149
      7.4.2.2 Supervised multi-label learning ................................... 151
      7.4.2.3 Unsupervised learning ................................................. 151
7.5 FSS on data and feature streams .................................................. 152
7.6 Open issues ....................................................................... 153
7.7 Conclusions ..................................................................... 155
8 Supervised feature subset selection and classification on streaming data with multi-dimensional continuous-time Bayesian network classifiers 157

8.1 Introduction ......................................................... 157
8.2 Dynamically updating Multi-CTBNCs in data streams .................. 159
  8.2.1 Local concept drift detection module .......................... 161
  8.2.2 Local update module ........................................... 162
8.3 Multi-dimensional FSS on feature streams ........................... 164
8.4 Experiments ......................................................... 166
  8.4.1 Data streams .................................................... 167
  8.4.2 Feature streams ............................................... 168
  8.4.3 Results ......................................................... 169
    8.4.3.1 Data streams .............................................. 169
    8.4.3.2 Feature streams .......................................... 173
8.5 Conclusions and future work ....................................... 175

IV CONCLUSIONS 181

9 Conclusions and future work ......................................... 183
  9.1 Summary of contributions ......................................... 183
  9.2 List of publications .............................................. 185
  9.3 Software ......................................................... 186
  9.4 Future work ..................................................... 186

References 191
List of Figures

2.1 Loan approval analysis modeled with a BN. ........................................ 14
2.2 D-separation and d-connection between $X_{j-1}$ and $X_{j+1}$ given $X_j$ is observed. 15
2.3 Drug effect graph as a CTBN (taken from Nodelman et al. [2002]). ........... 20

3.1 Illustration of the discussed BNCs, classified by the factorization of the JPD. 38
3.2 Extension of the drug effect graph to a multi-dimensional classification problem (class variables are highlighted in gray). ................................. 41
3.3 Example of the structure of a CTBNC with the class variable highlighted in gray (taken from Codecasa and Stella [2013]). ................................. 46

4.1 Categories of features with respect to a learning task. .......................... 55
4.2 Example of data (top) and feature (bottom) streams of phishing websites. 58

5.1 A Multi-CTBNC graph and its class, bridge and feature subgraphs. .......... 71
5.2 Examples of structures from different Multi-CTBNC families. ................. 78
5.3 Structures from which synthetic datasets were generated. ..................... 81
5.4 Diagram of the industrial machine showing the electrical consumers under study. X-Flex CNC machine architecture courtesy of Etxetar, S.A. ............ 83
5.5 Structure of a DAG-max1 Multi-CTBNC learned from the energy dataset with the BIC score. ................................................................. 88

6.1 Steps of the MB-CTPC algorithm. Dash lines represent removed arcs. ...... 96
6.2 Results of increasing features’ cardinality with sequences of 10 time units. ... 101
6.3 Results of increasing features’ cardinality with sequences of 20 time units. ... 102
6.4 Average increase in learning time, global and mean accuracy and macro F1 score of CTPC over MB-CTPC. ............................................... 103
6.5 Structure learned on the BHPS dataset with the MB-CTPC algorithm (class variables are highlighted in gray). ........................................... 109

7.1 Example of a framework of online FSS algorithms for data streams. ......... 114
7.2 General framework of FSS algorithms for feature streams. .................... 115
7.3 Classification of the analyzed incremental FSS methods. .................... 118
7.4 Multi-task feature selection learning scheme. ................................. 121
7.5 Weighted naive Bayes structure. .................................................. 123

8.1 Framework to locally detect concept drifts and update a Multi-CTBNC when
a new data batch is received from a data stream. ............................... 160
8.2 Framework to process feature streams with an FSS prior to model relearning. 165
8.3 Results of the different updating strategies over a synthetic data stream with
no concept drifts. The detection threshold is set to 1.2 for both the local and
global update methods. ............................................................... 171
8.4 Results of the different updating strategies over a synthetic data stream with
gradual concept drifts. The detection threshold is set to 1.2 for the local update
method and to 1.6 for the global update method. ............................. 173
8.5 Results of the different updating strategies over a synthetic data stream with
abrupt concept drifts. The detection threshold is set to 1.2 for the local update
method and to 1.4 for the global update method. ............................. 175
8.6 Comparison of the structures of a Multi-CTBNC before (a) and after (b) a
concept drift. The graphs at the bottom showcase the evolution of the average
local log-likelihood (c) and Page-Hinkley statistic (d) for each feature node
across different data batches, with a black dashed vertical line indicating the
onset of the concept drift. ................................................................. 176
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Classification of FSS techniques by search strategy.</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>Example dataset of phishing websites.</td>
<td>57</td>
</tr>
<tr>
<td>5.1</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets when learning the models with the BIC score.</td>
<td>84</td>
</tr>
<tr>
<td>5.2</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets when learning the models with the CLL score with BIC penalization.</td>
<td>84</td>
</tr>
<tr>
<td>5.3</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets when learning the models with the BDe score.</td>
<td>85</td>
</tr>
<tr>
<td>5.4</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets when learning using hyperparameters $\lambda_{c_j}^{pa(C_t)} = 1$, $\alpha_{x_j,x_t}^{pa(X_f)} = 100$ and $\tau_{x_j}^{pa(X_f)} = 10$.</td>
<td>85</td>
</tr>
<tr>
<td>5.5</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets when learning an empty-digraph Multi-CTBNC with the BDe score.</td>
<td>86</td>
</tr>
<tr>
<td>5.6</td>
<td>Model learning times in seconds (mean $\pm$ std. deviation) on all synthetic datasets.</td>
<td>86</td>
</tr>
<tr>
<td>5.7</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the energy dataset.</td>
<td>87</td>
</tr>
<tr>
<td>6.1</td>
<td>Parameters used to generate the datasets for the experiments.</td>
<td>98</td>
</tr>
<tr>
<td>6.2</td>
<td>Percentage of datasets in which structure learning algorithms achieve better, worse or identical results.</td>
<td>99</td>
</tr>
<tr>
<td>6.3</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over the synthetic datasets.</td>
<td>100</td>
</tr>
<tr>
<td>6.4</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) over synthetic datasets generated from Multi-CTBNCs with 30 and 5 feature and class variables, respectively.</td>
<td>102</td>
</tr>
<tr>
<td>6.5</td>
<td>Estimated performance measures (mean $\pm$ std. deviation) when increasing noise in synthetic datasets.</td>
<td>106</td>
</tr>
</tbody>
</table>
6.6 Estimated performance measures (mean ± std. deviation) with ten cross-validations on the BHPS dataset. ................................. 108

7.1 Supervised learning algorithms for FSS on data streams. ......................... 133
7.2 Unsupervised learning algorithms for FSS on data streams. ................. 135
7.3 Supervised learning algorithms for individual FSS on feature streams. ... 147
7.4 Unsupervised learning algorithms for individual FSS on feature streams. .. 149
7.5 Supervised learning algorithms for group FSS on feature streams. ......... 151
7.6 Supervised learning algorithms for FSS on data and feature streams. ...... 153

8.1 Estimated performance measures (mean ± std. deviation) over ten stationary synthetic data streams. ................................................. 170
8.2 Estimated performance measures (mean ± std. deviation) over ten synthetic data streams with gradual concept drifts. ................. 172
8.3 Estimated performance measures (mean ± std. deviation) over ten synthetic data streams with abrupt concept drifts. ...................... 174
8.4 Estimated performance measures (mean ± std. deviation) when using different methods to process the feature streams. ....................... 177
Abbreviations and acronyms

The following list provides an overview of the most relevant abbreviations and acronyms of this dissertation:

- AIC Akaike information criterion
- BDe Bayesian Dirichlet equivalent
- BIC Bayesian information criterion
- BN Bayesian network
- BAN Bayesian network-augmented naive Bayes
- BNC Bayesian network classifier
- BR Binary relevance
- BHPS British Household Panel Survey
- CIM Conditional intensity matrix
- CLL Conditional log-likelihood
- CPD Conditional probability distribution
- CPT Conditional probability table
- CTBN Continuous-time Bayesian network
- CTBNC Continuous-time Bayesian network classifier
- CTNBC Continuous-time naive Bayes classifier
- CTPC Continuous-time PC
- DAG Directed acyclic graph
- DBN Dynamic Bayesian network
- DTW Dynamic time warping
- FSS Feature subset selection
JPD Joint probability distribution

\(k\text{-DB}\) \(k\)-dependence Bayesian classifier

\(k\text{NN}\) \(k\)-nearest neighbours

LP Label powerset

LL Log-likelihood

LML Log-marginal-likelihood

MB Markov blanket

MB-CTPC Markov blanket-based continuous-time PC

MAP Maximum a posteriori

MLE Maximum likelihood estimation

MBC Multi-dimensional Bayesian network classifier

Multi-CTBNC Multi-dimensional continuous-time Bayesian network classifier

PC Peter-Clark

PGM Probabilistic graphical model

RAkEL Random k-labelsets

TAN Tree-augmented naive Bayes
Notation

The following list provides the main notations used throughout this dissertation:

\[ X = \{X_1, \ldots, X_m\} \] set of \( m \) feature variables
\[ C = \{C_1, \ldots, C_d\} \] set of \( d \) class variables
\[ \Omega_X = \{x_1, \ldots, x_k\} \] sample space of variable \( X \) with \( k \) distinct values
\[ I_X = \Omega_{X_1} \times \cdots \times \Omega_{X_m} \] joint sample space of variables \( X \)
\[ |x| \] absolute value of \( x \)
\[ |X| \] cardinality of set \( X \)
\[ z_i = (x_i, c_i) \] \( i \)-th instance
\[ x_i = (x_{i1}, \ldots, x_{im}) \] feature values for the \( i \)-th instance
\[ c_i = (c_{i1}, \ldots, c_{id}) \] class variable values (class configuration) for the \( i \)-th instance
\[ S_p = \{x_p^{t_1}, \ldots, x_p^{t_p}\} \] multivariate sequence (trajectory) \( p \) with \( T_p \) observations
\[ D \] a dataset, consisting of \( N \) instances or sequences as the context dictates
\[ z^t = (x^t, c^t) \] instance received at time \( t \) from a data stream
\[ F^t = \{X^t_j, 1 \leq j \leq m_t\} \] group of \( m_t \) features arriving at time \( t \) from a feature stream
\[ P(X) \] probability distribution of \( X \)
\[ P(x) \] shorthand for \( P(X = x) \)
\[ P(X_1, X_2) \] joint probability distribution of \( X_1 \) and \( X_2 \)
\[ P(X_1 | X_2) \] conditional probability of \( X_1 \) given \( X_2 \)
\[ X_1 \perp \perp X_2 \] \( X_1 \) is independent of \( X_2 \)
\[ X_1 \not\perp \not\perp X_2 \] \( X_1 \) is not independent of \( X_2 \)
\[ X_1 \perp \perp X_2 | X_3 \] \( X_1 \) is conditionally independent of \( X_2 \) given \( X_3 \)
\[ X_1 \not\perp \not\perp X_2 | X_3 \] \( X_1 \) is not conditionally independent of \( X_2 \) given \( X_3 \)
\[ \mathcal{G} \] structure of a probabilistic graphical model
\[ \text{Pa}(X) \] parent nodes of \( X \) in \( \mathcal{G} \)
\( \text{pa}(X) \) instantiation of nodes in \( \text{Pa}(X) \)

\( \text{Pa}_C(X) \) nodes in set \( C \) which are parents of \( X \) in \( G \)

\( \text{De}(X) \) descendant nodes of \( X \) in \( G \)

\( \text{Sp}(X) \) spouses (co-parents) of a node \( X \) in \( G \)

\( \mathbf{B} \) parameters of a Bayesian network

\( \hat{\mathbf{B}} \) maximum likelihood estimate of the parameters of a Bayesian network

\( \mathcal{Q}^\text{Pa}(X) \) conditional intensity matrix of \( X \) given \( \text{Pa}(X) \)

\( q \) parameters denoting the intensities of a continuous-time Bayesian network

\( \Theta \) parameters denoting the transition probabilities of a continuous-time Bayesian network

\( \beta^\text{pa}(X) \) parameter within set \( \mathbf{B} \) denoting the probability of a discrete variable \( X \) taking state \( x \) given \( \text{pa}(X) \)

\( q^\text{pa}(X) \) parameter within set \( q \) denoting the intensity of \( X \) leaving state \( x_a \) given \( \text{pa}(X) \)

\( \theta^\text{pa}(X) \) parameter within set \( \Theta \) denoting the probability of \( X \) transitioning from state \( x_a \) to \( x_b \) when a transition is known to occur and given \( \text{pa}(X) \)

\( N^\text{pa}(C) \) number of sequences for which a class variable \( C \) takes state \( c \) given \( \text{pa}(C) \)

\( N^\text{pa}(C) = \sum_c N^\text{pa}(C) \) number of sequences where parents of \( C \) have state \( \text{pa}(C) \) independently of the state of \( C \)

\( M^\text{pa}(X)_{x_a,x_b} \) number of transitions of feature \( X \) from state \( x_a \) to \( x_b \) when its parents’ state is \( \text{pa}(X) \)

\( M^\text{pa}(X)_{x_a,x_b} = \sum_{x_a \neq x_b} M^\text{pa}(X)_{x_a,x_b} \) number of transitions of \( X \) from state \( x_a \) to any other when its parents’ state is \( \text{pa}(X) \)

\( T^\text{pa}(X) \) time of \( X \) spent in state \( x \) when its parents’ state is \( \text{pa}(X) \)

\( S \subseteq \mathcal{X} \) subset of selected features or reduct

\( S^t \subseteq \mathcal{X} \) subset of selected features or reduct at time \( t \)

\( \mathcal{U} \) universe (set of instances in rough set theory)

\( \mathcal{U}/\text{IND}(\mathcal{X}) = \{ \mathcal{E}_1, \ldots, \mathcal{E}_v \} \) indiscernibility relation given a feature set \( \mathcal{X} \)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{E}$</td>
<td>Equivalence class</td>
</tr>
<tr>
<td>$\text{POS}_{X}(C)$</td>
<td>Positive region for feature set $X$ given class variables $C$</td>
</tr>
<tr>
<td>$\text{NEG}_{X}(C)$</td>
<td>Negative region for feature set $X$ given class variables $C$</td>
</tr>
<tr>
<td>$\text{BND}_{X}(C)$</td>
<td>Boundary region for feature set $X$ given class variables $C$</td>
</tr>
<tr>
<td>$\mathbf{w}$ ($\mathbf{W}$)</td>
<td>Bold lower (upper) case letters denote vectors (matrices)</td>
</tr>
</tbody>
</table>
Part I

INTRODUCTION
Chapter 1

Introduction

Machine learning algorithms are increasingly being deployed in critical decision-making domains, including healthcare diagnostics [Jothi et al., 2015], criminal justice [Kovalchuk et al., 2022], financial forecasting [Kumar et al., 2016], and industrial systems [Ruiz-Sarmiento et al., 2020], among others. While complex black-box models, such as neural networks, often deliver high levels of predictive accuracy, their lack of transparency hinders our ability to understand and validate the rationale behind their decisions. This opacity is particularly problematic in sensitive areas like healthcare and criminal justice [Rudin, 2019], where the consequences of decisions can have a significant impact on society and explaining algorithmic decisions may even be a legal requirement [Bibal et al., 2021]. Consequently, understanding how predictive models make decisions is becoming increasingly important. Researchers like Rudin [2019] discuss the need to develop inherently interpretable models instead of trying to explain black-box solutions, suggesting that, although complex models are believed to be more accurate, this is not always the case. To address these challenges, this dissertation focuses on developing a novel probabilistic graphical model (PGM) classifier that seeks both high accuracy and interpretability. Furthermore, we explore the role of feature subset selection (FSS) in dynamic data environments, aiming to provide models that are not only easier to understand but also more efficient and adaptable, particularly in settings with limited computational resources.

High-dimensional data is a common challenge in real-world problems, adding complexity to both modeling and analysis. Often, this high dimensionality includes irrelevant and redundant feature variables\(^1\) that could introduce noise rather than potentially improve the performance of machine learning algorithms. Reducing this dimensionality is essential for several reasons, such as computational efficiency and model performance, but also to enhance

\(^1\)Throughout this dissertation, the term feature variables will often be referred to simply as features for brevity and simplicity.
the interpretability of the models. FSS techniques are a key tool in this context, as they aim to select a subset of relevant features while removing the irrelevant and redundant ones. Unlike methods that transform the feature space, FSS maintains the original meaning and context of the variables, making it easier to understand and explain the decisions of the learning models. In the dynamic environments characterized by data streams and feature streams [Swan, 2012; Wang et al., 2014, 2017; Yu et al., 2016a], which are studied in this dissertation, the need for FSS techniques becomes even more pronounced. As new data continuously flow, the relevance of features can fluctuate, leading to the need for models that can adapt quickly and efficiently. Incremental FSS techniques, also known in some contexts as online, are particularly useful in this scenario, as they continuously identify and retain the most informative features. Apart from enhancing model interpretability and potentially improving performance, FSS helps to reduce computational resource usage, which is especially critical in dynamic environments where real-time decision-making is often required and resource utilization must be optimized.

The complexity of this temporal nature of real-world data does not end with the mere reception of new data over time. These data can contain complex temporal dependencies whose capture is essential to comprehend the underlying processes and successfully perform learning tasks [Tsay, 2005; Juang et al., 2017; Susto et al., 2018; Kim and Pyun, 2020]. Thus, the temporal dynamics within these data require specialized modeling techniques for effective understanding and reliable prediction. For this dissertation, the term streaming data seeks to encompass those contexts in which temporality has an important role in the learning processes, either because a time variable contains relevant information for the accurate modeling of the problem (time series data) or because the data are received progressively over time, and learning models must adapt to them (data and feature streams). In the case of time series data, this dissertation seeks to solve a novel real-world problem from the context of Industry 4.0, where modeling the temporal dependencies of the data is a requirement to properly solve a multi-dimensional classification task.

In this dissertation, one of our objectives is to introduce a new PGM specialized in the multi-dimensional classification of multivariate time series. Our focus on this learning task is motivated by the previously mentioned real-world problem faced by a company from the industrial sector. For this purpose, we will rely on the paradigm of continuous-time Bayesian networks (CTBNs), an extension of traditional Bayesian networks (BNs) capable of modeling time series in a continuous-time context. In addition to formally introducing this new model, the dissertation describes techniques for directly learning its parameters and structure from data. This includes the introduction of novel structure learning algorithms specifically designed for the proposed PGM. Moreover, this dissertation also studies the data stream and feature stream environments by conducting a comprehensive study on incremental FSS algorithms for these settings. The findings from this latter study enable us to adapt our model and a structure learning algorithm to the data stream setting and assess the utility of
implementing FSS within the feature stream context.

Chapter outline

This chapter is organized as follows. Section 1.1 presents the research hypotheses and main objectives of the dissertation. Then, Section 1.2 briefly describes the organization of the document.

1.1 Hypotheses and objectives

The work of this dissertation was conducted around the following five research hypotheses:

- **H1.** In the context of multi-dimensional classification of multivariate time series, applying a novel PGM classifier based on CTBNs, adapted to the multi-dimensional setting, will yield better results than state-of-the-art solutions in terms of both classification performance, which is assessed by performance measures specially designed for multi-dimensional classification problems, and also learning time.

- **H2.** When applied to a novel real-world problem in the context of Industry 4.0, the proposed classifier will not only outperform current state-of-the-art solutions in terms of classification performance but also provide more interpretable insights into the problem in a single model due to its ability to capture dependencies between class variables.

- **H3.** Under specified conditions, novel constraint-based and hybrid structure learning algorithms tailored for the multi-dimensional extension of continuous-time Bayesian network classifiers (CTBNCs) will outperform existing state-of-the-art solutions in selected performance metrics and structure learning time.

- **H4.** In specific scenarios characterized by non-stationary data streams where concept drifts occur, employing local updates to our proposed multi-dimensional CTBNC model will maintain or improve predictive accuracy and significantly reduce the time required for model updates compared to a global update method.

- **H5.** In a feature stream scenario, applying FSS algorithms to the categorical time series datasets prior to the learning of our proposed classifier will enable the processing of larger feature streams and reduce learning times.

Based on these hypotheses, the main objectives pursued in this dissertation are:

- **O1.** To develop a novel PGM classifier based on CTBNs specifically designed for the multi-dimensional classification of categorical time series data, capable of simultaneously
classifying multiple class variables while capturing their probabilistic dependencies.

- **O2.** To adapt existing constraint-based structure learning algorithms for the learning of CTBNCs and develop a novel algorithm of this kind specifically tailored for the proposed multi-dimensional classifier.

- **O3.** To develop a hybrid algorithm for learning the structure of CTBNs and their classification variants that attempts to combine the strengths of both score-based and constraint-based solutions.

- **O4.** To conduct a comprehensive review of FSS algorithms for data and feature streams, analyzing their functioning and shortcomings in diverse learning contexts and clarifying terminological ambiguities. This research also aims to identify current open issues to guide future work in improving algorithmic efficiency, robustness or versatility in increasingly diverse dynamic environments.

- **O5.** To adapt an online FSS algorithm from the state of the art to process categorical time series data in a feature stream scenario.

- **O6.** To develop a local update method for the proposed multi-dimensional classifier and a local concept drift detection technique to address multi-dimensional classification problems in non-stationary data streams of categorical time series data.

- **O7.** To empirically evaluate our proposed multi-dimensional classifier and learning algorithms using synthetic and real-world datasets against state-of-the-art methods. Performance measures specific to multi-dimensional classification will be used to assess classification performance, learning time, and classification time.

- **O8.** To demonstrate the practical applicability of the proposed classifier through its implementation on real-world datasets sourced from Industry 4.0 domains and longitudinal socio-economic studies, where multi-dimensional classification problems need to be addressed.

- **O9.** To implement an open-source software solution that allows the application and testing of the developed models and learning algorithms, as well as the visualization of the proposed models and their results for easier interpretation.

### 1.2 Document organisation

This dissertation is divided into four parts consisting of nine chapters, which are organized as follows.
Part I. Introduction

The first part of the manuscript introduces the dissertation.

- Chapter 1 provides an overview of the learning context of this dissertation, outlines its research hypotheses and main objectives, and presents the organization of the manuscript.

Part II. Background

This part consists of three chapters that introduce the basic theoretical foundations necessary to understand the contributions of this dissertation.

- Chapter 2 serves as a comprehensive introduction to BNs and CTBNs, covering their fundamentals along with methods for learning their parameters and structure from data.
- Chapter 3 introduces the basics of supervised classification, focusing on Bayesian network classifiers (BNCs) and their adaptations for multi-dimensional and time-series classification problems. The chapter also provides an overview of the performance measures used in the experimental research of this dissertation.
- Chapter 4 defines the purpose of FSS, presenting a detailed overview of some techniques. It offers formal probabilistic definitions of relevant, irrelevant, and redundant variables in a supervised context and discusses different criteria for feature relevance and redundancy in unsupervised settings. The chapter also introduces the application of FSS within the dynamic contexts of data streams and feature streams.

Part III. Contributions

This part contains the four chapters corresponding to the contributions of the dissertation.

- Chapter 5 studies multi-dimensional classification in the context of multivariate time series data. It introduces a novel multi-dimensional CTBNC (Multi-CTBNC) capable of capturing the temporal dynamics of features in continuous time while modeling probabilistic dependencies among multiple class variables and between these and the features. The chapter also extends learning algorithms of the state of the art for estimating the parameters and structure of this new predictive model directly from data.
- Chapter 6 focuses on the development of novel structure learning algorithms specifically designed for Multi-CTBNCs. This chapter introduces a constraint-based and a hybrid
structure learning algorithm, marking the first effort in this direction for CTBNCs.

- Chapter 7 provides a comprehensive review of FSS algorithms designed for dynamic environments. It categorizes dynamic data into data streams, feature streams or a hybrid of both, and reviews and classifies FSS algorithms for each scenario based on their application to different learning paradigms. This chapter also discusses limitations of existing methods and identifies open research areas for future studies in the field.

- Chapter 8 explores the application of Multi-CTBNCs to both data streams and feature streams, extending the study on dynamic data from Chapter 7. This chapter introduces two frameworks for learning Multi-CTBNCs in these specific environments. The frameworks address several challenges: in the case of data streams, the framework focuses on performing localized updates to the model whenever necessary; for feature streams, it employs online FSS algorithms prior to model learning. These strategies aim to reduce the complexity and computational resources required to relearn Multi-CTBNCs on dynamic data.

Part IV. Conclusions

This part serves as the conclusion of the dissertation, presenting the principal findings and suggesting future lines of research.

- Chapter 9 summarizes the most significant conclusions of our work, discusses possible lines of future research, and lists all the publications and software produced as part of this dissertation.
Part II

BACKGROUND
Chapter 2

Bayesian networks

2.1 Introduction

PGMs represent a powerful framework for modeling complex systems under uncertainty, a common aspect of real-world problems [Koller and Friedman, 2009]. These models combine probability and graph theory to provide an intuitive graphical tool to represent and reason about probabilistic relationships between variables. The application of PGMs in this dissertation is inspired by a real-world industrial classification problem presented by a partner company. Industrial experts required not only a predictive model that was accurate but also transparent enough to provide valuable information about their systems. In industrial environments, interpretable models not only facilitate informed decision-making for aspects such as process optimization but could also assist in anomaly detection for predictive maintenance [Larrañaga et al., 2018; Dalzochio et al., 2020; Vollert et al., 2021].

In those environments, data are often more than just a set of independent observations. Time series data can be generated from sensor readings that continuously monitor various aspects of their processes, such as temperature, hydraulic pressure, servo activation, machine vibrations, product quality or energy consumption. This temporal nature of industrial data goes beyond the mere ordering in time. They often contain complex temporal dependencies and patterns whose capture is crucial to understanding the underlying processes and applying learning tasks. Commonly used PGMs like BNs lack the capabilities to capture these temporal dynamics. Therefore, this dissertation focuses on CTBNs, a specialized extension of BNs capable of modeling time-dependent relationships, providing a more fitting framework for handling the previous real-world industrial problem.

This background chapter will study the basics of BNs and CTBNs and their learning from data. The theoretical fundamentals presented in the chapter serve as the foundation for introducing
our contributions in Chapters 5, 6 and 8. These chapters present and experimentally study a novel PGM, which is based on the CTBN framework, to solve multi-dimensional classification problems while capturing temporal dependencies in the data. Specifically, in Chapter 5, the proposed model is formally introduced and employed to solve the real-world industrial problem previously mentioned. This model not only successfully fulfills the classification task by improving a state-of-the-art solution but also provides an interpretable framework that offers valuable insights for industrial experts. Then, Chapter 6 introduces novel structure learning algorithms for the model, with one of them being particularly useful on another real-world dataset. Lastly, Chapter 8 evaluates the model’s performance in scenarios involving data or feature streams. This chapter presents an extension of one of the introduced structure learning algorithms, which is designed to address concept drifts in data streams by locally updating the model.

In this dissertation, we explore the application of BNs as classifiers. While inference in these models can be employed for a wide range of learning tasks, our focus is limited to employing inference to solve classification problems, whose fundamentals will be covered in Chapter 3. The reader is referred to works such as Koller and Friedman [2009], Nodelman et al. [2002] and Nodelman et al. [2005] for a deeper understanding of inference in BNs and CTBNs.

Chapter outline

The remainder of this chapter is as follows. Section 2.2 covers the fundamentals of BNs, highlighting how they capture conditional independence to model data relationships. Section 2.3 introduces the framework of CTBNs, elaborating on the specific types of temporal data they model and the motivations behind their conceptualization. Finally, Section 2.4 describes how to learn BNs and CTBNs from data, explaining some approaches for learning their parameters and structure.

2.2 Bayesian networks

A BN is a PGM that encodes conditional independence assumptions over some random variables to obtain a factorized version of their joint probability distribution (JPD) [Pearl, 1988]. These models have been widely used in a variety of domains [Beinlich et al., 1989; Luo et al., 2005; Correa et al., 2008; DeFelipe et al., 2013], and their learning from data with different algorithms constitutes an important active research area. Some reasons for their success are the graphical representation of uncertainty, from which we could even learn causal relationships under certain conditions, that they can handle incomplete data, which are common in real-world problems, or that they can combine expert and data-extracted
knowledge [Heckerman, 2008; Bielza and Larrañaga, 2014].

In probability theory, a JPD describes the likelihood of two (or more) random variables taking on specific values jointly, providing probabilities for every combination. Based on the definition of the conditional probability distribution (CPD) of $X_1$ given $X_2$:

$$P(X_1|X_2) = \frac{P(X_1, X_2)}{P(X_2)},$$

a JPD can be decomposed into a product of conditional probabilities by the chain rule of probability.

**Definition 2.1. (Chain rule).** Given an ordered sequence of random variables $\mathcal{X} = \{X_1, X_2, X_3, \ldots, X_m\}$, their JPD can be expressed as a product of the conditional probability of each variable given its predecessors:

$$P(\mathcal{X}) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \cdots P(X_m|X_1, X_2, \ldots, X_{m-1}).$$

In scenarios with multiple variables, dealing directly with JPDs and their parameters can quickly become complex due to the exponential growth of combinations of the variables’ values. Nevertheless, variables may exhibit conditional independence relationships with each other, greatly simplifying the chain rule decomposition and reducing the number of parameters. As the number of parameters required to specify a full JPD grows exponentially with the number of variables, assuming conditional independence relations between triplets of the variables simplifies the computation of the JPD and avoids its intractability in practical applications.

**Definition 2.2. (Conditional independence).** Two random variables $X_1$ and $X_2$ are said to be conditionally independent given a third variable $X_3$, i.e., $X_1 \perp \perp X_2|X_3$, with respect to a distribution $P$, iff:

$$P(X_1, X_2|X_3) = P(X_1|X_3)P(X_2|X_3).$$

The concept of conditional independence is essential in BNs, which provides a means to represent a JPD in a more compact and computationally efficient manner. This fact not only facilitates probabilistic reasoning but also makes BNs more interpretable.

**Definition 2.3. (Bayesian network).** A BN $\mathcal{B} = (\mathcal{G}, \mathcal{B})$ over a set of random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ consists of a directed acyclic graph (DAG) $\mathcal{G}$ encoding conditional independence assumptions among triplets of the variables and a set of parameters $\mathcal{B}$ defining the probabilistic relationships between them. This allows to factorise the JPD over $\mathcal{X}$ via the chain rule for BNs:

$$P(X_1, \ldots, X_m|\mathcal{G}, \mathcal{B}) = \prod_{j=1}^{m} P\left(X_j|\text{Pa}(X_j), \mathcal{B}_{\text{Pa}(X_j)}^{X_j}\right),$$
where $\text{Pa}(X_j)$ are the parent variables of $X_j$ in $\mathcal{G}$, i.e., variables pointing at $X_j$ in $\mathcal{G}$, and $B_{X_j}^{\text{Pa}(X_j)}$ represents the conditional probability (discrete or continuous) distribution of $X_j$ given $\text{Pa}(X_j)$.

**Example 2.1.** Figure 2.1 illustrates a hypothetical example of a BN for assessing loan approval probabilities. The model includes four categorical nodes. Each node is associated with a conditional probability table (CPT) that quantifies the probabilities of its outcomes given its parents’ states. For example, the CPT for Loan approval considers both Employment status and Has debts, providing a probabilistic assessment of loan approval outcomes based on these factors.

Once we have the structure of a BN, either because it has been previously inferred with expert knowledge or learned directly from data, we can extract the conditional independence relationships of the variables it models. The d-separation criterion defines a set of rules to identify conditional independencies [Geiger et al., 1990; Murphy, 2012].

---

1Employment status.
Definition 2.4. (D-separation). Given three sets of disjoint nodes $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ from graph $\mathcal{G}$, $\mathcal{X}_1$ and $\mathcal{X}_2$ are d-separated given $\mathcal{X}_3$, i.e., $\text{d-sep}(\mathcal{X}_1 \perp \mathcal{X}_2 | \mathcal{X}_3)_\mathcal{G}$, iff no active trail by $\mathcal{X}_3$ exists, i.e., all undirected sequences of edges (trails) between nodes in $\mathcal{X}_1$ and $\mathcal{X}_2$ are blocked by $\mathcal{X}_3$. A trail is said to be active between $\mathcal{X}_1$ and $\mathcal{X}_2$ given $\mathcal{X}_3$ if there is a flow of probabilistic influence from $\mathcal{X}_1$ to $\mathcal{X}_2$ (or vice-versa) via $\mathcal{X}_3$. Given all possible trails $\mathcal{T}$ between $\mathcal{X}_1$ and $\mathcal{X}_2$, a trail $T \in \mathcal{T}$ is said to be blocked by $\mathcal{X}_3$ if any of the following conditions are met:

- $T$ contains a chain $X_{j-1} \rightarrow X_j \rightarrow X_{j+1}$ such that node $X_j \in \mathcal{X}_3$ (as illustrated in Figure 2.2a).
- $T$ contains a fork $X_{j-1} \leftarrow X_j \rightarrow X_{j+1}$ such that node $X_j \in \mathcal{X}_3$ (as illustrated in Figure 2.2b).
- $T$ contains a $v$-structure or collider $X_{j-1} \rightarrow X_j \leftarrow X_{j+1}$ such that node $X_j \notin \mathcal{X}_3$ and none of its descendants $\text{De}(X_j) \notin \mathcal{X}_3$ (Figure 2.2c shows an active trail in a $v$-structure).

Conversely, if there is an active trail between two sets of nodes (as in Figure 2.2c), then these nodes are said to be d-connected.

When it comes to inferring information about a particular variable, it is not necessary to consider all the other nodes in $\mathcal{G}$. The Markov blanket of a node $X_j$ in a BN consists of all those nodes that d-separate $X_j$ from the remaining variables in $\mathcal{G}$ [Koller and Friedman, 2009].

Definition 2.5. (Markov blanket). Given a graph $\mathcal{G}$ over a set of nodes $\mathcal{X}$, the Markov blanket $\text{MB}(X_j)$ in $\mathcal{G}$ of a node $X_j$ is given by the set of parents $\text{Pa}(X_j)$, children $\text{Ch}(X_j)$,
Carlos Villa Blanco

and spouses\(^2\) \(\text{Sp}(X_j)\) of \(X_j\), i.e.:

\[
\text{MB}(X_j) = \text{Pa}(X_j) \cup \text{Ch}(X_j) \cup \text{Sp}(X_j),
\]

such that

\[
P(X_j|\mathcal{X} \setminus \{X_j\}) = P(X_j|\text{MB}(X_j)).
\]

If all the conditional independence relationships that are encoded in \(\mathcal{G}\) are also held in a given JPD \(P\), then \(\mathcal{G}\) is said to be an independence map (I-map) of \(P\) [Verma and Pearl, 1990a]. In that case, if two nodes \(X_1\) and \(X_2\) are d-separated by another node \(X_3\) in \(\mathcal{G}\), then \(X_1\) and \(X_2\) are also conditional independent given \(X_3\) in \(P\).

**Definition 2.6. (Independence map).** Given any three disjoint sets of nodes \(\mathcal{X}_1\), \(\mathcal{X}_2\) and \(\mathcal{X}_3\), the graph \(\mathcal{G}\) of a BN is said to be the I-map of a JPD \(P\) iff:

\[
d-sep(\mathcal{X}_1 \perp \perp \mathcal{X}_2|\mathcal{X}_3) \Rightarrow (\mathcal{X}_1 \perp \perp \mathcal{X}_2|\mathcal{X}_3)_{\mathcal{P}}.
\]

However, it is possible to observe that, under this definition, a complete graph is an I-map of any joint distribution since it does not encode any conditional independence relation. Thus, we may aim to find the minimal I-map (which may not be unique), which provides a concise representation of the probabilistic relationships in the data [Verma and Pearl, 1990a].

**Definition 2.7. (Minimal I-map).** A graph \(\mathcal{G}\) that is an I-map of \(P\) is said to be minimal if removing any of its arcs implies that it is no longer an I-map, i.e., there is no graph \(\mathcal{G}' \subseteq \mathcal{G}\) that is an I-map of \(P\).

### 2.3 Continuous-time Bayesian networks

The classical definition of a BN was designed for reasoning about static processes [Pearl, 1988], where samples are assumed to be independent of each other. Thus, such a model is unsuitable when a system exhibits temporal behavior. A time series is a collection of time-ordered observations over one (univariate) or multiple (multivariate) variables. The analysis and processing of time series data are present in practically any field of study, such as engineering, medicine, economics, signal processing or cybersecurity, and are essential to understand and automate many of their processes [Tsay, 2005; Juang et al., 2017; Susto et al., 2018; Liu et al., 2019]. Therefore, this limitation of BNs becomes an issue when addressing dynamic data with a temporal dimension since we would like to model a system that evolves over time.

Part of this dissertation focuses on studying multivariate time series with discrete-valued

\(^2\)The spouses (or co-parents) of a node \(X_j\) are the other parents of its children.
variables, meaning that, at a given time, variables can take a state from a finite set of possibilities. While variables' states might exhibit a natural order, this order is irrelevant to our contributions. Therefore, for simplicity, we will refer to these data as categorical time series [Fokianos and Kedem, 2003; Elzinga, 2010; Frühwirth-Schnatter and Pamminger, 2010]. The prediction model that will be proposed and experimentally studied in Chapters 5, 6 and 8 of this dissertation can be learned and applied over datasets of categorical time series. Therefore, the characteristics of those datasets need to be formally introduced.

**Definition 2.8. (Categorical time series dataset).** A categorical time series dataset \( \mathcal{D} = \{S_1, \ldots, S_N\} \) consists of multiple multivariate temporal sequences or trajectories\(^3\) \( S_i = \{x_{1}^{t_i}, \ldots, x_{T_i}^{t_i}\} \) \((i = 1, \ldots, N)\), which are time-ordered sets of \( T_i \) observations \( x_{t}^{i} = (x_{1}^{t_i}, \ldots, x_{T_i}^{t_i}) \) \((T_i - 1 \text{ transitions})\) over some variables \( \mathcal{X} = \{X_1, \ldots, X_m\} \). Each transition is represented by two pairs \( x_{t}^{i} \) and \( x_{t+1}^{i} \), where \( t, t+1 \in \mathbb{R}^+ \) and \( t < t+1 \), such that variables have value \( x_{t}^{i} \) in the time interval \([t, t+1)\) and value \( x_{t+1}^{i} \) in the interval \([t+1, t+2)\), where \( h + 2 \leq T_i \).

Dynamic Bayesian networks (DBNs) [Dean and Kanazawa, 1989; Dagum et al., 1992; Murphy, 2002] are the best-known extension of BNs to model temporal data, and they have been successfully used in real-world problems [Zweig and Russell, 1998; Perrin et al., 2003; Burge et al., 2009]. These PGMs also consist of a DAG to describe the relationships between variables, but in this case, they allow for modeling their influences across time. DBNs can be seen as a set of equal structure static BNs, each of them modeling a JPD during a specific time slice. DBNs also include inter-slice arcs between nodes of the different BNs to model the effect of past values on future ones [Koller and Friedman, 2009]. Definition 2.9 formally introduces the most basic kind of DBNs, which operates under stationarity and the first-order Markov assumptions [Koller and Friedman, 2009].

**Definition 2.9. (Dynamic Bayesian network).** A DBN over a set of variables \( \mathcal{X} \) is a pair \( \langle \mathcal{B}_0, \mathcal{B}_\to \rangle \), where \( \mathcal{B}_0 \) is a BN that models the initial state \( \mathcal{X}^0 \) of the variables, and \( \mathcal{B}_\to \) is a 2-time-slice BN that represents a transition model defining \( P(\mathcal{X}^{t+1}|\mathcal{X}^t) \) for any given \( t \geq 0 \). Given a time duration \( T \geq 0 \), the JPD over \( \mathcal{X}^{0:T} = \{\mathcal{X}^0, \mathcal{X}^1, \ldots, \mathcal{X}^T\} \) for a DBN is defined as follows:

\[
P(\mathcal{X}^{0:T}) = P(\mathcal{X}^0) \prod_{t=0}^{T-1} P(\mathcal{X}^{t+1}|\mathcal{X}^t).
\]

Nonetheless, DBNs are based on discretizing time, forcing us to define a uniform time granularity even when the described processes evolve at different rates. This discretization can, therefore, result in a loss of valuable information. Xu and Shelton [2010] discuss specific scenarios where applying DBNs is not trivial, highlighting the difficulty of establishing a suitable time granularity. To address these limitations, the continuous-time Bayesian network...
Carlos Villa Blanco

(CTBN) has been proposed to represent the temporal dynamics of continuous-time and discrete-state stochastic processes [Nodelman et al., 2002]. In this framework, the dynamics of each variable are described by finite-state, continuous-time, homogeneous Markov processes through intensity matrices.

**Definition 2.10.** *(Homogeneous Markov process).* Given a discrete random variable $X$, whose sample space is $\Omega_X = \{x_1, \ldots, x_k\}$, its transient behavior can be described as a homogeneous Markov process $X(t)$ with its intensity matrix denoted by $Q_X$:

$$
Q_X = \begin{pmatrix}
-q_{x_1} & q_{x_1,x_2} & \cdots & q_{x_1,x_k} \\
q_{x_2,x_1} & -q_{x_2} & \cdots & q_{x_2,x_k} \\
\vdots & \vdots & \ddots & \vdots \\
q_{x_k,x_1} & q_{x_k,x_2} & \cdots & -q_{x_k}
\end{pmatrix},
$$

where $q_{x_a, x_b}$ is the intensity of leaving state $x_a$ and arriving at $x_b$ and $q_{x_a} = \sum_{b \neq a} q_{x_a, x_b}$ is the intensity of variable $X$ leaving state $x_a$. The waiting time of variable $X$ in a given state $x_a$ is exponentially distributed with parameter $q_{x_a}$. Thus, variable $X$ is expected to transition from state $x_a$ at time $1/q_{x_a}$ and to a state $x_b$ with probability $q_{x_a, x_b}/q_{x_a}$ [Nodelman et al., 2002].

When modeling systems with multiple variables $\mathcal{X} = \{X_1, \ldots, X_m\}$, a simple solution would be to define an intensity matrix over the joint sample space of $\mathcal{X}$. However, this approach would only be feasible with a very limited number of variables, as the size of this intensity matrix would grow exponentially with the number of variables $m$ and their cardinality. Therefore, in order to model larger systems, a factored representation of Markov processes, known as conditional Markov process, should be employed.

**Definition 2.11.** *(Conditional Markov process).* A conditional Markov process is a type of inhomogeneous Markov process whose intensity matrix changes over time as a function of some conditioning variables’ state. Given a discrete random variable $X_j$ and a set of parents $\text{Pa}(X_j)$ of $X_j$ in a directed graph $\mathcal{G}$, a conditional intensity matrix (CIM) $Q_{X_j}^{\text{Pa}(X_j)}$ describes the temporal dynamics of the variable. A CIM is a set of homogeneous intensity matrices $Q_{X_j}^{\text{pa}(X_j)}$, each encoding the dynamics of $X_j$ given the state $\text{pa}(X_j)$ of its parents $\text{Pa}(X_j)$:

$$
Q_{X_j}^{\text{pa}(X_j)} = \begin{pmatrix}
-q_{x_1}^{\text{pa}(X_j)} & q_{x_1,x_2}^{\text{pa}(X_j)} & \cdots & q_{x_1,x_k}^{\text{pa}(X_j)} \\
q_{x_2,x_1}^{\text{pa}(X_j)} & -q_{x_2}^{\text{pa}(X_j)} & \cdots & q_{x_2,x_k}^{\text{pa}(X_j)} \\
\vdots & \vdots & \ddots & \vdots \\
q_{x_k,x_1}^{\text{pa}(X_j)} & q_{x_k,x_2}^{\text{pa}(X_j)} & \cdots & -q_{x_k}^{\text{pa}(X_j)}
\end{pmatrix}.
$$

CIMs can be summarized with the sets of parameters $q_{x_a}^{\text{pa}(X_j)}$ and $\theta_{x_a, x_b}^{\text{pa}(X_j)} = q_{x_a, x_b}^{\text{pa}(X_j)}/q_{x_a}^{\text{pa}(X_j)}$, the
latter defining the probability of transitioning from state $x_a$ to another $x_b$ when a transition is known to occur.

CTBNs describe the temporal dynamics of some discrete random variables and their dependencies on each other using conditional intensity matrices while providing a graphical representation of the modeled system. CTBNs have been successfully applied in a variety of real-world problems, including intrusion detection [Xu and Shelton, 2010], gene network reconstruction [Acerbi and Stella, 2014], speech-related facial action unit recognition [Meng et al., 2017] or disease monitoring [Liu et al., 2019], among others.

**Definition 2.12.** (*Continuous-time Bayesian network*). A CTBN $\mathbb{N} = (\mathcal{G}, \mathcal{Q}, P^0_\mathcal{X})$ over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ consists of:

- A continuous transition model specified by a directed (possibly cyclic) graph $\mathcal{G}$ over $\mathcal{X}$ and a set of CIMs $\mathcal{Q}$, so there is a CIM $\mathcal{Q}_{X_j}^{\text{Pa}(X_j)}$ for each variable $X_j$. Each CIM can be summarized with the sets of multinomial $^4 \Theta_{X_j}^{\text{Pa}(X_j)} = \{q_{x_a,x_b}^{\text{Pa}(X_j)}: x_a, x_b \in \Omega_{X_j}, x_a \neq x_b, \text{pa}(X_j) \in \mathcal{I}_{\text{Pa}(X_j)}\}$ and exponential $q_{X_j}^{\text{Pa}(X_j)} = \{q_{x_a}^{\text{Pa}(X_j)}: x_a \in \Omega_{X_j}, \text{pa}(X_j) \in \mathcal{I}_{\text{Pa}(X_j)}\}$ parameters, where $\mathcal{I}_{\text{Pa}(X_j)}$ denotes the joint sample space of the parents of $X_j$.

- An initial distribution $P^0_\mathcal{X}$, specified as a BN over $\mathcal{X}$, representing the initial state of a temporal process.

Unlike a BN, the graph $\mathcal{G}$ of a CTBN can be cyclic since its arcs represent the dependencies between variables across time, i.e., the state to which a variable will transition depends on the current state of other variables. This can be easily understood with the following example introduced by Nodelman et al. [2002].

**Example 2.2.** The graph in Figure 2.3 describes the uptake of a drug and the relationships of its concentration in blood with other variables across time. We can observe a cycle if we focus on the upper part of the graph. This cycle indicates that if a person who is not eating or hungry transitions from having a full stomach to an empty stomach, then the person will be more likely to shift from being satiated to hungry in a shorter period of time. This increased hunger then makes the individual more likely to start eating soon, eventually leading back to a full stomach state in the future.

### 2.4 Learning Bayesian networks from data

Chapter 5 of this dissertation will introduce a novel multi-dimensional classification model based on CTBNs and discrete BNs and how to learn it directly from data. Furthermore,
Chapter 6 will study new structure learning algorithms for the presented model. Given this context, this section on learning the parameters and structure of both BNs and CTBNs becomes a fundamental part of the background work to understand our contributions. It is important to note that throughout this dissertation, we operate under the complete data assumption, meaning that the data is assumed to be complete with no missing values and no hidden variables [Murphy, 2012].

### 2.4.1 Parameter learning

Given that we have the structure of a BN or a CTBN, parameter learning involves estimating the CPDs for each node directly from some observed data. Assuming that the data are complete, the two most common methods for parameter learning are maximum likelihood estimation (MLE) and Bayesian estimation.

#### 2.4.1.1 Bayesian networks

The MLE seeks the parameters \( \hat{\mathbf{B}} \) that maximize the likelihood of observing some given data. Let \( \mathcal{D} = \{ \mathbf{x}_i, 1 \leq i \leq N \} \) be the observed dataset, where \( \mathbf{x}_i \) denotes an instance \( i \) with a vector of values \( \mathbf{x}_i = (x_{i1}, \ldots, x_{im}) \) of a fixed dimension \( m \). Then, the likelihood of observing

\[ \mathcal{L}(\mathbf{B}; \mathcal{D}) = \prod_{i=1}^{N} f(x_i | \mathbf{B}) \]

where \( f(x_i | \mathbf{B}) \) is the probability of observing data \( x_i \) given the parameters \( \mathbf{B} \).

---

5Drug concentration.
\( \mathcal{D} \) given the structure and parameters of a BN is defined by the following function [Koller and Friedman, 2009]:

\[
L(\mathcal{G}, B : \mathcal{D}) = P(\mathcal{D} | \mathcal{G}, B) = \prod_{i=1}^{N} P(x_i | \mathcal{G}, B).
\]

As we seek the set of parameters that maximize the likelihood, in practice, it is common to maximize the log-likelihood (LL) instead since it eases the calculation of derivatives, and given that the logarithmic function is monotonically increasing, the results are not affected by this transformation [Koller and Friedman, 2009]:

\[
LL(\mathcal{G}, B : \mathcal{D}) = \log \left( \prod_{i=1}^{N} P(x_i | \mathcal{G}, B) \right) = \sum_{i=1}^{N} \log P(x_i | \mathcal{G}, B).
\]

Given the structure of a BN, the likelihood can be decomposed into the sum of the local LL of each variable \( X_j \) given its parents:

\[
LL(\mathcal{G}, B : \mathcal{D}) = \sum_{j=1}^{m} LL(B_{\text{Pa}(X_j)}^{X_j} : \mathcal{D}) = \sum_{i=1}^{N} \sum_{j=1}^{m} \log P(x_{ij} | \text{pa}(X_j)), \tag{2.1}
\]

where \( B_{\text{Pa}(X_j)}^{X_j} \) collects all the parameters of the CPDs of node \( X_j \) given any instantiation \( \text{pa}(X_j) \) of its parents \( \text{Pa}(X_j) \).

Considering that this dissertation only concerns BNs with categorical variables, our focus centers on estimating the parameters \( B_{\text{Pa}(X_j)}^{X_j} \) of multinomial distributions. As a result, the CPDs of each variable, given their parents, are represented by CPTs. Each entry of a CPT is formed by a parameter \( \beta_{\text{pa}(X_j)}^{x_j} \), which corresponds to the probability of variable \( X_j \) taking state \( x_j \) given the parents’ state \( \text{pa}(X_j) \). Therefore, the maximum likelihood estimator for the CPD of a categorical variable \( X_j \) given an instantiation \( \text{pa}(X_j) \) of its parents \( \text{Pa}(X_j) \) is [Koller and Friedman, 2009]:

\[
\hat{\beta}_{\text{pa}(X_j)}^{x_j} = \frac{N_{\text{pa}(X_j)}^{x_j}}{N_{\text{pa}(X_j)}},
\]

where \( N_{\text{pa}(X_j)}^{x_j} \) is the number of instances in which \( X_j \) takes state \( x_j \), while its parents are in state \( \text{pa}(X_j) \), and \( N_{\text{pa}(X_j)} = \sum_{x_j} N_{\text{pa}(X_j)}^{x_j} \) is the number of instances where parents of \( X_j \) have state \( \text{pa}(X_j) \) independently of the state of \( X_j \).

As the sets of parameters \( B_{\text{Pa}(X_j)}^{X_j} \) are assumed disjoint, the global decomposition property of the likelihood function can be applied to get the MLE for the entire BN by independently...
maximizing each local LL function [Koller and Friedman, 2009]:

\[
\hat{B} = \arg\max_B \text{LL}(G, B : D) = \{\arg\max_{B_1} \text{LL}(B_{X_1}^{\text{pa}(X_1)} : D), \ldots, \arg\max_{B_m} \text{LL}(B_{X_m}^{\text{pa}(X_m)} : D)\},
\]

where \( B_m \) represents the set of parameters for the CPD of the variable \( X_m \).

MLE provides a point estimate of the parameters. In contrast, in the Bayesian estimation, parameters are considered random variables, providing a prior distribution over them. This is a more interesting approach since we can add prior expert knowledge about the parameters while updating our beliefs on them using some observed data. This updating process is performed with the Bayes’ theorem, which describes how to calculate the posterior distribution of the parameters given the data based on the prior distribution of the parameters and the likelihood of the data. Bayesian estimation also avoids the zero-count problem, which occurs when estimating probabilities for events not observed in the data [Murphy, 2012]. The most common conjugate prior distribution for the multinomial parameters is the Dirichlet distribution [Heckerman et al., 1995]. As conjugate priors are used, the posterior distribution of the parameters given the observed data follows the same distribution and, therefore, can be obtained analytically. Then, the parameters can be estimated using their expected values, but including the hyperparameters of the Dirichlet prior distributions \( \lambda_{\text{pa}(X_j)} \), which can be seen as imaginary counts of the sufficient statistics that occur before any data is observed [Koller and Friedman, 2009]:

\[
\hat{\beta}_{x_i^{\text{pa}(X_j)}} = \frac{N_{x_i^{\text{pa}(X_j)}} + \lambda_{x_i^{\text{pa}(X_j)}}}{N_{\text{pa}(X_j)} + \sum_{x_z} \lambda_{x_z^{\text{pa}(X_j)}}}.
\]

### 2.4.1.2 Continuous-time Bayesian networks

In the case of a CTBN, datasets are formed by a set of sequences or trajectories \( D = \{S_1, \ldots, S_N\} \), as it was introduced in Definition 2.8. The likelihood of this dataset \( D \) can be expressed as a product of the likelihoods for individual transitions. Hence, given a variable \( X_j \in \mathcal{X} \), the likelihood for a transition \( u = < x_{uj}, t_{uj}, x'_{uj} > \in D \), where \( X_j \) transitions from state \( x_{uj} \) to \( x'_{uj} \) after an amount of time \( t_{uj} \) is [Nodelman et al., 2003]:

\[
L(G, q, \Theta : u) = L(G, q : u)L(G, \Theta : u),
\]

where \( q = \{q_{X_j}^{\text{pa}(X_j)} : X_j \in \mathcal{X}\} \) and \( \Theta = \{\Theta_{X_j}^{\text{pa}(X_j)} : X_j \in \mathcal{X}\} \) are the set of parameters of the CTBN over the set of variables \( \mathcal{X} \).

Thus, the likelihood of a transition can be decomposed into two components. The first component is temporal, represented by likelihood \( L(G, q : u) \), which captures the probability
Chapter 2. Bayesian networks

of variable $X_j$ remaining in state $x_{uj}$ for an amount of time $t_{uj}$ given an instantiation $\text{pa}(X_j)$ of its parents. This probability is modeled with an exponential distribution with parameter $q_{x_{uj}}^{\text{pa}(X_j)}$:

$$L(\mathcal{G}, q : u) = q_{x_{uj}}^{\text{pa}(X_j)} \exp(-q_{x_{uj}}^{\text{pa}(X_j)} t_{uj}).$$

The second component is a state transition component, represented by likelihood $L(\mathcal{G}, \Theta : u)$, corresponding to the probability of transitioning from state $x_{uj}$ to state $x'_{uj}$ given that a transition is known to occur and given an instantiation $\text{pa}(X_j)$ of the parents of $X_j$. In this case, a multinomial distribution with parameter $\theta_{x_{uj}x'_{uj}}^{\text{pa}(X_j)}$ is employed:

$$L(\mathcal{G}, \Theta : u) = \theta_{x_{uj}x'_{uj}}^{\text{pa}(X_j)}.$$

The observable dataset $\mathcal{D}$ can be summarized with the following sufficient statistics [Nodelman et al., 2003]:

- $M_{x_i,x_z}^{\text{pa}(X_j)}$, number of transitions of feature $X_j$ from state $x_i$ to $x_z$ when the parents’ state is $\text{pa}(X_j)$.
- $T_{x_i}^{\text{pa}(X_j)}$, time of $X_j$ spent in state $x_i$ when the parents’ state is $\text{pa}(X_j)$.

Then, the LL of dataset $\mathcal{D}$ for a CTBN can be decomposed by variable as follows [Nodelman et al., 2003]:

$$\text{LL}(\mathcal{G}, q, \Theta : \mathcal{D}) = \text{LL}(\mathcal{G}, q : \mathcal{D}) + \text{LL}(\mathcal{G}, \Theta : \mathcal{D})$$

$$= \sum_{j=1}^{m} \sum_{\text{pa}(X_j)} \sum_{x_i} \left[ M_{x_i}^{\text{pa}(X_j)} \log(\hat{q}_{x_i}^{\text{pa}(X_j)}) - \hat{q}_{x_i}^{\text{pa}(X_j)} T_{x_i}^{\text{pa}(X_j)} \right]$$

$$+ \sum_{x_i \neq x_z} M_{x_i,x_z}^{\text{pa}(X_j)} \log(\hat{\theta}_{x_i,x_z}^{\text{pa}(X_j)}).$$

(2.2)

where $M_{x_i}^{\text{pa}(X_j)} = \sum_{x_z \neq x_i} M_{x_i,x_z}^{\text{pa}(X_j)}$ is the number of transitions of $X_j$ from state $x_i$ to any other when the parents’ state is $\text{pa}(X_j)$.

From Equation (2.2), the MLE parameters of a CTBN can be derived [Nodelman et al., 2003]:

$$\hat{q}_{x_i}^{\text{pa}(X_j)} = \frac{M_{x_i}^{\text{pa}(X_j)}}{T_{x_i}^{\text{pa}(X_j)}}, \quad \text{and} \quad \hat{\theta}_{x_i,x_z}^{\text{pa}(X_j)} = \frac{M_{x_i,x_z}^{\text{pa}(X_j)}}{M_{x_i}^{\text{pa}(X_j)}}.$$

The zero-count problem may be more pronounced for CTBNs than for traditional BNs. This is motivated by the fact that it becomes more likely that not all possible transitions between states of a variable $X_j$, given its parents, occur in an observed dataset. For this reason and the possibility of adding prior knowledge, parameter learning with Bayesian estimation is introduced for CTBNs, as was the case with BNs. For CTBNs, we have the parameters
of multinomial and exponential distributions. Thus, the Dirichlet distribution is used as conjugate prior distribution for the multinomial parameters, while the Gamma distribution is an appropriate choice for the exponential parameter [Nodelman et al., 2003]. Then, the parameters are estimated as follows [Nodelman et al., 2003; Koller and Friedman, 2009]:

$$\hat{\theta}_{pa}(x_j) = \frac{M_{x_i}^{pa}(X_j)}{T_{pa}(x_j)} + \alpha_{x_i - x_j}^{pa}(X_j), \quad \text{and} \quad \hat{\theta}_{x_i - x_j}^{pa}(X_j) = \frac{M_{x_i - x_j}^{pa}(X_j)}{M_{x_i}^{pa}(x_j)} + \alpha_{x_i - x_j}^{pa}(X_j),$$

where $\alpha_{x_i}(X_j) = \sum_{x_z \neq x_i} \alpha_{x_i - x_z}^{pa}(X_j)$ and $\tau_{x_i}^{pa}(X_j)$ (Gamma), and $\alpha_{x_i - x_z}^{pa}(X_j)$ (Dirichlet) are the hyper-parameters of the prior distributions over the parameters.

### 2.4.2 Structure learning

Section 2.4.1 covered the learning of the parameters of BNs and CTBNs directly from data. This learning was possible because the structure of models was known in advance, so the variable distributions could be defined based on the conditional independence relations encoded in the structure. However, this strong assumption of knowing the structure in advance will not be fulfilled in most real problems. As a result, structure learning algorithms aim to learn the structure of PGMs automatically from some observed data, attempting to find the structure that best fits the data. Three common approaches to structure learning can be found, which are classified as score-based, constraint-based and hybrid structure learning algorithms. These three learning approaches will be explored during some contributions of this dissertation, so a brief introduction is given in this section.

#### 2.4.2.1 Score-based structure learning

Score-based algorithms define a score to evaluate how well a particular model, and therefore a structure, fits some observed data. The goal is to use an optimization algorithm to search, through a space of candidate structures, that structure that maximizes the chosen score.

**Score metric** Score metrics are commonly divided into those based on the likelihood function, which measures how likely the observed data is given a model structure, and Bayesian scores, which combine the likelihood with a prior belief about the structure.

The simplest approach to learning the structure of either a BN or CTBN is to maximize the LL of the observed data given the model, i.e., find the structures that maximize Equations (2.1) or (2.2) for BNs and CTBNs, respectively. Nonetheless, optimizing this score often leads to overfitting, as it tends to favor denser structures. Based on the number of model parameters, a network complexity penalty can be introduced to counteract this effect.
Popular penalty functions like the Akaike information criterion (AIC) [Akaike, 1974] and the Bayesian information criterion (BIC) [Schwarz, 1978] can be employed for both the LLs of a BN or a CTBN.

Bayesian scoring methods combine likelihood with prior beliefs about the model structure, allowing preferences over specific structures [Koller and Friedman, 2009]. A prior probability distribution over the structures \( P(G) \) is chosen, and Bayes’ theorem is applied to combine this prior with the marginal likelihood of the data given the structure \( P(D|G) \), resulting in the following Bayesian score that accounts for the posterior probability distribution over the structures:

\[
BS(G : D) = \log P(D|G) + \log P(G).
\]

The marginal likelihood term \( P(D|G) \) incorporates our uncertainty over the parameters by integrating over all of their possible values for \( G \), so no specific assignment of the parameters is considered. Thus, the term \( P(D|G) \) is defined for a BN as:

\[
P(D|G) = \int P(D|B, G)P(B|G)dB,
\]

while for a CTBN as:

\[
P(D|G) = \int P(D|q, \Theta, G)P(q, \Theta|G)dqd\Theta.
\]

The Bayesian Dirichlet equivalent (BDe) score is a typical example of a Bayesian score for BNs, which assumes a Dirichlet distribution over the parameter priors (with hyperparameters of Section 2.4.1.1). Making the common assumptions of global and local parameter independence [Heckerman et al., 1995], the log-marginal-likelihood (LML) of the data given the structure can be decomposed into the sum of local LML for each parameter type and variable. Therefore, the LML for the parameters \( B_{X_j}^{\text{pa}(X_j)} \) of a discrete BN can be decomposed as follows (for a derivation, see Cooper and Herskovits [1992]):

\[
\text{LML} \left( B_{X_j}^{\text{pa}(X_j)} : D \right) = \log \left( \prod_{\text{pa}(X_j)} \frac{\Gamma \left( \sum_{x_i} \lambda_{x_i}^{\text{pa}(X_j)} \right)}{\Gamma \left( \sum_{x_i} \left( \lambda_{x_i}^{\text{pa}(X_j)} + N_{x_i}^{\text{pa}(X_j)} \right) \right)} \prod_{x_i} \frac{\Gamma \left( \lambda_{x_i}^{\text{pa}(X_j)} + N_{x_i}^{\text{pa}(X_j)} \right)}{\Gamma \left( \lambda_{x_i}^{\text{pa}(X_j)} \right)} \right),
\]

where \( \Gamma(\cdot) \) is the gamma function.

In the case of a CTBN, the temporal evolution of variables is modeled by exponential distributions, which predict when the next state transition occurs, and multinomial distributions, which determine the direction of that state transition. We will refer to the Bayesian score introduced by Nodelman et al. [2003] for CTBNs as BDe for simplicity, as done in Bregoli et al. [2021]. The LML for the parameters \( \Theta_{X_j}^{\text{pa}(X_j)} \) and \( q_{X_j}^{\text{pa}(X_j)} \) can be estimated with the
following closed formulas (for a derivation, see Cooper and Herskovits [1992]; Nodelman et al. [2003]):

\[
\text{LML}^{\text{Pa}(X_j)} : \mathcal{D} = \log \left( \prod_{\text{pa}(X_j)} \prod_{x_i} \frac{\Gamma \left( \alpha_{x_i}^{\text{pa}(X_j)} \right)}{\Gamma \left( \alpha_{x_i}^{\text{pa}(X_j)} + M_{x_i}^{\text{pa}(X_j)} \right)} \prod_{x_i \neq x_j} \frac{\Gamma \left( \alpha_{x_i,x_j}^{\text{pa}(X_j)} + M_{x_i,x_j}^{\text{pa}(X_j)} \right)}{\Gamma \left( \alpha_{x_i,x_j}^{\text{pa}(X_j)} \right)} \right),
\]

and

\[
\text{LML}^{q_{X_j}} : \mathcal{D} = \log \left( \prod_{\text{pa}(X_j)} \prod_{x_i} \frac{\Gamma \left( \alpha_{x_i}^{\text{pa}(X_j)} + 1 \right) \left( \tau_{x_i}^{\text{pa}(X_j)} \right)^{\alpha_{x_i}^{\text{pa}(X_j)} + 1}}{\Gamma \left( \alpha_{x_i}^{\text{pa}(X_j)} + 1 \right) \left( \tau_{x_i}^{\text{pa}(X_j)} \right)^{\alpha_{x_i}^{\text{pa}(X_j)} + 1}} \right).
\]

If a uniform prior over the structures \( P(\mathcal{G}) \) is considered, the Bayesian score simply maximizes the LML of the observed data given the model. Thus, the BDe scores for a BN and a CTBN are defined, respectively, as:

\[
\text{BDe}(\mathcal{G} : \mathcal{D}) = \sum_{j=1}^{m} \text{LML} \left( B_{X_j}^{\text{Pa}(X_j)} : \mathcal{D} \right)
\]

and

\[
\text{BDe}(\mathcal{G} : \mathcal{D}) = \sum_{j=1}^{m} \left[ \text{LML} \left( q_{X_j}^{\text{Pa}(X_j)} : \mathcal{D} \right) + \text{LML} \left( \Theta_{X_j}^{\text{Pa}(X_j)} : \mathcal{D} \right) \right].
\]

In summary, Bayesian scores, whether for BNs or CTBNs, provide a method to evaluate the fit of a model structure to some observed data combining some prior beliefs about the structure. Thus, we can select models that are consistent with our empirical observations and our domain knowledge.

**Optimization algorithm and search space of structures** Typically, there are many possible model structures to evaluate. For example, in the case of BNs, the search space of possible DAGs is superexponential in the number of variables, as shown by the recursive formula of Robinson [1977]:

\[
f(m) = \sum_{j=1}^{m} (-1)^{j+1} \binom{m}{j} 2^{j(m-j)} f(m-j),
\]

where \( f(0) = 1 \). This vast search space contributes to the NP-hard nature of finding the DAG structure that maximizes a given score metric [Chickering et al., 1994; Koller and Friedman, 2009]. As it is not feasible to score all structures, heuristic optimization algorithms are used to search the space of possible structures. Some examples of these algorithms include greedy hill climbing [Heckerman et al., 1995; Nodelman et al., 2003], tabu search [Glover, 1986], K2
Algorithm 2.1: Hill climbing($X$, score)

1: Get initial candidate solution $G$ on node set $X$
2: Initialize best solution $G^*$ with $G$
3: repeat
4: Find a set of neighboring solutions $N$ by applying some operators on $G^*$
5: for each neighboring solution $G' \in N$ do
6: if $G'$ is a legal structure and score($G'$) $>$ score($G^*$) then
7: $G^* \leftarrow G'$
8: end if
9: end for
10: until no improvement in score($G^*$)
11: return structure $G^*$

[Cooper and Herskovits, 1992], simulated annealing [Bouckaert, 1995] or genetic algorithms [Larrañaga et al., 1996]. In the context of this dissertation, two optimization algorithms, hill climbing and tabu search, will be applied in Chapters 5 and 6. Therefore, the fundamentals of these algorithms are explained.

First, the greedy hill climbing algorithm, which is a local search method, is one of the simplest and most common optimization techniques for structure learning. This algorithm iteratively applies, through some operators, small changes to the structure of the models. Then, it moves along the chain of changes that, in each step, lead to the most immediate improvement in the model score on the observed data. Algorithm 2.1 shows a high-level pseudocode of the greedy hill climbing algorithm [Koller and Friedman, 2009]. In a BN, the operators of adding, deleting and reversing arcs are applied to find a neighbor solution for the entire structure, ensuring these modifications do not violate the acyclicity constraint of BNs. In the case of a CTBN, there is no acyclicity constraint due to arcs representing relations across time, reducing the search process’s complexity. The parent set of each node of a CTBN can be optimized separately using operators that add and delete arcs, so the search can be performed more efficiently compared to BNs [Nodelman et al., 2003]. Actually, if a maximum parent set size is established, an algorithm that exhaustively studies the parent sets in a CTBN can perform in polynomial time, depending on the number of variables and size of the dataset. Nonetheless, heuristic optimization algorithms are used in practice to avoid an exhaustive study of all possible solutions [Nodelman et al., 2003].

The hill climbing algorithm can get stuck in local optima, where further changes do not lead to immediate score improvements, and it might miss a globally optimal solution [Koller and Friedman, 2009]. Tabu search is a variation of this algorithm that aims to overcome the local optima problem. It allows the exploration of new directions in the search space, even if they do not lead to immediate or significant improvements in the score. The algorithm uses a tabu list to track recently applied operators during a history window of a predefined size,
preventing the application of operators that reverse the effect of those in the tabu list. This strategy keeps the algorithm from revisiting the same solutions in the near future, encouraging exploration beyond local optima.

2.4.2.2 Constraint-based structure learning

Constraint-based algorithms use conditional independence tests to determine the relationships between variables in the structure. The main idea is to test for conditional dependence and independence in the data and find the structure (or equivalence class of structures for BNs) that better explains the dependence and independence relationships found [Koller and Friedman, 2009].

The inductive causation (IC) algorithm, a pioneering methodology in the field of causal inference, has significantly contributed to the development of constraint-based approaches. Proposed by Verma and Pearl [1990b], the IC algorithm presents a framework for learning DAGs of BNs. The algorithm initiates with an empty graph, and it identifies conditional independence relationships between any pair of variables. If two variables are found to be conditionally dependent given a set of other variables (separating set), an undirected edge is placed between them. Once all conditional independence relations are identified, they define an initial skeleton of the structure formed by undirected edges. The algorithm then orients all possible edges based on certain deterministic rules that avoid introducing cycles in the structure. The result is a set of equivalent BNs, i.e., a collection of different BN structures that encode the same set of conditional independence relations, compactly represented by a partially DAG, i.e., a graph that contains both directed and undirected edges with no cycles. Unfortunately, applying the IC algorithm is impractical for scenarios with a large number of variables since it performs conditional independence tests given all possible separating sets.

In order to address the previous limitation, the Peter-Clark (PC) algorithm was proposed [Spirtes et al., 2000; Pearl, 2009; Colombo and Maathuis, 2014], which is one of the most well-known refinements of the IC algorithm. In contrast to the IC algorithm, the PC algorithm begins with a fully connected graph and employs a process of edge removal, considering conditional independence tests with separating sets of increasing size. Once conditional independence is found between two variables given a conditioning set, the edge between them is removed, and larger separating sets are not tested. This effectively reduces the number of required conditional independence tests compared to the IC algorithm. Algorithm 2.2 presents the pseudocode of the PC algorithm, showing in Steps 1 and 2 the definition of a skeleton of \( G \) after applying conditional independence tests, and then in Steps 3 and 4 the rules employed by the PC algorithm to orient all possible undirected edges without introducing any cycle.

Apart from the PC algorithm, other noteworthy constraint-based methods include the grow-
Algorithm 2.2: PC($\mathcal{X}$)

1: Build the complete undirected graph $\mathcal{G}$ on node set $\mathcal{X}$
2: Find the skeleton and separating sets of $\mathcal{G}$. For each pair $X_i, X_j \in \mathcal{X}$ of adjacent nodes, remove edge $X_i \rightarrow X_j$ from $\mathcal{G}$ iff a separating set $S_{X_i,X_j} \subseteq \mathcal{X} \setminus \{X_i, X_j\}$ exists, such that $X_i \perp \perp X_j|S_{X_i,X_j}$
3: Orient the colliders using the separating sets. For any path $X_i \rightarrow X_j \rightarrow X_k$ in $\mathcal{G}$, such that $X_i \neq X_k$, orient the edges as $X_i \rightarrow X_j \leftarrow X_k$ iff $X_i \perp \perp X_k|S_{X_i,X_k}$ and $X_j \notin S_{X_i,X_k}$
4: Orient all possible undirected edges:
   4.1: Given $X_i \rightarrow X_j \rightarrow X_k$ and $X_i \neq X_k$, orient the undirected edge as $X_j \rightarrow X_k$ to prevent introducing a new v-structure
   4.2: Given $X_i \rightarrow X_k$ and $X_i \rightarrow X_j \rightarrow X_k$, orient $X_i \rightarrow X_k$ as $X_i \rightarrow X_k$ to prevent introducing a cycle
   4.3: Given $X_i \rightarrow X_k$, $X_i \rightarrow X_j \rightarrow X_k$, $X_i \rightarrow X_w \rightarrow X_k$ and $X_j \neq X_w$, orient $X_i \rightarrow X_k$ as $X_i \rightarrow X_k$ to prevent introducing a new v-structure or a cycle
5: return partially directed acyclic graph $\mathcal{G}$

Structure learning for CTBNs has been traditionally addressed as an optimization problem [Nodelman et al., 2003; Codecasa and Stella, 2014; Villa and Stella, 2016]. Only recently, a constraint-based algorithm was proposed, which infers the structure by performing conditional independence tests. The continuous-time PC (CTPC) algorithm, introduced by Bregoli et al. [2021], is the first proposal of this kind, which adapts the classical PC algorithm for BNs to CTBNs. As CIMs describe temporal dynamics, classical statistical tests such as the chi-square or those based on measures like mutual information cannot be applied [Bregoli et al., 2021]. Thus, CTPC introduces a novel definition of conditional independence in CTBNs.

Definition 2.13. (Conditional independence in CTBNs). Given a CTBN over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$, a variable $X_i$ is conditionally independent of $X_j$ given a separating set $S_{X_i,X_j} \subseteq \mathcal{X} \setminus \{X_i, X_j\}$ iff:

$$Q_{X_i}^{y,s} = Q_{X_i}^s \quad \forall y \in \Omega_{X_j}, \forall s \in \mathcal{I}_{S_{X_i,X_j}}.$$  

Definition 2.13 is not symmetric, which means that while a variable $X_i$ could be conditionally independent of $X_j$ given $S_{X_i,X_j}$, the opposite does not necessarily have to be true. This fact has significant implications for learning the structure of a CTBN, as it may be necessary to
evaluate twice the number of arcs compared to a traditional BN.

The CTPC algorithm employs two different statistical tests to establish conditional independence between some variables $X_i$ and $X_j$ given a separating set. First, the time-to-transition null hypothesis (see Definition 2.14) is evaluated to determine if significant differences exist for the waiting times of variable $X_i$ when $X_j$ is added (or not) to its parents. Similarly, the state-to-state-transition null hypothesis (see Definition 2.15) is subsequently evaluated for significant differences in the probabilities of variable $X_i$ transitioning from one particular state to another. Conditional independence between $X_i$ and $X_j$ given a separating set $S_{X_i,X_j}$ is not established, i.e., $X_i \not\perp \hspace{-0.1cm} \perp X_j | S_{X_i,X_j}$, if any null hypothesis is rejected.

**Definition 2.14. (Time-to-transition null hypothesis).** Given two variables $X_i$ and $X_j$, and a separating set $S_{X_i,X_j} \subseteq \mathcal{X} \setminus \{X_i, X_j\}$, the time-to-transition null hypothesis of $X_j$ over $X_i$ is defined as:

$$q^{y,s}_{x} = q^{s}_{x} \quad \forall x \in \Omega_{X_i}, \forall y \in \Omega_{X_j}, \forall s \in \mathcal{I}_{S_{X_i,X_j}}.$$

**Definition 2.15. (State-to-state-transition null hypothesis).** Given two variables $X_i$ and $X_j$, and a separating set $S_{X_i,X_j} \subseteq \mathcal{X} \setminus \{X_i, X_j\}$, the state-to-state-transition null hypothesis of $X_j$ over $X_i$ is defined as:

$$\theta^{y,s}_{x_a,x_b} = \theta^{s}_{x_a,x_b} \quad \forall x_a \in \Omega_{X_i}, \forall x_b \in \Omega_{X_i} \setminus \{x_a\}, \forall y \in \Omega_{X_j}, \forall s \in \mathcal{I}_{S_{X_i,X_j}}.$$

Bregoli et al. [2021] proposed to test the time-to-transition null hypothesis using the F-test while they explored the use of the two-sample chi-square and Kolmogorov-Smirnov tests for the state-to-state-transition null hypothesis. Nevertheless, they found the two-sample chi-square test to be marginally better for testing the latter null hypothesis, so only this test will be employed in the present work.

Algorithm 2.3 shows the pseudocode of the CTPC algorithm. The CTPC algorithm starts by initializing in Step 1 the complete directed graph $\mathcal{G}$ of the CTBN and, between Steps 2 and 11, it iterates over all nodes $X_i \in \mathcal{X}$ to identify their parent sets by testing first for unconditional independencies and then for conditional independencies, gradually increasing the cardinality of the separating sets. If $X_i$ is found to be conditionally independent of $X_j$ given some separating set, the arc from $X_j$ to $X_i$ is removed, discarding $X_j$ from being a parent of $X_i$. As conditional independence relationships are not symmetric in CTBNs, the parent set of every variable can be defined independently of the rest of the conditional independence relations found in the graph, so the final result of this algorithm is a valid structure of a CTBN. Therefore, this algorithm mainly differs from the PC algorithm (see Algorithm 2.2) in that Steps 3 and 4 of the latter are not necessary [Bregoli et al., 2021], as CTBNs are not restricted to a DAG topology. Furthermore, the result of this algorithm is
Algorithm 2.3: CTPC($\mathcal{X}$)

1: Build the complete directed graph $\mathcal{G}$ on node set $\mathcal{X}$
2: for each variable $X_i \in \mathcal{X}$ do
3:    Set $\mathcal{U} = \{X_j \in \mathcal{X} | X_j \rightarrow X_i\}$
4:    for increasing values $s = 0, \ldots, |\mathcal{X}|$, until $s = |\mathcal{U}| - 1$ do
5:        for each variable $X_j \in \mathcal{U}$ and subset $S_{X_i} \subseteq \mathcal{U} \setminus \{X_j\}$, where $|S_{X_i}| = s$ do
6:            if $X_i \perp \perp X_j | S_{X_i} \cap X_j$ then
7:                Remove arc $X_j \rightarrow X_i$ from $\mathcal{G}$ and $X_j$ from $\mathcal{U}$
8:            end if
9:        end for
10:    end for
11: end for
12: return directed graph $\mathcal{G}$

not an equivalence class of structures as it happens with the BNs reported by Algorithm 2.2, but a final CTBN structure. CTBNs have a unique minimal structure for a given stochastic process, so two CTBNs with different structures cannot represent the same stochastic process [Nodelman et al., 2003].

The CTPC algorithm served as the basis in this dissertation for the subsequent development of a constraint-based structure learning algorithm aimed at learning CTBNs that apply to multi-dimensional classification problems. This learning problem will be introduced and exemplified in Chapter 3, while the novel algorithm will be formally presented in Chapter 6.

2.4.2.3 Hybrid structure learning

Hybrid structure learning algorithms combine the characteristics of score-based and constraint-based approaches to learn the structure of BNs, intending to exploit the strengths of both. These algorithms are commonly divided into two phases, known as the restriction and maximization phases. During the restriction phase, a constraint-based approach is applied to identify conditional independence relationships among variables and learn an initial skeleton of the structure. Thus, this phase reduces the potential search space of structures, so the maximization phase employs a score-based solution to search for the structures that maximize a score within the restricted search space [Scutari et al., 2019]. A prominent example of such hybrid approaches is the max-min hill-climbing [Tsamardinos et al., 2006], which uses the max-min parents and children algorithm to find the skeleton of the BN and then a greedy hill-climbing search to refine the solution of the first phase by orienting the edges of the skeleton. Other notable hybrid proposals include the CB [Singh and Valtorta, 1993], EGS [Dash and Druzdzel, 1999], BENEDICT-step [Acid and De Campos, 2001], IMAPR [De Campos et al., 2003], H2PC [Gasse et al., 2014] or ARGES [Nandy et al., 2018] algorithms.
To the best of our knowledge, no literature exists on hybrid structure learning algorithms for CTBNs until this dissertation. The first hybrid structure learning algorithm for CTBNs will be presented and experimentally studied in Chapter 6 of this dissertation.
Chapter 3

Classification algorithms

3.1 Introduction

A significant part of the contributions developed in this dissertation are set within the context of supervised classification. For this reason, it is necessary to introduce this learning paradigm and specific methods or algorithms that are fundamental pillars or have influenced our research.

Subsequent Chapters 5, 6 and 8 are performed in a supervised classification context, where a novel predictive model for multi-dimensional classification is introduced. This model is based on the CTBN framework. Thus, it is essential to introduce the basis of BNCs and their extensions to multi-dimensional classification and prediction on time series data. Furthermore, this background serves as valuable context for Chapter 7, which offers a comprehensive review of incremental FSS algorithms, especially highlighting their applications in supervised learning contexts.

Classification algorithms can be divided into three main categories: supervised, unsupervised, and semi-supervised classification. They differ primarily in their use of labeled data. Supervised classification exclusively uses labeled training instances, unsupervised classification, often termed clustering, operates without labels, while semi-supervised classification combines both labeled and unlabeled data. It is necessary to emphasize that the main focus of this dissertation is on supervised classification. Then, in the interest of clarity and conciseness, any mention of the term classification in this dissertation will implicitly refer to supervised classification unless otherwise specified.
Chapter outline

The remainder of this chapter is as follows. Section 3.2 introduces the concept of supervised classification and some of the most well-known classification algorithms for static stochastic processes. Section 3.3 presents adaptations of BNs for classification tasks, analyzing topological restrictions that could be introduced in their structure and how to predict the value of a class variable in a previously unseen data instance. Section 3.4 explores the more general and complex scenario of multi-dimensional classification, studying some strategies to handle this learning task, with a special focus on BNCs. Section 3.5 covers the problem of classification on time series data and studies a classifier based on the CTBN framework. Finally, Section 3.6 provides an overview of performance measures for multi-dimensional classification, which will be employed in the experimental studies of this dissertation.

3.2 Supervised classification

The supervised classification (or simply classification) task represents a learning paradigm with the objective of predicting a label or category (class variable) on a given input data. This input is characterized by a set of feature variables, also known as independent/predictive variables or attributes, or simply features, whose values are observed. Essentially, we aim to determine a mapping from the input features, \( \mathcal{X} \), to a class variable, \( \mathcal{C} \) (in the case of one-dimensional classification). This is achieved by training a prediction model on a labeled dataset, commonly called a training dataset, where both the input data (features) and the corresponding desired output (class variable) are provided.

**Definition 3.1. (Training dataset (one-dimensional classification)).** Let \( \mathcal{D} = \{ \mathbf{z}_i = (\mathbf{x}_i, c_i), 1 \leq i \leq N \} \) be a training dataset, where \( \mathbf{z}_i \) denotes the training instance \( i \) with a vector of feature values \( \mathbf{x}_i = (x_{i1}, \ldots, x_{im}) \) of a fixed dimension \( m \) and a class value \( c_i \). Each instance is then associated with a feature set \( \mathcal{X} = \{X_1, \ldots, X_m\} \) and a single class variable \( \mathcal{C} \), which take one of \( k \) distinct values, i.e., \( \Omega_C = \{c_1, \ldots, c_k\} \). The value of \( k \) determines whether the learning task is a binary (\( k = 2 \)) or a multi-class (\( k > 2 \)) classification problem.

The training dataset provides the necessary data observations from which the classification algorithm learns the patterns and relationships between the feature and class variables. Then, the objective is to generalize from this training data so that the learned classifier can make accurate predictions on new, previously unseen, unlabeled data.

**Definition 3.2. (Supervised classification).** Given a training dataset \( \mathcal{D} \), the supervised classification task first involves learning a function \( f : \Omega_{X_1} \times \cdots \times \Omega_{X_m} \rightarrow \Omega_C \) such that for every training instance \( \mathbf{z}_i \) in \( \mathcal{D} \), \( f(\mathbf{x}_i) \) accurately predicts the corresponding class value \( c_i \). The aim is to generalize this function to provide accurate predictions for new, previously unseen
instances from a dataset \( D_p = \{x_i, 1 \leq i \leq N_p\} \) where the actual labels or categories of the instances are unknown.

In the context of classification algorithms, many methodologies and techniques have been developed, each with unique strengths and applications. As the no free lunch theorem states [Wolpert, 1996], no single model is universally best across all possible learning problems. A classifier can stand out in specific tasks, but it will underperform in others, as assumptions in one domain might prove inadequate in another. Consequently, a diverse range of classification algorithms can be found to handle the different types of data we find in real-world problems [Murphy, 2012]: from linear models such as logistic regression [Berkson, 1944] and the perceptron [Rosenblatt, 1958], rule-based methods like classification trees [Breiman, 1984] and RIPPER [Cohen, 1995], ensemble algorithms such as adaptive boosting (AdaBoost) [Freund and Schapire, 1997] and random forests [Breiman, 2001], kernel-based approaches including support vector machines [Vapnik, 1995; Cortes and Vapnik, 1995] and kernel Fisher discriminant analysis [Mika et al., 1999], deep learning techniques such as convolutional neural networks [LeCun et al., 1998] and transformers [Vaswani et al., 2017], or instance-based classifiers like \( k \)-nearest neighbors (\( k \)NN) [Cover and Hart, 1967], to mention some. This dissertation focuses on Bayesian models, specifically BNCs.

### 3.3 Bayesian network classifiers

BNCs represent a specialized subset of BNs designed to solve classification tasks, offering some interesting advantages over other classification solutions [Bielza and Larrañaga, 2014]:

- **Interpretability.** Their graphical structure provides a clear and intuitive visualization of variable dependencies, making BNCs more interpretable than many black-box models, such as deep neural networks. This transparency of BNCs is especially relevant in domains such as medical decision support [Park et al., 2018]. Data-driven insights can give valuable information to medical professionals; however, it is essential to comprehend how predictions originate, ensuring the classifier aligns with domain expertise and real-world observations. Contributions of Chapters 5 and 6 showcase how these graphical capabilities are of great help to analyze two real-world problems.

- **Probabilistic reasoning.** The probabilistic nature of BNCs enables these predictive models to explicitly represent and reason about uncertainty, something not inherently present in algorithms such as decision trees or support vector machines. This probabilistic approach further allows BNCs to handle incomplete data scenarios.

- **Integration of prior knowledge.** As discussed in Section 2.2, BNs can incorporate domain-specific prior knowledge, so classifiers do not have to be built solely from
observed data.

- **Versatility.** BNCs are presented as versatile models, as they can cope with classification problems that involve handling diverse data types, undetermined or partial labels, predicting multiple class variables simultaneously, or being applied to a data streaming setting [Bielza and Larrañaga, 2014]. The possibility of applying this kind of classifier to perform simultaneous classification over multiple class variables (Chapters 5, 6 and 8) and efficiently adapting them to streaming data without retraining the entire classifier (Chapter 8) are two fundamental characteristics of these models for the contributions of this dissertation.

These advantages highlight the potential of BNCs and why the general principles of Bayesian network-based classification guided us during the development of this dissertation. For a more comprehensive list of benefits, readers are referred to Bielza and Larrañaga [2014].

The naive Bayes classifier stands out as one of the simplest forms of BNCs. It operates under the assumption of conditional independence among features given the class variable, simplifying the probabilistic relationships that are modeled. In its graphical representation, the class variable node has directed edges toward every feature node, and no other arc is allowed in this constrained topology. As the model structure is predefined, the learning of the CPDs of the variables is significantly simplified, making the naive Bayes classifier a very efficient model to train.

**Definition 3.3. (Naive Bayes classifier).** Given a training dataset $D$, a naive Bayes classifier models the JPD $P(\mathcal{X}, C)$ under the assumption of conditional independence between features in $\mathcal{X}$ given the class variable $C$. Consequently, the JPD can be simplified as follows:

$$P(\mathcal{X}, C) = P(C) \prod_{f=1}^{m} P(X_f|C),$$

where $P(C)$ is the prior probability of the class variable, and $P(X_f|C)$ represents the conditional probability of each feature $X_f$ given only the class variable$^1$.

Although simplistic, naive Bayes classifiers have been successfully applied in many scenarios [Sarkar and Sriram, 2001; Vitello et al., 2014; Queiroz et al., 2016; Razaque et al., 2017; Castro-Luna et al., 2020]. Nevertheless, their strong conditional independence assumption often comes at the expense of model accuracy if dependencies between features exist. To overcome this strict independence assumption, alternative models such as the tree-augmented naive Bayes (TAN) were introduced [Geiger, 1992; Friedman et al., 1997].

In a TAN structure, each feature is not only dependent on the class variable, but it can also

---

$^1$For clarity in equations and ease of reading, Chapters 3, 5 and 8, which focus on Bayesian network-based classifiers, primarily use $X_f$ and $C_y$ to denote feature and class variables, respectively.
depend on another feature. As a result, the arcs between the features form a tree-shaped structure that effectively captures some probabilistic relationships between them. Importantly, the tree structure of a TAN that maximizes the likelihood of some given data can be learned in polynomial time with respect to the number of features and training instances. This can be achieved using the Chow-Liu algorithm [Chow and Liu, 1968], in conjunction with Kruskal’s or Prim’s algorithms [Friedman et al., 1997; Cormen et al., 2022]. Thus, TAN allows for a relaxation of the conditional independence assumption of naive Bayes without significantly increasing computational complexity. Friedman et al. [1997] indicate in their studies that TAN compares to the naive Bayes in terms of robustness and computational complexity while offering superior accuracy in certain scenarios. Another characteristic to highlight from TAN is that it maintains the original set of features. This contrasts with seminaive Bayes models [Pazzani, 1998], which attempt to capture dependencies among features by extracting $K$ new variables $V_{R_v}$, each derived from the Cartesian products of original variables. For each variable $V_{R_v}$, where $v = 1, \ldots, K$, the set $R_v \subseteq \{1, \ldots, m\}$ denotes the indices of the original features involved in the product. Thus, a seminaive Bayes model is formed by $K + L$ feature nodes, where $L$ is the number of original features that remain unaltered. While this approach can represent dependencies among features, it could obscure the interpretability of the model, particularly if Cartesian products involve many variables.

**Definition 3.4. (Tree-augmented naive Bayes classifier).** Given a training dataset $D$, a TAN classifier extends the naive Bayes structure by allowing each feature $X_f$ to have at most another feature as its parent, in addition to the class variable $C$. The JPD is factorized as:

$$P(\mathcal{X}, C) = P(C) \prod_{f=1}^{m} P(X_f|C, Pa_X(X_f)),$$

where $Pa_X(X_f)$ denotes the single parent feature of $X_f$ in the TAN structure, except for the root node, which has only the class variable as its parent.

Modeling dependencies between features beyond those enclosed in a tree structure can be beneficial in specific situations [Friedman et al., 1997]. To potentially enhance classification performance, one could consider relaxing the strict topology of the TAN model, although this flexibility could come at the expense of computational efficiency. For example, the $k$-dependence Bayesian classifier ($k$-DB) [Sahami, 1996] allows structures where a feature depends on up to $k$ parent variables, in addition to the class variable. An even more general structure, known as Bayesian network-augmented naive Bayes (BAN), still maintains the class variable as the shared parent of all features but allows them to adopt a general BN structure without restricting their number of parents. Finally, unrestricted BNCs do not pose any restriction on their structure as long as they satisfy the acyclicity constraint, allowing for richer structures in their DAGs [Bielza and Larrañaga, 2014]. For an in-depth study on different BNCs, readers are referred to Friedman et al. [1997] and Bielza and Larrañaga [2014].
Definition 3.5. *(Unrestricted Bayesian network classifier)*. Consider a set of random variables \( V = \{ X_1, \ldots, X_m, C \} \), where \( C \) is a discrete class variable, and \( X = \{ X_1, \ldots, X_m \} \) are the features. An unrestricted Bayesian network classifier is a BN that is designed for classification purposes, represented by \( B = (G, B) \), where \( G \) is a DAG over \( V \) and \( B \) is the set of parameters. The DAG \( G \) can potentially have any structure, meaning that the class variable \( C \) can have parent variables and is not limited to being a root node. Thus, the JPD for an unrestricted BNC over \( V \) is given by:

\[
P(X, C) = P(C|Pa(C)) \prod_{f=1}^{m} P(X_f|Pa(X_f)),
\]

where \( Pa(X_f) \) denotes the set of parent variables of \( X_f \) in \( G \), which could include or not the class variable \( C \).

Figure 3.1: Illustration of the discussed BNCs, classified by the factorization of the JPD.

Learning the structure that maximizes the likelihood of a given dataset in a more general BNC is NP-hard, even when restricting the nodes to have up to \( k \) parents, where \( k \geq 3 \).
Chapter 3. Classification algorithms

[Chickering et al., 2004]. As the number of features increases, the time required to find the optimal structure grows exponentially. Therefore, the structure of these classifiers can be learned using algorithms such as those studied in Chapter 2 (Section 2.4.2), but considering the distinctions between feature and class variables.

Once a BNC is learned, it can be used to classify previously unseen instances. This classification process is essentially an inference task on the class variable given the observed feature values in the instances (evidence). Using the Bayes’ theorem, the posterior probabilities for each class variable value given the feature values are computed. Then, the class value that maximizes the posterior probability for a given instance, i.e., the maximum a posteriori (MAP) estimate for the class variable, is assigned to the instance.

**Definition 3.6. (Maximum a posteriori estimation).** Given a training dataset \( D \) from which a BNC has been learned, the BNC defines the JPD \( P(\mathcal{X}, C) \) over the features \( \mathcal{X} \) and a class variable \( C \). For a new, unseen instance \( \mathbf{x} \) from a dataset \( D_p \), the MAP estimate is given by:

\[
c^* = \arg \max_{c_j \in \Omega_C} P(C = c_j | \mathbf{x}),
\]

where \( c^* \) is the most probable class value for the instance \( \mathbf{x} \). Using Bayes’ theorem, the posterior probability of each class value is computed as follows:

\[
P(C = c_j | \mathbf{x}) = \frac{P(\mathbf{x} | C = c_j)P(C = c_j)}{P(\mathbf{x})},
\]

where each term is derived from the parameters of the BNC. In practice, the probability of the evidence \( P(\mathbf{x}) \) can be ignored during classification, as it remains constant across all class values, unless the posteriori probabilities of each class value are required.

### 3.4 Multi-dimensional classification

Certain classification problems require predicting simultaneously the state of multiple \( d \) class variables \( \mathcal{C} = \{C_1, \ldots, C_d\} \). While training individual classifiers for each class variable is feasible, this approach would fail to identify inter-class dependencies, which could provide relevant information for the classification task. Considering this limitation, more sophisticated solutions have been developed to exploit the dependencies between the class variables and enhance the prediction accuracy. In this scenario, we extend Definition 3.1 of a training dataset of the previous section to account for multi-dimensional classification.

**Definition 3.7. (Training dataset (multi-dimensional classification)).** Let \( D = \{\mathbf{z}_i = (\mathbf{x}_i, c_i), 1 \leq i \leq N\} \) be a training dataset for a classification task, where \( \mathbf{z}_i \) denotes the training instance \( i \) with a vector of feature values \( \mathbf{x}_i = (x_{i1}, \ldots, x_{im}) \) of a fixed dimension \( m \)
and a class vector \( \mathbf{c}_i = (c_{i1}, \ldots, c_{id}) \), where \( d \) is the number of class variables. When \( d = 1 \), the learning problem is a one-dimensional supervised classification, while \( d > 1 \) describes a multi-dimensional scenario. Each class variable \( C_y \) has \( k_y \) distinct values, so if \( k_y = 2 \), \( \forall y = 1, \ldots, d \), we have a binary \( (d = 1) \) or multi-label \( (d > 1) \) problem, while if \( k_y > 2 \) for some \( y \), it can be a multi-class \( (d = 1) \) or multi-dimensional \( (d > 1) \) classification.

Based on the problem presented by Nodelman et al. [2002] in Figure 2.3, the following example illustrates a possible multi-dimensional classification problem.

**Example 3.1.** Figure 3.2 presents a hypothetical multi-dimensional classification problem based on the drug effect graph of Figure 2.3. In this scenario, a doctor may have different observations taken on a patient during a given period of time, and they need to predict the state of multiple conditions of the patient. This prediction may include if the patient is physically active [yes or no], the stress level that could be suffering the patient [low, moderate or high], if they have an eating disorder [anorexia, bulimia or no disorder] [Codecasa and Stella, 2013], the medication tolerance as the body of the patient could adapt to a prolonged presence of the drug [low, moderate or high] or the overall health status that could be perceived by the patient [poor, fair or good].

This graph shows that the class variables to be predicted also present dependencies that could provide significant information for the overall prediction. For example, physical activity is known to reduce stress levels [Rimmel et al., 2009], which, in turn, might reduce the likelihood of stress-induced eating disorders [Strober, 1984]. In addition, patients who exhibit high tolerance to the drug may be susceptible to more frequent joint pain episodes, potentially impacting their overall health status.

Multi-dimensional classification tasks are primarily addressed through two main strategies: problem transformation and algorithm adaptation methods [Tsoumakas and Katakis, 2007]. While originally framed for multi-label classification, these strategies equally apply to multi-dimensional classification domains, as discussed by Gil-Begue et al. [2021]. First, problem transformation methods break down complex multi-dimensional problems into more straightforward one-dimensional tasks. The main idea is to adapt this complex problem to the plethora of algorithms available for one-dimensional classification. On the other hand, algorithm adaptation methods are designed to handle multi-dimensional classification problems directly without performing any transformation on the problem. Thus, they modify or extend established one-dimensional classification algorithms to inherently consider and manage the complexities of multi-dimensional classification problems. This section only covers a small proportion of problem transformation and algorithm adaptation methods. For a deeper study of the topic, readers are referred to Sorower [2010], Zhang and Zhou [2013] and Tarekegn et al. [2021].

Among the problem transformation methods, the binary relevance (BR) method stands out.
for its simplicity [Boutell et al., 2004; Godbole and Sarawagi, 2004]. BR trains separate one-dimensional classifiers for each class variable, offering a direct and simple solution but without capturing inter-class dependencies. Unlike BR, the label powerset (LP) method [Boutell et al., 2004; Tsoumakas and Katakis, 2007] can capture these dependencies by creating a compound class variable that captures all potential combinations of class values from the different class variables. Nonetheless, the LP method also comes with several disadvantages. One of its main problems is that as the number of class variables $d$ increases, class value combinations can grow exponentially, leading to imbalanced datasets where certain combinations have limited training instances. The random k-labelsets (RAkEL) method [Tsoumakas et al., 2011] divides the class variables into random smaller subsets of $k$ variables and employs the LP method in each of them. Then, for each subset, a one-dimensional classifier is trained, and the outputs of the classifier ensemble are merged for the final prediction. RAkEL offers a more scalable solution

---

Figure 3.2: Extension of the drug effect graph to a multi-dimensional classification problem (class variables are highlighted in gray).
than LP, avoiding its combinatorial complexity. Another possible solution to the exponential increase of combinations of LP is given by the pruned sets (PS) method [Read et al., 2008]. The PS method reduces class value combinations by only considering the most frequent found in the training data, pruning combinations based on a predefined threshold. This method effectively reduces the problem’s dimensionality and improves computational efficiency; however, there is a potential information loss if infrequent but important combinations are pruned. This can be mitigated by using an ensemble of classifiers derived from, possibly, different pruning combinations. Read et al. [2008] experimentally proved that using the ensemble of PS can further improve the predictive performance. However, both the RAkEL and PS methods come with their own challenges, particularly in choosing appropriate parameters for a given problem. Finally, another alternative to capturing inter-class dependencies while keeping the simplicity of the BR method is provided by the chain classifiers [Read et al., 2011]. This technique sequentially trains a set of classifiers, one for each class variable, but extends the feature spaces with the ground-truth classes of preceding classifiers. This solution’s main inconvenience is that it depends on the order in which classifiers are applied.

Although this dissertation will apply a problem transformation method in Chapter 8, the primary focus of Chapters 5, 6 and 8 is on the adaptation of existing algorithms to perform multi-dimensional classification. This approach tackles the problem more directly without the need for transforming or redefining the classification problem. Several learning algorithms, such as decision trees [Clare and King, 2001], support vector machines [Elisseeff and Weston, 2001], kNN [Zhang and Zhou, 2007], AdaBoost [Schapire and Singer, 2000] or BNCs [van der Gaag and de Waal, 2006; Bielza et al., 2011] have already been extended to this scenario.

Continuing the trajectory established in this dissertation, Bayesian classification models will be considered to address the multi-dimensional classification problem. In this scenario, the multi-dimensional Bayesian network classifier (MBC) [van der Gaag and de Waal, 2006; Bielza et al., 2011] extends traditional BNCs to handle multi-dimensional classification problems.

**Definition 3.8. (Multi-dimensional Bayesian network classifier).** Given a set of random variables $\mathcal{V} = \{X_1, \ldots, X_m, C_1, \ldots, C_d\}$, where each $C_y$ (for $y = 1, \ldots, d$) is a discrete class variable, and $\mathcal{X} = \{X_1, \ldots, X_m\}$ are the features, an MBC is a BN designed for multi-dimensional classification purposes, represented by $\mathcal{B} = (\mathcal{G}, \mathcal{B})$, where $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a DAG over $\mathcal{V}$ and $\mathcal{B}$ is the set of parameters. Unlike traditional BNCs with a single class variable, in an MBC, the DAG represents dependencies among multiple class variables and features. As a result, the set of arcs $\mathcal{A}$ is partitioned into three sets, dividing the structure of the MBC into three subgraphs:

1. The class subgraph $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{A}_C)$, which is formed by the arcs $\mathcal{A}_C$ between the vertices of the class variables $\mathcal{V}_C$.

2. The feature subgraph $\mathcal{G}_\mathcal{X} = (\mathcal{V}_\mathcal{X}, \mathcal{A}_\mathcal{X})$, which is formed by the arcs $\mathcal{A}_\mathcal{X}$ between the
vertices of the features $\mathcal{V}_{\mathcal{X}}$.

3. The bridge subgraph $\mathcal{G}_{\mathcal{C}X} = (\mathcal{V}, \mathcal{A}_{\mathcal{C}X})$, which is formed by the arcs $\mathcal{A}_{\mathcal{C}X}$ from the vertices of the class variables $\mathcal{V}_{\mathcal{C}}$ to the vertices of the features $\mathcal{V}_{\mathcal{X}}$.

Given a training dataset $\mathcal{D}$, where $d > 1$ class variables are present, an MBC models the JPD $P(\mathcal{X}, \mathcal{C})$ as follows:

$$P(\mathcal{X}, \mathcal{C}) = \prod_{y=1}^{d} P(C_y|\text{Pa}(C_y)) \prod_{f=1}^{m} P(X_f|\text{Pa}(X_f)),$$

where $\text{Pa}(X_f)$ denotes the set of parent variables of $X_f$ in $\mathcal{G}$ and $\text{Pa}(C_y)$ represents the set of parent variables for the class variable $C_y$, which can include other class variables. ■

Note that with Example 3.1, we anticipate one of our contributions to this dissertation, which proposes a novel predictive model to solve multi-dimensional classification tasks on categorical time series data using CTBNs. Thus, the structure shown in this example does not correspond with the graph of an MBC, as its restricted topology does not allow cycles in the feature subgraph. Nonetheless, the MBC model inspired this dissertation to explore the more complex setting of multi-dimensional classification of temporal sequences using PGMs. Before introducing our proposed model in Chapter 5, it is necessary to explore the application of CTBNs to the one-dimensional classification problem.

### 3.5 Continuous-time Bayesian network classifiers

As commented in Section 2.3, the classical definition of a BN is limited to model static data. Nonetheless, this dissertation considers dynamic data with a temporal dimension. In the case of Chapters 5 to 8, the focus is on how to exploit this temporal information to improve the classification of temporal sequences, which are time series data whose observations can be taken at irregular timestamps.

For the classification of temporal sequences, strategies are broadly divided into three primary categories [Xing et al., 2010]. A common strategy encompasses feature-based methods [Nanopoulos et al., 2001], which resemble the problem transformation methods used for static multi-dimensional classification. This strategy reformulates the original dataset by extracting new features that encapsulate the time series dynamics within a given time window, allowing the application of conventional static classifiers. However, this method may entail the costly extraction of a potentially large number of variables and the loss of relevant information for the classification process. Distance-based methods give another solution to the classification of time series, which are based on computing some form of distance or similarity measure between the sequences. A straightforward yet effective solution combines the $k$NN algorithm
with dynamic time warping (DTW) [Susto et al., 2018]. Here, DTW serves as an algorithm to compare and align sequences, while $k$NN uses these alignment scores to classify the time series data based on the closest or most similar labeled sequences in the training set. Finally, model-based methods work by directly training a suitable model on the time series data and using it for classification. As with BNs, the interest in CTBNs is not solely motivated by knowledge discovery, but these models can also be applied to the classification of unseen sequences. Therefore, we will focus on a model-based method to classify categorical time series data. In this scenario, commonly used methods such as $k$NN combined with DTW may yield limited results. This limitation arises because they require encoding the states of the features into numerical values. Such a conversion can be challenging or even meaningless, especially when the data that we study in this dissertation could lack a clear order of the variable states.

Definition 3.9. (Categorical time series classification (one-dimensional)). This task consists of training a classification model with a (multivariate) time series dataset, whose sequences $S_l = \{x_1^{t_1}, \ldots, x_m^{t_l}, c_l\}$ describe the transitions of $m$, time-dependent, discrete features $X$ and have assigned a unique state $c_l$ for a, time-independent, class variable $C$. The objective is to use the model to predict the state of $C$ on a previously unseen sequence $S_p = \{x_1^{t_p}, \ldots, x_m^{t_p}\}$, where class variable information is missing, by analyzing the state transitions of the features.

The family of classification models known as CTBNCs is able to perform classification over the data introduced in Definition 3.9 by incorporating a new, time-independent, class variable node to the CTBNs [Stella and Amer, 2012]. As in a conventional classification setting, this class variable represents either an unknown class value expected in the future or a static explanation of the temporal sequence [Codecasa, 2014]. Specifically, the static class variable of CTBNCs can have the following roles:

- **Predictive role.** The static class variable represents a future class value that is subject to change as the temporal sequence is still developing. Thus, the CTBNC is tasked with predicting the class value by analyzing existing time series data. For example, this could involve predicting the onset of a patient’s disease based on their currently available health-related data.

- **Descriptive role.** The objective is to determine the class value that best explains or corresponds to a temporal sequence, which is assumed to be complete, thereby offering a static explanation or categorization of this observed sequence for a specific context. An example would be classifying the type of a pre-existing medical condition based on a patient’s diagnostic tests.

Definition 3.10. (Continuous-time Bayesian network classifier). Given a set of discrete random variables $V = \{X_1, \ldots, X_m, C\}$, a CTBNC is a pair $\mathcal{C} = (\mathcal{N}, P(C))$, where
N is a CTBN over time-dependent features $\mathcal{X} = \{X_1, \ldots, X_m\}$ and $C$ is a time-independent class variable fully specified by the marginal probability $P(C)$ on states $\Omega_C = \{c_1, \ldots, c_k\}$. The graph of a CTBNC has the same properties as those of a CTBN but includes a class variable node with no parents, i.e., $Pa(C) = \emptyset$.

As for learning the structure of a CTBNC, this can be done using the structure learning algorithms for CTBNs of Section 2.4.2, only taking into account that the topology of a CTBNC does not allow the class variable to have parents. As with a general CTBN, it is possible to find the optimal structure of a CTBNC in polynomial time with respect to the number of variables and size of the dataset if a maximum number of parents is set for its nodes [Nodelman et al., 2003; Codecasa and Stella, 2014].

Stella and Amer [2012] introduced a method to perform classification on a fully observed sequence $S_p = \{x_{t_1}^{t_{p_1}}, \ldots, x_{t_{p_T}}^{t_{p_T}}\}$, where every transition is captured without any missing observation. Classification is then performed with CTBNCs by selecting the class value that maximizes the posterior probability (MAP estimate) given $S_p$ (evidence). This resembles the classification procedure in BNCs, as detailed in Definition 3.6:

$$P(c|S_p) = \frac{P(S_p|c)P(c)}{P(S_p)} \propto P(S_p|c)P(c).$$

Building on the approach described by Stella and Amer [2012] (see Section 5.4 for a more in-depth explanation of the derivation), the predicted class value $c^*$ for a sequence $S_p$ is determined with a CTBNC as follows:

$$c^* = \arg\max_{c \in \Omega_C} P(c) \prod_{j=1}^{T_p-1} \prod_{f=1}^m \exp \left( -q_{x_t^{t_j}}^{pa(X_f)} \delta_j \right) P(x_{t_{j+1}}^{t_{j+1}} | x_{t_j}^{t_j}, c),$$

where the feature $X_f$ stays in state $x_{t_{j}}^{t_{j+1}}$ during the time interval of length $\delta_j = t_{j+1} - t_j$ between two observations of the sequence.

Example 3.2. Figure 3.3 presents an example of the structure of a CTBNC as introduced by Codecasa and Stella [2013]. This model identifies potential eating disorders by analyzing the temporal dynamics of an individual’s eating behaviors. This example assumes that the presence or absence of an eating disorder would influence the evolution of the eating, full stomach and hungry features and the temporal relationships between them. ■
3.6 Performance measures

Performance measures for multi-dimensional classifiers should consider the performance of the model on multiple class variables simultaneously. Although the literature on this topic is limited, several performance measures have already been proposed for this context. The following measures will be used to compare the performance of classifiers in Chapters 5, 6 and 8 of this dissertation:

- **Global accuracy** [Bielza et al., 2011]. The ratio of sequences that were correctly classified for all class variables, i.e., a partially correct or completely incorrect classification is considered an error:

  \[
  Acc = \frac{1}{N} \sum_{l=1}^{N} \delta(c'_l, c_l),
  \]

  where \( c'_l \) and \( c_l \) are the predicted and actual classes of sequence \( l \), respectively, and \( \delta(\cdot, \cdot) \) is the Kronecker’s delta function, so \( \delta(c'_l, c_l) = 1 \) if \( c'_l = c_l \) and 0 otherwise.

- **Mean accuracy** [Bielza et al., 2011]. The mean of the accuracies obtained for each class variable separately:

  \[
  \overline{Acc} = \frac{1}{d} \sum_{y=1}^{d} Acc_y = \frac{1}{d} \sum_{y=1}^{d} \frac{1}{N} \sum_{l=1}^{N} \delta(c'_y c_l),
  \]

  where \( Acc_y \) is the accuracy for class variable \( C_y \).

- **Global Brier score** [Fernandes et al., 2013]. Calibration measure that assesses the accuracy of probabilistic classifiers by considering the probability that is assigned to
Chapter 3. Classification algorithms

multi-dimensional predictions:

\[ Bs = \frac{1}{N} \sum_{l=1}^{N} \sum_{g=1}^{|I_C|} \left( P(C = c_g|x_l^{t_1}, \ldots, x_l^{t_I}) - \delta(c_g, c_l) \right)^2, \]

where \( c_l \) is the actual class configuration of the sequence \( l \) and \( I_C = \Omega_{C_1} \times \cdots \times \Omega_{C_d} \) is the space of joint configurations of the class variables. The closer this measure is to zero, the more calibrated the classifier will be.

- **F_1 score.** Harmonic mean of the precision \( P \) and recall \( R \) on a class variable \( C \) with class value \( c_j \):

\[ F_1 = 2 \frac{PR}{P + R} = 2 \frac{tp_{c_j}}{2tp_{c_j} + fp_{c_j} + fn_{c_j}}, \]

where \( tp_{c_j} \), \( fp_{c_j} \) and \( fn_{c_j} \) are the counts for true positives, false positives and false negatives, respectively, for class value \( c_j \).

- **Learning time (Updating time).** The time required to train (update) a model on a given dataset.

- **Classification time.** The time required for a trained model to predict the class configurations of some previously unseen sequences.

Traditional equations for precision, recall and, therefore, F_1 score can only be used for a unique binary class variable. However, Gil-Begue et al. [2021] extended them for multiple, possibly non-binary, class variables. Let \( B \) be a function that computes any of these performance measures \( P, R \) or \( F_1 \) score from a confusion matrix. Then, the measures are obtained with macro- and micro-averaging as follows:

- **Macro-averaging.** This approach averages the scores of each class variable \( C_y \) with \( c_j \) values:

\[ B_{macro} = \frac{1}{d} \sum_{y=1}^{d} B_{C_y}, \text{ where } B_{C_y} = \begin{cases} \frac{1}{|\Omega_{C_y}|} \sum_{c_j} B(tp_{C_y}, fp_{C_y}, tn_{c_j}, fn_{c_j}) & \text{if } |\Omega_{C_y}| > 2 \\ B(tp_{C_y}, fp_{C_y}, tn_{C_y}, fn_{C_y}) & \text{otherwise,} \end{cases} \]

and \( tn_{c_j} \) is the counts for true negatives for class \( c_j \). Note that if the class variable \( C_y \) is binary, only the confusion matrix for one of its classes \((tp_{C_y}, fp_{C_y}, tn_{C_y}, fn_{C_y})\) is considered.

- **Micro-averaging.** This approach aggregates the counters of the confusion matrices \((TP_{C_y}, FP_{C_y}, TN_{C_y} \text{ and } FN_{C_y})\) of each class variable \( C_y \) before the overall performance
measure is computed:

\[ B_{\text{micro}} = B(\sum_{y=1}^{d} TP_{Cy}, \sum_{y=1}^{d} FP_{Cy}, \sum_{y=1}^{d} TN_{Cy}, \sum_{y=1}^{d} FN_{Cy}), \]

where

\[ \{ TP_{Cy}, FP_{Cy}, TN_{Cy}, FN_{Cy} \} = \begin{cases} \frac{1}{|\Omega_{Cy}|} \sum_{c_j} \{ tp_{c_j}, fp_{c_j}, tn_{c_j}, fn_{c_j} \} & \text{if } |\Omega_{Cy}| > 2 \\ \{ tp_{Cy}, fp_{Cy}, tn_{Cy}, fn_{Cy} \} & \text{otherwise.} \end{cases} \]

These two averaging methods will be used in our experiments to calculate the F\(_1\) score\(^3\). However, only the macro-averaging approach will be considered in some experiments to calculate the F\(_1\) score, as the micro-averaging is equivalent to the mean accuracy when the cardinality of all class variables is the same and greater than two [Villa-Blanco et al., 2021]. This is derived as follows. As a false positive for a particular class value is, at the same time, a false negative for another when a multi-class class variable \(C_y\) is considered, i.e., \(|\Omega_{Cy}| > 2\) and \(\sum_{c_j} fp_{c_j} = \sum_{c_j} fn_{c_j}\), both the micro-averaged precision and recall for \(C_y\) are equivalent and, therefore, equal to the micro-averaged F\(_1\) score. Given that the micro-averaged precision and the accuracy for \(C_y\) can be computed as follows:

\[
P_{\text{micro}}y = \frac{\sum_{c_j} tp_{c_j}}{\sum_{c_j} tp_{c_j} + \sum_{c_j} fp_{c_j}} = \frac{\sum_{c_j} tp_{c_j}}{N} = Acc_y, \tag{3.1}
\]

then we can infer that the mean accuracy of multi-class class variables with the same cardinality is equivalent to the micro-averaged F\(_1\) score:

\[
F_{1\text{micro}} = \frac{\sum_{y=1}^{d} \frac{1}{|\Omega_{Cy}|} \sum_{c_j} tp_{c_j}}{\sum_{y=1}^{d} \frac{1}{|\Omega_{Cy}|} \sum_{c_j} tp_{c_j} + \sum_{y=1}^{d} \frac{1}{|\Omega_{Cy}|} \sum_{c_j} fp_{c_j}} = \frac{\sum_{y=1}^{d} \sum_{c_j} tp_{c_j}}{\sum_{y=1}^{d} \left[ \sum_{c_j} tp_{c_j} + \sum_{c_j} fp_{c_j} \right]} = \frac{\sum_{y=1}^{d} \sum_{c_j} tp_{c_j}}{dN} = \frac{1}{d} \sum_{c_j} \frac{\sum_{y=1}^{d} tp_{c_j}}{N} = \overline{Acc},
\]

where in the last equality we have used Equation (3.1). Note that the normalization factors \(1/|\Omega_{Cy}|\) cancel each other out since the cardinality is assumed to be the same for each class variable. Otherwise, this equality would not hold.

\(^3\)Henceforth referred to as macro or micro F\(_1\) score for convenience.
Chapter 4

Feature subset selection and dynamic data

4.1 Introduction

Real-world problems are commonly characterized by a high feature dimensionality, which hinders the modeling and descriptive analysis of the data. However, some of these data may be irrelevant or redundant for the learning process. Different approaches can be used to reduce this information, improving not only the speed of building models but also their performance and interpretability. In this chapter, we focus on FSS techniques, which select a subset from the original set of features without making any transformation on them. The main focus is on the application of FSS algorithms in the context of data streams and feature streams, commonly referred to as incremental or online FSS algorithms.

Chapter 7 of this dissertation contributes with a comprehensive survey of state-of-the-art FSS techniques that were designed for their incremental application on data and feature streams. Traditional batch FSS algorithms may not be adequate to handle large volumes of data efficiently, either because memory problems arise or data are received sequentially. As this review will explore the previous learning problem in more depth, this background chapter will provide a brief but necessary introduction to the problem of FSS and the settings of data and feature streams.

Note that in this dissertation, we use the term streaming data to refer to all data that is collected on a temporal basis, including three main scenarios: time series data, data streams and feature streams. This background chapter and Chapter 7 will exclusively focus on the context of data and feature streams, which we will refer to as dynamic data contexts. However, these three scenarios are not mutually exclusive and can occur simultaneously, as we will later
study in Chapter 8.

As mentioned, this chapter will explore the topic of FSS on data and feature streams. For a comprehensive study of classification methods specific to data streams, the reader is referred to Gaber et al. [2007], Nguyen et al. [2015] and Gomes et al. [2017].

Chapter outline

The remainder of this chapter is as follows. Section 4.2 introduces the concept of FSS, highlighting its main benefits when applied to a learning task. This section offers an overview of some taxonomies of FSS techniques and formally introduces the definitions of relevant, irrelevant and redundant variables in a supervised context, and discusses different criteria for feature relevance and redundancy in unsupervised settings. Section 4.3 studies the data stream setting, where new training instances are dynamically received over time, addressing the impact of concept drifts. Section 4.4 discusses the characteristics of a feature stream, where the set of features dynamically evolves over a static set of instances. Finally, Section 4.5 introduces the rough set theory and its relationship with incremental FSS.

4.2 Feature subset selection

FSS is a specific type of dimensionality reduction technique that focuses on selecting the most informative (relevant) features of a dataset while discarding those considered redundant or irrelevant for a given learning task. The FSS task described in this dissertation should not be confused with feature extraction, which constructs new variables from the available ones to obtain a lower-dimensional feature space. FSS techniques address multiple challenges in data analysis and model learning that are inherent to high-dimensional data, providing interesting advantages [Kohavi and John, 1997; Guyon and Elisseeff, 2003; Saeys et al., 2007; Chandrashekar and Sahin, 2014; Miao and Niu, 2016; Khaire and Dhanalakshmi, 2022]:

- **Interpretability improvement.** Selecting a reduced subset of features facilitates the visualization and comprehension of the data, allowing for a deeper understanding of the processes that generated them. Moreover, predictive models learned on a reduced set of features are typically easier to interpret and understand, an especially important characteristic in applications such as decision support systems. As FSS maintains the original set of features, this is one of its main advantages with respect to feature extraction algorithms, which often generate a new, less intuitive feature space.

- **Model performance improvement.** FSS algorithms can enhance model performance by selecting the most relevant features for a given learning task while removing irrelevant
or redundant variables that can have a negative impact. The detriment of model
performance can be explained, for example, by the introduction of noise or the creation
of overly complex models. This complexity becomes particularly problematic when there
is a large number of features with respect to the number of data instances, increasing the
risk of overfitting the data. Consequently, models may not generalize well on previously
unseen datasets.

- **Computational requirements reduction.** Training models on fewer features can
significantly reduce the computational resources required in terms of time and storage.
Thus, models can be trained and make predictions more efficiently. This is especially
important for real-time applications or scenarios with limited computational resources.

FSS algorithms can be classified according to several criteria. One common categorization
is based on the learning problem they address, distinguishing them into supervised, semi-
supervised, and unsupervised methods [Miao and Niu, 2016]. This dissertation will only focus
on FSS for supervised and unsupervised learning tasks.

**Definition 4.1. (Supervised feature subset selection).** Given that our set of features
is represented by \( X = \{X_1, \ldots, X_m\} \) and our set of class variables by \( C = \{C_1, \ldots, C_d\} \), the
FSS task selects a subset of features \( S \subseteq X \) that can be used to obtain a mapping function
from \( S \) to \( C \) that is as good as possible for a certain criterion. This subset of features
\( S \) should contain the most relevant variables with respect to \( C \) while avoiding redundancies.

If the dataset is unsupervised, it may not be clear how to evaluate the importance of a feature
subset since there are no class variables (\( C = \emptyset \)) that could be used as a reference to define its
relevance. The ideal aim of unsupervised FSS is, therefore, to preserve the most important
characteristics of the data in a reduced feature space that contains the most discriminative
features for the task to be performed.

An exhaustive search over all possible \( 2^m - 1 \) feature subsets (excluding the empty set) to
find the best one is computationally infeasible even for datasets of moderate dimensionality.
The problem of finding the optimal feature subset is NP-hard, and the possible number of
subsets would grow exponentially with \( m \). Therefore, heuristic optimization methods are used
in practice, which provide a tractable way to find a reasonably good (though perhaps not
optimal) feature subset [Chandrashekar and Sahin, 2014].

As it can be seen in Table 4.1, FSS techniques are also commonly categorized into three types
based on their searching strategy and relationship with the model construction [Saey et al.,
2007; Guyon et al., 2008; Miao and Niu, 2016; Solorio-Fernández et al., 2020]:

- **Filter methods.** A score function is employed to evaluate each feature variable
or subset based on the training data alone. This approach ensures that the FSS
results are independent of learning models. Widely used score functions in supervised

51
scenarios include dependency measures like the Pearson (linear) correlation coefficient and statistical tests like the chi-square test [Biesiada and Duch, 2008], measures based on information theory, such as mutual information and symmetrical uncertainty [Biesiada and Duch, 2008], separability (or distance) measures such as the Fisher score [Gu et al., 2011] or consistency measures, such as the degree of dependency (consistency) in the context of rough-set theory [Pawlak and Skowron, 2007; Arauzo-Azofra et al., 2008]. In unsupervised learning, the application of these measures is more limited due to the absence of a target variable. However, some can still be used to evaluate the relationships between features themselves. Furthermore, statistical measures such as variance or mean absolute difference (MAD) could be valuable metrics in this unsupervised context [Ferreira and Figueiredo, 2012]. Features with higher variance or MAD might be more informative, as they indicate greater variability within the data [Ferreira and Figueiredo, 2012]. Nonetheless, note that more variability alone does not necessarily imply that a feature is inherently relevant for a learning task. Other measures considered by the unsupervised literature include a distance-based entropy measure [Dash et al., 2002] and the cosine similarity [Tabakhi et al., 2014], to name a few. Another unsupervised filter approach involves using a similarity measure to generate clusters within the feature space (rather than on the instance space) and select a representative feature from each of them [Liu and Motoda, 2007]. This representative feature can be, for example, the feature with the highest similarity to the cluster centroid [Almusallam et al., 2018]. Filter methods can be either univariate, assessing and ranking each feature individually, or multivariate, which, despite being more computationally demanding, take into account feature interactions. Thus, in certain cases, multivariate filtering can potentially remove redundancies between features and provide better results for the learning task. Examples of multivariate filter algorithms include the RELIEF-F [Kononenko, 1994], correlation-based feature selection (CFS) [Hall, 1999], fast correlation-based filter (FCBF) [Yu and Liu, 2004], minimum redundancy maximum relevance (mRMR) [Ding and Peng, 2005] and relevance redundancy feature selection (RRFS) [Ferreira and Figueiredo, 2012] algorithms.

**Wrapper methods.** These methods evaluate different feature subsets by training an individual model for each of them and report the subset whose model achieves the best testing performance. This performance measure could be, for example, the accuracy, area under the ROC curve or root-mean-square error in a supervised context [Kuhn et al., 2013], and trace criterion or silhouette score for evaluating cluster quality [Dy and Brodley, 2004; Hruschka and Covoes, 2005]. As we mentioned earlier, due to the exponential growth of possible subsets, evaluating each possible solution can quickly become computationally infeasible as the number of features grows. Thus, apart from defining a performance measure to evaluate the model, another key part of wrapper FSS is a heuristic procedure that guides the search in the space of possible feature
Table 4.1: Classification of FSS techniques by search strategy.

<table>
<thead>
<tr>
<th>Search strategy</th>
<th>Evaluation method</th>
<th>Model dependency</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>Score function</td>
<td>Model-independent</td>
<td>RELIEF-F [Kononenko, 1994], CFS [Hall, 1999], FCBF [Yu and Liu, 2004], mRMR [Ding and Peng, 2005], RRFS [Ferreira and Figueiredo, 2012]</td>
</tr>
<tr>
<td>Embedded</td>
<td>Inherent in model learning</td>
<td>Model-dependent</td>
<td>Decision trees [Quinlan, 1986], weighted naive Bayes [Zhang et al., 2021], Lasso regression [Tibshirani, 1996]</td>
</tr>
</tbody>
</table>

subsets [Saeys et al., 2007]. For this task, methods such as sequential forward selection (SFS) [Kittler, 1978], sequential backward elimination (SBE) [Kittler, 1978], and their corresponding floating variants [Pudil et al., 1994], or genetic algorithms [Holland, 1992; Yang and Honavar, 1998] can be employed. The advantage of this type of FSS is that it seeks to find the optimal feature subsets for a given learning task, as the model’s performance guides the FSS. Nonetheless, these techniques can be very computationally intensive, have a higher risk of data overfitting than filter solutions and depend on the selected model. As a result of this last limitation, the selected feature subset could perform poorly with other models [Saeys et al., 2007].

- **Embedded methods.** The FSS process is directly incorporated into the model learning, making it an inherent part of the learning algorithm. Embedded FSS methods have the advantage of being directly coupled with the learning of the predictive model, eliminating the need for a separate FSS procedure and typically being less computationally intensive than wrapper approaches [Saeys et al., 2007]. Nevertheless, similarly to wrapper methods, their solutions have the limitation of being model-specific. Examples of models with embedded FSS include decision trees [Quinlan, 1986], weighted naive Bayes [Zhang et al., 2021] or Lasso regression [Tibshirani, 1996].

Finally, some algorithms combine ideas from the previous categories, such as filter and wrapper approaches in Das [2001] or Solorio-Fernández et al. [2016]. These are often referred
to in the literature as hybrid or combined methods [Liu and Yu, 2005; Liu et al., 2018d; Solorio-Fernández et al., 2020].

## 4.2.1 Relevant and redundant features

There is no unique way to define relevant and redundant features: the definition depends on the learning task and objectives for which the FSS is conducted. For example, Yang et al. [2013] and Wang et al. [2014] learn sparse models, i.e., models where parameters of the least important features shrink to zero, by imposing certain constraints, such as L1-norm regularization, whereas Domingos and Hulten [2000], Yu et al. [2014] and Liang et al. [2014] explicitly define the goodness of the features based on measures such as entropy, mutual information or Gini index. By contrast, wrapper algorithms could simply report a feature subset based on the model that achieves, for example, the highest accuracy in supervised learning or silhouette score in clustering tasks. Thus, it becomes essential to identify important features that the chosen measures and models align with the specific goals of the learning task.

Formally, in a probabilistic context, feature relevance is commonly defined as in John et al. [1994], which states that, in supervised learning, relevant features can be divided into two groups: strongly relevant and weakly relevant features.

**Definition 4.2. (Strongly relevant feature).** These are variables that are indispensable for the final subset of selected features since their removal would imply a loss of prediction power. Given a feature $X_i$ and the rest of features $T = X \setminus \{X_i\}$, $X_i$ is strongly relevant to a class variable $C_j$, iff:

$$P(C_j|X_i, T) \neq P(C_j|T).$$

**Definition 4.3. (Weakly relevant feature).** This is a feature that may contribute to increasing the prediction power of a model, which depends on the other available features. A weakly relevant feature can become strongly relevant by removing other features. $X_i$ is considered to be weakly relevant iff it is not strongly relevant and there exists $T' \subset T$ such that:

$$P(C_j|X_i, T') \neq P(C_j|T').$$

If a feature is not strongly or weakly relevant, then it does not contribute to the prediction power, i.e., it is an irrelevant feature.

Selecting only the most relevant features may not be sufficient to perform an effective FSS. An essential part is to incorporate strategies that identify and reduce redundancy by excluding features that do not provide additional discriminative information that enhances the model’s
Chapter 4. Feature subset selection and dynamic data

<table>
<thead>
<tr>
<th>Irrelevant</th>
<th>Weakly relevant and redundant</th>
<th>Weakly relevant but non-redundant</th>
<th>Strongly relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevant feature subset</td>
<td>Optimal feature subset</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.1:** Categories of features with respect to a learning task.

Yu and Liu [2004] divide a set of features into four disjoint components as shown in Figure 4.1: (1) irrelevant features, (2) weakly relevant and redundant features, (3) weakly relevant but non-redundant features and (4) strongly relevant features, defining the optimal subset of features as formed by (3) and (4). The following Example 4.1, inspired by Yu and Liu [2004], attempts to clarify the concepts of irrelevant, strongly/weakly relevant and redundant features.

**Example 4.1.** Given a set of binary features \( \{X_1, X_2, X_3, X_4\} \), where \( X_2 \) is the complement of \( X_3 \), i.e., \( X_2 = X_\bar{3} \), and a class variable \( C \) defined by \( C = f(X_1, X_2) \) with function \( f(\cdot, \cdot) \), then the relevance of each feature for the learning task of \( f \) is as follows:

1. \( X_1 \) is a strongly relevant feature because it directly influences the outcome of \( C \).
2. Both \( X_2 \) and \( X_3 \) are weakly relevant. They provide similar information due to their complementary relationship, making one of them redundant. Hence, retaining only one would be optimal.
3. \( X_4 \) is irrelevant for the computation of \( C \).

Thus, the optimal feature subsets for classifying \( C \) are either \( \{X_1, X_2\} \) or \( \{X_1, X_3\} \).

Even though most studies focus on supervised problems, a wide range of areas involve high-dimensional unlabeled data. In these contexts, the challenge of identifying irrelevant and
redundant features is still present. This challenge is compounded in unsupervised settings since we do not have the information provided by class variables to guide the FSS [Dy and Brodley, 2004; Liu and Motoda, 2007]. Different objectives may guide the unsupervised FSS. For example, if the goal is dimensionality reduction for data compression or visualization [Mitra et al., 2002; Genender-Feltheimer, 2018], the subset of relevant features may be formed by those variables that preserve the underlying structure of the data, such as the overall distribution (global structure) or neighborhood relations (local manifold structure) [Du and Shen, 2015; Lei and Zhu, 2018; Zhu et al., 2018]. Meanwhile, in the case of clustering tasks, the relevance and redundancy of features could be defined based on their contribution to cluster discrimination [Liu and Motoda, 2007]. In other words, if the features preserve discriminative information. An irrelevant feature would not contribute to cluster discrimination as it provides no useful information. On the other hand, removing a redundant feature would not significantly impact the clustering outcome as its information is already captured by other variables [Dy and Brodley, 2004; Liu and Motoda, 2007].

In their comprehensive study on unsupervised FSS, Solorio-Fernández et al. [2020] identified some commonly used criteria for identifying relevant and redundant features in an unsupervised context. To determine feature relevance, they highlighted three main criteria:

1. The selection of features that most effectively preserve the manifold structure of the original data. Examples of this category are found in Banerjee and Pal [2014] and Luo et al. [2018].

2. Seeking cluster indicators, which can be used as a target variable. Then, the obtained target variable enables the transformation of the unsupervised FSS into a supervised one. Examples of this category are found in Li et al. [2012] and Qian and Zhai [2013].

3. The choice of features based on their correlation (feature dependency), aiming to define a subset of features that exhibit the highest or lowest correlation between them. Examples of this category are found in Haindl et al. [2006] and Wang et al. [2015b].

In the case of redundant features, two main approaches were distinguished:

1. Quantifying the redundancy by evaluating the degree of dependence, similarity, association or correlation among features using statistical or information-based measures. Examples of this category are found in Yen et al. [2010] and Ferreira and Figueiredo [2012].

2. Assessing feature redundancy considering a specific task or objective concept. The focus is on defining the relationships among the features by evaluating if they are redundant for an objective concept or task. Examples of this category are found in Zhao et al. [2010] and Zhu et al. [2016].
Chapter 4. Feature subset selection and dynamic data

### Table 4.2: Example dataset of phishing websites.

<table>
<thead>
<tr>
<th>LongURL</th>
<th>LinksToPage</th>
<th>AgeDomain</th>
<th>IsReliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>z_1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>z_2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>z_3</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>z_4</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>z_5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>z_6</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

### 4.3 Data streams

There are certain problems where not all the training instances are available at the beginning but are obtained dynamically from a data stream \( \mathcal{D} = \lim_{T \to \infty} \bigcup_{t=1}^{T} \{z_t^i = (x_t^i, c_t^i)\}, 1 \leq i \leq N_t \} = \lim_{T \to \infty} \bigcup_{t=1}^{T} \mathcal{D}^t \). Therefore, new training instances \( z_t^i \) can be received at each time \( t \), and the number of time steps \( T \) could be unknown and theoretically infinite. Chapter 7 will extensively study the application of FSS algorithms in such a dynamic setting. Subsequently, Chapter 8 will focus on experimentally updating our proposed multi-dimensional classifier under these conditions.

The task of FSS on data streams is to dynamically update a subset of features, \( S^t \), in response to the arrival of new training instances. This process leads to the formation of an updated subset, \( S^{t+1} \), as the underlying distribution of the data could have changed. Therefore, the FSS should account for possible changes in the relevance and redundancy of the features. Note that in the case of an unsupervised problem, a training instance would not have class variables that could guide the FSS, i.e., \( z_t^i = x_t^i \).

**Example 4.2.** For the following examples, we use the dataset for fraudulent website analysis from Table 4.2. As shown at the top of Figure 4.2, the instances of the dataset could have been sequentially received from a data stream until time \( t = 3 \). Then, at \( t = 4 \), a new website could be analyzed, and its information for the four available features would be included. These new data could be obtained from multiple sources, such as sensors, social networks and lab experiments.

#### 4.3.1 Concept drift

The evolving environment under study implies that the underlying distribution of the data can change over time, which produces the appearance of concept drifts. A concept at a time \( t \) is
<table>
<thead>
<tr>
<th>LongURL ($t = 1$)</th>
<th>LinksToPage ($t = 1$)</th>
<th>AgeDomain ($t = 1$)</th>
<th>IsReliable ($t = 1$)</th>
<th>DNSRecord ($t = 2$)</th>
<th>StatisticalReport ($t = 3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_1^t$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$z_2^t$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$z_3^t$</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$z_4^t$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$z_5^t$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$z_6^t$</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 4.2:** Example of data (top) and feature (bottom) streams of phishing websites.

typically defined as the JPD between the feature space and, in the case of supervised learning, the target space, i.e., $P^t(\mathcal{X}, \mathcal{C})$ [Gama et al., 2014]. Thus, concept drift occurs between $t$ and $t+1$ if the instances obtained from the data stream at those time instants are generated by different probability distributions:

$$P^t(\mathcal{X}, \mathcal{C}) \neq P^{t+1}(\mathcal{X}, \mathcal{C}).$$

Given the decomposition $P^t(\mathcal{X}, \mathcal{C}) = P^t(\mathcal{X})P^t(\mathcal{C}|\mathcal{X})$ of the joint probability distribution, two types of concept drift are of particular interest based on the nature of the change in the data: real concept drift (or class drift [Webb et al., 2016]), which implies changes in the underlying relationships between feature and class variables (supervised learning), i.e., $P^t(\mathcal{C}|\mathcal{X}) \neq P^{t+1}(\mathcal{C}|\mathcal{X})$, and virtual concept drift (or covariate drift [Webb et al., 2016]), which is related to changes in the distribution of features (supervised and unsupervised learning), i.e., $P^t(\mathcal{X}) \neq P^{t+1}(\mathcal{X})$. There is also the possibility that both types of concept drifts occur concurrently, which is not uncommon in real-world scenarios [Lu et al., 2018].
Research on concept drift mainly focuses on real concept drifts due to their direct impact on model decision boundaries and performance. Nonetheless, virtual concept drifts can also degrade performance, even if they do not affect the true decision boundaries, as they could introduce new observations in parts of the feature space that the model has not covered. Thus, the decision boundary may become insufficient or incorrectly learned considering the latest available data [Oliveira et al., 2023].

Note that, given the formulation of the Bayes’ theorem:

$$P^t(C|X) = \frac{P^t(X|C)P^t(C)}{P^t(X)}$$

the conditional probability $P^t(C|X)$ is directly influenced by the prior probabilities of the class variables $P^t(C)$ and the likelihood of observing the features given the class variables $P^t(X|C)$. Consequently, any changes in these probabilities can lead to a real concept drift [Xiang et al., 2023].

The interest in detecting these distribution changes is that they may alter the subset of features that should be considered relevant. This is known as feature drift and occurs when the relevant subset $S^t \subseteq X$ for a certain task at time $t$ differs from the subset obtained at another consecutive time instant $t + 1$, i.e., $S^t \not= S^{t+1}$ [Nguyen et al., 2012]. Example 4.3 illustrates how concept drifts can influence the feature subset.

Example 4.3. The following example explores a hypothetical scenario where it is predicted whether a customer will default on a loan. It illustrates how real and virtual concept drifts can impact the optimal feature subset for the learning task.

- **Initial scenario**.
  - **Class variable**: Loan default.
  - **Features**: Credit score, Employment status, Income and Loan amount.
  - **Optimal feature subset**: the Credit score and Income variables are strongly relevant and form the optimal feature subset. Employment status is weakly relevant and redundant as most customers have stable, full-time employment, so the Income variable gives sufficient information in this context. Due to the high average wealth of the customers, the Loan amount variable is initially considered irrelevant to the prediction task.

- **Real concept drift**.
  - **Change**: a major economic downturn occurs.

---

1In an ideal scenario, we may only be interested in changes affecting the optimal feature subset.
– **Effect**: the risk associated with larger loan amounts becomes more significant as customers find it harder to service large debts. Thus, the Loan amount variable becomes strongly relevant. Job losses also make the Employment status variable strongly relevant, as income alone no longer ensures financial stability. The minimum credit score for loan approval increases, so customers previously qualified as low-risk now fall into a higher-risk category. The Credit score variable is still strongly relevant, but its relationship with the class variable has changed.

– **Implication**: the model’s ability to predict loan defaults accurately has been compromised. The model needs to be updated to reflect the changed importance and interactions of the variables. The feature subset should now include new or previously discarded features.

- **Virtual concept drift**.
  - **Change**: the employment situation of customers becomes more diverse.
  - **Effect**: the Employment status variable is found to be strongly relevant given new data. Initially not included in the feature subset due to uniform employment status, it now provides significant predictive power. The relationship between this feature and the class variable does not change, as employment status was always an important indicator of loan default but could not be identified with the available data.

– **Implication**: the feature subset should be updated to include the Employment status variable to avoid a degradation of model accuracy.

In addition to the previously mentioned concept drift types, other forms and categorizations should be noted. An example is the novel class appearance, which occurs with the emergence of new class values [Masud et al., 2010]. Furthermore, concept drifts can be divided into abrupt, incremental, gradual or recurring, depending on their change patterns. The reader is referred to Gama et al. [2014], Webb et al. [2016], Barddal et al. [2017] and [Lu et al., 2018] for a more in-depth discussion of concept and feature drifts.

### 4.4 Feature streams

In the case of a feature stream, the number of instances will not increase, so \( N \) is a fixed known value. However, the set of features \( \mathcal{X} \) is not fixed, but \( m \) is incremented (or even reduced) over time. This learning setting receives special attention in the context of online FSS, which will be extensively examined in Chapter 7 and later experimentally studied in Chapter 8 in a classification context.
When a new feature from a feature stream becomes available, all its values for each instance are received, and the subset of selected features $S^t$ must be dynamically adapted to represent the most relevant features seen so far and avoid redundancies. Most of the existing approaches process new features individually, but it is also possible to receive the variables by groups. A group of features arriving at time $t$ is represented as $F^t = \{X^t_j, 1 \leq j \leq m_t\}$, where $X^t_j$ is its $j$-th feature and $m_t$ is the number of variables it contains. Thus, in the limit, we would obtain a set of feature groups $\mathcal{F} = \bigcup_{t=1}^{\infty} F^t$.

In the feature stream context, the problem of concept drift does not arise since the set of instances never varies. At two different points in time, the subset of relevant features may change, which is the definition of feature drift. However, this is caused by the appearance of new variables that make previous ones redundant.

**Example 4.4.** Continuing with the dataset of Table 4.2, but in this case, assuming all instances to be available in advance, new information about the websites could be obtained at times $t = 2$ and $t = 3$, as described at the bottom of Figure 4.2. In this case, the new data at $t = 2$ is related to whether the DNS record is found, and the data at $t = 3$ is the result of a new statistical report. The data are included as new features that contain values for all the available instances.

## 4.5 Rough set theory

When applying FSS algorithms, we are not restricted to a unique domain; we can handle very diverse data. That is why some studies seek approaches that do not need prior knowledge about the domain beyond the given data. This is the case for incremental FSS algorithms under approaches such as rough set theory, a mathematical tool to express vagueness (due to lack of information) by a boundary region [Pawlak, 1997] without needing any domain knowledge apart from the given data. Rough set theory enables working on problems where knowledge is incomplete, so it is ideal for the streaming environment under study since the complete dataset is unknown at the beginning of the process. Incremental FSS based on rough set theory, commonly known as incremental attribute reduction, is an important task for knowledge acquisition, and algorithms based on this theory are receiving considerable attention [Xu et al., 2011; Zhang et al., 2016]. This theory constitutes a fundamental basis for understanding a significant part of the contribution presented in Chapter 7. Therefore, it is essential to provide an introduction to this theory.

Rough set theory was originally proposed by Pawlak [1982], and it describes a subset of a universe using two subsets, the lower and upper approximations. This division allows hidden
knowledge to be discovered in datasets\(^2\) and expressed with decision rules. A dataset is defined as a tuple \(\mathcal{D} = \langle \mathcal{U}, \mathcal{A} \rangle\), where \(\mathcal{U}\) is a set of objects or instances, known as the universe, and \(\mathcal{A} = \{\mathcal{X} \cup \mathcal{C}\}\) is a set of attributes, where \(\mathcal{X}\) is the set of features and \(\mathcal{C}\) is the set of class variables, such that \(\mathcal{X} \cap \mathcal{C} = \emptyset\).

Rough set theory uses the indiscernibility relation between instances to divide them into disjoint sets of similar instances. This relationship is present when all instances of a set have the same values for all the studied attributes. Thus, each non-empty subset \(\mathcal{B} \subseteq \mathcal{A}\) determines a \(\mathcal{B}\)-indiscernibility relation \(\text{IND}(\mathcal{B}) = \{(z_i, z_j) \in \mathcal{U} \times \mathcal{U} \mid z_i^{|\mathcal{B}} = z_j^{|\mathcal{B}}\}\), where \(z_i^{|\mathcal{B}}\) represents the projection of instance \(z_i\) on attributes in \(\mathcal{B}\). This indiscernibility relation partitions \(\mathcal{U}\) into disjoint subsets of indistinguishable instances, which are known as equivalence classes or elementary sets. This partition is denoted as \(\mathcal{U}/\text{IND}(\mathcal{B}) = \{\mathcal{E}_1, \ldots, \mathcal{E}_v\}\), where \(v\) is the number of equivalence classes. To represent the equivalence class with respect to \(\mathcal{B}\) that contains a certain instance \(z_i\), we use the expression \([z_i]_\mathcal{B}\), i.e., \([z_i]_\mathcal{B} = \{z_j \in \mathcal{U} \mid z_i^{|\mathcal{B}} = z_j^{|\mathcal{B}}\}\).

Notably, equivalence classes formed only by features, i.e., \(\mathcal{U}/\text{IND}(\mathcal{X})\), or by class variables, i.e., \(\mathcal{U}/\text{IND}(\mathcal{C})\), are known as condition and decision classes, respectively.

Given any subset \(\mathcal{H} \subseteq \mathcal{U}\), rough set theory seeks to approximate \(\mathcal{H}\) using a lower and upper bound, which are defined with the equivalence classes induced by a feature set \(\mathcal{B}\) over \(\mathcal{U}\). These sets are the \(\mathcal{B}\)-lower approximation of \(\mathcal{H}\), denoted as \(\overline{\mathcal{B}}\mathcal{H}\), and the \(\mathcal{B}\)-upper approximation of \(\mathcal{H}\), denoted as \(\overline{\mathcal{B}}\mathcal{H}\). The \(\mathcal{B}\)-lower approximation contains all the instances that certainly belong to \(\mathcal{H}\) with the information given by \(\mathcal{B}\), i.e., \(\overline{\mathcal{B}}\mathcal{H} = \{z \mid [z]_\mathcal{B} \subseteq \mathcal{H}\}\), while the \(\mathcal{B}\)-upper approximation includes those instances that can possibly belong to \(\mathcal{H}\), i.e., \(\overline{\mathcal{B}}\mathcal{H} = \{z \mid [z]_\mathcal{B} \cap \mathcal{H} \neq \emptyset\}\). Finally, a rough set of \(\mathcal{H}\) with respect to \(\mathcal{B}\) is defined as the tuple formed by the \(\mathcal{B}\)-lower and \(\mathcal{B}\)-upper approximations, i.e., \(\langle \overline{\mathcal{B}}\mathcal{H}, \overline{\mathcal{B}}\mathcal{H} \rangle\). These approximations divide the universe into three disjoint regions: (1) the positive region \(\text{POS}^\mathcal{B}(\mathcal{H}) = \overline{\mathcal{B}}\mathcal{H}\), which contains all the instances from \(\mathcal{U}\) that can be classified as certainly belonging to \(\mathcal{H}\); (2) the negative region \(\text{NEG}^\mathcal{B}(\mathcal{H}) = \mathcal{U} \setminus \overline{\mathcal{B}}\mathcal{H}\), which represents the instances that for sure do not belong to \(\mathcal{H}\); and (3) the boundary region \(\text{BND}^\mathcal{B}(\mathcal{H}) = \overline{\mathcal{B}}\mathcal{H} \setminus \overline{\mathcal{B}}\mathcal{H}\), which consists of those instances that cannot be classified as belonging or not to \(\mathcal{H}\) due to a lack of knowledge. If \(\text{BND}^\mathcal{B}(\mathcal{H}) = \emptyset\), then \(\mathcal{H}\) is considered crisp (exact) since it is possible to define all instances as members or not of \(\mathcal{H}\). Otherwise, the set is rough (inexact) with respect to \(\mathcal{B}\), i.e., we cannot define the set precisely with the available knowledge.

For a certain dataset with class variables \(\mathcal{C}\) and given \(\mathcal{H} \in \mathcal{U}/\text{IND}(\mathcal{C})\), the positive, negative and boundary regions for a feature set \(\mathcal{B}\) given \(\mathcal{C}\) are: \(\text{POS}^\mathcal{B}(\mathcal{C}) = \cup \text{POS}^\mathcal{B}(\mathcal{H})\), \(\text{NEG}^\mathcal{B}(\mathcal{C}) = \cup \text{NEG}^\mathcal{B}(\mathcal{H})\) and \(\text{BND}^\mathcal{B}(\mathcal{C}) = \cup \text{BND}^\mathcal{B}(\mathcal{H})\), respectively. These regions are obtained by joining the resulting partitions of the universe for each of the decision classes induced by \(\mathcal{C}\).

As explained before, not all the available features may be necessary. Thus, attribute reduction

\(^2\)Commonly known as an information system or decision information system if there are class variables.
Chapter 4. Feature subset selection and dynamic data

seeks a minimal feature subset (or all possible subsets) that is sufficient to characterize the knowledge of a dataset, i.e., it discards all unnecessary features while preserving certain properties of the original dataset. This subset is called the minimal reduct, which could be required, for example, to maintain the same positive region as the one obtained with the original feature set. To respect the notation used so far, the reducts are denoted by $S$. As we are working in a dynamic environment, the reducts should be updated according to currently available data.

Example 4.5. Given the dataset of Table 4.2 with features $B = \{\text{LongURL}, \text{LinksToPage}, \text{AgeDomain}\}$ and $U = \{z_1, z_2, z_3, z_4, z_5, z_6\}$, it is possible to find that instances such as $z_1$ and $z_5$ are indiscernible with respect to $B$, so equivalence classes $\{z_1, z_5\}$, $\{z_2, z_4\}$ and $\{z_3, z_6\}$ are obtained. If the target set to approximate is $H = \{z | C(z) = 1\}$, i.e., $H = \{z_2, z_3, z_6\}$, where $C = \{\text{IsReliable}\}$, we can determine that $B \bar{H} = \{z_3, z_6\}$ and $\bar{B} \bar{H} = \{z_2, z_3, z_4, z_6\}$. Thus, $\text{POS}_B^U(H) = \{z_3, z_6\}$, $\text{NEG}_B^U(H) = \{z_1, z_5\}$, and $\text{BND}_B^U(H) = \{z_2, z_4\}$. As $\text{BND}_B^U(H) \neq \emptyset$, set $H$ is rough. If the values of $z_4$ are changed to, for example, $\text{LongURL}(z_4) = 0$, $\text{LinksToPage}(z_4) = 1$ and $\text{AgeDomain}(z_4) = 1$, the boundary region would be empty and $H$ would be crisp. This is easy to see in our example since those changes would make $B \bar{H} = \bar{B} \bar{H} = \{z_2, z_3, z_6\}$, allowing us to undoubtedly classify all the instances as belonging or not to $H$. Another way to achieve this could be by including new features to $B$.

Finally, whether we consider the original dataset or its version with $z_4$ modified, set $B$ can be further reduced to $S = \{\text{LongURL}, \text{LinksToPage}\}$ without losing significant information since the same approximation sets are obtained. This set cannot be further reduced while maintaining the same approximations. Thus, it can be assured that it is a minimal reduct.
Part III

CONTRIBUTIONS
Chapter 5

Multi-dimensional continuous-time Bayesian network classifiers

5.1 Introduction

Many classification problems imply the analysis of trends or dynamics that occur in sequences of time-ordered data to perform accurate predictions. Typical examples are found in finance, medicine, signal processing or industry, but more applications are emerging in virtually any domain [Dietrich et al., 2004; Jeong et al., 2011; Fulcher and Jones, 2014; Moskovitch and Shahar, 2015]. This chapter focuses on the complex scenario of multi-dimensional classification over multivariate time series, presenting a novel model for the task and a real-world problem where it is applied.

The multi-dimensional classification problem deals with the simultaneous classification of multiple class variables, i.e., it requires the definition of a mapping function that determines the output of several multi-class class variables based on a given input data. This learning problem is included in the more general multi-output paradigm, which also covers supervised learning problems with outputs of different data types, such as real-valued [Borchani et al., 2015] or ordinal [Ma and Chen, 2019]. The reader is referred to the comprehensive review of Xu et al. [2019] for a more in-depth reading about the multi-output learning paradigm.

Traditional classification algorithms are limited to the prediction of a unique variable, so they cannot be directly applied in the studied multi-dimensional context. Two simple approaches are commonly used to avoid this limitation: the definition of a compound class variable that collects all combinations of class values (LP method) and the learning of independent classifiers for each class variable (BR method). Nonetheless, both solutions involve a number of drawbacks, such as an exponential growth in the combinations for the LP method or the
impossibility of modeling the dependencies between class variables with the BR method. 

An alternative to avoid the independence assumption while keeping the simplicity of the 
BR method is given by chain classifiers [Read et al., 2011], which iteratively train a set of 
classifiers (one per class variable) whose feature spaces are extended with the ground-truth 
classes of their predecessors. However, this approach is really dependent on the order in which 
classifiers are applied, requiring the exploration of an intractable number of chain orders to 
find a suitable one. The performance of all these solutions could be improved by considering 
them along with frameworks such as that from Jia and Zhang [2020], which enriches our 
original data by extracting new features that encode information about the class variables.

The above methods transform the multi-dimensional problem into one or more one-dimensional 
classifications. Rather, we will study the adaption of existing algorithms, which tackle the 
problem more directly without requiring this preprocessing. Learning algorithms such as 
decision trees [Clare and King, 2001], support vector machines [Elisseeff and Weston, 2001], 
kNN [Zhang and Zhou, 2007] or BNCs [van der Gaag and de Waal, 2006] have already been 
extended to perform simultaneous classification of multiple class variables. However, some of 
these proposals focus on the multi-label classification subproblem, where all class variables 
are binary [Zhang and Zhou, 2013].

Apart from the multi-dimensional facet, the data studied in this chapter present a temporal 
dimension that cannot be ignored. In order to apply most of the above models over temporal 
data, it is common to follow a similar preprocessing strategy to that for static multi-dimensional 
data, called feature-based approach [Nanopoulos et al., 2001]. It transforms our original 
dataset by extracting new features that summarize the dynamics of time series during a time 
window. In this way, any traditional static classifier can be applied. Nonetheless, this implies 
the costly extraction of a probable large number of variables, which, in the end, may lose 
significant information. The adaptation of multi-dimensional classifiers to temporal data has 
not been extensively studied and few models can be directly applied in this context. Some of 
these algorithms include the multi-label kNN with DTW [Berndt and Clifford, 1994; Zhang 
and Zhou, 2007] and the long-short term memory recurrent neural networks [Hochreiter and 
Schmidhuber, 1997; Lipton et al., 2016]. Of these, the multi-label kNN stands out due to its 
simplicity and great performance [Batista et al., 2011]. Nonetheless, since it is a lazy classifier, 
it may imply a high space and classification complexity.

To the best of our knowledge, the direct application of PGMs to the multi-dimensional 
classification of time series has received no attention, although they provide interesting 
characteristics such as an intuitive representation of variable dependencies. DBNs [Dean 
and Kanazawa, 1989; Murphy, 2002] are widely studied in the literature, but they only 
model discrete time, later extended to CTBNs [Nodelman et al., 2002]. For one-dimensional 
classification, the CTBNCs were introduced by Stella and Amer [2012]. Here, we extend them 
to the multi-dimensional problem, as well as different methods for their learning from data.
We believe this is the first PGM able to perform time series multi-dimensional classification while explicitly modeling continuous time. More specifically, the main contributions of this chapter are the following:

- A novel multi-dimensional continuous-time Bayesian network classifier that is able to model discrete state multivariate time series data and classify them into multiple class variables. This proposal explicitly represents the temporal dynamics of features in continuous time and seeks to improve its predictions by modeling the dependencies between class variables.
- The introduction of algorithms for the learning of the parameters and structure of the presented model from data, as well as different structure constraints to adapt the model and its learning to the characteristics and demands of a certain problem.
- A comprehensive comparative study with 50 synthetic datasets and a real-world Industry 4.0 dataset to validate, under different conditions, the proposed model’s effectiveness and its performance improvements with respect to CTBNCs using the BR strategy.

This chapter includes the content of Villa-Blanco et al. [2021]. All developed software and studied datasets are freely available at https://github.com/carlvilla/Multi-CTBNCs.

Chapter outline

The remainder of this chapter is organized as follows. Section 5.2 introduces the multi-dimensional continuous-time Bayesian network classifier, while Section 5.3 presents the methods to learn its parameters and structure from data. Subsequently, Section 5.4 describes how the model performs multi-dimensional classification of unseen sequences. Section 5.5 presents synthetic and real-world experiments and discusses the results that the proposed model obtains. Finally, Section 5.6 concludes the chapter and highlights future lines of research.

5.2 Multi-dimensional continuous-time Bayesian network classifier

The existence of multi-dimensional classification data provoked the appearance of multi-dimensional Bayesian network classifiers (MBCs) [van der Gaag and de Waal, 2006] but in static settings. However, there are real-world problems where we need to classify temporal sequences into multiple class variables, i.e., a sequence $S_l = \{x_{t1}^l, \ldots, x_{t_T}^l, c_l\}$ now includes the state of $d$ class variables $c_l = \{c_{l1}, \ldots, c_{ld}\}$. Take as an example the problem presented in
Section 5.5.1.2, which requires identifying the power consumption state (discretized as high, low or inactive) of elements of an industrial machine based on the energy data it produces. Indeed, several individual models (one per element) can be used to predict each of them. However, this would not identify inter-element dependencies, which could provide valuable, or even crucial, information to classify certain class variables. For example, it is known that some elements always work together, while others cannot be active at the same time.

The multi-dimensional continuous-time Bayesian network classifier (Multi-CTBNC) that we introduce here seeks to extend CTBNCs to this more complex scenario, allowing us to model the inter-class interactions by capturing the probabilistic relationships of class variables with a BN.

Definition 5.1. *(Multi-dimensional continuous-time Bayesian network classifier).* A Multi-CTBNC $M = (G, B, Q, P^0_V)$ over a set of discrete-valued variables $V = \{X_1, \ldots, X_m, C_1, \ldots, C_d\}$ is formed by:

- A directed (possibly cyclic) graph $G = (V, A)$, where vertices $V$ are partitioned into those for features $V_X = \{X_1, \ldots, X_m\}$, $m \geq 1$, and for class variables $V_C = \{C_1, \ldots, C_d\}$, $d \geq 1$, and arcs $A = \{A_C, A_X, A_{CX}\}$ are divided into those between class variables $A_C \subseteq V_C \times V_C$, features $A_X \subseteq V_X \times V_X$ and from class to feature variables $A_{CX} \subseteq V_C \times V_X$.

- Class variable parameters $B$, which form CPTs.

- A set of CIMs $Q$, one for each feature $X_f$.

- An initial distribution $P^0_V$ to represent the initial state of a sequence. As more than one class variable is present, $P^0_V$ is specified as an MBC.

As the class variables do not depend on time, a Multi-CTBNC is based on capturing their probabilistic relationships with BNs. Thus, this model can be decomposed into a BN and a CTBN, which divide $G$ into three subgraphs:

1. The class subgraph $G_C = (V_C, A_C)$, which is defined by a BN that models the dependencies between class variables. This subgraph must be a DAG.

2. The feature subgraph $G_X = (V_X, A_X)$, which is defined by a CTBN that models the dependencies between features over time and, therefore, cycles may appear.

3. The bridge subgraph $G_{CX} = (V, A_{CX})$, which represents the dependencies of features on class variables and, therefore, it is defined by the same CTBN as the feature subgraph. It is also known as feature selection subgraph [van der Gaag and de Waal, 2006] since it specifies which features are relevant for classification. The bridge subgraph only contains arcs from class to feature variables.

Figure 5.1 shows a Multi-CTBNC graph and its constituent subgraphs. Note that the structure of this model shares many similarities with that of an MBC, despite the multiple differences
in their underlying paradigms, but the Multi-CTBNC allows the appearance of cycles in its feature subgraph.

5.3 Learning Multi-CTBNCs from data

5.3.1 Parameter learning

The parameters of a Multi-CTBNC are those of a BN and a CTBN. As we are assuming all variables to be discrete, Multi-CTBNC nodes would contain either CPTs (for class variables) or CIMs (for features). The parameters of the CPTs are:

- \( \beta^{\text{pa}(C_y)}_{c_j} \), probability of class variable \( C_y \) taking state \( c_j \) given the parents’ state \( \text{pa}(C_y) \).
  These are the parameters for the multinomial distribution over the class variables’ state.

In contrast, CIMs are summarized by two types of parameters:

- \( q^{\text{pa}(X_f)}_{x_j} \), intensity of feature \( X_f \) leaving state \( x_j \) when the parents’ state is \( \text{pa}(X_f) \). This is the parameter for the exponential distribution over the time a feature remains in a certain state.

- \( \theta^{\text{pa}(X_f)}_{x_j,x_z} = \frac{q^{\text{pa}(X_f)}_{x_j}}{q^{\text{pa}(X_f)}_{x_j}} \), probability of \( X_f \) transitioning from state \( x_j \) to \( x_z \), where \( x_j \neq x_z \), when a transition is known to occur and the parents’ state is \( \text{pa}(X_f) \). These are the parameters for the multinomial distribution over which state a feature will transition to.

Therefore, the parameters for each class variable, \( B^{\text{pa}(C_y)}_{C_y} = \{ \beta^{\text{pa}(C_y)}_{c_j} : c_j \in \Omega_{C_y}, \text{pa}(C_y) \in \} \).
\( I_{\text{Pa}(C_y)} \), and each feature, \( q^\text{Pa}_{X_f}(x_f) = \{ q^\text{Pa}_{x_j}(x_f) : x_j \in \Omega_{X_f}, \text{pa}(X_f) \in I_{\text{Pa}(x_f)} \} \) and \( \Theta^\text{Pa}_{X_f} = \{ \theta^\text{Pa}_{x_j,y}(x_f) : x_j, x_y \in \Omega_{X_f}, x_j \neq x_y, \text{pa}(X_f) \in I_{\text{Pa}(x_f)} \} \), define the parameters \( B = \{ B^\text{Pa}_{C_y} : C_y \in \mathcal{C}_y \}, q = \{ q^\text{Pa}_{x_j}(x_f) : x_f \in \mathcal{V}_{X_f} \} \) and \( \Theta = \{ \Theta^\text{Pa}_{X_f} : X_f \in \mathcal{V}_{X_f} \} \) of a Multi-CTBNC. To estimate these parameters, some sufficient statistics that summarize all observable data are recorded:

- \( N^\text{pa}(C_y) \), number of sequences where class variable \( C_y \) takes state \( c_j \) while the parents’ state is \( \text{pa}(C_y) \).
- \( N^\text{pa}(C_y) = \sum_{c_j} N^\text{pa}(C_y) \), number of sequences where parents of \( C_y \) have state \( \text{pa}(C_y) \) independently of the state of \( C_y \).
- \( M^\text{pa}_{x_j,x_z}(x_f) \), number of transitions of feature \( X_f \) from state \( x_j \) to \( x_z \) when the parents’ state is \( \text{pa}(X_f) \).
- \( M^\text{pa}_{x_j}(x_f) = \sum_{x_z \neq x_j} M^\text{pa}_{x_j,x_z}(x_f) \), number of transitions of \( X_f \) from state \( x_j \) to any other when the parents’ state is \( \text{pa}(X_f) \).
- \( T^\text{pa}_{x_j}(x_f) \), time of \( X_f \) spent in state \( x_j \) when the parents’ state is \( \text{pa}(X_f) \).

Given the structure of a Multi-CTBNC, its parameters can be estimated with methods like maximum likelihood estimation or Bayesian estimation. The first approach assumes that the parameters are constants, seeking those values that maximize the probability of the observable data, i.e., the maximum likelihood estimates. Parameters are then estimated as follows [Nodelman et al., 2003; Koller and Friedman, 2009]:

\[
\hat{\beta}^\text{pa}_{C_y} = \frac{N^\text{pa}(C_y)}{N_{\text{Pa}(C_y)}}, \quad \hat{\theta}^\text{pa}_{x_j}(x_f) = \frac{M^\text{pa}_{x_j}(x_f)}{T^\text{pa}_{x_j}(x_f)}, \quad \text{and} \quad \hat{\beta}^\text{pa}_{x_j,x_z}(x_f) = \frac{M^\text{pa}_{x_j,x_z}(x_f)}{M^\text{pa}_{x_j}(x_f)}.
\]

In the case of the Bayesian estimation, the parameters are considered random variables and a prior distribution is defined over them. This is a more interesting approach, as we discussed in detail in Section 2.4.1. Conjugate prior distributions are defined for the two types of distributions used by the Multi-CTBNC. The Dirichlet distribution serves as the conjugate prior for the multinomial distribution [Heckerman et al., 1995], and the Gamma distribution functions as the conjugate prior for the exponential distribution [Nodelman et al., 2003]. As conjugate priors are used, the posterior distribution of the parameters given the observed data follows the same distribution and, therefore, can be obtained analytically. Then, the parameters can be estimated using, e.g., their expected values, in the same way as the maximum likelihood estimation, but including the hyperparameters of the Dirichlet prior distributions \( \lambda^\text{pa}_{C_y} \) and \( \alpha^\text{pa}_{x_j,x_z}(x_f) \), and of the Gamma prior distribution \( \alpha^\text{pa}_{x_j}(x_f) \) and \( T^\text{pa}_{x_j}(x_f) \).
\[
\hat{\beta}_{c_j}^{pa}(C_y) = \frac{\lambda^{pa}(C_y) + \lambda^{pa}(C_y)}{N^{pa}(C_y) + \sum_{c_z} \lambda^{pa}(C_y)}, \quad \hat{q}_{x_j}^{pa}(X_f) = \frac{M^{pa}(X_f) + \alpha^{pa}(X_f)}{T^{pa}(X_f) + \tau^{pa}(X_f)},
\]

and

\[
\hat{\theta}_{x_j,x_z}^{pa}(X_f) = \frac{M^{pa}(X_f) + \alpha^{pa}(X_f)}{M^{pa}(X_f) + \alpha^{pa}(X_f)}.
\]

These hyperparameters can be seen as imaginary counts of the sufficient statistics that occur before any data is observed, i.e., \(\lambda^{pa}(C_y)\) is the number of times a class variable \(C_y\) takes state \(c_j\), \(\alpha^{pa}(X_f)\) is the number of transitions of a feature \(X_f\) from state \(x_j\) to state \(x_z\), \(\alpha^{pa}(X_f)\) is the number of transitions from state \(x_j\) to any other different state and \(\tau^{pa}(X_f)\) is the time \(X_f\) remains in state \(x_j\), all before a dataset \(D\) is considered.

Although the hyperparameters are commonly treated directly as the imaginary counts (see, for example, Nodelman et al. [2003]), we need to point out that the latter should also be defined in terms of the states of the nodes. Otherwise, the size of the imaginary dataset would be significantly influenced by the cardinalities of the nodes and their parents. In order words, the hyperparameters should be divided by the cardinality of the feature \(|\Omega_{X_f}|\) or class variable \(|\Omega_{C_y}|\) nodes and the cardinality of the space of joint states of their parent nodes \(I^{pa}(X_f)\) or \(I^{pa}(C_y)\), such that:

\[
\frac{\lambda^{pa}(C_y)}{|I^{pa}(C_y)||\Omega_{C_y}|}, \quad \frac{\alpha^{pa}(X_f)}{|I^{pa}(X_f)||\Omega_{X_f}|^2}, \quad \text{and} \quad \frac{\tau^{pa}(X_f)}{|I^{pa}(X_f)||\Omega_{X_f}|},
\]

are the values of the imaginary counts\(^1\). The hyperparameters can be defined by using expert knowledge and/or optimization techniques [Yang and Shami, 2020], such as random search [Bergstra and Bengio, 2012] or Bayesian optimization [Snoek et al., 2012].

### 5.3.2 Structure learning

As discussed in Section 2.4.2, the problem of learning the structure of a CTBN has been traditionally approached as an optimization problem. Therefore, this section focuses on adapting some common scores to the multi-dimensional classification problem with a Multi-CTBNC. Chapter 6 will further study other structure learning algorithms for Multi-CTBNCs, including novel constraint-based and hybrid solutions. The following sections will introduce three different scores for learning the structure of Multi-CTBNCs.

\(^1\)This is and will be omitted from the equations for the sake of simplicity.
5.3.2.1 Log-likelihood

The simplest approach to learn the structure of a Multi-CTBNC is to maximize the likelihood of the observed data given the model, \( P(D|M) \). The likelihood function of a Multi-CTBNC is obtained by incorporating the appearance and dependencies of class variables to the likelihood function of a CTBN [Nodelman et al., 2003]:

\[
L(M : D) = \prod_{y=1}^{d} \prod_{pa(C_y)} \prod_{c_j} (\hat{\beta}_{c_j}^{pa(C_y)})^{N_{c_j}^{pa(C_y)}} \prod_{f=1}^{m} \prod_{pa(X_f)} \prod_{x_j} (\hat{\varphi}_{x_j}^{pa(X_f)})^{M_{x_j}^{pa(X_f)}} \exp \left(-\hat{\varphi}_{x_j}^{pa(X_f)} T_{x_j}^{pa(X_f)} \right) \prod_{x_z \neq x_j} \prod_{pa(X_f)} (\hat{\varphi}_{x_j,x_z}^{pa(X_f)})^{M_{x_j,x_z}^{pa(X_f)}}. \tag{5.1b}
\]

The equation above shows that the likelihood for a Multi-CTBNC is decomposed into those for a BN (5.1a) and a CTBN (5.1b). This means that the learning of the class subgraph structure (BN) and the feature and bridge subgraphs (CTBN) of a Multi-CTBNC can be performed separately since they do not influence each other. In the case of a BN, the search space is limited to DAG, while the search space of a CTBN is simpler since the graph can be cyclic. As the bridge subgraph encodes the dependencies of the features on the class variables, it is also defined during the learning of the CTBN. As explained above, its structure has to be restricted to only allow arcs from class variables to features. This is the same restriction found in a CTBNC, but here it is extended to more than one class variable.

In practice, instead of using the likelihood function, a better approach is to maximize the log-likelihood (LL), which is monotonically related [Koller and Friedman, 2009]. The reason for this is that the LL is much easier to maximize and it helps to prevent underflows and overflows caused by the multiplication of small numbers and the exponential function:

\[
\text{LL}(M : D) = \sum_{y=1}^{d} \sum_{pa(C_y)} \sum_{c_j} N_{c_j}^{pa(C_y)} \log(\hat{\beta}_{c_j}^{pa(C_y)}) \tag{5.2a}
+ \sum_{f=1}^{m} \sum_{pa(X_f)} \sum_{x_j} \left[ M_{x_j}^{pa(X_f)} \log(\hat{\varphi}_{x_j}^{pa(X_f)}) \right. \\
- \hat{\varphi}_{x_j}^{pa(X_f)} T_{x_j}^{pa(X_f)} + \sum_{x_z \neq x_j} M_{x_j,x_z}^{pa(X_f)} \log(\hat{\varphi}_{x_j,x_z}^{pa(X_f)}) \right]. \tag{5.2b}
\]

This score tends to overfit the data by favoring densely connected networks. Therefore, a penalization factor over the complexity of the network, that is, the number of parameters, can be included. For example, the LL of a Multi-CTBNC with BIC penalization is:

\[
\text{BIC}(M : D) = \text{LL}(M : D) - \frac{\dim(G_c) + \dim(G_x \cup G_{cx})}{2} \log(N),
\]
where \( \text{dim}(G_C) = \sum_{y=1}^{d} r_{C_y}(|\Omega_{C_y}| - 1) \) and \( \text{dim}(G_X \cup G_{CX}) = \sum_{f=1}^{m} r_{X_f} z_{X_f} \) are the dimension (number of independent parameters) of the BN and CTBN, respectively, \( r_{C_y} \) is the number of possible instantiations of \( \text{Pa}(C_y) \) and \( z_{X_f} = (|\Omega_{X_f}| - 1)|\Omega_{X_f}| \) is the number of possible transitions from each state of \( X_f \) to any other.

### 5.3.2.2 Conditional log-likelihood

When facing a multi-dimensional classification problem, the LL can be defined as:

\[
\text{LL}(\mathbb{M} : \mathcal{D}) = \sum_{l=1}^{N} \log P(c_l | x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T}) + \sum_{l=1}^{N} \log P(x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T}).
\] (5.3)

Equation (5.3) divides the LL into a first term representing the model’s ability to classify a sequence and a second term describing the dependencies among features. If the number of features is large, the LL is dominated by the second term, which may negatively impact the performance of the classifier. Consequently, Friedman et al. [1997] proposed to focus only on the first term, which is the conditional log-likelihood (CLL), thereby following a discriminative approach. The CLL function was previously proposed by Codecasa and Stella [2014] for the learning of CTBNCs, and here we extend it for Multi-CTBNCs.

The idea is to specialize the LL of the CTBN (Equation (5.2b)) in the classification task since it defines the bridge subgraph and, therefore, the features that are relevant for classification. The LL of the BN (Equation (5.2a)) remains unchanged, as inter-class dependencies may be relevant for the classification. This results in the following score:

\[
\text{CLL}(\mathbb{M} : \mathcal{D}) = \sum_{l=1}^{N} \log P(c_l) + \sum_{l=1}^{N} \log P(c_l | x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T}),
\]

where

\[
\sum_{l=1}^{N} \log P(c_l | x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T}) = \sum_{l=1}^{N} \log \left( \frac{P(x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T} | c_l) P(c_l)}{P(x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T})} \right) = \sum_{l=1}^{N} \left[ \log P(x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T} | c_l) + \log P(c_l) - \log \left( \sum_{c'} P(x_{t_l}^{t_1}, \ldots, x_{t_l}^{t_T} | c') P(c') \right) \right].
\]

Most notable differences with respect to the CLL for a CTBNC are found in the class probability and denominator terms, which take into account the multiple class variables and
their dependencies. These are estimated, respectively, as follows:

$$\sum_{l=1}^{N} \log P(c_l) = \sum_{y=1}^{d} \sum_{c_y} N_{c_{j}}^{\text{pa}(c_y)} \log \left( \beta_{c_{j}}^{\text{pa}(c_y)} \right),$$

and

$$\sum_{l=1}^{N} \log \left( \sum_{c'} P(x_{t1}^{l}, \ldots, x_{tTl}^{l} | c') P(c') \right) = \log \left( \sum_{y=1}^{d} \beta_{c_{j}}^{\text{pa}(c_y)} \prod_{f=1}^{m} \prod_{x_{j}} \left( \beta_{x_{j}}^{\text{pa}(X_{f})} \right)^{M_{x_{j}}^{\text{pa}(X_{f})}} \exp \left( -\theta_{x_{j}}^{\text{pa}(X_{f})} T_{x_{j}}^{\text{pa}(X_{f})} \right) \prod_{x_{z} \neq x_{j}} \left( \beta_{x_{j}}^{\text{pa}(X_{f})} \right)^{M_{x_{j}}^{\text{pa}(X_{f})}} \right).$$

(5.4)

In the case of the likelihood term $P(x_{t1}^{l}, \ldots, x_{tTl}^{l} | c_l)$, the only difference is that parameters and sufficient statistics are defined based on the state of, potentially, multiple class variables that are parents of the features. Unfortunately, the denominator term cannot be further decomposed, as the logarithm is applied to a sum over all class configurations. Therefore, the log-sum-exp trick may be required in practice to prevent underflows and overflows [Murphy, 2012].

5.3.2.3 Bayesian Dirichlet equivalent score

This section presents a Bayesian score function for learning Multi-CTBNCs, the BDe score, which was first presented for CTBNs by Nodelman et al. [2003]. As discussed more thoroughly in Section 2.4.2.1, Bayesian scores are defined as:

$$BS(G : D) = \log P(D|G) + \log P(G),$$

which are derived from the logarithm of the Bayes’ rule to obtain the model structure with the highest probability given the data:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} \propto P(D|G)P(G).$$

They incorporate a prior probability over the model structures, $P(G)$, which is maximized together with the marginal likelihood of the data given the structure, $P(D|G)$, which, in contrast to the likelihood, does not consider a specific assignment of the parameters. It rather adds uncertainty about them by integrating over all their possible values for $G$:

$$P(D|G) = \int P(D|B, q, \Theta, G)P(B, q, \Theta|G)dBdqd\Theta.$$
This has the advantage over the previous scores of hopefully reducing overfitting by evaluating over all possible values of the parameters.

Making the common assumptions of global and local parameter independence [Heckerman et al., 1995], the LML of the data given the structure (i.e., log \( P(\mathcal{D}|\mathcal{G}) \)) can be decomposed into the sum of local LMLs for each type of parameter and variable:

\[
\log P(\mathcal{D}|\mathcal{G}) = \sum_{y=1}^{d} \text{LML} \left( B_{C_y}^{\text{Pa}(C_y)} : \mathcal{D} \right) + \sum_{j=1}^{m} \left[ \text{LML} \left( q_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) + \text{LML} \left( \Theta_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) \right].
\]

Given that the BDe score assumes a Dirichlet distribution over the parameter priors \( P(B|\mathcal{G}) \) and \( P(\Theta|\mathcal{G}) \) (with hyperparameters of Section 5.3.1), the LMLs for the parameters \( B_{C_y}^{\text{Pa}(C_y)} \) and \( \Theta_{X_f}^{\text{Pa}(X_f)} \) can be decomposed, respectively, as follows (for a derivation, see Cooper and Herskovits [1992] and Nodelman et al. [2003]):

\[
\text{LML} \left( B_{C_y}^{\text{Pa}(C_y)} : \mathcal{D} \right) = \log \left( \prod_{\text{pa}(C_y)} \frac{\Gamma \left( \sum_{c_j} \lambda_{c_j}^{\text{pa}(C_y)} \right)}{\Gamma \left( \sum_{c_j} \lambda_{c_j}^{\text{pa}(C_y)} + N_{c_j}^{\text{pa}(C_y)} \right)} \prod_{c_j} \frac{\Gamma \left( \lambda_{c_j}^{\text{pa}(C_y)} + \lambda_{c_j}^{\text{pa}(C_y)} \right)}{\Gamma \left( \lambda_{c_j}^{\text{pa}(C_y)} \right)} \right),
\]

and

\[
\text{LML} \left( \Theta_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) = \log \left( \prod_{\text{pa}(X_f)} \prod_{x_f} \frac{\Gamma \left( \alpha_{x_f}^{\text{pa}(X_f)} \right)}{\Gamma \left( \alpha_{x_f}^{\text{pa}(X_f)} + M_{x_f}^{\text{pa}(X_f)} \right)} \prod_{x_{x_f} \neq x_f} \frac{\Gamma \left( \alpha_{x_f,x_{x_f}}^{\text{pa}(X_f)} + M_{x_f,x_{x_f}}^{\text{pa}(X_f)} \right)}{\Gamma \left( \alpha_{x_f,x_{x_f}}^{\text{pa}(X_f)} \right)} \right),
\]

where \( \Gamma(\cdot) \) is the gamma function.

In the case of \( P(q|\mathcal{G}) \), a Gamma distribution is assumed (with hyperparameters of Section 5.3.1). Then, the LML for \( q_{X_f}^{\text{Pa}(X_f)} \) can be estimated with the following closed formula (for a derivation, see Nodelman et al. [2003]):

\[
\text{LML} \left( q_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) = \log \left( \prod_{\text{pa}(X_f)} \prod_{x_f} \frac{\Gamma \left( \alpha_{x_f}^{\text{pa}(X_f)} + M_{x_f}^{\text{pa}(X_f)} + 1 \right) \left( T_{x_f}^{\text{pa}(X_f)} \right)^{\alpha_{x_f}^{\text{pa}(X_f)} + M_{x_f}^{\text{pa}(X_f)} + 1}}{\Gamma \left( \alpha_{x_f}^{\text{pa}(X_f)} + 1 \right) \left( T_{x_f}^{\text{pa}(X_f)} + T_{x_f}^{\text{pa}(X_f)} \right)^{\alpha_{x_f}^{\text{pa}(X_f)} + M_{x_f}^{\text{pa}(X_f)} + 1}} \right).
\]

Finally, if a uniform prior over the structures \( P(\mathcal{G}) \) is considered, the BDe score simply maximizes the LML of the observed data given a Multi-CTBNC:

\[
\text{BDe}(\mathcal{G} : \mathcal{D}) = \sum_{y=1}^{d} \text{LML} \left( B_{C_y}^{\text{Pa}(C_y)} : \mathcal{D} \right) + \sum_{j=1}^{m} \left[ \text{LML} \left( q_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) + \text{LML} \left( \Theta_{X_f}^{\text{Pa}(X_f)} : \mathcal{D} \right) \right].
\]

In order to further penalize complex structures, a non-uniform structure prior \( P(\mathcal{G}) \), such as
a Binomial prior distribution [Andrews et al., 2018], can be considered.

### 5.3.2.4 Structure constraints

Due to the complexity of finding and learning all possible Multi-CTBNCs, some assumptions can be made about the structure, i.e., the hypothesis space can be reduced. Furthermore, these assumptions could help to learn models with better performance since they can prevent overfitting the data.

Similarly to MBCs, a variety of Multi-CTBNC families can be proposed considering different search spaces for the class and feature subgraphs. For example, they can be limited to be empty, a tree, a forest, a polytree, a max\(K\) (nodes have \(K\) parents at most), a DAG or, in the case of the feature subgraph, a directed graph (digraph). Following the notation proposed by Bielza et al. [2011], the different families are denoted as \{class subgraph structure\} - \{feature subgraph structure\} Multi-CTBNC. Some examples like the empty-empty Multi-CTBNC, tree-max\(K\) Multi-CTBNC and DAG-digraph Multi-CTBNC are shown in Figure 5.2.

A well-known subclass of classifiers is the naive Bayes family, which assumes conditional independence between features given the class variables. In the case of the Multi-CTBNC, a fully naive model is an empty-empty Multi-CTBNC (see Figure 5.2a) with a complete bridge subgraph. The benefit of this model over independent continuous-time naive Bayes classifiers (CTNBCs) [Stella and Amer, 2012] is that the number of parameters can be drastically reduced for certain datasets. Take as an example the parameters that would be learned from a dataset with eight ternary variables, three of them class variables. If three CTNBCs are built, the total number of independent parameters would be 276 since there would be 45 intensity matrices (three for each of the 15 feature nodes), each of them with six degrees of freedom, plus three CPTs with a degree of freedom of two. However, if there are exclusively five possible class configurations, a fully naive Multi-CTBNC would only require 156 parameters since the number of intensity matrices is reduced to 25 (five for each of the five feature nodes). Therefore, if the number of possible states of the class variables is large, but the number of class configurations is relatively small, a fully naive Multi-CTBNC would need a potentially much smaller number of parameters.
The \( \ast \)-max\( K \) Multi-CTBNC family is of special interest since, for fixed \( K \), the learning of a CTBN can be performed in polynomial time depending on the number of variables and dataset size [Nodelman et al., 2003]. This is possible since the parent set of each CTBN feature can be optimized individually without having to worry about avoiding cycles in the resulting structure. Unfortunately, the complexity of learning the bridge and feature subgraphs increases rapidly with the inclusion of more class variables. A \( \ast \)-max\( K \) Multi-CTBNC does not limit the number of class variables that can be parents of the features, in the same way as a \( \ast \)-max\( K \) MBC does, so the total number of parents could potentially be \( K + d \) for each feature. Evidently, this problem could be alleviated by imposing restrictions on the number of class variables [de Waal and van der Gaag, 2007].

Regarding the learning of the class subgraph, if a general DAG or even a max2 are considered, finding its optimal structure would be NP-hard due to the acyclicity constraint [Chickering et al., 1994]. If the number of class variables is relatively large, a tree structure may be a better option since polynomial-time learning algorithms can be applied [Chow and Liu, 1968; Heckerman et al., 1995; Friedman et al., 1997].

### 5.4 Classification

Given a sequence \( S_p = \{ x_{p1}^{t1}, \ldots, x_{p}^{tp} \} \), whose transitions are fully observed, classification is performed by choosing the class configuration that maximizes the posterior probability, i.e., the MAP estimate of

\[
P(c|S_p) = \frac{P(S_p|c)P(c)}{P(S_p)} \propto P(S_p|c)P(c) = \prod_{j=1}^{T_p-1} \prod_{f=1}^{m} P(x_{pf}^{t_j}|c)P(x_{pf}^{t_j+1}|x_{pf}^{t_j}, c) \prod_{y=1}^{d} P(c_y|pa(C_y)).
\]

The term \( P(x_{pf}^{t_j}|c) \) is the probability that feature \( X_f \) stays in state \( x_{pf}^{t_j} \) during the time interval of length \( \delta_j = t_{j+1} - t_j \) given the class configuration \( c \):

\[
P(x_{pf}^{t_j}|c) = \exp \left( -q_{x_{pf}^{t_j}}^{pa(X_f)} \delta_j \right),
\]

while \( P(x_{pf}^{t_j+1}|x_{pf}^{t_j}, c) \) is the probability that \( X_f \) transitions from state \( x_{pf}^{t_j} \) to state \( x_{pf}^{t_j+1} \) when it is known that a transition occurs and given the class configuration \( c \):

\[
P(x_{pf}^{t_j+1}|x_{pf}^{t_j}, c) = \begin{cases} 
1 - \exp \left( -q_{x_{pf}^{t_j}}^{pa(X_f)} \theta_{x_{pf}^{t_j}, x_{pf}^{t_j+1}} \right), & \text{if } x_{pf}^{t_j} \neq x_{pf}^{t_j+1} \\
1 & \text{otherwise},
\end{cases}
\]
where $\epsilon$ is a small positive number. Finally, $P(c_y|\text{pa}(C_y))$ represents the probability of class variable $C_y$ taking state $c_y$ given the parents’ state $\text{pa}(C_y)$:

$$P(c_y|\text{pa}(C_y)) = \beta_{c_y}^{\text{pa}(C_y)}.$$ 

Therefore, the predicted class configuration $c^*$ for a sequence $S_p$ is:

$$c^* = \arg\max_c \prod_{j=1}^{T_p-1} \prod_{f=1}^m \exp \left( -q_{x_fj}^{\text{pa}(X_f)} \delta_j \right) P \left( x_{pfj+1}^{t_fj} | x_{pfj}^{t_fj}, c \right) \prod_{y=1}^d \beta_{c_y}^{\text{pa}(C_y)}. \quad (5.6)$$

As it happened before, the logarithm of the argmax argument in Equation (5.6) is used instead for convenience.

If estimating the a posteriori probabilities of each class configuration is needed, the marginal likelihood of the sequence $S_p$ must be computed, i.e., the denominator term of the posterior probability (Equation (5.5)) cannot be ignored:

$$P(S_p) = \sum_{c'} P(S_p|c') P(c').$$

Unfortunately, as shown in Equation (5.4), this term cannot be further decomposed.

### 5.5 Experiments

This section will empirically compare the performance of the Multi-CTBNC against multiple independent CTBNCs for the multi-dimensional classification of synthetic and real-world time series data. The assessment will be performed with several performance measures estimated with a 5-fold cross-validation scheme to guarantee an honest and fair comparison. The learning of the model structures will be done by hill-climbing optimization, using the aforementioned scores and an empty initial structure. Parameters will be learned with Bayesian estimation. For the synthetic datasets, the hyperparameters were defined as $\lambda_{c_y}^{\text{pa}(C_y)} = 1$, $\alpha_{x_fj,x_z} = 1$ and $\tau_{x_fj} = 0.001$, which were found to provide interesting results. In the case of the real-world problem, Bayesian optimization with 100 iterations was used to tune the hyperparameters separately for the CTBNCs and the Multi-CTBNC on a validation dataset. The hyperparameters that led to the best global accuracy on the validation dataset were selected.

All experiments were run on an Intel Core i7-7700K at 4.20GHz with 32 GB of RAM using Windows 10 operating system. The Multi-CTBNC and CTBNC were developed in Java.
5.5.1 Datasets

The proposed model will be first evaluated over randomly generated synthetic data. Then, a real-world dataset from an industrial machine (from now on referred to as energy dataset) is used to prove the usefulness of the model in a real-world scenario.

5.5.1.1 Synthetic datasets

Fifty synthetic time series datasets were randomly generated from the structures of Figures 5.1a and 5.3a to 5.3i (five datasets per structure). The objective is to compare the performance of the Multi-CTBNC against independent CTBNCs when the data is obtained under diverse conditions. The class variables ($C_1$, $C_2$, $C_3$, $C_4$ and $C_5$) are assumed ternary, while the features ($X_1$, $X_2$, $X_3$, $X_4$ and $X_5$) can take four values, so they have three possible transitions from a certain state.

The datasets are sampled via probabilistic logic sampling [Henrion, 1988] from random CPTs and CIMs. In order to sample a sequence, a class configuration obtained from a BN and an initial observation are first defined for the sequence. From this, we can sample the time that each feature will remain in its current state. This time gives the order in which the transitions of the features will occur, obtaining a new observation for the sequence after the transition of only one of them. The new state to which the feature transitions is sampled by taking into account the current state of its parent features, as well as the classes of the parent
class variables. Once the transition is done, a duration time for the new state of the feature is sampled (as well as for its children), and the above process will be carried on until a sequence of a predefined duration is obtained. In our case, each dataset contains 10000 sequences that last a bit more than 10 time units (an average of 96 transitions).

Note that although synthetic datasets introduced in [Villa-Blanco et al., 2021] consist of categorical time series, their sampling was performed differently than in this chapter. Specifically, the transition times of the features were not re-sampled when their parents underwent a transition. This chapter addresses this issue so synthetic datasets are correctly sampled from Multi-CTBNs by following the forward sampling algorithm introduced by Fan and Shelton [2008]. Nonetheless, the main findings and conclusions of the original article remain intact. The adjustments only led to the BIC score being able to better reconstruct the underlying structures of the synthetic datasets.

5.5.1.2 Energy dataset

The energy dataset contains electrical measurements extracted in collaboration with a partner company from an industrial machine working in a real environment. These variables include intensity (I), voltage (V), active power (P), reactive power (Q) and apparent power (S), which were observed at a sampling frequency of 500Hz and discretized into 30 states with an equal width discretization. The industrial machine is composed of different three-phase motors and, therefore, a measure in each of the three phases (A, B and C) was obtained for each energy variable. In total, the dataset has 15 features. This dataset is an updated version of the one previously introduced in Villa-Blanco et al. [2021], for which the data collection process was improved, and data from two new motors were added.

The task to perform with this dataset is to classify the power consumption state (high, low or inactive) of eight motors (M1, M2, M3, M4, M5, M6, M7 and M8), which constitute eight class variables, by using the energy consumption of the machine as a whole. Note that the M7 and M8 motors can only take the low or inactive states. Figure 5.4 shows a diagram of the industrial machine and the studied motors. In this diagram, we can appreciate that the motors are related to each other, as M1 and M2, as well as M5 and M6, work together on very similar tasks. At the same time, M3 and M4 work synchronously with the motor pairs M1/M2 and M5/M6, respectively. Finally, the work of the motor groups M1/M2/M3 and M4/M5/M6 is conditioned by that of motors M7 and M8, respectively.

As in a real application we cannot obtain sequences where the motors have a unique state for all transitions, the extracted training sequences have a fixed duration of 0.3 seconds, determined based on the needs of the company, and the consumption state assigned for the motors is the one that occurs the most.
5.5.2 Results

5.5.2.1 Synthetic datasets

Average results obtained from 5-fold cross-validations over the five datasets generated from each structure are shown in Tables 5.1, 5.2 and 5.3. Each table includes the performance of the models when they are learned with different scores. The mean and standard deviation for each performance measure are reported, and the best results are written in bold.

Interesting results have been obtained in all experiments except with the CLL score (Table 5.2). This score did not lead to the expected models, at least in the performed experiments, since most learned structures have an empty bridge subgraph even when no penalization on their complexity is applied. The reason is the difference between the likelihood and denominator terms, as the latter tends to be larger (and therefore the score smaller) with the inclusion of dependencies in this subgraph. The CLL score then favors very sparse or even empty structures. Consequently, the following analysis will focus on the results achieved with the BIC and BDe scores.

As we can see in Tables 5.1 and 5.3, the Multi-CTBNC outperforms the independent CTBNCs in all synthetic experiments using both the BIC and BDe scores. As averages are susceptible to outliers, the Wilcoxon signed-rank test was applied over all the results of the 50 datasets to verify that there are statistical differences in the performance of the Multi-CTBNC and the CTBNCs. This results in p-values smaller than 0.001 for both scores in all performance measures. Therefore, the differences are highly significant, and the null hypothesis that both methods perform equally well is rejected in favor of the proposed model. The Multi-CTBNC has an important advantage in those contexts where class variables have a very weak or
non-existent relationship with the features that we were able to collect. This can be seen in some class variables from Figures 5.3a, 5.3b, 5.3e and 5.3f. The reason for this is that the Multi-CTBNC can model the dependencies of those class variables with others, while the CTBNCs may only rely on their prior probabilities.

The influence of the scores on the model performance was also studied. When comparing the Multi-CTBNC results in Tables 5.1 and 5.3, no significant differences were observed across most performance metrics. However, there were statistical differences in the mean accuracy and F1 score, suggesting slight improvements when employing the BIC score. Nevertheless, defining other hyperparameters for the prior distributions on the parameters, such as $\lambda_{C_j}^{pa} = 1$, $\alpha_{x_j|x_2}^{pa}(X_j) = 100$ and $\tau_{x_j}^{pa}(X_j) = 10$, led to significant improvements with the BDe score for all performance measures (except learning time). All these findings were confirmed using the Wilcoxon signed-rank test at a 0.05 significance level. This indicates that, if we have relevant expert knowledge about priors, we can potentially enhance the results obtained with the Bayesian scores. Table 5.4 presents a comparison of the Multi-CTBNCs learned using either the BIC or BDe score with these new hyperparameters, showing that, overall, models trained with the BDe score performed equally or better for all cases.

84
Chapter 5. Multi-dimensional continuous-time Bayesian network classifiers

Table 5.3: Estimated performance measures (mean ± std. deviation) over the synthetic datasets when learning the models with the BDe score.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro $F_1$ score</th>
<th>Global Brier score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CTBNCs</td>
<td>Multi-CTBNCs</td>
<td>CTBNCs</td>
<td>Multi-CTBNCs</td>
</tr>
<tr>
<td>Figure 5.1a</td>
<td>0.4399 ± 0.0860</td>
<td>0.6589 ± 0.1226</td>
<td>0.5818 ± 0.0265</td>
<td>0.9220 ± 0.0268</td>
</tr>
<tr>
<td>Figure 5.2a</td>
<td>0.3255 ± 0.0665</td>
<td>0.3347 ± 0.0581</td>
<td>0.7924 ± 0.0254</td>
<td>0.8291 ± 0.0195</td>
</tr>
<tr>
<td>Figure 5.3a</td>
<td>0.4340 ± 0.0635</td>
<td>0.5033 ± 0.0445</td>
<td>0.8494 ± 0.0200</td>
<td>0.8830 ± 0.0130</td>
</tr>
<tr>
<td>Figure 5.4a</td>
<td>0.5075 ± 0.0411</td>
<td>0.8855 ± 0.0130</td>
<td>0.8991 ± 0.0112</td>
<td>0.9679 ± 0.0039</td>
</tr>
<tr>
<td>Figure 5.5a</td>
<td>0.5832 ± 0.0477</td>
<td>0.9161 ± 0.0157</td>
<td>0.8804 ± 0.0136</td>
<td>0.9810 ± 0.0034</td>
</tr>
<tr>
<td>Figure 5.6a</td>
<td>0.4866 ± 0.0837</td>
<td>0.6184 ± 0.0492</td>
<td>0.8603 ± 0.0203</td>
<td>0.9162 ± 0.0103</td>
</tr>
<tr>
<td>Figure 5.7a</td>
<td>0.2714 ± 0.0912</td>
<td>0.3997 ± 0.0720</td>
<td>0.7745 ± 0.0467</td>
<td>0.8492 ± 0.0239</td>
</tr>
<tr>
<td>Figure 5.8a</td>
<td>0.5504 ± 0.0677</td>
<td>0.8455 ± 0.0311</td>
<td>0.8877 ± 0.0195</td>
<td>0.9566 ± 0.0111</td>
</tr>
<tr>
<td>Figure 5.9a</td>
<td>0.5766 ± 0.0667</td>
<td>0.9070 ± 0.0126</td>
<td>0.9872 ± 0.0175</td>
<td>0.9779 ± 0.0031</td>
</tr>
<tr>
<td>Figure 5.10a</td>
<td>0.6343 ± 0.0671</td>
<td>0.8836 ± 0.0242</td>
<td>0.9102 ± 0.0169</td>
<td>0.9685 ± 0.0071</td>
</tr>
</tbody>
</table>

Results obtained with the datasets from Figure 5.3h are of special interest since there are no dependencies among the class variables and a more even result between the independent classifiers and the Multi-CTBNC was expected. However, the Multi-CTBNC significantly improves all performance measures while correctly not defining any association between class variables. The fact of knowing the simultaneous dependencies of the features on different class variables allows the Multi-CTBNC to learn more accurate models. For this same reason, even an empty-digraph Multi-CTBNC may obtain better results than independent CTBNCs. Table 5.5 shows the performance of an empty-digraph Multi-CTBNC on the synthetic datasets, highlighting in bold the results that improve those of the CTBNCs (Table 5.3). Overall, significant improvements were found in all performance measures when optimizing both the BIC and BDe scores to learn the structures. These improvements were validated with the Wilcoxon signed-rank test at a significance level of 0.001. This justifies the use of Multi-CTBNCs even when it is clear that there are no dependencies between the class variables.

Finally, Table 5.6 shows the learning times for the Multi-CTBNC and the CTBNCs when they are built with the BIC and BDe scores. The learning of a Multi-CTBNC is significantly faster than multiple CTBNCs when using the proposed datasets. Additionally, no significant differences were detected between the learning times achieved with the BIC and BDe scores.
Table 5.5: Estimated performance measures (mean ± std. deviation) over the synthetic datasets when learning an empty-digraph Multi-CTBNC with the BDe score.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F₁ score</th>
<th>Global Brier score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5.1a</td>
<td>0.5771 ± 0.1579</td>
<td>0.9027 ± 0.0352</td>
<td>0.8719 ± 0.0642</td>
<td>0.5326 ± 0.1691</td>
</tr>
<tr>
<td>Figure 5.3a</td>
<td>0.1973 ± 0.0494</td>
<td>0.7716 ± 0.0193</td>
<td>0.6777 ± 0.0065</td>
<td>0.8736 ± 0.0155</td>
</tr>
<tr>
<td>Figure 5.3b</td>
<td>0.3160 ± 0.0432</td>
<td>0.8442 ± 0.0157</td>
<td>0.7886 ± 0.0185</td>
<td>0.7307 ± 0.0233</td>
</tr>
<tr>
<td>Figure 5.3c</td>
<td>0.8563 ± 0.0124</td>
<td>0.9599 ± 0.0036</td>
<td>0.9456 ± 0.0155</td>
<td>0.2102 ± 0.0164</td>
</tr>
<tr>
<td>Figure 5.3d</td>
<td>0.8867 ± 0.0081</td>
<td>0.9742 ± 0.0016</td>
<td>0.9657 ± 0.0030</td>
<td>0.1668 ± 0.0099</td>
</tr>
<tr>
<td>Figure 5.3e</td>
<td>0.4643 ± 0.0829</td>
<td>0.8798 ± 0.0209</td>
<td>0.8038 ± 0.0144</td>
<td>0.6459 ± 0.0496</td>
</tr>
<tr>
<td>Figure 5.3f</td>
<td>0.2975 ± 0.0860</td>
<td>0.8116 ± 0.0318</td>
<td>0.6826 ± 0.0045</td>
<td>0.8225 ± 0.0505</td>
</tr>
<tr>
<td>Figure 5.3g</td>
<td>0.7976 ± 0.0404</td>
<td>0.9428 ± 0.0139</td>
<td>0.9245 ± 0.0104</td>
<td>0.2870 ± 0.0545</td>
</tr>
<tr>
<td>Figure 5.3h</td>
<td>0.9070 ± 0.0126</td>
<td>0.9779 ± 0.0031</td>
<td>0.9605 ± 0.0100</td>
<td>0.1374 ± 0.0181</td>
</tr>
<tr>
<td>Figure 5.3i</td>
<td>0.8452 ± 0.0263</td>
<td>0.9574 ± 0.0075</td>
<td>0.9474 ± 0.0042</td>
<td>0.2239 ± 0.0364</td>
</tr>
</tbody>
</table>

Table 5.6: Model learning times in seconds (mean ± std. deviation) on all synthetic datasets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4822 ± 1.1002</td>
<td>9.3767 ± 1.0333</td>
<td>8.9041 ± 1.3125</td>
<td><strong>8.6955 ± 1.2550</strong></td>
</tr>
</tbody>
</table>

This lack of difference, as well as the advantage of Multi-CTBNC over multiple CTBNCs, was assessed with the Wilcoxon signed-rank test and a significance level of 0.05. Note that the learning of both a CTBNC and a Multi-CTBNC, as well as the different CTBNCs, is performed in parallel.

5.5.2.2 Energy dataset

Due to the high cardinality of the energy dataset features, their nodes are limited to having at most one feature as a parent. Therefore, this section compares the differences in performance between independent max1 CTBNCs and a DAG-max1 Multi-CTBNC.

Fifteen datasets with different sequence order were extracted from the original energy dataset and a 5-fold cross-validation was performed on each of them. The average results of the cross-validations are shown in Table 5.7, which indicate that the DAG-max1 Multi-CTBNC outperforms the independent classifiers in all performance measures. In this study, the most interesting models were obtained with the LL penalized with BIC. In contrast to the previous findings reported in Villa-Blanco et al. [2021], where BDe identified empty bridge subgraphs,
Table 5.7: Estimated performance measures (mean ± std. deviation) over the energy dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F1 score</th>
<th>Micro F1 score</th>
<th>Global Brier score</th>
</tr>
</thead>
<tbody>
<tr>
<td>max1 CTBNCs [BIC]</td>
<td>0.5738 ± 0.0019</td>
<td>0.8611 ± 0.0011</td>
<td>0.7597 ± 0.0025</td>
<td>0.9039 ± 0.0007</td>
<td>0.8524 ± 0.0039</td>
</tr>
<tr>
<td>max1 CTBNCs [BDe]</td>
<td>0.5738 ± 0.0019</td>
<td>0.8611 ± 0.0011</td>
<td>0.7597 ± 0.0025</td>
<td>0.9039 ± 0.0007</td>
<td>0.8524 ± 0.0039</td>
</tr>
<tr>
<td>DAG-max1 Multi-CTBNC [BIC]</td>
<td><strong>0.7349 ± 0.0023</strong></td>
<td><strong>0.8963 ± 0.0014</strong></td>
<td><strong>0.8677 ± 0.0024</strong></td>
<td><strong>0.9276 ± 0.0009</strong></td>
<td><strong>0.5180 ± 0.0051</strong></td>
</tr>
<tr>
<td>DAG-max1 Multi-CTBNC [BDe]</td>
<td>0.7312 ± 0.0029</td>
<td>0.8879 ± 0.0020</td>
<td>0.8569 ± 0.0030</td>
<td>0.9229 ± 0.0014</td>
<td>0.5375 ± 0.0058</td>
</tr>
</tbody>
</table>

we have employed Bayesian optimization on this occasion to determine the hyperparameters. This was done by optimizing the global accuracy on a validation dataset. As a result, we have observed a significant accuracy improvement using both BIC and BDe when learning Multi-CTBNCs (note that results from Table 5.7 include the two new class variables M7 and M8). Wilcoxon signed-rank tests show significant differences in the commented comparisons for all performance measures with a significance level of 0.001.

Figure 5.5 shows an example of a Multi-CTBNC structure learned on the energy dataset with the BIC score. Unsurprisingly, most class variables have similar features as children since they all represent similar motors. The main differences between the class variables lie in their influence on the total consumption of the machine and their relationships with each other. Regarding the latter, the results are very accurate since most dependencies match the description of Section 5.5.1.2. In this problem, the differential factor that makes the Multi-CTBNC perform better than independent CTBNCs is its ability to model these dependencies. Expected relationships among features are also detected, such as between voltage and intensity, power measurements or the three different phases of the electrical system. The reason for this last kind of relationship is that the current and voltage are balanced, with the same magnitudes and 120 degrees displacement, between each phase. Nonetheless, this balance could be affected in terms of magnitude and/or phase angular displacement by the type of the electrical consumers (e.g., single phase elements) or anomalous behaviors (e.g., failure of electronic components, such as phase diode, fuse or isolation). The advantage of using a graphical model is that these machine malfunctions could be easily detected by analyzing the learned structures.

5.6 Conclusions and future work

This chapter introduces the Multi-CTBNC, a new PGM that is able to capture the dependencies of multivariate temporal sequences and perform multi-dimensional classification over them. The learning of its parameters and structure from data is discussed, and its usefulness is justified by solving a novel real-world engineering problem.

Score-based learning algorithms for Multi-CTBNCs were studied, as this is the common
Figure 5.5: Structure of a DAG-max1 Multi-CTBNC learned from the energy dataset with the BIC score.

approach for learning CTBNs. However, a constraint-based algorithm for CTBNs has been recently proposed by Bregoli et al. [2021], where it was experimentally shown to perform better in certain scenarios. Therefore, its extension to learn classification models and, more specifically, the use of its fundamentals to develop a constraint-based algorithm for Multi-CTBNCs are possible lines of research, which will be covered in Chapter 6 of this dissertation.

The adaptation of Multi-CTBNCs to handle other types of predictive problems where data and sources have different characteristics is currently being studied. First, an important limitation of the presented model is that the features are assumed to be discrete, while the problem of discretizing time series still requires more attention [Villa and Stella, 2016]. Nonetheless, we would like to study a possible adaptation of Multi-CTBNCs (and, therefore, CTBNs) for continuous variables, thus avoiding this discretization. Second, this work assumes no missing data or hidden variables, i.e., complete data assumption, which in practice is not always the case. Third, the classifier can easily use other distributions to model the waiting times of variables in a certain state. This is the case of hypoexponential distributions, previously used with CTBNs [Liu et al., 2018c], which are more appropriate than the exponential for certain applications.

The study of structure constraints for CTBNs has not received a lot of attention and, to the best of our knowledge, only the learning of naive Bayes and structures where nodes have a maximum number of parents were considered. It could be interesting to analyze the possible benefits in terms of computational complexity and performance of learning CTBNs with other constraints, such as being trees or DAGs.
Finally, although a really limited number of class variables is commonly assumed, multiple applications stand out for having a large number of them. For example, the industrial problem introduced in this chapter could require classifying many more motors. Therefore, we plan to study the inclusion of FSS approaches to the Multi-CTBNC to make classification under these conditions less prohibitive. This would allow the inclusion of restrictions over the complexity of the bridge subgraph. With a similar objective, it would be interesting to design class-bridge decomposable [Bielza et al., 2011] Multi-CTBNCs to reduce the computational cost of estimating the most probable class configuration of a sequence.
Chapter 6

Constraint-based and hybrid structure learning of multi-dimensional continuous-time Bayesian network classifiers

6.1 Introduction

Time series data are characterized by their large size and high dimensionality [Fu, 2011], which are expected to keep growing as they become easier to collect and at higher granularity. Therefore, designing algorithms capable of modeling these data more accurately and efficiently is becoming more necessary.

In this chapter, we focus on learning the structure of CTBNs from data and, more specifically, structures of CTBNs that can be applied to the multi-dimensional classification of time series, i.e., Multi-CTBNCs. CTBNCs have been successfully applied for one-dimensional problems such as post-stroke rehabilitation [Codecasa and Stella, 2014] or multi-dimensional problems like detecting energy consumption states [Villa-Blanco et al., 2021]. Nevertheless, the study of structure learning algorithms for these classifiers remains largely unexplored in the literature.

Learning the structure of CTBNs directly from data has traditionally been performed using score-based structure learning algorithms. Only recently has a constraint-based method been proposed, proving to be more suitable under specific settings, as in modeling systems with variables having more than two states [Bregoli et al., 2021]. As a result, studying diverse structure learning algorithms is essential to learn the most appropriate models according to data characteristics and task-related priorities, such as learning speed or accuracy. This
chapter proposes alternative algorithms for learning Multi-CTBNCs, introducing, for the first time, constraint-based and hybrid algorithms for these models. Nevertheless, these contributions also apply to the simpler one-dimensional classification problem for which only score-based solutions exist in the literature. More specifically, the aforementioned constraint-based structure learning algorithm by Bregoli et al. [2021] is first adapted to the supervised classification setting. Then, a novel algorithm of this kind, specifically tailored for the multi-dimensional classification problem, is presented to improve the learning times for the induction of multi-dimensional classifiers. Finally, a hybrid algorithm is introduced, attempting to combine the strengths of the score- and constraint-based approaches. Experiments with synthetic and real-world data are performed not only to validate the capabilities of the proposed algorithms but also to conduct a comparative study of available structure learning solutions.

The main contributions of this chapter are the following:

- The development of the first constraint-based structure learning algorithm for CTBNCs, conceived to learn Multi-CTBNCs.
- The introduction of the first hybrid structure learning algorithm for CTBNCs and Multi-CTBNCs.
- A comprehensive comparative study to evaluate the strengths and weaknesses of the state-of-the-art algorithms and those proposed in this work.
- The presentation of a multi-dimensional classification problem that uses publicly available data to demonstrate the usefulness of the proposed constraint-based algorithm in a real-world setting.
- The development and integration of the presented algorithms into a software tool introduced in Villa-Blanco et al. [2021].

This chapter includes the content of Villa-Blanco et al. [2022] and Villa-Blanco et al. [2023b]. All developed software and studied datasets are freely accessible at https://github.com/carlvilla/Multi-CTBNCs.

Chapter outline

The remainder of this chapter is as follows. Section 6.2 describes the state-of-the-art constraint-based structure learning algorithm available to date for CTBNs and discusses how it could be adapted for learning Multi-CTBNCs. Section 6.3 and 6.4 introduce novel constraint-based and hybrid algorithms for Multi-CTBNCs, respectively. Section 6.5 presents experiments and discusses the results of multiple structure learning algorithms. Section 6.6 concludes the
chapter and discusses future research lines.

**Algorithm 6.1: CTPC(\mathcal{X}, \mathcal{V})^a**

1: for each feature $X_i \in \mathcal{X}$ do
2:    Set $\mathcal{U} = \{ V_j \in \mathcal{V} \mid V_j \rightarrow X_i \}$
3:    for increasing values $s = 0, 1, \ldots, |\mathcal{U}| - 1$ do
4:        for each variable $V_j \in \mathcal{U}$ and subset $S_{X_i,V_j} \subseteq \mathcal{U} \setminus \{V_j\}$, where $|S_{X_i,V_j}| = s$ do
5:            if $X_i \perp \perp V_j \mid S_{X_i,V_j}$ then
6:                Remove arc $V_j \rightarrow X_i$ from $G$ and delete $V_j$ from $\mathcal{U}$
7:            end if
8:        end for
9:    end for
10: end for
11: return directed graph $G$

^aAdaptation of the algorithm by Bregoli et al. [2021]. The first step of the original algorithm, which forms a complete directed graph, was omitted for convenience (see Algorithm 2.3).

### 6.2 Constraint-based structure learning of CTBNs

Structure learning of CTBNs has been traditionally addressed as an optimization problem [Nodelman et al., 2003; Codecasa and Stella, 2014; Villa and Stella, 2016], where a structure is selected from a candidate space by maximizing a score (score-based algorithms). The motivation may be that standard score-based algorithms can be straightforwardly applied to learn CTBNs. As CTBNs have no acyclicity constraints, the parent set of each node can even be conveniently defined in parallel without concern for reporting erroneous structures. Only recently has a constraint-based algorithm been proposed, which infers their structures by performing conditional independence tests. The CTPC algorithm, introduced by Bregoli et al. [2021], is the first proposal of this kind, which adapts the classical PC algorithm, described in Algorithm 2.2, to CTBNs. As CIMs describe temporal dynamics, classical statistical tests cannot be applied. Thus, CTPC introduces a novel definition of conditional independence in CTBNs.

The CTPC algorithm can be easily adapted to learn the bridge and feature subgraphs of Multi-CTBNCs since only the parent sets of nodes with CIMs are learned. The algorithm only has to meet the topology constraints of the model. Algorithm 6.1 shows the pseudocode of the CTPC algorithm used in this work, where $\mathcal{V} = \mathcal{X} \cup \mathcal{C}$. As for learning the class subgraph, traditional constraint-based solutions for discrete BNs can be used. We call this adaption “naive” since it tests all possible dependencies between features, even those irrelevant to the classification task. This considerably increases the learning time, a problem aggravated in
6.3 Markov blanket-based continuous-time PC algorithm

This section introduces a novel constraint-based structure learning algorithm called Markov blanket-based continuous-time PC (MB-CTPC), specially designed to learn Multi-CTBNCs. This algorithm aims to evaluate only those dependencies relevant to the Markov blanket\(^1\) of the class variables, which provides sufficient information to infer these variables. To this end, MB-CTPC defines a set of rules to ignore irrelevant dependencies based on ancestor class variables of features.

Algorithm 6.2 describes the pseudocode of MB-CTPC. Step 1 finds the probabilistic relationships between class variables using a traditional constraint-based algorithm, such as PC. Step 2 forms the complete bridge and feature subgraphs of \(G\). Then, Step 3 defines the descendants of the class variables via conditional independence tests between feature and class variables without considering other features in the separating set. Thus, a dependency of a feature on a class variable might exist because it is its child or there is a flow of information through intermediate feature nodes. This step defines a preliminary bridge subgraph that provides valuable information to reduce the statistical tests for the feature subgraph. If a pair of features does not share the same parent class variables, information is not flowing in at least one direction, and at least one dependency can be removed. Steps 4 to 14 use three rules to reduce the number of conditional independence tests:

- **Rule 1 (Steps 6 and 7).** Given adjacent features \(X_i\) and \(X_j\), arc \(X_i \rightarrow X_j\) is removed iff \(\text{Pa}_C(X_j) = \emptyset\), where \(\text{Pa}_C(X_j)\) denotes the parent class variables of \(X_j\).

- **Rule 2 (Steps 8 and 9).** Given adjacent features \(X_i\) and \(X_j\), arcs \(X_i \rightarrow X_j\) and \(X_j \rightarrow X_i\) are removed iff \(\text{Pa}_C(X_i) \cap \text{Pa}_C(X_j) = \emptyset\), \(\text{Pa}_C(X_i) \neq \emptyset\) and \(\text{Pa}_C(X_j) \neq \emptyset\).

- **Rule 3 (Steps 10 and 11).** Given adjacent features \(X_i\) and \(X_j\), arc \(X_i \rightarrow X_j\) is removed iff \(\text{Pa}_C(X_i) \setminus \text{Pa}_C(X_j) \neq \emptyset\), \(\text{Pa}_C(X_i) \neq \emptyset\) and \(\text{Pa}_C(X_j) \neq \emptyset\).

Finally, Step 15 further identifies conditional independence relationships using the CTPC algorithm. However, its execution time is significantly reduced, as it is limited to evaluating only those arcs the previous rules could not discard.

**Example 1.** Given some data sampled from the Multi-CTBNC of Figure 6.1a, then Figures 6.1b to 6.1h show the steps of MB-CTPC to learn the Markov blankets of the class variables. First, Figure 6.1b represents the learning of the class subgraph (Step 1). Then, the complete

---

\(^1\)Given a Multi-CTBNC, the Markov blanket of a node consists of its parents, children and spouses.
Algorithm 6.2: MB-CTPC(\(\mathcal{X}, \mathcal{C}\))

1: \(\mathcal{G} \leftarrow \text{PC}(\mathcal{C})\)
2: Build the complete bridge and feature subgraphs of \(\mathcal{G}\) on node set \(\mathcal{X} \cup \mathcal{C}\)
3: \(\mathcal{G} \leftarrow \text{CTPC}(\mathcal{X}, \mathcal{C})\)
4: for each feature \(X_i \in \mathcal{X}\) do
5:   for each feature \(X_j \in \mathcal{X}\), where \(X_j \neq X_i\) do
6:     if \(\text{Pa}_C(X_i) = \emptyset\) then
7:       Remove arc \(X_j \rightarrow X_i\) from \(\mathcal{G}\)
8:     else if \(\text{Pa}_C(X_i) \cap \text{Pa}_C(X_j) = \emptyset\) AND \(\text{Pa}_C(X_j) \neq \emptyset\) then
9:       Remove arcs \(X_i \rightarrow X_j\) and \(X_j \rightarrow X_i\) from \(\mathcal{G}\)
10:     else if \(\text{Pa}_C(X_i) \setminus \text{Pa}_C(X_j) \neq \emptyset\) AND \(\text{Pa}_C(X_j) \neq \emptyset\) then
11:       Remove arc \(X_i \rightarrow X_j\) from \(\mathcal{G}\)
12:   end if
13: end for
14: end for
15: \(\mathcal{G} \leftarrow \text{CTPC}(\mathcal{X}, \mathcal{C} \cup \mathcal{X})\)
16: return directed graph \(\mathcal{G}\)

bridge and feature subgraphs are built in Figure 6.1c (Step 2). Afterward, conditional independence tests, which consider only class variables in the separating sets, find the descendant features of the class variables (Step 3). For example, Figure 6.1d shows an arc from \(C_4\) to \(X_6\), as a direct path exists through \(X_7\) in the original structure. Subsequently, Rule 1 removes incoming arcs of \(X_2\) and \(X_5\) in Figure 6.1e since they have no dependencies on class variables. Then, Rule 2 discards dependencies between features not sharing parent class variables. That is the case for pairs like \(X_3\) and \(X_6\) or \(X_1\) and \(X_7\) in Figure 6.1f. In Figure 6.1g, Rule 3 removes a dependency from one feature to another if the former has parent class variables that the latter does not, such as arcs \((X_1, X_3)\) and \((X_6, X_7)\). These rules discard 27 out of 42 arcs (64%) of the feature subgraph without performing any conditional independence test. Finally, tests are performed on the remaining arcs in Figure 6.1h. The resulting structure of Figure 6.1i shows that arcs from the original structure providing no information about the Markov blankets of class variables (those between \(X_2\) and \(X_5\)) are discarded.

6.4 Hybrid structure learning algorithm

Score- and constraint-based algorithms have their own strengths and weaknesses, making them more or less useful in different classification contexts. Therefore, as already done for other PGMs [Acid and De Campos, 2001; Tsamardinos et al., 2006; Trabelsi et al., 2013; Liu et al., 2017], we study hybrid algorithms for Multi-CTBNCs. These methods aim to combine the
Figure 6.1: Steps of the MB-CTPC algorithm. Dash lines represent removed arcs.
advantages of both approaches, such as faster learning speed of constraint-based algorithms and higher accuracy of score-based solutions (see Section 6.5.1.2). This section presents the first hybrid algorithm to learn both one-dimensional and multi-dimensional CTBNCs. The algorithm is divided into a restriction phase where conditional independence tests find an initial structure and a maximization phase that refines it. Two variants are used depending on the subgraph:

- **Class subgraph.** The PC algorithm is used to reconstruct the skeleton of the class subgraph. Then, a hill climbing procedure searches for a solution, starting from the empty subgraph but only allowing arcs included in the skeleton.

- **Bridge and feature subgraphs.** The CTPC algorithm defines an initial structure during the restriction phase, which serves as the initial solution for a hill climbing algorithm in the maximization phase. Two aspects balance the influence of these algorithms. First, a maximum separating set size is established for conditional independence tests. Second, hill climbing only removes or adds arcs that the restriction phase has not discarded. The maximum size of the separating set dictates each algorithm’s influence and range of action.

Although the scope of this chapter is learning classifiers, this hybrid algorithm can also be applied to learn CTBNs, making it their first hybrid proposal to the best of our knowledge.

### 6.5 Experiments

This section empirically compares the performance of Multi-CTBNCs learned with five different structure learning algorithms in a variety of contexts, including synthetic datasets and a real-world problem. Score-based algorithms are represented by hill climbing and tabu search (tabu list of size 5), whose scores (BIC or BDe; Villa-Blanco et al. [2021]) are indicated in square brackets, e.g., hill climbing [BDe]. Regarding constraint-based algorithms, (the naive) CTPC and the presented MB-CTPC are evaluated. Significance levels of 0.05 (class subgraph) and $10^{-5}$ (bridge and feature subgraphs) are used to test conditional independence. Finally, the proposed hybrid algorithm is studied using the hill climbing [BIC] algorithm in the maximization phase and separating sets of zero (hybrid $|S_{V_iV_j}| = 0$) and one (hybrid $|S_{V_iV_j}| = 1$) maximum size.

In order to guarantee an honest and fair comparison, the learned models are evaluated using several performance measures and a 5-fold cross-validation scheme. The measures under consideration are global and mean accuracy, global Brier score, macro and micro $F_1$ score,

\(^2\)The large number of hypothesis tests used to compare parameters from exponential distributions requires a very small significance level to avoid inaccurately dense structures.
Table 6.1: Parameters used to generate the datasets for the experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Studied values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of features</td>
<td>5, 10, 20</td>
</tr>
<tr>
<td>Cardinality of features</td>
<td>2, 3, 4, 8</td>
</tr>
<tr>
<td>Number of class variables</td>
<td>4</td>
</tr>
<tr>
<td>Cardinality of class variables</td>
<td>2, 3</td>
</tr>
<tr>
<td>Density of class subgraph</td>
<td>30%</td>
</tr>
<tr>
<td>Density of bridge subgraph</td>
<td>5%, 10%, 20%</td>
</tr>
<tr>
<td>Density of feature subgraph</td>
<td>5%, 10%, 20%</td>
</tr>
</tbody>
</table>

and learning and classification time. For the comparisons, the Wilcoxon signed-rank test is used with a significance level of 0.05 to verify that the results are statistically significant. Regarding parameter learning, Bayesian estimation is used with the following hyperparameters for their prior distributions (see Section 5.3.1): \( \lambda_{C_{y}}^{pa} = 1 \) and \( \alpha_{X_{f}}^{pa} = 1 \) for the Dirichlet prior distribution, and \( \tau_{x_{j}}^{pa} = \tau_{x_{j}}^{pa} (X_{f}) = 0.001 \) (synthetic experiments) or \( \tau_{x_{j}}^{pa} = 1 \) (real-world experiment) for the gamma prior distribution.

The experiments were run on a 4.20GHz Intel Core i7-7700K with 32 GB of RAM using Windows 10. The structure learning algorithms were developed in Java.

6.5.1 Experimental results on synthetic data

Due to the limited progress in the literature on the classification problem under study, publicly available real-world datasets are scarce. Thus, this section uses synthetic datasets drawn from various contexts to effectively evaluate structure learning algorithms’ advantages and disadvantages.

Synthetic datasets are sampled via probabilistic logic sampling [Henrion, 1988; Fan and Shelton, 2008] from Multi-CTBNs whose structures and parameters are randomly generated. Five datasets have been sampled from each combination of parameters’ values (216 combinations) shown in Table 6.1 (1080 datasets), each with 5000 sequences that last 20 time units. The generated structures have at least one arc in the bridge subgraph, and features are restricted to a maximum of three children to avoid memory problems.

Table 6.2 presents the results for all datasets of specific comparisons of structure learning algorithms that we found most relevant, whereas Table 6.3 includes the average results for each algorithm. In the following sections, we discuss some conclusions drawn from these tables and perform a more exhaustive analysis.

98
Table 6.2: Percentage of datasets in which structure learning algorithms achieve better, worse or identical results.

<table>
<thead>
<tr>
<th>Performance measure</th>
<th>Results of ... are</th>
<th>Better</th>
<th>Worse</th>
<th>Same</th>
<th>Same or Better</th>
<th>Than</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>10.56%</td>
<td>5.93%</td>
<td>83.52%</td>
<td>94.07%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>1.20%</td>
<td>0.83%</td>
<td>97.96%</td>
<td>99.17%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>17.69%</td>
<td>42.31%</td>
<td>40.00%</td>
<td>57.69%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>Global Brier score</td>
<td>1.76%</td>
<td>39.26%</td>
<td>58.98%</td>
<td>60.74%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>1.76%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>5.56%</td>
<td>34.54%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>22.96%</td>
<td>20.56%</td>
</tr>
<tr>
<td></td>
<td>Mean accuracy</td>
<td>11.39%</td>
<td>6.48%</td>
<td>82.13%</td>
<td>93.52%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>1.20%</td>
<td>1.20%</td>
<td>97.59%</td>
<td>98.80%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>21.39%</td>
<td>39.44%</td>
<td>39.17%</td>
<td>60.56%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>Hil climbing [BIC]</td>
<td>1.67%</td>
<td>40.28%</td>
<td>58.06%</td>
<td>59.72%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>1.85%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>9.35%</td>
<td>31.85%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>22.95%</td>
<td>21.76%</td>
</tr>
<tr>
<td></td>
<td>Macro F score</td>
<td>12.13%</td>
<td>7.59%</td>
<td>80.28%</td>
<td>92.41%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>1.30%</td>
<td>1.30%</td>
<td>97.41%</td>
<td>98.70%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>22.41%</td>
<td>41.11%</td>
<td>36.48%</td>
<td>58.89%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>MB-CTPC</td>
<td>2.31%</td>
<td>40.28%</td>
<td>57.41%</td>
<td>59.72%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>1.85%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>9.35%</td>
<td>31.24%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>23.33%</td>
<td>22.78%</td>
</tr>
<tr>
<td></td>
<td>Global Brier score</td>
<td>24.81%</td>
<td>18.89%</td>
<td>56.30%</td>
<td>81.11%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>5.19%</td>
<td>5.28%</td>
<td>89.54%</td>
<td>94.72%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>19.91%</td>
<td>59.81%</td>
<td>20.28%</td>
<td>40.19%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>MB-CTPC</td>
<td>11.85%</td>
<td>53.52%</td>
<td>34.63%</td>
<td>46.85%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>3.52%</td>
<td>2.59%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>7.41%</td>
<td>49.26%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>34.54%</td>
<td>23.80%</td>
</tr>
<tr>
<td></td>
<td>Learning time (s)</td>
<td>37.04%</td>
<td>62.06%</td>
<td>0.00%</td>
<td>37.04%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>37.96%</td>
<td>62.04%</td>
<td>0.00%</td>
<td>37.96%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>99.54%</td>
<td>0.46%</td>
<td>0.00%</td>
<td>99.54%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>MB-CTPC</td>
<td>98.15%</td>
<td>1.85%</td>
<td>0.00%</td>
<td>98.15%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>36.11%</td>
<td>63.89%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>90.28%</td>
<td>9.72%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>4.63%</td>
<td>95.37%</td>
</tr>
<tr>
<td></td>
<td>Classification time (s)</td>
<td>31.33%</td>
<td>68.67%</td>
<td>0.00%</td>
<td>31.33%</td>
<td>Hill climbing [BDe]</td>
</tr>
<tr>
<td></td>
<td>Hill climbing [BIC]</td>
<td>34.41%</td>
<td>65.59%</td>
<td>0.00%</td>
<td>34.41%</td>
<td>Tabu search [BIC]</td>
</tr>
<tr>
<td></td>
<td>CTPC</td>
<td>58.64%</td>
<td>41.36%</td>
<td>0.00%</td>
<td>58.64%</td>
<td>Hill climbing [BIC]</td>
</tr>
<tr>
<td></td>
<td>MB-CTPC</td>
<td>80.25%</td>
<td>19.75%</td>
<td>0.00%</td>
<td>80.25%</td>
<td>CTPC</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>44.75%</td>
<td>55.25%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>70.99%</td>
<td>29.01%</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td></td>
<td>S_{V'}</td>
<td>= 0</td>
<td>57.41%</td>
<td>42.59%</td>
</tr>
</tbody>
</table>
Table 6.3: Estimated performance measures (mean ± std. deviation) over the synthetic datasets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F₁ score</th>
<th>Global Brier score</th>
<th>Learning time (s)</th>
<th>Classification time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hill climbing [BIC]</td>
<td>0.6287 ± 0.2091</td>
<td>0.8681 ± 0.1156</td>
<td>0.7753 ± 0.1971</td>
<td>0.4612 ± 0.3068</td>
<td>108.1529 ± 138.1886</td>
<td>6.6264 ± 9.8329</td>
</tr>
<tr>
<td>Tabu search [BIC]</td>
<td>0.6287 ± 0.2091</td>
<td>0.8681 ± 0.1156</td>
<td>0.7752 ± 0.1971</td>
<td>0.4612 ± 0.3068</td>
<td>107.2197 ± 137.5624</td>
<td>6.5689 ± 9.7887</td>
</tr>
<tr>
<td>Hill climbing [BDe]</td>
<td>0.6229 ± 0.2702</td>
<td>0.8655 ± 0.1365</td>
<td>0.7711 ± 0.1980</td>
<td>0.4678 ± 0.3076</td>
<td>106.4439 ± 136.8045</td>
<td>6.4251 ± 9.5402</td>
</tr>
<tr>
<td>CTPC</td>
<td>0.6351 ± 0.2704</td>
<td>0.8729 ± 0.1125</td>
<td>0.7821 ± 0.1952</td>
<td>0.4537 ± 0.3084</td>
<td>41.8573 ± 47.3675</td>
<td>7.5145 ± 10.5733</td>
</tr>
<tr>
<td>MB-CTPC</td>
<td>0.6121 ± 0.2725</td>
<td>0.8658 ± 0.1165</td>
<td>0.7696 ± 0.1999</td>
<td>0.4689 ± 0.3091</td>
<td>49.7194 ± 57.5774</td>
<td>6.4525 ± 9.6591</td>
</tr>
</tbody>
</table>

6.5.1.1 Hill climbing and tabu search

Table 6.2 shows that hill climbing optimizing the BIC score obtains better results in all performance measures, except learning and classification time, for more datasets than the BDe score, improvements that were found statistically significant. As a result, subsequent comparisons will mainly consider the BIC score. Meanwhile, learning and classification time differences between hill climbing [BIC] and tabu search [BIC] were statistically significant, with the latter being faster in more than 60% of the datasets, while no statistically significant differences were detected for other performance measures. The reason is that, thanks to the restrictions on the structure search imposed by the tabu list, the tabu search algorithm finds the structure by analyzing fewer arcs. However, note that these time differences may not be very substantial since tabu search [BIC] achieves a sub-second improvement in 60% and 96% of the cases in terms of learning and classification time, respectively.

6.5.1.2 CTPC and hill climbing

Although the mean results of Table 6.3 suggest that CTPC may outperform hill climbing, statistical tests show that hill climbing significantly improves all performance measures, except learning time. This was the case independently of the score optimized by hill climbing. Nevertheless, this is the case when evaluating the algorithms on all datasets. If the datasets are divided according to the cardinality of the features, hill climbing obtains better results only when the cardinality is relatively low (two, three or four states). Statistically significant improvements are made with CTPC for all performance measures when features have eight possible states.

To further study the influence of features’ cardinalities, we have analyzed the results of the performance measures when the cardinality is increased from two to 30. Five datasets have been sampled for each possible cardinality (145 datasets) from a single randomly generated structure with ten features, four class variables, and bridge and feature subgraph densities of 10%. Figure 6.2 shows that classifiers learned with constraint-based algorithms (CTPC and MB-CTPC) are more robust than score-based and hybrid solutions as feature cardinality...
increases. The number of examples for each possible state transition declines as the cardinality of features increases, making models learned with score-based algorithms less accurate. Figure 6.3 shows the results of the last experiment but using sequences with twice the duration. Increasing the sequence duration enables score-based algorithms to achieve better results with features of a higher cardinality. Nevertheless, they still show worse robustness than constraint-based solutions for all the studied measures. At first glance, we thought that the BIC penalization negatively influenced the models’ accuracy. However, this behavior is even more severe for the BDe score. We can then conclude that for problems where features have high cardinality and the sequence duration is relatively small, constraint-based algorithms might be more convenient due to their robustness. This is consistent with the findings of Scutari et al. [2019] and Bregoli et al. [2021] for BNs and CTBNs, respectively.

Finally, it is worth noting that constraint-based algorithms achieve much shorter learning times since the estimated parameters can be cached and quickly retrieved for future statistical tests. The usefulness of a cache is more limited for scored-based algorithms as they iteratively evaluate previously unseen parent set configurations.

6.5.1.3 MB-CTPC and CTPC

The MB-CTPC algorithm, compared to CTPC, achieves the same or, in a few cases, better results in about 60% of the datasets for global and mean accuracy and macro F₁ score. Simultaneously, it reduces the learning time on most datasets (98%), which was its main
Figures 6.3: Results of increasing features’ cardinality with sequences of 20 time units.

Table 6.4: Estimated performance measures (mean ± std. deviation) over synthetic datasets generated from Multi-CTBNCs with 30 and 5 feature and class variables, respectively.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F₁ score</th>
<th>Global Brier score</th>
<th>Learning time (s)</th>
<th>Classification time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTPC</td>
<td>0.9874 ± 0.0149</td>
<td>0.9975 ± 0.0030</td>
<td>0.9597 ± 0.0851</td>
<td>0.0238 ± 0.0286</td>
<td>340.2399 ± 14.1575</td>
<td>217.4570 ± 37.9228</td>
</tr>
<tr>
<td>MB-CTPC</td>
<td>0.9590 ± 0.0403</td>
<td>0.9917 ± 0.0081</td>
<td>0.9546 ± 0.0792</td>
<td>0.0675 ± 0.0665</td>
<td>147.0353 ± 22.6594</td>
<td>150.1994 ± 29.0154</td>
</tr>
</tbody>
</table>

The CTPC algorithm obtains better results than MB-CTPC on multiple datasets, significantly improving all performance measures except learning and classification time. Nevertheless, these differences may not be significant enough if our priority is to speed up the model.
learning. Figures 6.4c and 6.4d show that the mean improvement of the global accuracy can go from as little as 1% to a maximum of 6% in the performed experiments. These differences are even lower for the mean accuracy (see Figures 6.4e and 6.4f) and macro $F_1$ score (see Figures 6.4g and 6.4h). As for the global Brier score, differences in the percentages are more significant; however, they are less than 0.01 in 50% of the datasets. The slightly lower accuracy of classifiers learned with MB-CTPC arises from the incorrect definition of some class variable descendants (Step 3 of Algorithm 6.2). Possible causes behind this may include weak relationships between variables, training datasets not sufficiently representative of the underlying problem or the assumption that waiting times of features conditioned on a non-parent ancestor follow an exponential distribution.
Overall, the classification time is also reduced when using the MB-CTPC algorithm. Table 6.4 shows a 31% reduction in the classification time of CTPC in the previous experiment, while a significant improvement is also seen in Tables 6.2 and 6.3. Alongside the learning time, this characteristic could make MB-CTPC more convenient for a streaming environment where models need to be incrementally updated and real-time response may be required. However, this result has to be interpreted with caution since, on certain occasions, the classification time improvement may benefit from the spouses of class variables not being defined correctly.

We can conclude that the MB-CTPC algorithm is a good choice when execution time is a priority, especially when dealing with high-dimensionality datasets. We also believe this kind of structure learning algorithm could be extended to real-time scenarios where our models have to be dynamically updated, and solutions should be provided as quickly as possible. Nevertheless, we should also consider that a trade-off exists between assuring better accuracy or significantly reducing the learning and classification times, which has to be assessed depending on the particular problem.

6.5.1.4 Hybrid $|S_{V_i V_j}| = 0$ and hybrid $|S_{V_i V_j}| = 1$

Overall, varying the maximum separating set size of the hybrid algorithm from zero to one results in no change in most experiments except for execution time. For example, the global accuracy improves in just 1.76% of the datasets when only testing for unconditional independence. However, 95% of these latter datasets have in common the presence of binary features. This outcome is coherent, given that the constraint-based algorithm, which is less accurate than the score-based approaches when features are binary, has more influence on the solution as the maximum size of the separating set increases. For this reason, statistically significant improvements were obtained for most performance measures with the hybrid $|S_{V_i V_j}| = 0$ algorithm. The hybrid $|S_{V_i V_j}| = 1$ solution succeeded in reducing the learning and classification times significantly. Nevertheless, differences between using both parameter values are generally negligible in the performed experiments.

6.5.1.5 Hybrid $|S_{V_i V_j}| = 0$ vs. hill climbing [BIC] and CTPC

Considering all synthetic datasets, no significant differences were found in terms of classification capabilities between the hybrid $|S_{V_i V_j}| = 0$ and CTPC algorithms, but an improvement in global Brier score and classification time by the former and in learning time by the latter. When comparing the hybrid $|S_{V_i V_j}| = 0$ and hill climbing [BIC] algorithms, the classification capabilities of the former are significantly diminished, yet the learning and classification times are substantially improved.

Considering only datasets with low cardinality features (two, three and four states), the
statistical tests report an improvement in all performance measures, except execution times, by
the hybrid algorithm with respect to CTPC while still notably improving the learning and, to a
lesser extent, classification times of hill climbing [BIC]. Thus, the proposed algorithm achieves
its intended purpose as an intermediate solution, as it combines the better classification
capabilities of score-based solutions with the faster learning time of constraint-based approaches
in the case of dealing with low cardinality features. As discussed in Section 6.5.1.2, the CTPC
algorithm performs significantly worse than the score-based approaches when dealing with low
cardinality features. Then, combining these two approaches in a hybrid algorithm improves
the CTPC performance.

6.5.1.6 Robustness to noisy data

In this section, we evaluate the robustness of each structure learning algorithm by progressively
increasing the presence of noise in the data. To do so, ten Multi-CTBNCs with 20 features,
four class variables and bridge and feature subgraph densities of 10% were randomly defined.
Four datasets were sampled from each of these models with the following degrees of noise:

- **Noise-free dataset.** No noise was added when sampling this dataset.

- **Low noise dataset.** The 5% of features’ transitions and class variables’ states were
  randomly sampled, while a Gaussian noise with zero mean and a standard deviation of
  0.1 was added to features’ transition times.

- **Medium noise dataset.** The 10% of features’ transitions and class variables’ states
  were randomly sampled, while a Gaussian noise with zero mean and a standard deviation
  of 0.2 was added to features’ transition times.

- **High noise dataset.** The 20% of features’ transitions and class variables’ states
  were randomly sampled, while a Gaussian noise with zero mean and a standard deviation
  of 0.5 was added to features’ transition times.

Overall, most algorithms were similarly affected by the increased noise in the datasets.
However, this was different for the MB-CTPC algorithm. Table 6.5 compares the results of
the CTPC and MB-CTPC algorithms for the given noise levels and shows a greater difference
between them as the noise increases. As we have discussed in Section 6.5.1.3, the definition
of the class variables’ descendants is a crucial step for the correct functioning of the algorithm,
which is being increasingly affected by adding more noise. On the other hand, the CTPC
algorithm is more robust, as it simply tests the dependencies between all possible pairs of
features. Nevertheless, this translates into a much higher cost in the learning time, which is
almost twice that of the MB-CTPC algorithm, and in the classification time.
Table 6.5: Estimated performance measures (mean ± std. deviation) when increasing noise in synthetic datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F1 score</th>
<th>Global Brier score</th>
<th>Learning time (s)</th>
<th>Classification time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise-free dataset</td>
<td>0.7927 ± 0.2217</td>
<td>0.9458 ± 0.0595</td>
<td>0.9202 ± 0.0878</td>
<td>0.2797 ± 0.2926</td>
<td>102.5735 ± 4.0715</td>
<td>25.0581 ± 2.9236</td>
</tr>
<tr>
<td>Low noise dataset</td>
<td>0.7898 ± 0.2228</td>
<td>0.9443 ± 0.0618</td>
<td>0.9198 ± 0.0899</td>
<td>0.2891 ± 0.2977</td>
<td>93.1212 ± 4.0769</td>
<td>22.5176 ± 2.6766</td>
</tr>
<tr>
<td>Medium noise dataset</td>
<td>0.7781 ± 0.2171</td>
<td>0.9411 ± 0.0605</td>
<td>0.9163 ± 0.0879</td>
<td>0.3028 ± 0.2857</td>
<td>76.6488 ± 3.8178</td>
<td>19.1597 ± 2.1732</td>
</tr>
<tr>
<td>High noise dataset</td>
<td>0.7151 ± 0.1841</td>
<td>0.9198 ± 0.0558</td>
<td>0.8885 ± 0.0796</td>
<td>0.3882 ± 0.2350</td>
<td>52.2272 ± 2.4204</td>
<td>13.0160 ± 1.3833</td>
</tr>
<tr>
<td>MB-CTPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise-free dataset</td>
<td>0.7743 ± 0.2057</td>
<td>0.9412 ± 0.0556</td>
<td>0.9151 ± 0.0833</td>
<td>0.3065 ± 0.2692</td>
<td>54.2652 ± 6.0504</td>
<td>19.1903 ± 1.9290</td>
</tr>
<tr>
<td>Low noise dataset</td>
<td>0.7611 ± 0.2075</td>
<td>0.9360 ± 0.0594</td>
<td>0.9107 ± 0.0851</td>
<td>0.3284 ± 0.2737</td>
<td>47.5664 ± 3.8766</td>
<td>17.1980 ± 1.8136</td>
</tr>
<tr>
<td>Medium noise dataset</td>
<td>0.7234 ± 0.2000</td>
<td>0.9244 ± 0.0585</td>
<td>0.8982 ± 0.0811</td>
<td>0.3755 ± 0.2601</td>
<td>38.0962 ± 2.9495</td>
<td>14.2572 ± 1.4947</td>
</tr>
<tr>
<td>High noise dataset</td>
<td>0.6477 ± 0.1960</td>
<td>0.8933 ± 0.0696</td>
<td>0.8595 ± 0.0836</td>
<td>0.4655 ± 0.2388</td>
<td>25.0125 ± 1.5644</td>
<td>9.8222 ± 0.9120</td>
</tr>
</tbody>
</table>

6.5.2 Experimental results on a real-world dataset

This section evaluates the effectiveness of the introduced structure learning algorithms when solving a real-world problem. For this purpose, the British Household Panel Survey (BHPS) dataset has been used, which is the result of a longitudinal study gathering, among others, information about the finances, health, household, work life, opinions and other personal aspects of UK citizens [University of Essex, 2018]. The surveys were conducted annually from 1991 to 2009 in a total of 18 waves, all of which are used in this work, except for the first, fourth and ninth waves, as some variables relevant to this study were not collected.

The objective of this experiment is to predict the state of certain variables related to personal information, given the evolution of individuals’ responses over the years. The usefulness of this work lies in the possibility of extracting relationships from a particular dataset and applying them to infer new knowledge in, for example, other surveys in which specific information was not collected. The following seven class variables were defined for this experiment:

- **Dental check-up.** Denotes whether the individual has had a dental check-up in the last year.

- **Employment status.** Specifies the individual’s current employment status. This variable takes ten states: self-employed, in paid employment, unemployed, retired, maternity leave, looking after family or home, full-time student, long-term sick or disabled, on a government training scheme or others.

- **Limb, back or neck problems.** Indicates whether the respondent has problems or disabilities related to the arms, legs, hands, feet, back or neck (including arthritis and rheumatism).

- **Lives with spouse or partner.** Indicates whether the respondent is living with their spouse or partner.
Chapter 6. Constraint-based and hybrid structure learning of Multi-CTBNCs

- **Responsible adult for child.** Indicates whether the individual is responsible for a child under 16 years old.
- **Sex.** Refers to the sex of the individual.
- **Smoker.** Indicates if the individual smokes.

The BHPS dataset collects information from 29702 individuals on more than 1300 variables. For the definition of the features, we focused on a subset of discrete-state variables related to health, work life, household and other personal information, such as marital status or residence region. Due to the nature of these data, the BHPS dataset is largely incomplete, as respondents could be unable or refuse to answer certain questions. Therefore, we have kept those variables containing no more than 3% of missing data. Altogether, 26 features of different kinds have been used (see Figure 6.5).

Each sequence extracted from the BHPS dataset contains the survey results of a single individual, results that are ordered according to the date they were collected and taking into account that the state of the class variables (class configuration) does not vary along a sequence. Therefore, for each individual, as many sequences have been extracted as changes in the class configuration plus one. Sequences containing only one observation were discarded since they do not provide any information. This may occur, for example, if an individual took the survey on a single occasion. In total, 14925 sequences were extracted.

Ten cross-validations with random shuffles have been performed to statistically compare the performance of the structure learning algorithms on the BHPS dataset. The average results of this experiment are shown in Table 6.6. Despite obtaining better results in more synthetic experiments with score-based algorithms (see Figure 6.2 and Section 6.5.1.2), a substantial improvement in classification performance was achieved by constraint-based algorithms on this occasion. In addition, hill climbing results were significantly improved by optimizing the BDe score instead of BIC, which was not the case with the synthetic datasets. These circumstances illustrate the importance of exploring structure learning algorithms of different natures, as they could obtain better results in diverse settings. As for the differences between the CTPC and MB-CTPC algorithms, the proposed method significantly improved all performance measures. The MB-CTPC algorithm reduced the learning time of CTPC by more than 50% while delivering a similar classification performance. Finally, the hybrid algorithm exhibited the worst overall results for this experiment, except for the classification time when using a maximum separating set size of two. If the size is reduced, the results worsen for this experiment and the computational time increases. In the case of an empty separating set, the available computational resources were not enough to learn the structures. This is because the hybrid algorithm’s restriction phase can only remove a few arcs, reporting a very dense structure to the maximization phase. This causes estimating a large set of parameters for nodes with multiple parents.
Table 6.6: Estimated performance measures (mean ± std. deviation) with ten cross-validations on the BHPS dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F1 score</th>
<th>Micro F1 score</th>
<th>Global Brier score</th>
<th>Learning time (s)</th>
<th>Classification time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hill climbing [BIC]</td>
<td>0.2029 ± 0.0014</td>
<td>0.7772 ± 0.0009</td>
<td>0.5614 ± 0.0013</td>
<td>0.7536 ± 0.0011</td>
<td>0.6991 ± 0.0004</td>
<td>11.4602 ± 0.5081</td>
<td>3.7173 ± 0.0877</td>
</tr>
<tr>
<td>Tabu search [BIC]</td>
<td>0.2029 ± 0.0014</td>
<td>0.7772 ± 0.0009</td>
<td>0.5614 ± 0.0013</td>
<td>0.7536 ± 0.0011</td>
<td>0.6991 ± 0.0004</td>
<td>10.9979 ± 0.3562</td>
<td>3.7071 ± 0.1446</td>
</tr>
<tr>
<td>Hill climbing [BDe]</td>
<td>0.2184 ± 0.0023</td>
<td>0.7902 ± 0.0008</td>
<td>0.5668 ± 0.0026</td>
<td>0.7668 ± 0.0009</td>
<td>0.8555 ± 0.0017</td>
<td>10.8260 ± 0.2263</td>
<td>3.1707 ± 0.0616</td>
</tr>
<tr>
<td>CTPC</td>
<td>0.3799 ± 0.0020</td>
<td>0.8601 ± 0.0009</td>
<td>0.6608 ± 0.0022</td>
<td>0.8439 ± 0.0009</td>
<td>0.7901 ± 0.0014</td>
<td>47.5145 ± 7.7559</td>
<td>5.5352 ± 0.0872</td>
</tr>
<tr>
<td>MB-CTPC</td>
<td>0.3849 ± 0.0012</td>
<td>0.8614 ± 0.0005</td>
<td>0.6638 ± 0.0015</td>
<td>0.8445 ± 0.0005</td>
<td>0.7866 ± 0.0008</td>
<td>23.4605 ± 3.0436</td>
<td>5.2359 ± 0.0718</td>
</tr>
<tr>
<td>Hybrid [</td>
<td></td>
<td>S_Vi</td>
<td>= 0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid [</td>
<td></td>
<td>S_Vi</td>
<td>= 1]</td>
<td>0.1573 ± 0.0021</td>
<td>0.7653 ± 0.0007</td>
<td>0.4898 ± 0.0033</td>
<td>0.7303 ± 0.0025</td>
</tr>
<tr>
<td>Hybrid [</td>
<td></td>
<td>S_Vi</td>
<td>= 2]</td>
<td>0.1597 ± 0.0022</td>
<td>0.7691 ± 0.0007</td>
<td>0.5072 ± 0.0031</td>
<td>0.7338 ± 0.0023</td>
</tr>
</tbody>
</table>

Figure 6.5 shows a structure learned with the MB-CTPC algorithm. Although an exhaustive analysis of the variables’ dependencies is beyond the scope of this work, some meaningful relationships can be easily identified thanks to the graphical capabilities of the model:

- The class variable *Limb, back or neck problems* is directly related to other medical variables, including those indicating limitations in daily and work activities for medical reasons; for example, individuals who claim to have these health problems are more likely to report that their health is limiting their work in the upcoming years.

- The individuals’ employment status was found to be related to their health condition, type of household or if they are of working age; for example, given that an individual is retired but still of working age (pre-retired), they are expected to be above working age earlier than if they had any other employment status. Specifically, they are expected to do so in about 4 to 5 years.

- The evolution of an individual’s household type is naturally affected by variables such as the number of people living in the household or whether the individual lives with a spouse or partner. However, this variable is also affected by other factors such as employment status; for example, given an employed person living with a partner in a two-person household, the likelihood of the household type changing from couple without children to couple with dependent children from one survey to the following is greater than if, for example, the individual is retired.

- Given an individual whose health limits their work, it is more likely that their health will also affect their daily tasks in a shorter time than if they did not have difficulties at work. The same is true when changing the occurrence order.

- The evolution of the household type provides information on whether an individual is living with a spouse or partner over a period of time; for example, it is evident that a household transition from single-parent with dependent children to single-parent with non-dependent children only occurs for individuals who are not living with a spouse or partner.
Figure 6.5: Structure learned on the BHPS dataset with the MB-CTPC algorithm (class variables are highlighted in gray).
The interest in using a Multi-CTBNC to model the BHPS dataset lies in the fact that dependencies also exist between the class variables. As an example, the following relationships have been found:

- Given that an individual has had a dental check-up in the last year, they are likelier to be a non-smoker.
- Individuals are more likely to be responsible for a child when they had a dental check-up last year. We could hypothesize that this is due to the individual being more responsible when becoming a parent and the intention of being a healthy role model for their children.
- The probability that an individual is responsible for a child is higher if they are employed or self-employed than unemployed. Moreover, this probability is significantly increased if the individual has indicated that their employment status is family or home care. On the opposite side, the probability of being responsible for a child is very low if the individual is retired or a student.

6.6 Conclusions and future work

This chapter introduces, for the first time, constraint-based and hybrid structure learning algorithms for CTBNCs, which were specially designed to learn Multi-CTBNCs. The novel constraint-based algorithm, named MB-CTPC, aims to learn the structure of these classifiers by performing conditional independence tests only on dependencies that could be relevant to the Markov blankets of class variables. Then, the hybrid algorithm, a solution not even studied for CTBNs, seeks to combine the strengths of the score- and constraint-based methods.

Synthetic experiments show that the MB-CTPC algorithm significantly improves the learning and classification times of Multi-CTBNCs compared to other structure learning algorithms while maintaining a competitive classification performance. This algorithm is especially convenient when learning from high-dimensionality datasets, but significant improvements were obtained regardless of the number of variables, their cardinality or structure density. Furthermore, the proposed algorithm was statistically proven to be the best choice when used on a real-world dataset, as it matched the best results achieved by the constraint-based algorithm CTPC while significantly reducing the learning and classification time of the models. Finally, the hybrid algorithm provides, in certain scenarios, an intermediate solution that significantly improves the results of CTPC and drastically reduces the learning and classification times of score-based techniques.

Multiple areas of open research were found while conducting this work:
• **Improving descendant identification of MB-CTPC.** MB-CTPC may struggle to identify descendants of class variables. Using a phase distribution, such as Erlang [Liu et al., 2018c], to model the transition times of features conditioned on ancestor class variables may improve this task and the model’s accuracy. We would also like to study whether this improvement could contribute to mitigating the impact of noise.

• **Distinguishing children from other descendants of class variables.** Discerning between descendants and children of class variables may improve MB-CTPC learning time, as arcs to feature nodes with no parent class variables are irrelevant for the classification task.

• **Class-bridge decomposable Multi-CTBNC.** When dealing with datasets of high dimensionality and size, multiple class variables and an underlying structure of consider- able density, the classification phase can be computationally expensive (see, for example, Table 6.4). Thus, a class-bridge decomposable [Bielza et al., 2011] Multi-CTBNC could be appropriate to improve this aspect.

• **Improving the hybrid structure learning algorithm.** A first hybrid algorithm was proposed to solve the structure learning problem for CTBNs. However, the improvements achieved compared to other solutions are limited to specific scenarios. Further study of these algorithms is needed to determine their usefulness in real situations, as well as to improve their results in more settings. As a starting point, attempts could be made to extend some ideas used in existing hybrid algorithms for other PGMs, such as Tsamardinos et al. [2006] or Gasse et al. [2014].

• **Adapting the HITON algorithm for Multi-CTBNC structure learning.** The HITON algorithm [Aliferis et al., 2003] could be adapted for learning the structure of Multi-CTBNCs. This algorithm has already been successfully applied to the learning of MBCs [Borchani et al., 2012]. Therefore, we are interested in assessing the performance of this proposal with respect to the solutions studied in this chapter.

• **FSS for categorical time series.** Employing FSS techniques prior to structure learning could improve execution times and classification results since irrelevant and redundant variables that may confuse learning algorithms are ignored. However, there is little literature on FSS for categorical time series and even less on a supervised context. This approach would be useful for problems such as the one introduced with the BHPS dataset, which consists of more than 1300 features. This issue will be further studied in Chapter 8.

• **Online extension of MB-CTPC.** The proposed MB-CTPC structure learning algorithm could be suitable for dynamic data contexts where models are continuously updated, as the importance and relationships between variables can evolve over time,
and solutions are provided as fast as possible. In Chapter 8, we propose an online extension of the MB-CTPC algorithm to perform structure learning on data streams.
Chapter 7

Feature subset selection for data and feature streams: A review

7.1 Introduction

FSS is the task of choosing a subset of features from a complete dataset with the objective of improving the efficiency, precision and interpretability of models. This can be achieved using fewer variables since FSS seeks to discard irrelevant and redundant features that may be confusing and harmful for learning algorithms.

The review presented in this chapter focuses on incremental FSS algorithms, i.e., approaches used in environments where new training instances, features or both are received progressively over time and, therefore, the subset of candidate features should be updated dynamically. Traditional batch approaches cannot be directly applied in a streaming context, implying that models have to be completely retrained when new data are received, a solution that may be inefficient and poorly scalable when processing big data [Wu et al., 2017]. In addition, incremental algorithms are not only useful when the entire set of features or instances (examples) are not available beforehand, but they can also be considered in the preprocessing step when all the data are available and computational resources are limited [Jing et al., 2018].

Proposals that incrementally work with information flows are known by several terms, such as online or incremental. There is no well-established definition for these concepts and they are used interchangeably by some authors. However, we believe it is appropriate to consider online learning as a subtype of incremental learning, which imposes stricter time and space requirements and should be able to work endlessly in a streaming environment [Gama et al., 2014; Losing et al., 2018]. It can be seen that the definition of an algorithm as online is truly
Figure 7.1: Example of a framework of online FSS algorithms for data streams.

dependent on its complexity when addressing real-time problems. An FSS algorithm could be adequately efficient to handle a specific online problem, given solutions at the expected time, but it may not be appropriate in a more demanding environment. If we focus on the complexity of the algorithm to define it as online or incremental, the definition would depend on the characteristics of the problem being addressed. Therefore, we concluded that online learning literature focuses more on defining algorithms that can be applied to real-time problems, while incremental algorithms simply seek to improve the efficiency obtained by batch proposals. In return, online algorithms, at least in the case of receiving new instances, may find their effectiveness reduced as the model complexity is additionally bounded [Losing et al., 2018]. However, this definition does not perfectly fit when new features are received. If we face a theoretically infinite stream of features, where an infinite number of them could be considered relevant and non-redundant, it is inevitable that a memory problem will appear at some point. Nevertheless, algorithms that could face this problem are considered to be online algorithms in the literature. This chapter reviews algorithms defined as online or incremental by their authors that can be applied in a context where new data instances or features can be received, i.e., on data streams, feature streams or a combination of both. The objective is not to present an unequivocal way of defining an algorithm as online or incremental but to study diverse proposals that are capable of incrementally updating a current selection of variables with the arrival of new information.

Dynamic data can be categorized into three main classes: data streams, feature streams and a hybrid known as data and feature stream. When working with data streams, new instances arrive over time, while the feature set remains fixed. Thus, the subset of selected features and the models should be adapted according to the new data. Figure 7.1 shows a framework commonly followed by online FSS algorithms when working with data streams, where a set of feature weights is dynamically updated according to certain rules. Given the arrival of a new instance (or instance group), the applied rules take into account, for example, whether a misclassification occurs or if the value of a loss function is above a certain threshold. If we consider other algorithms commonly defined as incremental, it is common to use temporal windows in conjunction with information theory measures, such as mutual information, to evaluate the feature importance. The appearance of new instances represents several challenges, such as model accuracy reduction due to changes in the underlying data relationships. This problem is known as concept drift, one of the main obstacles when considering data streams. There are real-world problems that involve handling data streams...
Figure 7.2: General framework of FSS algorithms for feature streams.

in very diverse areas, such as spam detectors [Wang et al., 2014], personalized medicine [Swan, 2012] or analysis of currency exchange rates and demographic data [Yi et al., 2000].

Unlike data streams, feature streams are characterized by a fixed set of instances, but the feature set evolves over time; thus, the objective is to maintain a feature subset with the most relevant features that have arrived so far, avoiding redundancies. The general approach for incremental/online FSS on feature streams includes a relevance and redundancy analysis, as shown in Figure 7.2, to assess whether a new variable (or variables) is relevant and, in a positive case, to check for redundancies caused by its inclusion in the subset of selected features. This framework varies since some approaches, for example, do not include a redundancy analysis or require additional steps (see, for example, Section 7.4.2). Feature streams may be useful when (1) the generation of all the features is expensive and it is not feasible to wait for all of them to be generated, as could occur in some scientific experiments, or (2) when we do not have the means to obtain all possible features at the moment since, for example, they could be obtained from a social network and be based on current news. We can find several examples of FSS applications over feature streams, such as in medical learning problems [Wang et al., 2017], statistical relational learning [Zhou et al., 2005], texture-based image segmentation [Perkins and Theiler, 2003], edge detection in grayscale images [Glocer et al., 2005] and analysis of social network data [Yu et al., 2016a]. For example, consider a medical context where different costly experiments can be performed to decide on a treatment, but not all of them are necessary to learn a confident enough model. Then, features (experiment results) would arrive over time, and our predictive model can be updated accordingly.

Finally, a hybrid type of stream called a data and feature stream, which implies the appearance of both new instances and features over time, will be discussed.

FSS techniques are commonly categorized into three types depending on their relationship with the model construction: the filter, embedded and wrapper methods [Saëys et al., 2007; Guyon et al., 2008]. A filter algorithm uses only the training data and a metric to score each feature or subset of features, so the result is independent of learning models. Embedded techniques include the FSS process as a part of the model definition, i.e., embedded in the
modeling algorithm. Wrapper algorithms evaluate different feature subsets by training a specific model for each of them and return the feature combination whose model achieves the best testing performance (e.g., accuracy). The creation of several models every time new data are available may be computationally expensive and incompatible with the interest of online algorithms to evaluate new data, ideally, in real time. This scenario is even worse if new features are received since, without considering the use of any heuristic search, there would be \(2^m - 1\) (for \(m\) features) possible feature combinations to evaluate and, therefore, models to build. We believe this is the reason why wrapper algorithms do not receive much attention for incremental learning, especially when considering online scenarios with more demanding time and memory constraints.

The comprehensive study presented in this chapter is motivated by the fact that reviews on incremental/online FSS are scarce, so the vast majority of studies focus primarily on batch techniques [Pereira et al., 2018; Urbanowicz et al., 2018; Bolón-Canedo and Alonso-Betanzos, 2019; Venkatesh and Anuradha, 2019; Zhang et al., 2019; Solorio-Fernández et al., 2020; Pintas et al., 2021]. However, the increased interest in algorithms with incremental mechanisms requires an in-depth study of the state of the art on this topic. Recent publications considering the streaming scenario only explain a really small and not very diverse set of methods compared to the number of existing contributions in this field [Hu et al., 2016; Tommasel and Godoy, 2016; Li et al., 2017; Cai et al., 2018; Ma et al., 2018; Somasundaram and Mylsamy, 2018; AlNuaimi et al., 2020]. Most of the algorithms studied in these publications were proposed several years ago, so it is also necessary to contemplate new proposals that can handle the performance demands and learning problems that we may face in increasingly massive datasets and faster processes. This chapter aims to provide a big picture of incremental FSS methods, presenting the reader with a detailed study of proposals with different characteristics and for a wide variety of learning problems. In addition, this comprehensive analysis has allowed us to detect an interesting variety of open issues in several areas of study. Nevertheless, this chapter not only reviews the most recent and prominent algorithms but also clarifies and structures an area of study in which there is confusion both in the acronyms and definitions used and in the categorization of some proposals. In summary, the main objectives of this work are the following:

- A comprehensive and structured study of a wide variety of incremental/online FSS algorithms for data and/or feature streams.
- A comparison of the studied algorithms, describing in detail their functioning and offering alternatives for some of their shortcomings.
- The study of an extensive variety of learning problems where incremental/online FSS is interesting to be applied. These include, among others, supervised (binary, multi-class and multi-label), unsupervised, multi-task, multi-view or ensemble learning, discrete
and continuous features, receiving instances/features individually or in groups, or rough
set-based approaches.

This chapter includes the content of Villa-Blanco et al. [2023a].

Chapter outline

The remainder of this chapter is organized as follows. Section 7.2 presents the classification
criteria used throughout this chapter for the incremental FSS algorithms and a brief review of
the terminology used in the literature to refer to FSS algorithms for data or feature streams.
In Section 7.3, several FSS algorithms that can be used with data streams are introduced
and categorized as supervised or unsupervised approaches. In Section 7.4, FSS algorithms for
feature streams are described and categorized into those that process new features individually
or by groups. In turn, these two categories are divided into supervised and unsupervised
algorithms. In Section 7.5, the incremental incorporation of both new instances and features
is analyzed, and the few proposals that exist are studied. In Section 7.6, future work on
incremental/online FSS is outlined. Finally, in Section 7.7, some conclusions achieved during
the realization of this review are presented.

7.2 Classification of incremental FSS approaches

Figure 7.3 summarizes the classification criteria for the incremental FSS algorithms used
throughout this review. Different algorithms for data streams and feature streams, which
are classified depending on how the learning is performed (supervised or unsupervised), the
number of available class variables and the arrival of new features (individual or group), will
be studied. Additionally, some algorithms for contexts where both instances and features
appear simultaneously are commented on.

The terminology used for FSS algorithms that work on dynamic data differs depending on
the author. After a study of the literature, the following terms have been found to reference
FSS algorithms for data streams:

- Dynamic feature selection [Barddal et al., 2017].
- Incremental attribute reduction [Hu et al., 2005].
- Incremental feature selection [Liang et al., 2014].
- Instance-based online streaming feature selection [Rahmaninia and Moradi, 2018].
- Online feature selection [Wang et al., 2014].
• Standard online feature selection [Eskandari and Javidi, 2016].

Proposals that can be applied to feature streams can be found as follows:

• Feature-based online streaming feature selection [Rahmaninia and Moradi, 2018].
• Incremental attribute reduction [Jing et al., 2016].
• Incremental feature selection [Zeng et al., 2015].
• Online feature selection [Yu et al., 2014].
• Online streaming feature selection [Wu et al., 2013].
• Streaming feature selection [Zhou et al., 2005].
• Streamwise feature selection [Dhillon et al., 2010].

To encompass under a single term all the algorithms that perform FSS in an incremental manner, we think that it is appropriate to use incremental feature subset selection. In turn, the proposals could be additionally classified as online feature subset selection algorithms to be consistent with the analysis of online and incremental learning in Section 7.1.

7.3 FSS on data streams

FSS over data streams aims to determine which features to add to and delete from the current set of selected features with the arrival of new instances. In this section, several algorithms will be discussed for supervised learning tasks and for unsupervised environments where it is
not possible to evaluate the feature relevance against a class variable.

### 7.3.1 Supervised learning

As mentioned in Figure 7.1, a common approach to perform online FSS is to define weights for each feature that are updated with the arrival of new instances without having to repeatedly process or store instances already seen. These weights are then used to select the most important features. In this category, two algorithms that use regularization and truncation techniques are presented [Wang et al., 2014]: online feature selection by learning with full inputs (OFS) and the online feature selection by learning partial inputs (OFS$_p$). Both fit a linear classifier and update it when a new instance is misclassified.

OFS assumes the full input of every new training instance to be received, including a binary class variable $C \in \{-1, +1\}$. Then, it can update the model using two different approaches: (1) the modified perceptron by truncation for OFS, which, given a misclassified instance $z^t = (x^t, c^t)$, updates the current weights $w^t \in \mathbb{R}^m$ by simply increasing them with the vector $x^t c^t$ (Perceptron rule); and (2) the OFS via sparse projection, which updates the weights by online gradient descent and projects them to an $L_2$ ball. After updating the weights, both methods truncate them to zero except for the $b$ largest, where $b$ is a predefined value. The first approach is a simple method that does not guarantee that discarded features have sufficiently small weights, which can result in many classification mistakes. The second proposal avoids this problem by projecting the weight vector to an $L_2$ ball before the truncation, so its norm is ensured to be bounded.

Meanwhile, OFS$_p$ further limits the problem to cases where learners can access only a small and fixed feature set of the training instances, an interesting setting when the entire feature set for all instances is expensive to obtain. OFS$_p$ does not simply choose $b$ features with nonzero weights, but it defines the feature subset by alternating phases of exploration, where $b$ features are chosen randomly, and exploitation, which chooses the $b$ features whose weights are nonzero. In that way, the method avoids getting stuck in a configuration with poor classification performance.

Recent studies have focused on reducing the computational and memory costs of FSS algorithms for applications with large-scale and high-dimensional data. For example, Wu et al. [2017] presented a second-order online feature selection (SOFS) algorithm that seeks to improve previous first-order algorithms in terms of effectiveness, efficiency and scalability over high-dimensional sparse data. While first-order algorithms update the feature weights using only the first-order derivative information of the gradient, second-order proposals can also use second-order information, such as geometrical properties of data, to improve the FSS. SOFS is based on the confidence-weighted method introduced by Dredze et al. [2008], which
maintains not only feature weights but also an estimate of their confidence. The inclusion of this additional information may be positive for the performance of online FSS algorithms since it retains information about past instances. As a result, confidence can be used to guide the updating rule of the feature weights, decreasing those of the less confident features. Different measures can be used to represent the confidence of the weights. In the case of SOFS, the weight vector is assumed to be modeled by a Gaussian distribution $w \sim N(\mu, \Sigma)$ with mean vector $\mu$ and a diagonal covariance matrix $\Sigma$, parameters that are updated with the misclassification of new instances. The $j$-th entry in the diagonal of $\Sigma$ represents the confidence in $w_j$. Therefore, the lower the value of $\Sigma_{jj}$ is, the higher the confidence in the weight of feature $j$. Note that instead of drawing a weight vector $w$ every time an instance is received, SOFS simply uses the mean vector $\mu$ to perform predictions.

Exploiting the second-order information of feature weights implies an additional high computational cost. Thus, the authors reduced the complexity of SOFS by two means. First, only features with nonzero values in the new instance are updated and evaluated if a misclassification occurs. In this way, SOFS is linearly dependent on the number of nonzero features instead of on $m$, as OFS is. However, this advantage is limited to those problems where the data are sparse. Second, SOFS uses a max heap-based approach, i.e., it maintains a binary tree of features where the parent nodes have a covariance that is greater than or equal to that of their children. Selecting the most relevant features by searching in the covariance diagonal can be a time-consuming task. Therefore, this tree data structure allows more efficient retrieval of the $b$ currently most confident weights without having to sort all of them in every step. The max heap combined with the monotonic decreasing property of the covariance, i.e., its value can only decrease or stay the same when it is updated, helps to further reduce the computational cost of the algorithm. Thus, when the weight and confidence of a certain feature in the tree are updated, the tree is modified only from the position of this feature towards its children since its parent is guaranteed to have a larger covariance. Additionally, those features that were not updated will not be part of the max heap since the covariance of the root node cannot increase. Then, the last step is to check whether the covariance of those features that were updated and are not in the tree is less than that of the root node. In such a case, the feature of the root node is removed and its weight is set to zero, while the tree is updated with the inclusion of this new feature with a lower covariance.

SOFS (like OFS) focuses on binary classification problems. Thus, Wu et al. [2017] extended it to handle a multi-class variable, meaning that $C \in \{0, \ldots, k - 1\}$ is a classification problem with $k$ classes. This extension, which is referred to as SMOFS for simplicity, is based on using the one-vs-rest strategy. SMOFS essentially turns the multi-class problem into several binary problems, so $w \in \mathbb{R}^{km}$ and $\Sigma \in \mathbb{R}^{km \times km}$, and at a time $t$, the total confidence of the $j$-th feature is defined by a combination of each of its confidences in the different binary tasks.

FSS can be performed across multiple related tasks, which is known as multi-task feature
selection (MTFS). The objective of multi-task learning is to simultaneously build models for tasks that are sufficiently different but still have commonalities, with the intention of obtaining more precise models than if they were obtained separately. For example, in spam filtering, users could need different features to label an email as spam; however, they would still share common features [Weinberger et al., 2009]. A single classifier would perform correctly generalizing among different users but would fail to learn the specific preferences of a single user [Dredze and Crammer, 2008]. Then, learning all tasks simultaneously could improve the performance of each of their models, thanks to sharing their related information. Note that this learning problem differs from transfer learning, which aims to solve a single target task by transferring knowledge from a very similar source task. Transfer learning does not optimize the performance of all the tasks involved and, therefore, does not allow the transfer of information among them in all possible directions [Torrey and Shavlik, 2010].

The multi-task problem consists of $Q$ tasks whose data come from the same space, with each task having $N_q$ instances, $q = 1, \ldots, Q$. The entire dataset can be represented as $\mathcal{D} = \bigcup_{q=1}^Q \mathcal{D}_q$, where $\mathcal{D}_q = \{(x_i^q, c_i^q)\}_{i=1}^{N_q}$ is a sample from distribution $P_q$. The final task is to learn $q$ functions, $f_q$, one for each task, such that $f_q(x^q)$ approximates $c^q$ as accurately as possible. These functions are parametrized by a weight vector, i.e., $f_q(x^q) = (w_q)\top x^q$, so most MTFS methods seek to learn a weight matrix $W$, whose dimension is $m \times Q$ (see Figure 7.4), by minimizing an empirical risk and a regularizer [Yang et al., 2013]. This matrix defines the importance of each feature for the different tasks.

MTFS has been studied mainly from a batch perspective. However, online FSS can be applied in multi-task problems by using proposals such as the dual-averaging for multi-task feature selection (DA-MTFS) framework [Yang et al., 2013], which selects relevant features across tasks. At iteration $t$, DA-MTFS needs to receive $Q$ instances, one for each task, with the objective of updating the weight matrix $W^t$. This is performed by computing the subgradient

![Figure 7.4: Multi-task feature selection learning scheme.](image-url)
of a chosen loss function on the current weights, which is used to store an average of all the computed subgradients \( G_t \) calculated until time \( t \). The choice of the loss function depends on the problem to solve. For example, Yang et al. [2013] proposed the logit and hinge losses for binary classification and the square loss for regression tasks. Finally, the average subgradient is used along with a regularization term to compute the new weight matrix \( W^{t+1} \). DA-MTFS uses a mixed \( L_{p,1} \)-norm in the regularization term, i.e., a hybridization of the \( L_1 \)-norm and an \( L_p \)-norm, to perform a joint regularization across multiple tasks. In other words, the mixed norm groups the weights of a specific feature for every task using an \( L_p \)-norm and applies an \( L_1 \)-norm over the resultant vector. The general form of the mixed \( L_{p,r} \)-norm is computed over a matrix \( W \) as:

\[
||W||_{p,r} = \left( \sum_{i=1}^{m} ||w_i||_r^p \right)^{\frac{1}{r}},
\]

where \( w_i \) is the \( i \)-th row of \( W \). Three mixed norms were studied: (1) \( p = 1 \), which promotes sparse solutions but only observes the instances of each task individually; (2) \( p = 2 \), which uses the information across tasks simultaneously, with the inconvenience of resulting in non-sparse solutions; and (3) a novel \( L_{1/2,1} \)-norm regularization, i.e., a linear combination of the \( L_1 \)-norm and \( L_2 \)-norm, which allows sparse solutions while still considering the information provided by all tasks.

DA-MTFS assumes the arrival of one instance per task in each iteration, which may not be realistic. The authors proposed not updating those tasks that do not receive an instance in a certain iteration; however, this would make them have less influence on the weights. Currently, there is no proposal that avoids this inconvenience. Additionally, it would be interesting to extend the online FSS algorithms to other multi-task settings, for example, to cases where there is no common set of features across all tasks due to outlier tasks [Gong et al., 2012]. Finally, we would like to clarify that this online learning scenario differs from the lifelong learning problem, as the latter receives new tasks over time (no instances), which it tries to solve using knowledge acquired from previous tasks [Chen and Liu, 2018].

Feature weights can also be included in naive Bayes classifiers to perform embedded FSS on data streams. Naive Bayes is an interesting probabilistic classifier algorithm based on applying Bayes’ theorem with a strong conditional independence assumption between the features given the class variable. Despite its simplicity, the low complexity of naive Bayes makes it attractive for predicting data streams [Klawonn and Angelov, 2006; Salperwyck et al., 2015]. Naive Bayes can be improved by avoiding its attribute conditional independence assumption, which is not commonly held in real-world problems. This is possible by, for example, weighting the predictive variables. These algorithms are known as weighted naive Bayes classifiers, and they introduce a weight for each variable (see Figure 7.5) to relax the attribute conditional independence assumption.
An online weighted naive Bayes proposal was introduced by Salperwyck et al. [2015], which we will refer to as OWNB. Although it does not focus on the online FSS problem by assigning feature weights, as previous algorithms do, the relevance of each variable is established. This algorithm defines a weight $w_{ij}$ per variable $X_i$ and per class value $c_j$, computing them using stochastic gradient descent for a certain cost function. Thus, given a new training instance $z^t = (x^t, c^t)$, the weights are updated as follows:

$$w_{ij}^{t+1} = w_{ij}^t - \eta \frac{\partial \text{Cost}^{z^t}}{\partial w_{ij}} ,$$

where $\text{Cost}^{z^t}$ is the cost function applied to the new training instance, $\frac{\partial \text{Cost}^{z^t}}{\partial w_{ij}}$ is the partial derivative of the cost function with respect to the weights and $\eta$ is the learning rate.

Instead of relying on a single FSS algorithm, we could combine the knowledge extracted from different methods to define a more robust feature subset. This is the approach followed by the multi-objective automated negotiation based online feature selection (MOANOFS) system introduced by BenSaid and Alimi [2021], which identifies relevant and redundant features based on the feature weights assigned by different online FSS algorithms. This system is divided into two decision levels. The first level selects the most confident learners by means of a trust model [Das and Islam, 2011]. Subsequently, in the second level, relevant features are defined with a novel multilateral automated negotiation method according to the feature weights and prediction error reported by each learning algorithm selected in the first level. The final weight assigned to a feature will be determined by certain rules depending on the number of algorithms that find the feature as relevant and the weight they report. This system can, therefore, be considered an ensemble technique, as it combines the outcomes of multiple learning algorithms to try to obtain a better result. Unfortunately, it should be noted that it is limited to FSS methods that are embedded in binary classifiers.

In a multi-class context, we can resort to the simpler online bagging feature selectors (OFS-Bag) and online boosting feature selectors (OFS-Boo) ensemble algorithms presented by Ditzler et al. [2017]. As the names of the algorithms suggest, the ensembles are built using variants of online bagging and boosting [Oza and Russell, 2001]. These proposals do not rely on a negotiation method to define the final feature weights. Instead, they are defined as a linear combination
of the weights given by the base online FSS algorithms. Therefore, it is straightforward to apply techniques such as the one-vs-rest strategy (as the previous SMOFS does) in order to use binary classifiers in a multi-class classification problem. Although the authors report the best classification results when using the boosting approach in their experiments, it is worth noting that the possibility of parallelizing the learning of the base models when using the bagging-based algorithm may be of great interest in certain online learning environments.

Previous proposals do not explicitly define the relevance of features but rather adapt the currently selected features by monitoring the performance of the models and updating some feature weights. However, incremental FSS does not only involve the use of feature weighting approaches since other algorithms explicitly quantify the importance of the features for the class variable. This is the case of tree-based learning algorithms [Domingos and Hulten, 2000; Hulten et al., 2001; Gama et al., 2006; Bifet and Gavaldà, 2009], which decide the splitting feature based on, for example, mutual information or the Gini index. A well-known algorithm of this kind is the very fast decision tree (VFDT) [Domingos and Hulten, 2000], which defines the best feature to perform a split by using the most recent instances that were received from the stream. VFDT may omit the use of some features in the decision tree, so it implements embedded FSS. The set of instances studied to select the splitting feature is given by the Hoeffding bound, from which the statistics for the splitting are extracted for each leaf node. In this way, instances are not stored and, therefore, they are not revisited.

The drawback of the VFDT algorithm is that it cannot efficiently deal with concept drift since the built tree is immutable and it is only possible to continue growing the tree in its leaves. Thus, the features selected for splits are never revoked. For those cases, the concept-adapting very fast decision tree (CVFDT) can be considered [Hulten et al., 2001]. CVFDT keeps the tree updated by replacing sub-trees that were outdated after a concept drift occurs. To do so, a sliding window of instances is considered. The oldest instances can be forgotten from the tree, while the newest instances are used to build alternate sub-trees and decide if they should replace the old ones.

The above algorithms have in common that they embed FSS into a model. However, we may be interested in a filter approach, which can be applied regardless of the model. A well-known algorithm of this kind is the fast correlation-based filter (FCBF), which defines the relevance and redundancy of features by analyzing their correlation [Yu and Liu, 2003]. If the correlation between a feature and a class variable is higher than a predefined threshold, then that feature is considered to be relevant. However, if its correlation with another feature is higher than that with the class variable, then it is removed since it is redundant. This is measured using the symmetrical uncertainty, which is a normalized version of the mutual information that does not favor features with more values. Given two variables $X_1$ and $X_2$,
the symmetrical uncertainty between them is defined as:

$$SU(X_1, X_2) = 2 \left( \frac{I(X_1, X_2)}{H(X_1) + H(X_2)} \right),$$

where the mutual information is:

$$I(X_1, X_2) = H(X_1) - H(X_1 | X_2),$$  \hspace{1cm} (7.1)

which measures the reduction in the entropy (H(·)) of $X_1$ when $X_2$ is known.

FCBF is not an incremental algorithm since it considers that all training instances are available at the beginning when computing the relevance of each feature. However, the use of time windows allows us to apply the method to data streams. Articles such as Nguyen et al. [2012] prove the interest in applying this window version of FCBF on data streams, where it was embedded into a heterogeneous ensemble algorithm that is incrementally updated. Thanks to the inclusion of FCBF, the processing time to build the base models is reduced, while the accuracy of the ensemble is improved. This simple approach to adapt the traditional FSS algorithm to incremental scenarios is commonly used, and other famous algorithms, such as Relief [Kira and Rendell, 1992] or its extensions [Kononenko, 1994], could benefit from it.

### 7.3.1.1 Rough set-based FSS

One inconvenience of working with data streams is that the knowledge is incomplete since new instances appear over time. As explained in Chapter 4, rough set theory is ideal for this setting, as it needs only the dataset as input, and the objective of the attribute reduction is to generate minimal reducts that are sufficient to characterize the knowledge of a dataset. However, this process is an NP-hard task [Skowron and Rauszer, 1992], so heuristic methods have been proposed. Most of the proposed algorithms follow the same approach, which consists of analyzing whether the current feature subset meets a certain criterion and, if not, the most significant features are included until the criterion is met. Afterward, variables that do not compromise the criterion when removed are defined as redundant. We would like to emphasize that it is common to use the term significance when working with rough sets to refer to the importance/relevance of a feature. This section studies incremental FSS algorithms based on rough set theory, classifying them according to the heuristic used to compute the reducts.

Classical rough set theory can be directly applied to multi-label problems; however, this approach has several limitations [Li et al., 2016]. No specific proposals for incremental attribute reduction for multi-label data streams were found, although articles such as Liu et al. [2018a] show that such an adaptation should not be complex. That said, any of the
following algorithms is considered to be appropriate for multi-label tasks. In addition, due to
the nature of this theory, practically all the proposed algorithms need to store the previously
seen instances. This inconvenience can be avoided with the use of an appropriate time window,
although it may not be trivial to define [Yang et al., 2018].

**FSS based on the positive region**

One way to define a new reduct due to variation in
the number of instances is to study the changes in the positive region. In this category, we
find algorithms such as the *incremental attribute reduction algorithm based on the element set*,
which we refer to as IARES [Hu et al., 2005]. To understand IARES, first, the concepts of
positive \( \mathcal{P} \) and negative \( \mathcal{N} \) elementary sets must be introduced. \( \mathcal{P} \) contains all the condition
classes of \( \mathcal{U}/\text{IND}(\mathcal{X}) \) that are consistent, i.e., all instances in these equivalence classes share
the same value for the class variable. The opposite is true for the \( \mathcal{N} \) set, in which all the
condition classes of \( \mathcal{U}/\text{IND}(\mathcal{X}) \) are inconsistent, i.e., each of them has at least two instances
with different values for the class variable. Knowing this, a subset \( \mathcal{B} \subseteq \mathcal{X} \) is a reduct iff there
are no collision equivalence classes on \( \mathcal{B} \) in \( \mathcal{P} \). This is an equivalence class that (1) shares
the same values for the features in \( \mathcal{B} \) with all condition classes in \( \mathcal{P} \) but not the same class
variable value or (2) shares the same values for features in \( \mathcal{B} \) with one equivalence class in
\( \mathcal{N} \). The reason for avoiding collision equivalence classes on \( \mathcal{B} \) in \( \mathcal{P} \) is that, in other cases,
\( \text{POS}_{\mathcal{B}}(C) = \text{POS}_{\mathcal{X}}(C) \) would not hold, which would imply a loss of information since the
positive region given by \( \mathcal{X} \) is not preserved by \( \mathcal{B} \).

When processing a new instance \( \mathbf{z}^t \), IARES considers three scenarios: (1) there exists an
equivalence class \( \mathcal{E}_j \in \mathcal{N} \) to which \( \mathbf{z}^t \) belongs, (2) there exists an equivalence class \( \mathcal{E}_j \in \mathcal{P} \) to
which \( \mathbf{z}^t \) belongs, and (3) there is no equivalence class \( \mathcal{E}_j \) to which \( \mathbf{z}^t \) belongs. It is only in the
third situation where the current reduct \( \mathcal{S}^t \) may need to be updated since a new equivalence
class \( \mathcal{E}_{\text{new}} \) is created for \( \mathbf{z}^t \) (and added to \( \mathcal{P} \) since it is a unit set), which could be a collision
equivalence class on \( \mathcal{S}^t \) in \( \mathcal{P} \). If this is the situation, IARES solves the collision by adding a
feature (or features) from \( \mathcal{X} \setminus \mathcal{S}^t \) that eliminates the collision, i.e., which makes \( \mathcal{E}_{\text{new}} \) have
different feature values with respect to any equivalence class on \( \Hat{\mathcal{S}}^t = \mathcal{S}^t \cup \mathcal{X}_i \) in \( \mathcal{P} \). In this
way, a transitional subset \( \Hat{\mathcal{S}}^t \) is obtained that satisfies \( \text{POS}_{\Hat{\mathcal{S}}^t}(C) = \text{POS}_{\mathcal{X}}(C) \) after adding \( \mathbf{z}^t \) to the dataset. To avoid confusion, the notation \( \Hat{\mathcal{S}}^t \) will be used to refer to the
current state of the reduct, i.e., \( \mathcal{S}^t \) modified by the inclusion or removal of features, which is
prior to the final result \( \mathcal{S}^{t+1} \). Finally, redundant features are removed from \( \Hat{\mathcal{S}}^t \). A feature
\( \mathcal{X}_i \in \Hat{\mathcal{S}}^t \) is considered to be redundant if \( \text{POS}_{\Hat{\mathcal{S}}^t \setminus \{\mathcal{X}_i\}}(C) = \text{POS}_{\mathcal{X}}(C) \). Once all redundant
features are removed, the remaining features in \( \Hat{\mathcal{S}}^t \) form the reduct \( \mathcal{S}^{t+1} \).

Shu and Qian [2015] go a step further and propose the *incremental attribute reduction
algorithm for the immigration of multiple objects* (IARM-I) for environments where multiple
instances are added simultaneously to incomplete datasets, i.e., datasets where some feature
values are missing for certain instances. IARM-I is able to process new instances that arrive
by groups, something not possible with IARES, which receives groups but processes instances one by one. The interest in processing groups instead of single instances is based on the speed at which large volumes of data can be generated in some problems. Therefore, processing instances individually may be inefficient. When a group of instances $U_a$ arrives, IARM-I compares the positive regions obtained from $U \cup U_a$ given the reduct $S^t$ and the complete feature set $X$. If they do not have the same positive region, features from $X \setminus S^t$ are included in $S^t$ until they do. The addition of features is performed in descending order of significance value in the current $\tilde{S}^t$.

There is no common way to define the significance of a variable; however, it is usually divided into outer and inner significance, so the effect of adding (outer) or removing (inner) a feature from a set is assessed. Given a feature $X_i \notin \tilde{S}^t$ and a measure $M(\cdot, \cdot)$, the outer significance of $X_i$ in $\tilde{S}^t$ relative to $C$ can be defined as:

$$\text{sig}_{\text{outer}}(X_i, \tilde{S}^t, C) = |M(\tilde{S}^t, C) - M(\tilde{S}^t \cup \{X_i\}, C)|.$$(7.2)

If $X_i \in \tilde{S}^t$, the inner significance can be obtained as:

$$\text{sig}_{\text{inner}}(X_i, \tilde{S}^t, C) = |M(\tilde{S}^t, C) - M(\tilde{S}^t \setminus \{X_i\}, C)|.$$(7.3)

It is common to use outer significance to define the most important features to include in the reduct, while inner significance indicates the redundant features that should be removed. However, note that these approaches just compute the difference in a measure when a feature is considered or not. Therefore, this is not a practice common to all proposals.

In the case of Shu and Qian [2015], the variation in the cardinality of the positive region is used as the measure to compute the significance of the variables, i.e., $M(\tilde{S}^t, C) = |\text{POS}_{\tilde{S}^t}(C)|$. Then, features with higher outer significance are included in the reduct until $\text{POS}_{\tilde{S}^t \cup \tilde{U}_a}(C) = \text{POS}_{X \setminus \tilde{S}^t}(C)$. Once this condition is met, redundant features are removed from $\tilde{S}^t$ if their inner significance in $\tilde{S}^t$ is zero. The resultant $\tilde{S}^t$ is the new reduct $S^{t+1}$.

Previous algorithms expect new instances to be added to the dataset. However, FSS can also be performed when a set of instances $U_r$ is removed from $U$. The existence of these algorithms is justified by possible erroneous instances that need to be removed, for example, erroneous diagnoses of patients [Shu et al., 2019]. The work done by Shu and Qian [2015] contemplates this possibility by proposing the incremental attribute reduction algorithm for the emigration of multiple objects (IARM-E), which is a reduced version of IARM-I in which only the existence of irrelevant features in the reduct is checked. The elimination of instances is also considered by Shu et al. [2019], where two novel algorithms are presented, one for the inclusion of new instances and another for the deletion of instances. These algorithms, which are known as the incremental approach for feature selection in the decision system under
multiple objects being added (IFSA) and under multiple objects being deleted (IFSD), perform an incremental updating of the reduct using the dependency function. Each time a new set of objects is received or deleted, IFSA and IFSD incrementally update the dependency functions $\gamma^{U'}(S^t, C)$ and $\gamma^{U'}(X, C)$, which represent to what degree $C$ depends on $S^t$ and $X$, respectively, where $U'$ is the set of instances after the addition or elimination. The degree of dependency compares the number of instances in the positive region with the number in the studied universe:

$$\gamma^{U'}(S^t, C) = \frac{|\text{POS}^{U'}_{S^t}(C)|}{|U'|}. \quad (7.4)$$

The closer its value is to one, the more instances in the universe can be correctly classified with the available information. If $\gamma^{U'}(S^t, C) \neq \gamma^{U'}(X, C)$, the reduct is incremented with the sequential addition of features with the highest outer significance. Once $\gamma^{U'}(S^t, C) = \gamma^{U'}(X, C)$, redundancies are removed. A feature $X_i \in \tilde{S}^t$ is redundant if its inner significance in $\tilde{S}^t$ is equal to zero. The outer and inner significances are computed as in Equations (7.2) and (7.3), where $M(\tilde{S}^t, C) = \gamma^{U'}(\tilde{S}^t, C)$.

**FSS based on the discernibility matrix** The incremental updating algorithm of attribute reduction set (IUAARS) [Guan, 2009] uses the discernibility matrix element set to report all the possible reducts at a certain moment and is able to process new instances that arrive by groups. The discernibility matrix indicates the subset of features that distinguish each pair of instances, and the discernibility matrix element set is formed by the nonempty elements of this matrix. Thus, when a new instance arrives, IUAARS updates the current reduct by analyzing the changes in this set due to inconsistencies or repetitions or because the instance was not seen before. If a change is necessary, IUAARS uses the set to find indispensable and dispensable features.

The dynamic attribute reduction algorithm based on 0-1 integer programming [Xu et al., 2011], which will be called DARIP for convenience, uses a similar approach. DARIP is based on creating a set of inequality constraints over a new instance set, where each constraint indicates the necessary features to discern each instance pair, therefore reporting the most relevant features. Although the authors considered DARIP to be an algorithm to address multiple instances simultaneously, it generates constraints by pairs of new instances. This ultimately implies analyzing each new instance one by one against the rest. The constraints grow quadratically with the number of instances; thus, preprocessing is introduced to remove redundant constraints. Once this is done, features in the remaining constraints form a reduct for the incoming instance set, which is used to increase $S^t$ and obtain the final reduct $S^{t+1}$. The disadvantage of DARIP is that it does not eliminate features from $S^t$, so it cannot efficiently handle concept drift since features that become irrelevant or redundant are not removed. However, unlike previous rough set-based approaches, DARIP does not need to
store previously seen instances.

**FSS based on knowledge granularity** Few algorithms use knowledge granularity to quantify the discernibility of features. Given that $\mathcal{U}/\text{IND}(\mathcal{X}) = \{ \mathcal{E}_1, \ldots, \mathcal{E}_v \}$, the knowledge granularity of $\mathcal{X}$ is computed as:

$$
\text{GK}_\mathcal{U}(\mathcal{X}) = \sum_{i=1}^{v} \frac{|\mathcal{E}_i|^2}{|\mathcal{U}|^2}.
$$

(7.5)

As can be deduced from Equation (7.5), the smaller the knowledge granularity is, the stronger the discernibility since the partitioning is finer and, therefore, the size of the equivalence classes is smaller. This measure can be used to dynamically find a reduct when adding or deleting instances since the goal is to use a feature subset that maintains the knowledge granularity achieved with the complete feature set.

Notable members of this category include the updating attribute reduction algorithm when adding (UARAO) and deleting (UARDO) some objects with a multi-granulation view [Jing et al., 2017], two algorithms that focus on improving the efficiency when addressing large-scale datasets that are increased or reduced dynamically, respectively. To do so, they adapt the knowledge granularity formula to be incrementally updated for datasets with a multi-granulation view, which is based on dividing a dataset into $g$ small sub-tables (small granularities), i.e., $\mathcal{U} = \bigcup_{i=1}^{g} \mathcal{U}_i$. Approaches to determine the size of the sub-tables were studied by Liang et al. [2012]. The equivalence classes from each sub-table are obtained and merged to obtain the equivalence classes from the complete dataset. This strategy enables the equivalence classes to be obtained in a more efficient way since computing them directly from a large dataset is more costly in terms of computational time and space.

UARAO follows the same approach as the previously studied algorithms. Upon the arrival of a new instance set $\mathcal{U}_a$, the algorithm checks whether the conditional knowledge granularity of $C$ under $\mathcal{X}$ and $\mathcal{S}^t$ in $\mathcal{U}_a$ is the same, i.e., $\text{GK}^{\mathcal{U}_a}(C|\mathcal{S}^t) = \text{GK}^{\mathcal{U}}(C|\mathcal{X})$. The conditional knowledge granularity is computed as:

$$
\text{GK}^{\mathcal{U}_a}(C|\mathcal{S}^t) = \text{GK}^{\mathcal{U}_a}(\mathcal{S}^t) - \text{GK}^{\mathcal{U}_a}(\mathcal{S}^t \cap C).
$$

(7.6)

If the knowledge granularity is the same, the reduct is not altered since the equivalence classes obtained in $\mathcal{U}_a$ by $\mathcal{S}^t$ and $\mathcal{X}$ are identical and, therefore, the discernibility is the same in $\bigcup_{i=1}^{g} \mathcal{U}_i \cup \mathcal{U}_a$. If the equality is compromised, relevant features from $\mathcal{X} \setminus \hat{\mathcal{S}}^t$ are included in the reduct based on their outer significance measure in $\hat{\mathcal{S}}^t$, which is computed with Equation (7.2), where $M(\hat{\mathcal{S}}^t, C) = \text{GK}^{\mathcal{U}_{\mathcal{X}\setminus\hat{\mathcal{S}}^t}}(C|\hat{\mathcal{S}}^t)$. When $\text{GK}^{\mathcal{U}_a}(C|\mathcal{S}^t) = \text{GK}^{\mathcal{U}_a}(C|\mathcal{X})$, the final reduct $\mathcal{S}^{t+1}$ is obtained after the removal of redundant features, i.e., any feature in $\mathcal{S}^{t+1}$ that would not alter the knowledge granularity.
In the case of removing a set of instances from $U$, UARDO simply evaluates whether the features in $S^t$ are still relevant and non-redundant and removes those that do not alter the knowledge granularity when discarded.

**FSS based on entropy information** Entropy is a common uncertainty measure that can be used in rough set theory to define the significance of features. Some common entropies that can be applied within rough set theory are Shannon’s [Wierman, 1999], complementary [Liang et al., 2002] and combination [Qian and Liang, 2008]. Let $B \in X$ and $U/IND(B) = \{E_1, ..., E_v\}$; then, the Shannon $H^U(B)$, complementary $E^U(B)$ and combination $CE^U(B)$ entropies of $B$ for the instance set $U$ can be defined, respectively, as:

$$H^U(B) = -\sum_{i=1}^{v} \frac{|E_i|}{|U|} \log \frac{|E_i|}{|U|},$$

$$E^U(B) = \sum_{i=1}^{v} \frac{|E_i|}{|U|} \left(1 - \frac{|E_i|}{|U|}\right),$$

and

$$CE^U(B) = \sum_{i=1}^{v} \frac{|E_i|}{|U|} \left(1 - \frac{R^2_{E_i}}{R^2_t}\right),$$

where $R^2_{E_i} = |E_i|(|E_i| - 1)/2$ is the number of indistinguishable instance pairs in $E_i$.

The conditional version of these three entropies $ME(\cdot|\cdot)$ was provided by Liang et al. [2014] for data streams, so they are updated incrementally with changes in the equivalence classes due to the inclusion of new instances. Furthermore, they introduced the group incremental algorithm for reduct computation (GIARC), an algorithm that uses these measures to find a reduct when a new instance group is received.

Upon the arrival of a group of instances $U_a$, GIARC updates the current reduct only if instances in $U_a$ are indistinguishable using the features in $S^t$ but distinguishable when using $X$. This occurs when the conditional entropies $ME(\cdot|\cdot)$ (Shannon, combination or complementary) of $S^t$ and $X$ relative to $C$ are not the same, i.e., $ME^U(C|S^t) \neq ME^U(C|X)$. In that case, the features from $X \setminus S^t$ with the highest outer significance in $S^t$ are included in the reduct until $ME^{U_a,U_a}(C|\tilde{S}^t) = ME^{U_a,U_a}(C|X)$, where $ME^{U_a,U_a}(\cdot|\cdot)$ represents one of the three novel formulas used to recompute the entropy after adding $U_a$. The outer significance is computed as in Equation (7.2), defining $M(\tilde{S}^t, C)$ as the previously selected incremental conditional entropy. Once relevant features are added, redundant ones are removed by checking whether their inner significance in $\tilde{S}^t$ is zero. This is computed with Equation (7.3), employing the same conditional entropy as in the outer significance. The resulting set is the new reduct $S^{t+1}$.

Experiments in Liang et al. [2014] proved that GIARC produces results similar to those of
algorithms based on the positive region, such as IARES, but the computational time is always shorter.

### 7.3.1.2 FSS for rough set extensions

Rough set theory was extended with proposals that attempted to overcome its shortcomings. One of the main problems is that it is not convenient to handle hybrid (discrete and continuous) attributes, performing efficiently only if the attributes are discrete. This problem can be avoided by discretizing continuous features, although this approach can cause information loss [Zeng et al., 2015]. Thus, it may be more interesting to use fuzzy rough sets, a fuzzy generalization of rough sets that allows to handle uncertainty when data are real-valued [Dubois and Prade, 1990]. The fuzzy rough set theory introduces the fuzzy similarity relation, which measures how similar two instances are. Thus, instances are grouped into equivalence classes using soft boundaries depending on their similarities [Cornelis et al., 2008].

Two recent algorithms for fuzzy rough set-based FSS were presented by Yang et al. [2018]. These proposals adapt the reduct in an incremental manner using the relative discernibility relations of each feature and the feature set, which are updated every time a new set of instances arrives. The first novel algorithm is the first incremental version for fuzzy rough set-based feature selection (IV-FS-FRS-1), which checks whether the discernibility relation of the original feature set and that obtained with the reduct is the same. When it is not, new features are added to the reduct until they are equal. Finally, features that do not alter the previous equality when removed are discarded. The second novel algorithm is the second incremental version for fuzzy rough set-based feature selection (IV-FS-FRS-2), which follows the same strategy, except that it begins finding a reduct when there are no more new instance sets. Clearly, IV-FS-FRS-2 has the advantage of being faster than IV-FS-FRS-1; however, it cannot provide a feature subset at any time.

Another extension of rough set theory is variable precision rough set theory, which is used to overcome the sensitivity to noise and errors in data. Defining the $\mathcal{B}$-lower approximations of $\mathcal{H}$ using instances that share the same feature and class variable values may be a strict requirement, which does not reflect possible errors, such as in the observation of the data. Therefore, variable precision rough set theory includes a degree of uncertainty that relaxes the requirements to admit instances in the $\mathcal{B}$-lower approximation, thereby allowing a level of misclassification. The smaller this degree is, the smaller the boundary region [Ziarko, 1993].

The first incremental attribute reduction algorithms for variable precision rough sets are the incremental algorithm for $\beta$-upper distribution reduct (Incremental-U) and for $\beta$-lower distribution reduct (Incremental-L) [Chen et al., 2016]. These algorithms use discernibility matrices to dynamically find a reduct when a new instance is received.
In conclusion, Table 7.1 summarizes the analyzed supervised FSS algorithms for data streams. For each algorithm, the following are indicated: (1) the type of class variables the algorithm was designed for, (2) whether the algorithm can process a new group of instances, (3) whether the algorithm allows the elimination of previously received examples, and (4) whether the algorithm re-analyzes past instances.

### 7.3.1.3 Distributed FSS

Previous proposals follow a centralized approach, i.e., the FSS is performed by just one process in a single machine. It could be interesting to address big data processing from a distributed perspective since these algorithms can meet the necessary performance requirements by using multiple computing units. For instance, Fong et al. [2016] attempted to overcome computational and memory problems with the use of nature-inspired metaheuristics, specifically swarm algorithms. However, this approach is based on using a batch-accelerated particle swarm optimization and defining the feature subset by maximizing the fitness obtained with an incremental classifier. Thus, it may not be able to efficiently handle a real-time data stream. Particle swarm optimization is one nature-inspired metaheuristic that could be applied to incremental FSS [Diao et al., 2013], but there is a lack of other nature-inspired approaches to this problem, such as genetic [Leardi et al., 1992; Oh et al., 2004], harmony search [Diao and Shen, 2010, 2012] and particle swarm optimization variants such as the competitive swarm optimizer [Gu et al., 2018].

It is worth noting the distributed FSS technique for multi-label classification problems presented by Gonzalez-Lopez et al. [2020], which is based on the estimation of mutual information measures. Although it does not propose an online algorithm, we believe it may be of interest when processing high-dimensional data streams with time windows. Its main idea is to compute the mutual information between variables in a distributed environment with Apache Spark, as this is a computationally expensive task in a multi-label classification problem, especially with high-dimensional data.

### 7.3.2 Unsupervised learning

Performing FSS over unsupervised data faces the challenge of not being able to quantify the relevance of features by using the information provided by a class variable. Section 4.2.1 discusses some main criteria used to define relevant and redundant features when data are unlabeled. In this section, we examine two specific algorithms that perform unsupervised FSS in a data streaming context.

One of the first unsupervised FSS proposals for data streams is StreamFeatWeight [Huang
### Table 7.1: Supervised learning algorithms for FSS on data streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class</th>
<th>Process groups</th>
<th>Instance removal</th>
<th>No instance revisit</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS and OFS&lt;sub&gt;p&lt;/sub&gt;</td>
<td>Use instances with full and partial inputs to update feature weights.</td>
<td>Binary</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Wang et al. [2014]</td>
</tr>
<tr>
<td>SOFS and SMOFS</td>
<td>Second-order algorithms that use a MaxHeap-based approach.</td>
<td>Binary and Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Wu et al. [2017]</td>
</tr>
<tr>
<td>DA-MTFS</td>
<td>Performs online FSS over multiple related tasks simultaneously.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Yang et al. [2013]</td>
</tr>
<tr>
<td>OWNB</td>
<td>Constructs a weighted naive Bayes classifier.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Salperwyck et al. [2015]</td>
</tr>
<tr>
<td>MOANOFS</td>
<td>Online ensemble FSS algorithm with a multi-objective negotiation technique.</td>
<td>Binary</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>BenSaid and Alimi [2021]</td>
</tr>
<tr>
<td>OFS-Bag and OFS-Boo</td>
<td>Bagging- and boosting-based online FSS algorithm.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Ditzler et al. [2017]</td>
</tr>
<tr>
<td>VFDT</td>
<td>Decision tree algorithm for data streams with embedded FSS.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Domingos and Hulten [2000]</td>
</tr>
<tr>
<td>CVFDT</td>
<td>Extension of VFDT that is able to handle concept drifts.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Hulten et al. [2001]</td>
</tr>
<tr>
<td>FCBF&lt;sup&gt;c&lt;/sup&gt;</td>
<td>Filter-based algorithm that employs symmetrical uncertainty to define relevant and redundant features.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Yu and Liu [2003]</td>
</tr>
</tbody>
</table>

**Rough set-based**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class</th>
<th>Process groups</th>
<th>Instance removal</th>
<th>No instance revisit</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>IARES</td>
<td>Studies equivalence classes to determine the reducts.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Hu et al. [2005]</td>
</tr>
<tr>
<td>IARM-(I/E)</td>
<td>Uses positive region on incomplete decision systems.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>Shi and Qian [2015]</td>
</tr>
<tr>
<td>IFS(A/D)</td>
<td>Compares positive regions to find a reduct in incomplete decision systems.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>Shi et al. [2019]</td>
</tr>
<tr>
<td>IUAARS</td>
<td>Updates the reduct by using the discernibility matrix element set.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>Guan [2009]</td>
</tr>
<tr>
<td>DARIP</td>
<td>Based on 0-1 programming to determine the reduct through inequality constraints.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>Xu et al. [2011]</td>
</tr>
<tr>
<td>UAR(AO/DO)</td>
<td>Uses knowledge granularity to find a reduct with a multi-granulation view.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>Jing et al. [2017]</td>
</tr>
<tr>
<td>GIARC</td>
<td>Incrementally updates information entropies to define reducts.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>Liang et al. [2014]</td>
</tr>
<tr>
<td>IV-FS-FRS-(1/2)</td>
<td>Fuzzy rough set-based algorithms based on the relative discernibility relations.</td>
<td>Multi-class</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Yang et al. [2018]</td>
</tr>
<tr>
<td>Incremental-(U/L)</td>
<td>Attribute reduction for variable precision rough sets which use discernibility relation matrices.</td>
<td>Multi-class</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Chen et al. [2016]</td>
</tr>
</tbody>
</table>

<sup>a</sup> Different loss functions can be employed depending on the learning task.

<sup>b</sup> Does not explicitly allow the removal of instances, it forgets old ones with time windows.

<sup>c</sup> Batch algorithm that can be implemented with time windows.

<sup>d</sup> The time window could be formed by old and new instances or only by new ones.

<sup>e</sup> Needs an instance group to generate constraints, but instances are analyzed one by one.

<sup>f</sup> New instances are analyzed more than once, but they are not stored for future iterations.
Carlos Villa Blanco et al., 2015], also known as FSDS. The key idea of this algorithm, to enable its application to data streams, is to use matrix sketching to generate a low-rank approximation, defined as a matrix sketch, of all the observed data (or the data contained in a time window) up to time $t$. In other words, instances received by time $t$ are reduced into a matrix of a predefined rank $k$ that most faithfully represents the original data. This rank-$k$ approximation is updated with the inclusion of new data, which enables the algorithm to adapt to concept drift and make only one pass over the data. FSDS performs this step with a modified version of the frequent-directions algorithm [Liberty, 2013]. Therefore, singular value decomposition is performed on the cosine similarity matrix of the matrix sketch augmented with new data to define its spectrum, i.e., eigensystem, and the new approximation. Then, the leading $k$ eigenvectors, i.e., those corresponding to the $k$ largest eigenvalues, are used as the target in a regression analysis with regularization to report new feature coefficients. These coefficients are finally used to obtain the importance score of each feature. The intuition behind this approach is that the leading eigenvectors contain information about the structure of the instance distribution, so the features that better predict the eigenvectors are those that have a stronger capability to maintain this structure [Zhao et al., 2010].

FSDS can be classified as a spectral feature selection algorithm, which comprises methods that define the relevance of features by their ability to preserve the structure of the original data. The main idea is to analyze the consistency of the features with the graph obtained from the similarities between instances. Specifically, the consistency between the features and the spectrum of a matrix obtained from the similarity is measured [Zhao and Liu, 2011]. For example, the SPEC framework uses the eigensystem of its normalized Laplacian matrix [Zhao and Liu, 2007]. The similarity can be computed with different measures, which can include or not include information about a class variable.

FSDS assumes that all the data come from a single data stream. However, data could be generated from multiple sources with distinct feature sets. One example is dividing image information into two views, visual and metadata, a problem known as multi-view learning. The main difference of multi-view learning with respect to the multi-task problem is that, in the former, different views provide complementary information about the same instances while, in the latter, each task has its own instances that share a common feature space. Different datasets have co-occurring $N$ instances in $n_v$ views, $D_v = \{x_i\}_{i=1}^N$, $v = 1, 2, ..., n_v$, where $D_v \in \mathbb{R}^{N \times m_v}$ represents the dataset of the $v$-th view and $m_v$ represents its feature dimension. Views can have high dimensionality; thus, FSS is used to obtain a feature subset from each view based on the information provided by the different views to improve the selection. In addition, multi-view data often include a large number of instances, so this information could be processed in a streamwise manner to avoid memory problems. The online unsupervised multi-view feature selection (OMVFS) algorithm is the first approach that covers the unsupervised multi-view problem from an online perspective [Shao et al.,
Table 7.2: Unsupervised learning algorithms for FSS on data streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Process groups</th>
<th>Instance removal</th>
<th>No instance revisit</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSDS</td>
<td>Uses matrix sketching and regression analysis with regularization to weight features at every time step.</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Huang et al. [2015]</td>
</tr>
<tr>
<td>OMVFS</td>
<td>Performs unsupervised FSS over streaming multi-view data.</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>Shao et al. [2016]</td>
</tr>
</tbody>
</table>

2016]. OMVFS includes FSS into an online nonnegative matrix-based clustering algorithm, processing data chunks in an online manner and aggregating their information into different small matrices without storing all the previous data. To do so, a parameter is used to define the maximum size of the matrices and remove excess old information; i.e., a time window is used to reduce the memory requirements without losing the capability to capture concept drifts. The main objective of OMVFS is to generate a cluster indicator matrix that integrates information from all the views and a feature selection matrix for each of the $n_v$ views, which is updated in each iteration with new and some old data. It is from this feature selection matrix that the importance of each feature in every view is obtained.

A summary of the unsupervised FSS algorithms for data streams is provided in Table 7.2.

7.4 FSS on feature streams

This section focuses on different approaches to perform FSS on feature streams, from algorithms that evaluate new features one by one in one-dimensional tasks to algorithms that evaluate feature groups in multi-label environments. Individual and group FSS for unsupervised environments is also studied. Although, at first glance, it may be thought that the inclusion of new instances is more common, most recent studies have focused on working with feature streams.

The arrival of new features could imply that those features already selected are no longer the most relevant, i.e., weakly relevant, and even redundant. Therefore, it is necessary to adapt the subset of selected features over time to always contain the most relevant and non-redundant features. FSS algorithms for feature streams are useful not only in those scenarios where our set of features is, theoretically, infinite but also in those cases where the generation of features is expensive, so it could be more interesting to include them in our model progressively [Perkins and Theiler, 2003]. A well-known example is the detection of a large number of small craters on the surface of Mars from high-resolution images [Ding et al., 2011]. This analysis involves tracking many texture features that cannot be pre-generated in
a reasonable time; thus, they may be considered as soon as they are obtained.

7.4.1 Individual FSS

In this section, we analyze supervised and unsupervised FSS algorithms that process new features at the individual feature level.

7.4.1.1 Supervised one-dimensional learning

We start by discussing scenarios where data are labeled so it is possible to compute the relevance of the features with respect to one or several class variables. First, the algorithms that work with one-dimensional learning problems and feature streams are introduced.

The feature stream problem, introduced by Perkins and Theiler [2003], considers how to efficiently solve FSS problems in a dynamic environment with new features appearing one by one while training instances remain static. The authors proposed an adaptation of the grafting algorithm [Perkins et al., 2003] for feature streams in Perkins and Theiler [2003], which uses a fast gradient-based heuristic to quickly find which features are most likely to improve an existing model. This technique can be used with models parametrized by a weight vector that is subject to Lasso regularization, so the inclusion of new features will result in the augmentation of the model with new weights. Grafting is presented along with two different models, a linear and non-linear model, ensuring that it combines the speed of filters and the accuracy of wrappers.

Another interesting algorithm to consider is alpha-investing [Zhou et al., 2005], which is an (adaptive) complexity penalty method based on dynamically adapting a threshold \( \alpha^t \) to control which new features are added in the future. Alpha-investing is implemented with a model that is able to take features sequentially and report a p-value that represents the probability that a new feature is included in the model when it should be discarded, i.e., a false positive. The p-value of a feature is associated with a t-statistic, whose value is equivalent to the difference in the log-likelihood of the data given a model with the feature and another without it. Then, a new variable is included in the model if its p-value is less than \( \alpha^t \). This threshold represents the probability of including spurious features at time step \( t \) and is dynamically computed using the wealth of the current iteration \( w^t \):

\[
\alpha^t = \frac{w^t}{2t}.
\]

The wealth indicates the currently acceptable number of false positives. Therefore, if a feature is accepted, the wealth is increased, while it is reduced if the feature is rejected. The
alpha-investing algorithm is based on the information-investing algorithm [Ungar et al., 2005], a similar approach that adds a new feature if it reduces the entropy of the model sufficiently with respect to an adaptive threshold.

As with multi-view learning on data streams, our problem could involve not only one feature stream but multiple of them. In this case, we can use an extension of alpha-investing, the multiple streamwise feature selection (MSFS) algorithm [Dhillon et al., 2010]. MSFS stores the wealth for each of the streams and selects the next feature from the stream with the most permissive threshold since it should be the most beneficial for the model.

Alpha-investing has the advantage over grafting in that it does not need prior information about the global feature set. Grafting requires this information to choose a good regularization parameter, a major limitation for some scenarios since we can be considering, theoretically, infinite streams of features [Wu et al., 2013]. However, both algorithms suffer from the so-called nesting effect, meaning that they cannot discard redundant features that were previously selected [Pudil et al., 1994] since they do not re-evaluate them. To avoid the nesting effect, we can use the online streaming feature selection (OSFS) algorithm [Wu et al., 2013], which is capable of selecting relevant features in real time and removing redundant ones in two major steps. First, an online relevance analysis determines whether a new feature is relevant to the class variable. Second, and the novelty of this approach, online redundancy analysis is performed to identify and remove previously selected features that became redundant due to the inclusion of a new relevant feature. The OSFS algorithm uses conditional independence and dependence tests implemented with the $G^2$ test to determine relevant and redundant features. Thus, the conditional dependency test is computed between the new feature and the class variable to determine if it is relevant. A feature is identified as redundant if there exists a subset from the set of selected features that makes this feature and the class variable conditionally independent. Thus, this process is performed for every selected feature, removing those that become redundant. OSFS uses the Markov blanket criterion to define redundant features, which guarantees that a removed feature will still be considered redundant even after the posterior removal of others. However, it is computationally expensive to check all possible subsets of features every time a new feature is received. Therefore, the authors assume a predefined size for the checked subsets. This would reduce the time complexity but will also affect the results. Depending on the characteristics of the problem being addressed, it may be necessary to compromise between obtaining more reliable results and the efficiency of the algorithm.

Maintaining a good feature subset in a real-time environment demands efficient approaches, and in the case of OSFS, the analysis of redundancies can be time consuming. To accelerate the computation, the redundancy analysis was divided into two parts in a novel algorithm called fast-OSFS [Wu et al., 2013]. The first part of the redundancy analysis only checks whether a new relevant feature $X_t$ is redundant with respect to any other subset of selected
features. In that case, the new feature is discarded, and the algorithm moves directly to the next feature of the stream. Otherwise, fast-OSFS checks whether the inclusion causes other features to become redundant, just as OSFS does. However, in this case, the computational cost of conditional independence tests is reduced since only those subsets from the selected features that include the new feature $X_t$ are considered.

OSFS and fast-OSFS not only avoid the nesting effect but also do not need prior information about the feature set. However, they do have a disadvantage since the use of conditional independence tests entails a need for a large number of instances to obtain reliable results. This is even more important in this environment since the set of features grows over time [Javidi and Eskandari, 2019].

Despite significant improvements in the online FSS research area, Yu et al. [2014] noted that previous algorithms, such as alpha-investing and fast-OSFS, do not perform efficiently enough when applied to data of extremely high dimensionality. This is also the case of rough set-based and fuzzy rough set-based algorithms for feature streams, such as CIE-OSFS or FRSA-IFS-HIS(AA), which are commented on later in this section. Therefore, proposals such as the scalable and accurate online approach for feature selection (SAOLA) focus on the scalability of FSS methods [Yu et al., 2014]. SAOLA reduces the computational cost of identifying non-relevant and redundant features by employing only pairwise comparisons to compute the correlations between attributes, where the mutual information is used as a metric (see Equation (7.1)).

SAOLA considers a new feature $X_t$ relevant to the class variable $C$ if their mutual information $I(X_t, C)$ is greater than a predefined relevance threshold. In that case, SAOLA checks whether the feature introduces redundancy. Thus, if the mutual information between $X_t$ and any other already selected feature $X_j \in S'$ is greater than or equal to $I(X_t, C)$ or $I(X_j, C)$, one of the features contains at least as much information about the class variable as the other. Therefore, the feature with lower relevance (mutual information) to the class variable is removed, i.e., the redundant feature. This approach was proposed by Yu and Liu [2004] as a way to approximate the Markov blankets that still guarantees that the revisiting of discarded features is not necessary.

SAOLA is more efficient than OSFS or fast-OSFS, even if the latter defines a maximum size for the subsets to evaluate. Experiments in Yu et al. [2014] indicate that SAOLA does not appear to significantly improve the prediction accuracy, while fast-OSFS reports smaller feature subsets. However, fast-OSFS could require days to run in datasets of extremely high dimensions, while SAOLA would need only a few seconds or minutes to find a solution. In addition, the use of mutual information has the advantage, with respect to the rough set approaches, of being able to handle discrete and continuous features.

A disadvantage of SAOLA is that it requires a predefined relevance threshold to determine
relevant features, and it is not trivial to define an appropriate value. This problem is avoided with online stream feature selection based on mutual information (OSFSMI) and the online stream feature selection based on mutual information with fixed number of features (OSFSMI-$k$), two algorithms that also employ mutual information to assess the relevance and redundancy of features [Rahmaninia and Moradi, 2018]. Both algorithms start by discarding irrelevant new features, which are those whose mutual information with the class variable is zero. If a new feature is considered relevant, redundant features are detected in $S^t$ by evaluating their effectiveness. The effectiveness of a feature $X_i$ is computed by taking into account its relevance and redundancy with respect to the remaining features $S^t \setminus X_i$:

$$\lambda(X_i, S^t) = I(X_i, C) - \beta \sum_{X_j \in S^t \setminus X_i} \frac{I(X_j, C)}{H(X_j)} I(X_i, X_j),$$

where the parameter $\beta \in (0, 1]$ controls the redundancy penalty. Features with lower effectiveness values than the newly included feature are removed, while the effectiveness values of features that still remain are updated. In the case of OSFSMI, the redundancy analysis stops when no more features of lower effectiveness are found or if only one feature remains. OSFSMI-$k$ returns the best feature subset of a predetermined size $k$. Then, the feature subset $S^{t+1}$ is obtained, and both algorithms wait for the next feature.

OSFSMI and OSFSMI-$k$ appear to give more importance to new features since they cannot be discarded upon arrival if their relevance is greater than zero. A major problem with this behavior is that weakly relevant features would be accepted and possibly discarded in the next iteration, resulting in a waste of resources. Thus, it could be more interesting to include a threshold, as SAOLA does, since expecting the mutual information to be zero is a very strict requirement. Experiments in Rahmaninia and Moradi [2018] show that these algorithms can find feature subsets that provide higher classification accuracy than previous proposals while obtaining similar or better run times. In addition, OSFSMI is more stable than those algorithms that also perform redundancy analysis, i.e., more similar subsets of features are obtained on independent executions and with different feature orders, and smaller feature subsets than those of algorithms such as SAOLA are produced.

Note that hybridizations of previously proposed algorithms can take advantage of their benefits. For example, a new hybrid approach between the wind driven dynamic optimization algorithm (WD2O) [Boulesnane and Meshoul, 2017], a nature-inspired dynamic optimization algorithm, and the previously explained OSFS algorithm was introduced by Boulesnane and Meshoul [2018]. This novel algorithm is the dynamic online streaming feature selection (DOSFS), and the authors demonstrate, through several experiments, that the use of dynamic optimization for the selection of a feature subset significantly improves the accuracy obtained by OSFS for specific problems while taking advantage of the latter’s speed. This improvement is the result of the recovery of the features discarded by OSFS that is performed by WD2O. The final
feature subset is a combination of the subsets found by OSFS and WD2O.

**Rough set-based FSS**  As feature streams maintain a fixed number of instances, the application of rough set-based proposals can be of great interest. However, until recently, FSS was not considered from a rough set perspective, and the dimension incremental algorithm for reduction computation (DIA-RED) was among the first proposals [Wang et al., 2013]. DIA-RED uses the Shannon, combination or complementary entropy to incrementally adapt the reduct without recomputing the entropies for the whole dataset each time a new set of features is received. DIA-RED includes incremental mechanisms that enable us to update the existing entropies based on the changes in the condition and decision classes. It follows a similar strategy to that of GIARC (see Section 7.3.1.1), which considers the use of the same entropies and outer and inner significance measures to determine relevant and redundant features.

DIA-RED considers only the information contained in the positive region. However, this approach could result in defining sets of variables as independent due to noise. Given a set of instances $\mathcal{U}$, the online streaming noise-resistant-aided rough set attribute reduction using significance analysis (OS-NRRSAR-SA) algorithm [Eskandari and Javidi, 2016] defines the reduct using the degree of dependency between attributes $\gamma^\mathcal{U}(\cdot, \cdot)$ (see Equation (7.4)) and a novel noise-resistant dependency measure that additionally evaluates the boundary region:

$$\rho^\mathcal{U}(S^t, C) = \frac{\tau^\mathcal{U}(S^t, C) + \gamma^\mathcal{U}(S^t, C)}{2}. \quad (7.7)$$

The measure $\tau^\mathcal{U}(\cdot, \cdot)$ includes information about the proximity of the boundary and positive region and the possibility of transferring instances from one to another by removing noisy instances.

If the degree of dependency between $C$ and the selected features $S^t$ is less than one, i.e., the dataset is inconsistent using $S^t$, OS-NRRSAR-SA includes a new feature $X_t$ in $S^t$ as long as (1) its outer significance is greater than zero or (2) the noise-resistant dependency of $C$ on $X_t$ is nonzero. The outer and inner significances of features are defined in Equations (7.2) and (7.3), employing the degree of dependency as a measure and normalizing the difference in the case of the inner significance. If the degree of dependency is one, i.e., the dataset is consistent using $S^t$, the new feature $X_t$ is not simply ignored since it can replace a subset of $S^t$ that becomes redundant due to its inclusion. This subset is defined based on the inner significance, and if its size is greater than one, it is replaced by $X_t$, making the reduct more compact while keeping the dataset consistent. In the case that the subset is a unit set, the feature with the highest noise-resistant dependency is retained. The number of subsets to evaluate grows exponentially with the size of $S^t$ when looking for the largest redundant one. Therefore, sequential backward elimination is also proposed to obtain a more efficient solution.
However, the removal order that is employed affects the result.

Experiments in Eskandari and Javidi [2016] compare the performance of OS-NRRSAR-SA, grafting, information-investing, fast-OSFS and DIA-RED, with OS-NRRSAR-SA showing superiority in terms of the compactness of the selected subsets, computational time and classification accuracy achieved with the selected feature subsets.

OS-NRRSAR-SA was later extended [Javidi and Eskandari, 2019] because the original approach is not efficient enough when the number of selected features is not small. This is because it extracts equivalence classes considering all the currently selected features, which is a computationally expensive task. To make this approach slightly more scalable, the proposed extension seeks to obtain more compact results by including a new filter method that removes redundancies before the significance analysis. As this new step may be computationally expensive, a user-defined parameter is proposed to define the maximum size of the redundant subsets. The extension was compared against the original OS-NRRSAR-SA, and some improvements were detected in terms of compactness, run time and classification accuracy. Although the extension adds a new step to OS-NRRSAR-SA, the run time was reduced for certain datasets because of the more compact results.

As a last note about OS-NRRSAR-SA, the same strategy was used in another approach called SFS-RS [Javidi and Eskandari, 2018], whose only difference is that it was conceived with six different measures of dependency, including the noise-resistant measure of Equation (7.7). This work was useful to determine the effects of using different measures.

Other approaches use the knowledge granularity to incrementally update the reduct. This is the case of Jing et al. [2016], where matrix-based and non-matrix-based algorithms called the matrix-based incremental reduction algorithm (MIRA) and incremental algorithm for reduct computation (IARC), respectively, are proposed. This research seeks not only to analyze the benefits of incremental algorithms with respect to batch versions but also to prove the inefficiency of matrix-based algorithms for large datasets. MIRA and IARC use the same strategy: first, given a new group of features $F^t$, they check that the new granularity $\text{GK}^d(C|\mathcal{X} \cup F^t)$ (see Equations (7.5) and (7.6)) is still the same as the granularity of the current reduct $\text{GK}^d(C|S^t)$. If this is not the case, the features with the highest outer significance from $\{\mathcal{X} \cup F^t \setminus S^t\}$ are included in the reduct until the granularities are equalized. The final step is to remove features from the reduct as long as their removal does not alter the previous equality, i.e., redundant features, to obtain the new reduct $S^{t+1}$.

The only difference between MIRA and IARC is the method used to calculate the knowledge granularity. MIRA makes use of relation matrices, which may not be efficient enough for large datasets due to the memory and computational time needed. A more efficient method is presented in IARC, which computes the knowledge granularity with a non-matrix-based approach using the new partitions that occur in the dataset with the inclusion of new features.
As features are unknown until they are received, the order of their arrival may affect the obtained feature subsets. The online streaming feature selection algorithm based on conditional information entropy (CIE-OSFS) considers this problem and is robust to changes in the order because of the implementation of a sorting mechanism [Wang et al., 2017]. CIE-OSFS divides its strategy into two phases. First, an independence test checks whether a new feature is relevant to the class variable. In the second phase, redundancies are removed by searching for a reduct in the set of relevant features. During the redundancy analysis, features are sorted by their correlation coefficient against the class variable, so CIE-OSFS is more stable to changes in the feature arrival order. The new reduct $S^{t+1}$ is generated by computing the outer significance of each relevant feature, which is calculated using Equation (7.2), where $M(\cdot, \cdot)$ is the Shannon conditional entropy. The features are considered not redundant only if their outer significance in the current reduct is greater than zero.

**FSS for rough set extensions** As noted in Section 7.3.1.2, some extensions of classical rough set theory were proposed to overcome its problems. This includes its inefficiency in handling continuous features, which can be solved using fuzzy rough sets. In the case of feature streams, Zeng et al. [2015] proposed two fuzzy rough set approaches for incremental FSS on hybrid information systems (HIS), i.e., datasets that include different types of attributes, such as binary, continuous, categorical and set-valued, a common scenario in real-world problems. These novel algorithms are known as the fuzzy rough sets approach for incremental feature selection in HIS under one attribute being added (FRSA-IFS-HIS(AA)) and under one attribute being deleted (FRSA-IFS-HIS(AD)). Both algorithms use a novel hybrid distance function $HD(\cdot, \cdot)$ to compute the distance between instances with hybrid and incomplete variables. This distance is based on the Euclidean distance, but it uses five different equations to compute the value difference of each type of variable. Given two instances $x_i$ and $x_j$, their distance is computed as follows:

$$HD(x_i, x_j) = \sqrt{\sum_{r=1}^{m} vd(x_{ir}, x_{jr})^2},$$

(7.8)

where $vd(\cdot, \cdot)$ computes the value difference depending on the value type of $x_{ir}$ and $x_{jr}$.

FRSA-IFS-HIS(AA) is based on the positive region, and it uses the fuzzy dependency to define relevant and redundant features. This algorithm takes into account the feature order in the reduct, so those with a higher degree of dependency (with the class variable) are located in the first positions. This is done to modify only those elements of $S^t$ that are in a lower position than the new feature $X_t$ since the lower the dependency value is, the lower the importance. When $X_t$ is received, FRSA-IFS-HIS(AA) first incrementally computes its fuzzy relations, which are obtained by using the Gaussian kernel function with the hybrid distance of Equation (7.8). These relations are then used to obtain fuzzy positive regions necessary
to compute the dependencies between $C$ and the feature subsets (see Equation (7.4)). If $X_t$ increases the dependency for a certain subset of $S^t$, it will be included in $S^{t+1}$, and the remaining features can be added from its position in the reduct. These features are added in descending order of dependency value as long as they increase the dependency of the current reduct by more than a certain threshold; otherwise, they are considered redundant.

FRSA-IFS-HIS(AD) updates the reduct after the elimination of a feature $X_{del} \in S^t$. Then, following the same approach as FRSA-IFS-HIS(AA), it merely removes $X_{del}$ and adds the remaining features from its position in the reduct to obtain $S^{t+1}$. Previously, only unnecessary features were removed by the algorithms. However, this work introduces the possibility of removing features due to an external decision, even if they were considered to be relevant. We think this possibility may be useful, for example, in the case that a feature was detected as inappropriately collected. Thus, the feature can be eliminated and the reduct incrementally updated. Notably, FRSA-IFS-HIS(AD) (as well as FRSA-IFS-HIS(AA)) stores all the variables that were available at some point, which, for memory reasons, is not good practice when working with the incremental addition of new features. Therefore, how to remove more efficiently relevant features while recovering those that were redundant due to the features being removed remains an open issue.

The neighborhood rough set is another extension of the classical rough set theory commonly used to handle continuous features without requiring a discretization [Hu et al., 2008]. This is achieved by using a neighborhood relation to define the lower and upper approximations rather than equivalence classes that group instances with the same values for the studied attributes. Traditionally, this neighborhood relation is based on the use of a distance function to find for each instance the $k$ nearest neighbor instances ($k$-nearest neighborhood relation) or all those within a given $\delta$ distance ($\delta$ neighborhood relation) [Zhou et al., 2019b]. The K-OFSD [Zhou et al., 2017] is an example of a neighborhood rough set-based algorithm, which uses the $k$-nearest neighborhood relation to perform an online FSS that improves the separability between majority and minority classes of imbalanced datasets.

The main drawback of K-OFSD is that it requires the hyperparameter $k$ to be tuned beforehand. An inconvenience that also occurs with the $\delta$ hyperparameter of the $\delta$ neighborhood relation. Therefore, with the intention of avoiding this problem, Zhou et al. [2019b] introduced a gap neighborhood relation, which automatically selects the number of neighbors considered based on the distribution of the surrounding instances. Basically, it places the neighbors of an instance in ascending order of distance and defines a cutoff when there is a certain “gap” (distance value) between two consecutive instances. This neighborhood relation was devised to be used in the novel OFS-A3M algorithm, which performs online FSS by analyzing the degree of dependency of a new feature on the class variable (see Equation (7.4)) and the effect on the dependency of the currently selected feature subset when this new feature is included.
Subsequently, the above authors proposed the OFS-Density algorithm [Zhou et al., 2019a] with the main objective of including a more aggressive removal of redundant features than OFS-A3M. Instead of only performing a redundancy analysis when a new relevant feature does not improve the feature subset dependence degree (a rare occurrence in a real scenario), OFS-Density considers this task when the improvement is smaller than a given threshold. This can certainly increase the number of features detected as redundant, but it also implies setting a new hyperparameter. In addition, OFS-Density introduces a density neighborhood relation that automatically selects the number of neighbors. Similar to the gap neighborhood relation, this new relation sorts the neighbors in ascending order of distance and sets a cutoff not only taking into account the distances but also the number of instances already selected as neighbors. According to the experiments conducted by Zhou et al. [2019a], OFS-Density selects feature subsets that are significantly smaller than those of OFS-A3M, and classifiers trained with those subsets report, in general, higher accuracy. However, this comes at the cost of significantly slower performance, a handicap in certain online scenarios.

7.4.1.2 Supervised multi-label learning

Most state-of-the-art algorithms were designed for one-dimensional supervised learning problems. However, multi-dimensional learning has multiple applications in real-world problems, such as document analysis [Zhu et al., 2005], bioinformatics [Elisseeff and Weston, 2001] and music [Trohidis et al., 2011]. Any information can be classified into several class variables simultaneously. Thus, it is necessary to consider online FSS algorithms for this environment. A simple solution would be to use a transformation approach in which independent one-dimensional algorithms are used for each class variable. However, this approach would ignore the possible relations between class variables, information that could be important to find the best subset of features. Multi-label learning over feature streams brings several new challenges to overcome, such as label correlation, high dimensionality, class imbalance and label-specific features [Lin et al., 2017]. Note that the multi-label problem is a subtype of the multi-dimensional problem, where all the class variables are binary.

To the best of our knowledge, Lin et al. [2017] proposed the first online FSS solution for multi-label problems, the multi-label streaming feature selection algorithm, which we refer to as ML-SFS. This algorithm focuses on the label correlation and high-dimensionality problems in an online scenario and introduces the fuzzy mutual information $\text{FI}(\cdot, \cdot)$ as the evaluation criterion to define relevant and redundant features in a multi-label context. Correlations between class variables and features are stored in two separate similarity matrices, which include all influences of one variable on the others. The fuzzy mutual information is computed using these similarity matrices, so the mutual dependence between a feature $X_t$ and the label
space $C$ is defined as follows:

$$FI(X_t, C) = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{||x_i||_{X_t} \cdot ||x_i||_{C}}{N \cdot ||x_i||_{X_t \cap C}}.$$ 

where $[x_i]_{X_t}$ is the fuzzy equivalence class of $x_i$ under $X_t$, which is determined with the similarity degrees between $x_i$ and every instance under feature space $X_t$. Therefore, given a predefined relevance threshold $\delta$, a new feature $X_t$ is classified as relevant if $FI(X_t, C) > \delta$. If this is the case, ML-SFS removes possible redundancies caused by the inclusion of $X_t$ in $S^t$, considering a feature $X_i \in S^t$ as redundant if its fuzzy mutual information is lower, i.e., $FI(X_t, C) > FI(X_i, C)$.

A different approach to address multi-label problems is given in the multi-label online streaming feature selection algorithm based on spectral granulation and mutual information (ML-OSMI) [Wang et al., 2018], which seeks to capture the correlations among labels by transforming the label set $C = \{C_1, ..., C_d\}$ into a new set of multi-class variables $C' = \{C'_1, \ldots, C'_h\}$ with lower dimensionality, i.e., $h < d$. This step is performed only once before the relevance and redundancy analysis since the label set is assumed to be fixed. Labels are first clustered using spectral clustering with cosine similarity. Thus, each cluster is formed by labels with a high correlation. Then, each cluster is transformed into a multi-class variable by applying the LP framework. After the label transformation, when ML-OSMI receives a new feature $X_t$, its relevance against each of the created class variables $C'_i \in C'$ is assessed using the normalized mutual information for continuous variables:

$$nI(X_t, C'_i) = \frac{2 \int \int P(X_t, C'_i) \log \frac{P(X_t, C'_i)}{P(X_t)P(C'_i)} dX_t dC'_i}{-\int P(X_t) \log P(X_t) dX_t - \int P(C'_i) \log P(C'_i) dC'_i}.$$ 

If the normalized mutual information of $X_t$ and any class variable $C'_i$ is greater than a predefined threshold, then $X_t$ is considered relevant and included in $S^t$. In such a case, its inclusion can produce redundancy. A feature $X_i \in S^t$ is redundant if its significance on any class variable $C'_j$ given another feature $X_j$ is zero. This significance is defined as the maximum conditional mutual information between $X_i$ and each class variable in $C'$ given $X_j$.

The LP framework is useful to reduce the problem dimensionality or even to apply one-dimensional algorithms to multi-dimensional datasets. However, LP-based methods can suffer from class-imbalance problems if the number of labels is large [Wang et al., 2018] and they imply working with multi-class variables with high cardinality.

**Rough set-based FSS** Rough set-based FSS algorithms designed specifically to address multi-label feature streams are also available. This is the case for the online multi-label streaming feature selection based on neighborhood rough set (OM-NRS) algorithm [Liu et al.,
Carlos Villa Blanco

2018a], which has the advantage of supporting continuous features since it is based on using the neighborhood rough set theory to define positive regions. OM-NRS starts by measuring the outer significance of a new feature $X_t$ in $S^t$ relative to the label set $C$ to determine its relevance. To do so, Equation (7.2) is used with a multi-label adaptation of the degree of dependency (see Equation (7.4)), which is defined as:

$$\gamma^U(S^t, C) = \frac{\sum_{k=1}^d |POS_{S^t}^{l}(C_k)|}{d \cdot |U|}.$$ 

$X_t$ is included in $S^{t+1}$ if its outer significance is greater than zero. Otherwise, online redundancy analysis is performed since there exists at least one feature in $S^t$ that is redundant with $X_t$. In this case, and to reduce the number of calculations, it is first checked whether $\gamma^U(X_t, C)$ is less than a predefined relevance threshold, rejecting $X_t$ in such cases since it is considered weakly relevant. If this does not hold, OM-NRS has to evaluate $X_t$ against every feature $X_i \in S^t$, discarding the feature with a lower degree of dependency with $C$ as long as $\text{sig}^\text{outer}(X_t, S^t, C) = 0$. This last step continues until all $X_i$ are evaluated or $X_t$ is discarded.

Table 7.3 summarizes the discussed algorithms.

### 7.4.1.3 Unsupervised learning

FSS algorithms can also be applied over feature streams with no classes. A typical example is social media data, where features are usually generated dynamically and it is expensive to collect label information [Li et al., 2015].

The unsupervised streaming feature selection (USFS) framework is one of the first FSS algorithms for feature streams to handle unsupervised data [Li et al., 2015]. USFS performs FSS through a regression model using link information in social media. Therefore, it works on datasets with $N$ linked data instances, whose link information is encoded into a matrix $M \in \mathbb{R}^{N \times N}$ such that if $M_{ij} = 1$, instances $x_i$ and $x_j$ are linked. USFS extracts a predefined number $s$ of social latent factors from $M$ to which each instance is associated with a certain probability. This information is included for each instance in a vector $\pi_i \in \mathbb{R}^s$, so $\Pi = [\pi_1, ..., \pi_N]$. For example, a social latent factor could be hobbies or job positions shared between people. These factors should have a relationship with certain features; for example, age could be related to hobbies, so they are used to decide if a new feature $X_t$ should be included in the model. In some sense, the missing information about class variables is replaced with these social latent factors, so relevant features are defined by their ability to differentiate them. Then, FSS is performed through several regression models with regularization, where the social latent factors are the dependent variables, i.e., one regression model for each social latent factor, so the obtained regression coefficients at time step $t$, $W^t \in \mathbb{R}^{t \times k}$, represent the
## Table 7.3: Supervised learning algorithms for individual FSS on feature streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class variable</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Grafting</strong></td>
<td>Uses a fast gradient-based heuristic to find the most likely feature to improve a model.</td>
<td>Depends*</td>
<td>Perkins and Theiler [2003]</td>
</tr>
<tr>
<td><strong>Alpha-investing</strong></td>
<td>Uses the p-value of new features to determine their inclusion in a model.</td>
<td>Depends*</td>
<td>Zhou et al. [2005]</td>
</tr>
<tr>
<td><strong>Information-investing</strong></td>
<td>Adds new features to a model if they reduce the entropy sufficiently with respect to a threshold.</td>
<td>Depends*</td>
<td>Ungar et al. [2005]</td>
</tr>
<tr>
<td><strong>MSFS</strong></td>
<td>Extension of alpha-investing that works simultaneously with multiple feature streams.</td>
<td>Depends*</td>
<td>Dhillon et al. [2010]</td>
</tr>
<tr>
<td><strong>(fast-)OSFS</strong></td>
<td>Identifies relevant and redundant features by using conditional independence and dependence tests.</td>
<td>Multi-class</td>
<td>Wu et al. [2013]</td>
</tr>
<tr>
<td><strong>SAOLA</strong></td>
<td>Reduces the computational cost of identifying redundant features by computing correlations via pairwise comparisons with mutual information.</td>
<td>Multi-class</td>
<td>Yu et al. [2014]</td>
</tr>
<tr>
<td><strong>OSFSMI(-k)</strong></td>
<td>Uses mutual information to discard new features and compute their effectiveness.</td>
<td>Multi-class</td>
<td>Rahmaninia and Moradi [2018]</td>
</tr>
<tr>
<td><strong>DOSFS</strong></td>
<td>Hybridization of the WD2O dynamic optimization algorithm and OSFS.</td>
<td>Multi-class</td>
<td>Boulesnane and Meshoul [2018]</td>
</tr>
<tr>
<td><strong>ML-SFS</strong></td>
<td>Performs online relevance and redundancy analysis considering the correlations between labels.</td>
<td>Multi-label</td>
<td>Lin et al. [2017]</td>
</tr>
<tr>
<td><strong>ML-OSMI</strong></td>
<td>Captures correlations among labels by transforming them into a set with a lower dimensionality.</td>
<td>Multi-label</td>
<td>Wang et al. [2018]</td>
</tr>
</tbody>
</table>

### Rough set-based

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class variable</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DIA-RED</strong></td>
<td>Incremental attribute reduction algorithm proposed to be used with three different entropy measures.</td>
<td>Multi-class</td>
<td>Wang et al. [2013]</td>
</tr>
<tr>
<td><strong>OS-NR3SAR-SA and extensions</strong></td>
<td>Uses the degree of dependency with different dependency measures.</td>
<td>Multi-class</td>
<td>Eskandari and Javidi [2016]; Javidi and Eskandari [2018, 2019]</td>
</tr>
<tr>
<td><strong>MIRA and IARC</strong></td>
<td>Matrix and non-matrix based attribute reduction algorithms that make use of knowledge granularity.</td>
<td>Multi-class</td>
<td>Jing et al. [2016]</td>
</tr>
<tr>
<td><strong>CIE-OSFS</strong></td>
<td>Defines reduct using conditional information entropy and independence test. It is robust to changes in feature arrival.</td>
<td>Multi-class</td>
<td>Wang et al. [2017]</td>
</tr>
<tr>
<td><strong>FRSA-IFS-HIS( AA)/(AD)</strong></td>
<td>Fuzzy rough set-based algorithms for hybrid datasets. FRSA-IFS-HIS(AD) considers the removal of features from the dataset.</td>
<td>Multi-class</td>
<td>Zeng et al. [2015]</td>
</tr>
<tr>
<td><strong>K-OFSD</strong></td>
<td>Neighborhood rough set-based algorithm using the k-nearest neighborhood relation.</td>
<td>Multi-class</td>
<td>Zhou et al. [2017]</td>
</tr>
<tr>
<td><strong>OFS-A3M</strong></td>
<td>Neighborhood rough set-based algorithm using a new gap neighborhood relation.</td>
<td>Multi-class</td>
<td>Zhou et al. [2019b]</td>
</tr>
<tr>
<td><strong>OFS-Density</strong></td>
<td>Neighborhood rough set-based algorithm using a new density neighborhood relation.</td>
<td>Multi-class</td>
<td>Zhou et al. [2019a]</td>
</tr>
<tr>
<td><strong>OM-NRS</strong></td>
<td>Based on using neighborhood rough set theory to compute the positive region and a multi-label version of the degree of dependency.</td>
<td>Multi-label</td>
<td>Liu et al. [2018a]</td>
</tr>
</tbody>
</table>

* Different models can be used with the algorithm.
importance of every feature on each social latent factor. Thus, USFS performs FSS by solving at time $t$ the minimization problems for each social latent factor $\pi_i$ ($i$-th column of $\Pi$):

$$
\min_{w_{\cdot i}} J(w_{\cdot i}^t) = \frac{1}{2} ||X^t w_{\cdot i}^t - \pi_i||^2_2 + \alpha ||w_{\cdot i}^t||_1
$$

$$
+ \frac{\beta}{2} ||w_{\cdot i}^t||^2_2 + \frac{\gamma}{2} ||(X^t w_{\cdot i}^t)^T(L^t)\frac{1}{2}||^2_2,
$$

(7.9)

where $X^t \in \mathbb{R}^{N \times t}$ is the available data, $L^t$ is a Laplacian matrix extracted from a similarity graph of the instances, and $\alpha$, $\beta$ and $\gamma$ are parameters to control the penalty terms of Lasso regularization (second term) and elastic net regularization (third term) and to balance the link information and feature information (fourth term), respectively.

To define a new feature $X_t$ as relevant or not, USFS first computes for each social latent factor $\pi_i$ the derivative of Equation (7.9) with respect to $w_{\cdot i}^{t+1}$, which is an augmentation of $w_{\cdot i}^t$ with a nonzero value for $X_t$. If the derivative of $\pi_i$ is greater than the parameter $\alpha$, the inclusion of $X_t$ helps to reduce the objective function in Equation (7.9) when predicting $\pi_i$. Thus, $X_t$ is included in the model, and Equation (7.9) is optimized with the current weights using the Broyden-Fletcher-Goldfarb-Shanno algorithm. This approach may cause some regression coefficients to shrink to zero; therefore, a feature is not included in $S^{t+1}$ iff $w_{\cdot i}^{t+1}$ is an empty vector.

A disadvantage of USFS is that it does not consider the update of the link information $M$. The authors justify this strategy by noting that this information does not change as often as new features are received. However, this may not be the case for some applications.

The USFS framework has two other limitations: it is valid only for those problems where link information can be established and the dataset grows with the inclusion of new features, i.e., features are not totally discarded, but their weights are set to zero, which could lead to memory problems. The unsupervised feature selection for streaming features (UFSSF) algorithm avoids these issues by extending the k-means algorithm to include linearly dependent similarity measures and to work with streaming features [Almusallam et al., 2018]. UFSSF clusters incoming features and selects a representative feature for each cluster. Therefore, the subset of features in one cluster is approximated by only one feature, and the original set of features is reduced to equal the number of clusters. A new feature $X_t$ is assigned to the cluster with which it has the highest similarity to the centroid, and it is selected as the representative feature of that cluster at time $t + 1$ if its similarity to the centroid (updated with the information of $X_t$) is higher than that of the representative feature at time $t$. UFSSF was designed to be used with three similarity measures, the Pearson correlation coefficient, the least squares regression error and the maximal information compression index, which have the advantage of not being sensitive to the feature order and the scatter of feature distributions.


Table 7.4: Unsupervised learning algorithms for individual FSS on feature streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>USFS</td>
<td>Selects the most informative features for social media feature streams by making use of the link information of the data.</td>
<td>Li et al. [2015]</td>
</tr>
<tr>
<td>UFSSF</td>
<td>Representative features of each cluster, generated with an extension of the k-means algorithm, are selected as the most informative features.</td>
<td>Almusallam et al. [2018]</td>
</tr>
</tbody>
</table>

UFSSF incrementally computes the centroids as a weighted mean of the features that were assigned to their clusters, giving more importance to those features that were received more recently. This algorithm is, therefore, indicated for problems where the most recent features are more relevant. Other inconveniences are that the number of clusters is expected to be provided and, therefore, the size of the subset of selected features must be predefined, and that the solution is highly dependent on the initial clusters. Note that UFSSF could be an efficient solution in a streaming scenario, but it simply reports a feature subset using linear dependent measures.

Table 7.4 summarizes all the unsupervised individual FSS algorithms for feature streams that were analyzed in this section.

7.4.2 Group FSS

Features may exhibit group structures; however, previous algorithms evaluate them individually. In this section, proposals that exploit this information for supervised and unsupervised learning are reviewed. Group FSS chooses feature groups that are relevant, in the case of supervised learning, to class variables, removing irrelevant features and avoiding redundancies both at the individual and group feature levels. There are several real-world problems where it is preferable to evaluate features in a group manner. For example, in image analysis, there are certain properties of images, such as color information, that are described by a set of different features [Wang et al., 2015a]. Therefore, group FSS should not be confused with the simultaneous addition of new features that is performed by algorithms such as DIA-RED or MIRA (see Section (7.4.1.1)). These latter proposals do not consider the information provided by the group structures in which the features are received.

7.4.2.1 Supervised one-dimensional learning

The online group feature selection (OGFS) algorithm may be the first approach that takes into account the group structure to perform online FSS [Wang et al., 2015a]. This method is based on performing two phases, the intra-group and the inter-group selection. In the intra-group selection, each feature of an incoming group $\mathcal{F}^t$ is processed individually to select
a subset $\tilde{F}^t$ by using spectral feature selection. Traditional spectral feature selection was adapted for the online environment, introducing a criterion that has to be satisfied to select a feature from an incoming group. This criterion seeks to include new features that increment the between-class while reducing the within-class distances between instances. This first phase evaluates the features individually. Thus, the inter-group selection phase defines an optimal subset $\hat{F}^t \subseteq \tilde{F}^t$ that reduces the redundancies between features of different groups. This step is performed by using linear regression with Lasso regularization, which results in setting to zero the coefficients of features considered redundant.

Another algorithm to consider is group feature selection with streaming features (GFSSF) [Li et al., 2013], which, in this case, uses mutual information to select and discard features or groups. When a new group of features arrives, a feature-level selection step uses the mutual information between the features in this group to select a subset of relevant ones, while redundancies are discarded. Once this subset is defined, a group-level selection phase is performed to evaluate the subset against already selected groups. This step determines whether the information provided by the subset is more than the penalty and if previously included groups should be removed. This penalty is defined based on the number of features, so large groups that provide less information for the class variable are more likely to be discarded. Unlike OGFS, GFSSF replaces or removes feature groups completely, i.e., once a group is included, its features are only discarded if the complete group is eliminated. As GFSSF uses conditional mutual information to define redundancies, this approach is useful for reducing the number of comparisons to be made. However, this could lead to losing important features in a deleted group.

Finally, there are also adaptations of algorithms that evaluate features individually so that they can handle incoming feature groups. This is the case of group-SAOLA [Yu et al., 2016b], which extends the SAOLA algorithm. Group-SAOLA checks the relevance of a new feature group by analyzing the mutual information between each of its features and the class variable. If the mutual information for all the new features is less than or equal to a predefined threshold, the group of features is discarded. Otherwise, the redundancies between features within the group and between this group and previous ones are removed. Group-SAOLA eliminates existing redundancies between features of different groups, so the groups are not eliminated unless all their features are redundant. Nevertheless, this algorithm uses a more scalable solution than GFSSF since it performs pairwise comparisons of the features instead of computing conditional mutual information. Regarding the OGFS algorithm, the experiments performed in Yu et al. [2016b] achieve similar or, in most cases, higher prediction accuracies using the solutions proposed by group-SAOLA, which always produces smaller feature subsets. Additionally, group-SAOLA is more efficient than OGFS on datasets of extremely high dimensionality.
Table 7.5: Supervised learning algorithms for group FSS on feature streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class variable</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGFS</td>
<td>Uses spectral feature selection to perform intra-group selection and Lasso regression to select a final subset among different groups.</td>
<td>Multi-class</td>
<td>Wang et al. [2015a]</td>
</tr>
<tr>
<td>GFSSF</td>
<td>Performs FSS at individual and group feature levels by using mutual and conditional mutual information.</td>
<td>Multi-class</td>
<td>Li et al. [2013]</td>
</tr>
<tr>
<td>group-SAOLA</td>
<td>Adaptation of SAOLA that is able to evaluate incoming groups of features.</td>
<td>Multi-class</td>
<td>Yu et al. [2016b]</td>
</tr>
<tr>
<td>OMGFS</td>
<td>Allows online group FSS over multi-label feature streams.</td>
<td>Multi-label</td>
<td>Liu et al. [2018b]</td>
</tr>
</tbody>
</table>

7.4.2.2 Supervised multi-label learning

The aforementioned group FSS algorithms were created for one-dimensional problems. In the multi-label setting, we can use algorithms such as the online multi-label group feature selection (OMGFS) [Liu et al., 2018b]. OMGFS starts by performing an online group selection when a new feature group $F_t$ is received, where $F_t$ is classified as relevant or not by comparing its correlation with the label set $C$ and a predefined threshold $\delta$. This correlation is computed using a normalized multi-label neighborhood mutual information, which takes into account the neighborhood of each instance induced by the variables under comparison. If the correlation of $F_t$ with $C$ is larger than $\delta$, then $F_t$ is added to a buffer pool. This buffer pool is increased with new strongly and weakly relevant groups until it reaches its predefined maximum capacity. At that moment, an inter-group selection detects the most relevant features and redundancies by defining interaction weights between features from the different selected groups. Each interaction weight represents the relationship of two features with respect to the class variable in such a way that the smaller the weight is, the more information the features provide about the class variables separately than together. The feature with the highest relevance, i.e., multi-label neighborhood mutual information, with $C$ is used to compute the interaction weight with the rest of the features. This feature and those with interaction weights larger than zero are included in the final subset.

Table 7.5 summarizes the supervised group FSS algorithms on feature streams.

7.4.2.3 Unsupervised learning

We did not find any interesting method that performs group FSS in an unsupervised feature stream context. Nevertheless, it could be possible to adapt the strategies followed by algorithms such as FSDS (see Section 7.3.2), OGFS or Group-SAOLA (see Section 7.4.2.1) to the unsupervised scenario. For example, spectral feature selection with an unsupervised measure and regularized regression could be used in an intra-group phase to select the most
relevant features from an incoming group. Then, the feature redundancy among groups could be reduced by means of information theory measures. This idea would require further research to determine for which real-world problems it could be useful.

7.5 FSS on data and feature streams

Several approaches that allow incremental FSS, either for data streams or feature streams, have been discussed. However, the appearance of new instances and features can occur simultaneously in real-world problems. An example is text clustering, where the number of documents (instances) and vocabulary (features) could evolve over time [Zhang et al., 2015]. This is, of course, a more complex situation where the algorithm should adapt the feature selection depending not only on the available features at a certain moment but also on the evolution of their values given newly received instances.

To the best of our knowledge, no attention was given to this topic until the proposal of the sparse trapezoidal streaming data learning (STSD) algorithm [Zhang et al., 2015]. STSD dynamically maintains a linear model for binary classification, whose weights are updated based on whether a misclassification is produced, and includes an embedded FSS by projection and truncation. This algorithm is based on the strategy followed by the OFS algorithm (see Section 7.3.1), but in this case, the feature dimension of an incoming training instance is larger than or equal to that of the current classifier. Given a new instance \( z_t = (x_t, c_t) \in \mathbb{R}^{m_t} \), where \( c_t \in \{-1, +1\} \) and \( m_t \) is the size of the feature space at time \( t \), and a weight vector \( w_t \in \mathbb{R}^{m_t-1} \), STSD predicts the label of \( x_t \) using only the values \( \tilde{x}_t \) of the features that were considered in the current classifier, i.e., those features that have nonzero weight in \( w_t \). Therefore, \( x_t = (\tilde{x}_t, \hat{x}_t) \), where \( \hat{x}_t \) is the values of features with zero weights. This approach will result in a loss \( l_t \), computed with the hinge loss function, which will be used to update the weights according to the following rule:

\[
w_{t+1} = (w_t + \tau^t c^t \tilde{x}_t, \tau^t c^t \hat{x}_t),
\]

where \( \tau^t = l_t / \|x_t\|^2 \). If the instance is correctly classified, i.e., \( l_t = 0 \), then STSD considers that the new features are not currently necessary and that the weights should not be modified. We can see in Equation (7.10) that \( w_{t+1} \) would be an augmentation of \( w_t \) with a weight vector \( \tilde{w}_t = \tau^t c^t \tilde{x}_t \in \mathbb{R}^{m_t-m_t-1} \), i.e., \( w_{t+1} = (w_t, \tilde{w}_t) \), which is empty. Notably, the authors also proposed two variants of STSD that modify the computation of \( \tau_t \) to make the classifier less sensitive to noise. Once the weights are updated, redundant features are removed by selecting a predefined proportion \( b \) of features. This process is implemented by projecting the weights to an \( L_1 \)-ball and then truncating to zero those weights whose values are not in the highest \( b \) percent.
Table 7.6: Supervised learning algorithms for FSS on data and feature streams.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Class variable</th>
<th>No instance revisit</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSD</td>
<td>Maintains a linear classifier, which is updated with never-seen features depending on the loss of a new training instance.</td>
<td>Single-label</td>
<td>✓</td>
<td>Zhang et al. [2015]</td>
</tr>
<tr>
<td>Rough set-based</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAMRCD and IARCD</td>
<td>Matrix and non-matrix rough set-based FSS proposals that incrementally update the knowledge granularity when new data and features are received simultaneously.</td>
<td>Multi-class</td>
<td>✗</td>
<td>Jing et al. [2018]</td>
</tr>
</tbody>
</table>

Another attempt to perform FSS in this environment is found in Jing et al. [2018], where the incremental algorithm based on matrix reduction computation for dynamic data mining (IAMRCD) and the incremental reduction algorithm for reduct calculation of dynamic data set (IARCD) are proposed. These rough set-based proposals are able to generate reducts when features and instances vary simultaneously, with the main difference being that IAMRCD is a matrix-based approach, while IARCD is a non-matrix method. This research appears to be based on Jing et al. [2016], who compared the performance of matrix- and non-matrix-based algorithms under feature streams. As already proven in that article, matrix-based approaches for incremental feature reduction can be useful when processing different types of data. However, they are only good at handling small datasets since the run time is really affected by their size. Thus, the IARCD algorithm, whose run times are shorter in the performed experiments, was also proposed. The authors follow the same strategy as that of the MIRA and IARC algorithms (see Section 7.4.1.1), with the difference that the equations used to calculate the incremental knowledge granularity are adapted to accept variation of both features and instances.

Note that the aforementioned algorithms were designed for supervised classification problems that do not consider any group structure when several new features are received. A summary of them is given in Table 7.6.

7.6 Open issues

FSS for streams of data and/or features is a relatively new branch of study, and the continuous growth of information requires constant improvement. This section discusses some possible directions for future work.

- Simplicity of data stream FSS algorithms. The discussed online algorithms for data streams seek a simple solution to be efficient for real-time problems. However, this can make them of interest for only a small set of problems. Therefore, more research is
expected in this area.

- **Semi-supervised learning and infinitely delayed labels.** Several supervised and unsupervised algorithms were discussed. However, no proposal that focuses on semi-supervised learning was identified. One solution could be to use spectral feature selection in an incremental manner, where the similarity measure between instances considers that class information may or may not exist in some pairwise comparisons. Furthermore, it could be useful to perform FSS on data streams with infinitely delayed labels, i.e., supervised problems where the availability of class variables is delayed. This context was already considered for the classification of data streams [Souza et al., 2015a,b]. Unsupervised approaches could be used to perform FSS until the class variable is available, at which point a supervised method could improve the current feature subset.

- **Other multi-task settings.** Incremental FSS algorithms could be extended to other multi-task settings not covered by DA-MTFS. For example, there are problems where there is no common set of features due to outlier tasks [Gong et al., 2012].

- **Past instance removal in rough set-based algorithms.** The majority of works based on rough set theory do not implement a mechanism for the removal of past instances. That is why the possibility of including time windows is advocated in works such as Yang et al. [2018].

- **Multi-dimensional classification problems.** Existing algorithms focus mainly on one-dimensional problems, while few multi-label approaches have been reported. To the best of our knowledge, FSS on multi-dimensional data streams remains an open issue, as current approaches simply address the problem by transforming the data into a one-dimensional learning task. FSS algorithms for feature streams were proposed for multi-label learning but not for the more general multi-dimensional case.

- **Removal of relevant features by an external agent.** It may be interesting to research the possibility that a feature considered relevant is removed by an external agent (e.g., if it was detected as erroneous) during the processing of a feature stream. An efficient strategy to recover previously removed features using constant memory could be of interest in this situation.

- **Online ensemble FSS.** Ensemble FSS approaches have received some attention for batch environments [Bolón-Canedo and Alonso-Betanzos, 2019], but very little progress has been made for dynamic data problems. For example, to the best of our knowledge, no ensemble proposal exists for FSS on feature streams. This could be understandable due to the sophistication of some algorithms, which may be unhelpful for real-time data processing. Following an ensemble strategy to define a combined feature selection through different FSS algorithms could lead to inefficient approaches to processing
dynamic data. However, its robustness could help to obtain more stable solutions.

- **Algorithms to handle both data and feature streams.** Only two proposals that simultaneously handle streams of data and features were found. However, this kind of algorithm is of special interest since it combines the benefits of algorithms for data streams and feature streams into a unique model. When working in study areas that involve, for example, lab experiments, it is common that data not only grow in size but also in dimensionality. Thus, there is a clear absence of research on this topic, which lacks extensions of the classical rough set-based theory to handle, among others, hybrid (discrete and continuous) attributes, apart from algorithms that are not linked to these mathematical theories and deal with, for example, unsupervised, regression, multi-class, multi-dimensional or non-linear learning problems.

- **Distributed online FSS algorithms.** As the search for more efficient algorithms will continue to be an open issue, we believe that this trend makes it interesting to pay more attention to distributed online FSS. Centralized algorithms may not meet the necessary performance requirements for multiple real-world problems. A starting point could be the development of online FSS methods based on nature-inspired metaheuristics, such as genetic algorithms.

### 7.7 Conclusions

Incremental algorithms are characterized by their ability to adapt to the appearance of new data, which forces them to make a strict reduction of consumed resources to address real-time applications. Here, tasks such as dimensionality reduction, and more specifically, FSS, are implemented due to the inefficiency of maintaining irrelevant and redundant features. Incremental FSS is an important task not only when new features or instances are received over time but also when static datasets are massive since batch algorithms may not be efficient enough to process them. Thus, this review has studied a variety of FSS proposals from two main perspectives, depending on whether they are capable of incrementally adapting their solution to data or feature streams. Additionally, a more complex environment where both instances and features are received was discussed.

Although an ideal online FSS algorithm should analyze the data once, some literature methods need to re-examine and store past information, some with the justification of, for example, being able to correctly adapt to concept drifts. It is common to find these algorithms defined simply as incremental. Therefore, proposals described as incremental and online were analyzed in this review. Note that there is some ambiguity in the literature about the differences between these concepts. Some works define online learning algorithms as those capable of working in a streaming context endlessly, which seems reasonable, but the classification of an algorithm
may depend on the characteristics of the problem being addressed under this definition. A proposal could be considered online if it is able to handle a specific real-time problem but may not be suitable for a more demanding context or when there are, theoretically, an infinite number of incoming relevant and non-redundant features. Consequently, we proposed a thorough study of what should be considered online and incremental FSS when working with data and feature streams, probably using different definitions depending on the type of stream being processed.

There is no unique way to perform FSS, but the definition of a variable as relevant or redundant is dependent on the problem we are facing. Therefore, the FSS task was studied for supervised (from one-dimensional to multi-label) and unsupervised stream learning, as well as for more specific problems, such as multi-task learning, data with hybrid attributes or even environments where instances or features could be removed as they were detected as erroneous. In addition, the interest in providing as few hyperparameters as possible, since the whole dataset is unknown, led us to dedicate several sections to the study of rough set-based algorithms and extensions of this theory.

Some of the presented algorithms should be studied more deeply. The experiments performed in the respective articles may not be sufficiently exhaustive to draw some of their conclusions. For instance, some works lack trivial experiments, such as comparing the performance of models built using the entire feature set versus the reported subset. Moreover, the suitability of each algorithm is closely tied to the specific characteristics of the problem addressed in each study, as well as to the trade-offs between efficiency and effectiveness we are willing to accept. Given that no single algorithm suits all cases, it remains imperative to evaluate the proposals under a wide range of datasets and conditions to fully understand their true scope and limitations.

Through the review of the literature, it was found that the considerable attention the field of big data and real-time systems is receiving has led to an increase in research on incremental/online algorithms for streams of new instances and features. Among the FSS algorithms studied in this chapter, 47 of them were published in the past decade (2014-2023). These alone account for 75% of all the algorithms reviewed. This trend is expected to increase in the following years since more efficient strategies will be needed to address growing data in terms of the number of instances and dimensionality.
Chapter 8

Supervised feature subset selection and classification on streaming data with multi-dimensional continuous-time Bayesian network classifiers

8.1 Introduction

Previous Chapters 5 and 6 proposed novel ways of modeling and solving multi-dimensional classification problems over categorical time series. During the course of these studies, we were able to observe the relationship between time series and the streaming context discussed in Chapter 7, particularly with data stream learning. The latter is because both time series and data streams involve a set of data points that are collected sequentially over time. Thus, this chapter will explore that, although time series and data streams represent different concepts, they are not mutually exclusive; in fact, time series data can be present in a data stream environment. Similarly, we will study how online FSS algorithms can benefit the application of classification models on feature streams.

Time series data are commonly processed retrospectively, as they are acquired in a finite time period. Meanwhile, data streams are a continuous and potentially infinite flow of data with no predefined endpoint. Consequently, the data stream setting poses new challenges to the traditional time series analysis, such as a one-pass constraint over new data, (near-)real-time processing, the incremental update of learning models, as concept drifts can undermine their classification performance, and having to cope with potential memory issues [Gaber et al., 2007; Read et al., 2020]. Time series data streams are not uncommon, as real-world data
streams often manifest temporal dependencies [Read et al., 2020]. On the other hand, feature streams are characterized by a fixed set of instances and an evolving feature set. Feature streams may arise when the generation of all features is costly, and it is impractical to wait for their complete generation, or when it is currently impossible to obtain all variables, as their generation may depend on external events [Villa-Blanco et al., 2023a].

This chapter explores the application of the Multi-CTBNC [Villa-Blanco et al., 2021] to perform classification on data streams and feature streams, presenting two frameworks for their respective processing. First, in the case of data streams, their dynamic nature requires the study of methods that detect concept drifts and decide which parts of the model need to be updated. We aim to explore techniques that allow models to be locally updated to maintain high accuracy while learning time is as fast as possible. In the case of the feature streams, and as a continuation of the work in Chapter 7, our focus will be on how the application of online FSS algorithms can be beneficial for their processing and, therefore, for the dynamic learning of the model.

The frameworks introduced are versatile, as they apply not only to scenarios where data is naturally arriving from a data or feature stream but also in situations where data is fully available and computational resources are limited. The application of batch algorithms can be unfeasible in these cases since, among other issues, many proposals may assume unlimited memory resources [Gama et al., 2009]. Interest in processing data and feature streams is found across multiple areas of study, such as Internet of Things analytics [Kumari et al., 2019], anomaly detection [Qi et al., 2021], financial analysis [M. S. et al., 2023], analysis of social network data [Yu et al., 2016a], medical learning problems [Wang et al., 2017] or spam detectors [Almusallam et al., 2021]. Multiple problems of such areas can involve, for example, time series data streams, as is the case of Zhang and Thorburn [2022], where time is not only relevant to the order of data arrival but temporal information is key to successfully solving the learning problem.

The main contributions of this chapter are the following:

- The study of two frameworks applying Multi-CTBNCs for classifying categorical time series in data and feature streams, focusing on learning Multi-CTBNCs effectively in response to the dynamic characteristics of the streams.

- The adaptation of a local concept drift detection method for data streams of categorical time series data.

- The introduction of an online extension of the MB-CTPC algorithm (see Chapter 6) that performs local updates of Multi-CTBNCs to address concept drifts in data streams.

- The adaptation of an online FSS algorithm to handle categorical time series data to enhance the learning of Multi-CTBNCs within feature streams.
Chapter 8. Supervised FSS and classification on streaming data with Multi-CTBNCs

- An extensive experimental study to validate the effectiveness of the proposed methods. All the software and datasets used in the experiments are freely available at https://github.com/carlvilla/Multi-CTBNCs.

Chapter outline

The remainder of this chapter is as follows. Section 8.2 presents the problem of classification and adaptation on data streams and introduces an extension of the MB-CTPC algorithm (Section 6.3) to update Multi-CTBNCs (Section 5.2) locally when a concept drift is detected. Section 8.3 describes the feature stream problem and discusses a framework that includes online FSS to alleviate the learning of Multi-CTBNCs. Section 8.4 describes the experimental setting and compares the results obtained by the presented methods against other solutions. Section 8.5 concludes the chapter and discusses future research lines.

8.2 Dynamically updating Multi-CTBNCs in data streams

The first part of this chapter focuses on processing data streams, which consist of a potentially infinite flow of discrete-valued time series (referred to as sequences), with the aim of performing classification tasks. The objective is to model the dynamic nature of these data streams using the Multi-CTBNC model introduced in Chapter 5 while allowing any time prediction requests.

The work presented in this section is based on the research of Borchani et al. [2016], extending the local and global concept drift adapters from MBCs to Multi-CTBNCs and experimentally studying the adaptation and application of Multi-CTBNCs to concept-drifting data streams. Figure 8.1 depicts the framework employed in this chapter to perform the local adaptation of a Multi-CTBNC. Once a new data batch consisting of $q$ sequences is received, two separate modules are triggered in sequential order. First, to detect if a concept drift occurs and which nodes are affected and, second, to update the parent set of those drifting nodes using the newly available data. For illustration, the figure highlights local concept drifts in the most recent data batch, specifically in the features $X_1$ and $X_3$, and in the class variable $C_2$. Consequently, the structure of the Multi-CTBNC is locally updated around the nodes associated with these three variables to address the identified concept drifts.
Figure 8.1: Framework to locally detect concept drifts and update a Multi-CTBNC when a new data batch is received from a data stream.
8.2.1 Local concept drift detection module

The local concept drift detection module, proposed for MBCs by Borchani et al. [2016], is based on estimating the average local log-likelihood scores of each class and feature node on each data batch received. Then, it employs a change point detection method to identify time points where these scores undergo significant changes. This section will adapt this solution to detect local concept drifts on Multi-CTBNCs.

The log-likelihood of some observed data batch \( D^t \) given a Multi-CTBNC is computed as follows (see Section 5.3 for a detailed explanation):

\[
\text{LL}(\mathbb{M} : \mathcal{D}^t) = \sum_{y=1}^{d} \sum_{\text{pa}(C_y)} \sum_{c_j} N_{\text{pa}(C_y)} \log(\hat{\beta}_{\text{pa}(C_y)}) + \sum_{f=1}^{m} \sum_{\text{pa}(X_f)} \sum_{x_j} \left[ M_{\text{pa}(X_f)} \log(\hat{q}_{\text{pa}(X_f)}) - \hat{q}_{\text{pa}(X_f)} T_{\text{pa}(X_f)} \right] + \sum_{x_z \neq x_j} M_{\text{pa}(X_f)} \log(\hat{\theta}_{\text{pa}(X_f)})
\]

where \( d \) and \( m \) are the number of class and feature variables, respectively.

The idea is to monitor the average log-likelihood score, calculated by dividing the log-likelihood by the total number of sequences \( N_t \) in the data batch \( \mathcal{D}^t \). Using the average log-likelihood instead of the raw log-likelihood offers the advantage of effectively comparing data batches of different sizes [Borchani et al., 2016]. Specifically, the average local log-likelihood score of the received data batches given each node is tracked to detect concept drifts locally in each node. As a Multi-CTBNC comprises class and feature variables modeled using different distribution types, the average local log-likelihood score function varies depending on the node type. In the case of the class variables, the average local log-likelihood of a class variable \( C_y \) is computed as follows:

\[
\Pi_y(\mathbb{M} : \mathcal{D}^t) = \frac{1}{N_t} \sum_{\text{pa}(C_y)} \sum_{c_j} N_{\text{pa}(C_y)} \log(\hat{\beta}_{\text{pa}(C_y)})
\]

while for a feature \( X_f \) is determined by:

\[
\Pi_f(\mathbb{M} : \mathcal{D}^t) = \frac{1}{N_t} \sum_{\text{pa}(X_f)} \sum_{x_j} \left[ M_{\text{pa}(X_f)} \log(\hat{q}_{\text{pa}(X_f)}) - \hat{q}_{\text{pa}(X_f)} T_{\text{pa}(X_f)} \right] + \sum_{x_z \neq x_j} M_{\text{pa}(X_f)} \log(\hat{\theta}_{\text{pa}(X_f)})
\]

The concept drift detection is performed on sequences of average local log-likelihoods. The
idea is that if a concept drift occurs on a node, the average local log-likelihood on a new data batch, given that node, should significantly decrease. This decrease would suggest that the new data batch does not fit the CPD of the node as well as previous data batches, indicating that the data might come from a different distribution. Change point detection methods can be employed to perform concept drift detection, which are statistical techniques used to detect time points in sequences of observations where the underlying statistical properties of the data change. These techniques have been widely used for diverse applications, such as fault detection in manufacturing processes or signal processing for identifying seismic activity [Basseville and Nikiforov, 1993]. This work will employ the Page-Hinkley test [Page, 1954] to locally detect concept drifts. This is a sequential analysis technique that, in general terms, is based on calculating cumulative sums of the differences between consecutive average local log-likelihood scores and their mean until the present time. The Page-Hinkley statistic is computed based on the difference between the largest cumulative sum so far and the current cumulative sum. When this statistic exceeds a predefined threshold, a concept drift is detected.

8.2.2 Local update module

Once drifting nodes are detected, a local update algorithm is employed to update the current Multi-CTBNC dynamically. For this task, Algorithm 8.1 is introduced, which is an extended version of the algorithm MB-CTPC [Villa-Blanco et al., 2023b], presented in Chapter 6, to perform structure learning on data streams. This algorithm receives three inputs: the set of features $\mathcal{X}$, the set of class variables $\mathcal{C}$ and a set of drifting variables $\mathcal{N}$, which is provided by the local concept drift detection module.

In the first part of the algorithm, Steps 1 to 10 update the class subgraph of the Multi-CTBNC. This process involves comparing the children and parent sets of the drifting class nodes with the new sets reported by the HITON-PC algorithm [Aliferis et al., 2003, 2010] and the new data batch. The idea is that arcs from variables that are no longer parents of the drifting nodes are removed while arcs from new parents are added. Note that, in order not to violate the acyclicity constraint of the class subgraph, a new arc is only included if it does not introduce any cycle. Then, Steps 11 to 30 update the bridge and feature subgraphs of the Multi-CTBNC. Step 11 creates a temporal copy of the graph for the following operations. First, Steps 12 and 13 define the class variables’ descendants similarly as done in the algorithm MB-CTPC. As relationships between some features may have changed, class variables could have new descendants that are non-drifting features. Then, those relationships could provide relevant information to define the parent sets of drifting feature nodes with the rules of Steps 15 to 25. These rules are the same as those in the MB-CTPC algorithm, but they only study arcs from any feature node towards drifting feature nodes (previously defined in Step 14 as
Algorithm 8.1: Online MB-CTPC($\mathcal{X}, \mathcal{C}, \mathcal{N}$)

1: for each drifting class variable $C_i \in \mathcal{N}$ do
2:   Get current class variables $CP_{C_i}^t$ that are children or parents of $C_i$
3:   $CP_{C_i}^{t+1} \leftarrow$ HITON-PC($C_i$)
4:   for each class variable $C_j \in \{CP_{C_i}^t \mid CP_{C_i}^{t+1}\}$ do
5:     Remove arc $C_j \rightarrow C_i$ from $\mathcal{G}$
6:   end for
7:   for each class variable $C_j \in \{CP_{C_i}^{t+1} \mid CP_{C_i}^t\}$ do
8:     Add arc $C_j \rightarrow C_i$ to $\mathcal{G}$ if no cycles are introduced
9:   end for
10: end for
11: $\mathcal{G}' \leftarrow \mathcal{G}$
12: Build the complete bridge subgraph of $\mathcal{G}'$ on node set $\mathcal{X} \cup \mathcal{C}$
13: $\mathcal{G}' \leftarrow$ CTPC($\mathcal{X}, \mathcal{C}$) \{Algorithm applied to $\mathcal{G}'$\}
14: Build in $\mathcal{G}'$ the complete set of parent feature nodes for drifting feature nodes in $\mathcal{N}$
15: for each drifting feature $X_i \in \mathcal{N}$ do
16:   for each feature $X_j \in \mathcal{X}$, where $X_j \neq X_i$ do
17:     if $\text{Pa}_C(X_i) = \emptyset$ then
18:       Remove arc $X_j \rightarrow X_i$ from $\mathcal{G}'$
19:     else if $\text{Pa}_C(X_i) \cap \text{Pa}_C(X_j) = \emptyset$ AND $\text{Pa}_C(X_j) \neq \emptyset$ then
20:       Remove arc $X_j \rightarrow X_i$ from $\mathcal{G}'$
21:     end if
22:   end for
23: end for
24: $\mathcal{G}' \leftarrow$ CTPC($\mathcal{N}_\mathcal{X}, \mathcal{C} \cup \mathcal{X}$) \{Algorithm applied to $\mathcal{G}'$\}
25: for each drifting feature $X_i \in \mathcal{N}$ do
26:   $\mathcal{G}[X_i].\text{parents} \leftarrow \mathcal{G}'[X_i].\text{parents}$
27: end for
28: return directed graph $\mathcal{G}$

children of all other feature nodes). Then, Step 27 further identifies conditional independence relationships between drifting nodes that could not be removed with the rules. Finally, Steps 28 to 30 save the new parent set of the drifting feature nodes in graph $\mathcal{G}$ to prevent any modification of the parent set of non-drifting feature nodes.

With the aim of comparing the effectiveness and efficiency of the presented framework to locally update Multi-CTBNCs, the results achieved when using a global concept drift detector will also be considered. In this case, we monitor the evolution of the average log-likelihood score for the whole model. If a concept drift is detected, the MB-CTPC algorithm is applied to relearn the entire model.
8.3 Multi-dimensional feature subset selection on feature streams of categorical time series

In the second part of this chapter, we experimentally study a framework for learning Multi-CTBNC on feature streams. The proposed framework incorporates an online FSS algorithm adapted for categorical-state time series data.

The main objective behind the study of this framework lies in that time series classification problems can require significant computational resources for processing, occasionally motivated by the existence of multiple features, despite being common the existence of redundancies and irrelevant variables [Ircio et al., 2020]. This fact is especially noteworthy in feature streams, where the problem will likely be of high dimensionality. Thus, incorporating FSS as a previous step before training a classification model can be very useful to discard variables that can harm the learning algorithms, ease model interpretability, improve learning times and prevent memory problems.

Figure 8.2 illustrates the framework employed in this chapter for processing feature streams. First, FSS is performed in a dynamic environment with new features appearing individually, while training sequences remain static. These features are categorical time series, and it is assumed that all observations for all sequences are available when received. Second, if the new feature is found relevant and non-redundant, the current predictive model, a Multi-CTBNC, is updated.

Algorithm 8.2 describes the pseudocode of the online FSS algorithm. This algorithm, known as ConInd, was proposed by You et al. [2018] and performs a conditional independence analysis to define the relevancy and redundancy of the features in a supervised context. Due to the categorical time series data under study, the definition of conditional independence introduced by Bregoli et al. [2021] will be used (see Section 2.4.2.2). Thus, this FSS algorithm considers the temporal information of the time series to make decisions. Once a new feature is received, Step 2 of the algorithm evaluates its relevancy using unconditional independence tests. If this variable is found to be relevant for the classification task, Step 3 uses single-conditional independence tests to determine if there exists any feature \( X_i \) in the current subset of selected features \( S \) that makes the new feature \( X_t \) redundant. In other words, if \( X_t \) provides no extra information to classify the class variable. Then, Steps 4 to 18 look for redundancies that could occur due to the inclusion of the new feature. First, Steps 4 to 8 employ single-conditional independence tests to remove redundant features in \( S \) given only the new feature \( X_t \). Then, conditional independence tests employing higher cardinality separating sets are performed in Steps 10 to 18. Single-conditional independence tests are performed first in an attempt to reduce the time complexity of Steps 10 to 18, as more time-consuming multi-conditional independence tests are potentially performed on a smaller, more refined set of features.
that other online FSS algorithms employing conditional independence tests, such as Fast-OSFS (discussed in Chapter 7), could also be adapted to fit this learning context.

To apply the previous FSS algorithm in a multi-dimensional classification context, the LP method will be employed. This method creates a compound class variable that collects all combinations of the class values. Although there exist measures that have been adapted to compute the relevancy and redundancy of features in a multi-dimensional context (see, for example, Lin et al. [2017]), they are unsuitable for the categorical time series data that is under study. This is because these measures do not consider the temporal order of time series observations, which is essential for accurate analysis. More complex methods building upon the LP method could also be employed, such as clustering the class variables to define as many multi-class variables as clusters and performing one-dimensional FSS on each multi-class variable separately [Wang et al., 2018]. This conceptually resembles the RAkEL method, but the underlying motivations and methodology are different as the idea is to capture and exploit the inter-class dependencies instead of simply breaking down the problem by randomly defining smaller subsets. In this way, we could reduce the number of class variables by aggregating those that are most correlated and alleviate the class imbalance problem that the LP method suffers if the number of classes is too large. Nevertheless, we discarded employing such an approach in this chapter to avoid adding excessive complexity to an online process that should be conducted as fast as possible.

To evaluate the effectiveness of the studied framework, we compare it against other more
Algorithm 8.2: ConInd($S, C$)

1: for each new feature $X_t$ do
2:   if $X_t \not\perp \not\perp C$, where $C$ is the class variable then
3:     if $X_t \not\perp \perp C|X_i$, $\forall X_i \in S$, where $S$ is the current subset of selected features then
4:       for each feature $X_i \in S$ do
5:         if $X_i \perp \perp C|X_t$ then
6:           $S = S \setminus \{X_i\}$
7:         end if
8:       end for
9:     end if
10:   end for
11: for each feature $X_i \in S$ do
12:   for increasing values $s = 2, \ldots, |S \setminus \{X_i\}|$ do
13:     for each subset $T \subseteq S \setminus \{X_i\}$, where $|T| = s$ do
14:       if $X_i \perp \perp C|T$ then
15:         $S = S \setminus \{X_i\}$
16:       end if
17:     end for
18:   end for
19: end if
20: end for
21: end for

straightforward solutions, such as waiting for the entire feature stream before training the model or retraining the model without including any FSS technique. The goal is to demonstrate the advantages of the online FSS approach in efficiently adapting to feature streams.

### 8.4 Experiments

This section presents the experimental analysis of the proposed frameworks for learning Multi-CTBNCs in dynamic environments. The experiments will be divided into those performed on data streams and feature streams.

The data are sampled via probabilistic logic sampling [Henrion, 1988; Fan and Shelton, 2008] from Multi-CTBNCs whose structures and parameters are randomly generated. Noise is added to the sampling of class variable states and transitions of the features to make the experiments more realistic and challenging for the concept drift detection methods and the FSS algorithm. Approximately 1% of features’ transitions and class variables’ states were randomly sampled, while a Gaussian noise with zero mean and a standard deviation of 0.05 was added to features’ transition times. The feature nodes have four possible states, and their
parent set is randomly defined from zero to a maximum of five parents. Meanwhile, the class subgraphs of the Multi-CTBNCs are always formed by four class variables with three possible states.

The online MB-CTPC (data streams) and the CTPC (feature streams) algorithms are employed to learn the structures of the Multi-CTBNC. Significance levels of 0.05 (class subgraph) and $1 \times 10^{-5}$ (bridge and feature subgraphs) are used to test conditional independence. Regarding parameter learning, Bayesian estimation is used with the following hyperparameters for their prior distributions (see Section 5.3.1): $\lambda_{c_j}^{pa(C_y)} = 1$ and $\alpha_{x_j,x_z}^{pa(X_f)} = 1$ for the Dirichlet prior distribution, and $\tau_{x_j}^{pa(X_f)} = 0.001$ for the gamma prior distribution.

The Wilcoxon signed-rank test is used with a significance level of 0.05 to assess whether the differences between the results of the methods are statistically significant. In the tables, an asterisk (*) next to a result denotes that this result is statistically better than all the others for a given performance measure. When two or more results in a given category have an asterisk, they were found to be the best, and no statistically significant difference exists between them. Additionally, the best absolute results are highlighted in bold font.

The experiments were conducted on an Intel Core i7-7700K at 4.20GHz with 32 GB of RAM using Windows 10 operating system. All the algorithms were developed in Java.

### 8.4.1 Data streams

To evaluate the performance of the update methods under different concept drift scenarios, we have employed the experimental approach from Borchani et al. [2016], which defines three rates of concept drifts depending on the percentage of drifting nodes:

- **No concept drift.** A single model samples a stationary data stream. The objective is to evaluate the robustness of the compared methods to false positive detections.

- **Gradual concept drift.** Every ten data batches, the parent set of 20% of the model nodes is changed, simulating a gradual concept drift. These data streams simulate a scenario in which the underlying model is slowly changed over time. The objective is to evaluate the accuracy of the local detection and update algorithm in this scenario and thus assess its usefulness against solutions involving a global update of the trained model.

- **Abrupt concept drift.** Every ten data batches, the parent set of 50% of the model nodes is changed, simulating an abrupt concept drift. These data streams simulate a scenario where the underlying model is changed abruptly, so the final model is more...
likely to completely differ from the initial one. The objective is to evaluate if the global update method could be of better use in this situation.

For each concept drift rate, ten data streams, each consisting of 60000 sequences with a duration of 5 time units, are sampled. The prequential evaluation is used to evaluate the performance of the update methods [Gama et al., 2009]. This evaluation method consists of training the models on an initial dataset and updating and testing them with the incoming data batches from the data streams. More precisely, the model first predicts the class configuration of the newly received sequences, and then the model is updated using the actual class configurations of those sequences.

An essential aspect when processing the data stream is to establish an appropriate detection threshold for the local and global update methods. In these experiments, we will evaluate these methods using five thresholds: 1.2, 1.4, 1.6, 1.8, and 2.0.

8.4.2 Feature streams

In the case of the feature streams, our experiments start with a model where only the class subgraph is modeled, and features are progressively received. Note that, in these experiments, the actual relationships among class variables are stable and not influenced by new features. Multi-CTBNCs will be trained on the feature streams, and their performance will be assessed on a test dataset. Four different methods to process the feature stream will be compared:

- **Batch training.** This method assumes that the feature stream is finite and waits until all features are received before training the model. In most contexts, this may not be feasible, implying waiting for the entire feature stream to be received to start making predictions. Additionally, learning the model from a static dataset could be time-consuming or impossible due to memory constraints.

- **Model retraining.** The model will be relearned every time a new feature is received. Unlike the previous approach, this method allows the model to be employed without waiting for the entire feature stream to be received.

- **Online FSS with batch training.** Online FSS is employed over the entire feature stream to retain only relevant and non-redundant features. Once the entire feature stream is received, the model is trained using only the selected features. This method does not generate an anytime model, as it is only learned once the entire feature stream is processed. This method could be suitable when real-time prediction is unnecessary and the data is processed as a feature stream to avoid memory problems.

- **Online FSS with model retraining.** Online FSS is performed every time a new feature is received from a feature stream, and the model is relearned whenever a new
relevant feature is found. This method allows real-time prediction while keeping in memory only those features that were found relevant for the classification task.

This experimental analysis will study feature streams with sizes of 25, 50, 75, 100 and 125 features, and ten feature streams will be sampled for each feature stream size. It will be assumed that approximately 20% of the features received from the feature streams are relevant for the classification task, i.e., those features are included in the Markov blanket of at least one class variable.

As feature streams involve a constant data flow, algorithms may need to prioritize efficiency while balancing it with performance. In the case of the online FSS algorithm under study, two parameters can be tuned to balance the robustness and execution time of the solutions. First, to alleviate the redundancy phase of Steps 10 to 18 of Algorithm 8.2, the online FSS will consider separating sets up to a cardinality of 2. Increasing the size may help to detect more redundant variables, but it could also negatively affect the execution time of the algorithm. Another parameter that could be tuned is the significance level in the conditional independence tests. Using a lower significance level for the FSS may yield smaller subsets of features, as the algorithm needs stronger evidence to reject the null hypothesis of conditional independence between two features given a separating set. Therefore, fewer features could be considered relevant, while more redundancies could be detected. This would likely imply that smaller subsets of features are reported, improving the overall time needed to process the feature stream. However, more variables could be erroneously defined as irrelevant or redundant. A significance level of $5 \times 10^{-6}$ will be used for our experiments when performing the online FSS. This parameter was arbitrarily chosen, but on the condition that it was lower than the significance level used by the constraint-based algorithm to learn the bridge and feature subgraphs of the models.

8.4.3 Results

8.4.3.1 Data streams

Tables 8.1, 8.2 and 8.3 show the results of not updating and locally or globally updating a Multi-CTBNC over 30 different data streams, ten for each table, using different concept drift detection thresholds.

The first key finding from our experiments reveals that both the local and global update methods outperform not updating the model in the presence of both gradual and abrupt concept drifts. These results validate the effectiveness of the concept drift detection and model update methods presented in this chapter. This improvement holds across all the detection thresholds that are studied. However, it is worth noting that tuning this parameter
Table 8.1: Estimated performance measures (mean ± std. deviation) over ten stationary synthetic data streams.

<table>
<thead>
<tr>
<th>Detection threshold</th>
<th>Update method</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro F1 score</th>
<th>Global Brier score</th>
<th>Updating time&lt;sup&gt;a,b&lt;/sup&gt;</th>
<th>Total updating time&lt;sup&gt;c&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>No update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.2470*</td>
<td>0.2470*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.3144 ± 0.1386</td>
<td>0.7088 ± 0.0801</td>
<td>0.5799 ± 0.1337</td>
<td>0.8048 ± 0.1195</td>
<td>35.8770</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>No update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.3759 ± 0.1495</td>
<td>0.7433 ± 0.0750</td>
<td>0.6438 ± 0.1243</td>
<td>0.7523 ± 0.1409</td>
<td>0.5987 ± 0.0589*</td>
<td>2.950*</td>
</tr>
<tr>
<td>1.6</td>
<td>No update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td>1.8</td>
<td>No update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td>2.0</td>
<td>No update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.4132 ± 0.1498*</td>
<td>0.7614 ± 0.0837*</td>
<td>0.6683 ± 0.1132*</td>
<td>0.7221 ± 0.1413*</td>
<td>0.0000*</td>
<td>0.0000*</td>
</tr>
</tbody>
</table>

<sup>a</sup> The "-" symbol indicates that there is no associated updating time, as the model is not updated.

<sup>b</sup> The mean updating time is computed excluding batches where no updates occurred (in seconds).

<sup>c</sup> Total updating time to process the ten data streams (in seconds).

proves to be crucial, as setting it too low or too high could result in a significant number of false positives or negatives in concept drift detection.

The results of Table 8.1 correspond to those obtained on stationary data streams. From this experiment, we can highlight that if the detection threshold is not big enough (in this case, 1.2 or 1.4), the performance of the models learned with the global update may suffer significant deterioration. Since the entire model is relearned using relatively small data batches, more errors will likely be introduced in the resultant solution. In the case of local updating, errors could be introduced in more localized parts of the model, which could have a minor impact on its performance. This occurs in Figure 8.3, where we can see that the first false positive detected by the global update method (data batch number nine) derived from an incorrect update of the model and a degradation in the performance for the rest of the data stream. In the case of the local update, the single erroneously detected concept drift did not seem to have a relevant impact on the results. Nevertheless, when an appropriate detection threshold is set, no significant differences are found between not updating the model and both the local and global update methods, as no false positive detections occur.

Then, Table 8.2 shows the results when processing data streams with gradual concept drifts. This experiment shows a significant improvement in the results obtained by the local and global update methods compared to not updating the model. As mentioned above, these results show the importance of establishing a correct detection threshold, which must be tuned individually for each update method. If we consider the best results obtained by the local and global update methods, i.e., using a detection threshold of 1.2 and 1.6, respectively, the local update method obtains statistically better results for all performance measures, except
for the mean accuracy and global Brier score, where no significant differences were found. Thus, apart from obtaining better or the same overall results as the global update method when gradual concept drifts are present, the local update method demonstrated a statistically significant advantage regarding mean update time and total update time. Figure 8.4 provides an example of the processing of a data stream with gradual concept drifts. It is interesting to see how both the local and global update methods correctly detect a concept drift in the tenth data batch, but the global update method introduces errors that significantly hinder the performance due to relearning the entire model. On the other hand, the local update method also identifies a concept drift in that data batch but incorporates updates to the model without causing any issues. Although only a slight improvement is observed compared to not updating the model, the local update method handles the concept drift better. As previously mentioned, completely updating the model with a relatively small data batch while few or no changes occur makes it more likely to introduce more errors in the resultant model.

Finally, Table 8.3 presents the results obtained from data streams with abrupt concept drifts where half of the nodes are drifting. In this scenario, the global update method demonstrates superior accuracy performance. Models are significantly affected by concept drifts in this situation, so it may be easier for the global update to detect them and more convenient to relearn the entire model. Figure 8.5 further illustrates these results, showing the processing of one such data stream using a detection threshold of 1.2 and 1.4 for the local and global update, respectively, which reported the best overall result for each method. Although, in this
Table 8.2: Estimated performance measures (mean ± std. deviation) over ten synthetic data streams with gradual concept drifts.

<table>
<thead>
<tr>
<th>Detection threshold</th>
<th>Update method</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro $F_1$ score</th>
<th>Global Brier score</th>
<th>Updating time$^a$</th>
<th>Total updating time$^a$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No update</td>
<td>0.2467 ± 0.1781</td>
<td>0.6245 ± 0.1609</td>
<td>0.5455 ± 0.1533</td>
<td>0.9295 ± 0.2008</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1.2</td>
<td>Local update</td>
<td>0.3347 ± 0.2075*</td>
<td>0.6862 ± 0.1516*</td>
<td>0.5978 ± 0.1624*</td>
<td>0.7928 ± 0.1931*</td>
<td>0.2846 ± 0.1320*</td>
<td>12.5240*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2436 ± 0.1446</td>
<td>0.6015 ± 0.0944</td>
<td>0.5235 ± 0.1435</td>
<td>0.8575 ± 0.1156</td>
<td>0.6454 ± 0.1219</td>
<td>43.8800</td>
</tr>
<tr>
<td></td>
<td>No update</td>
<td>0.2467 ± 0.1781</td>
<td>0.6245 ± 0.1609</td>
<td>0.5455 ± 0.1533</td>
<td>0.9295 ± 0.2008</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1.4</td>
<td>Local update</td>
<td>0.2902 ± 0.1751*</td>
<td>0.6671 ± 0.1415*</td>
<td>0.5819 ± 0.1414*</td>
<td>0.8525 ± 0.1748</td>
<td>0.2908 ± 0.1305*</td>
<td>7.8520*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2892 ± 0.1564*</td>
<td>0.6761 ± 0.0995*</td>
<td>0.5578 ± 0.1503</td>
<td>0.8374 ± 0.1326*</td>
<td>0.6288 ± 0.1273</td>
<td>28.0590</td>
</tr>
<tr>
<td></td>
<td>No update</td>
<td>0.2467 ± 0.1781</td>
<td>0.6245 ± 0.1609</td>
<td>0.5455 ± 0.1533</td>
<td>0.9295 ± 0.2008</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1.6</td>
<td>Local update</td>
<td>0.2828 ± 0.1811</td>
<td>0.5650 ± 0.1529</td>
<td>0.5723 ± 0.1494*</td>
<td>0.8616 ± 0.1807</td>
<td>0.3069 ± 0.1369*</td>
<td>7.3660*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2867 ± 0.1582*</td>
<td>0.6084 ± 0.1014*</td>
<td>0.5531 ± 0.1533</td>
<td>0.8322 ± 0.1350*</td>
<td>0.6426 ± 0.1230</td>
<td>17.9920</td>
</tr>
<tr>
<td></td>
<td>No update</td>
<td>0.2467 ± 0.1781</td>
<td>0.6245 ± 0.1609</td>
<td>0.5455 ± 0.1533</td>
<td>0.9295 ± 0.2008</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1.8</td>
<td>Local update</td>
<td>0.2816 ± 0.1812*</td>
<td>0.6549 ± 0.1521</td>
<td>0.5706 ± 0.1492*</td>
<td>0.8627 ± 0.1807</td>
<td>0.2894 ± 0.1181*</td>
<td>5.7880*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2796 ± 0.1596*</td>
<td>0.6710 ± 0.1077*</td>
<td>0.5451 ± 0.1585</td>
<td>0.8387 ± 0.1361*</td>
<td>0.6458 ± 0.1250</td>
<td>16.1440</td>
</tr>
<tr>
<td></td>
<td>No update</td>
<td>0.2467 ± 0.1781</td>
<td>0.6245 ± 0.1609</td>
<td>0.5455 ± 0.1533</td>
<td>0.9295 ± 0.2008</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2.0</td>
<td>Local update</td>
<td>0.2808 ± 0.1816*</td>
<td>0.6542 ± 0.1521*</td>
<td>0.5608 ± 0.1496*</td>
<td>0.8634 ± 0.1810*</td>
<td>0.2928 ± 0.1146*</td>
<td>5.5630*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2845 ± 0.1727</td>
<td>0.6502 ± 0.1271*</td>
<td>0.5251 ± 0.1705</td>
<td>0.8659 ± 0.1659*</td>
<td>0.6384 ± 0.1399</td>
<td>12.7270</td>
</tr>
</tbody>
</table>

$^a$ The "*" symbol indicates that there is no associated updating time, as the model is not updated.
$^b$ The mean updating time is computed excluding batches where no updates occurred (in seconds).
$^c$ Total updating time to process the ten data streams (in seconds).

example, the local update method correctly detects concept drifts in expected data batches and achieves significantly better results than not updating the model, it still falls short in predictive power compared to the global update method. This limitation arises from the local update method's inability to detect all drifting class and feature nodes. This is especially evident in nodes that lose parents after the concept drift, as the average local log-likelihood score does not decrease in some of these situations; in fact, it might even increase. This latter behavior is depicted in Figure 8.6 for feature nodes $X_2$ and $X_5$. Even when their parent set is no longer accurate for the newly received data, nodes $X_2$ and $X_5$ exhibit an increased average local log-likelihood score, suggesting that their parent set is a better fit after the concept drift. In the case of $X_7$, an increase is not observed. Still, despite losing multiple parents, the average local log-likelihood for this node remains relatively stable, which makes it difficult to detect a local concept drift. Nevertheless, assuming a threshold of 1.3 or less, concept drifts are correctly detected for nodes $X_1$, $X_4$ and $X_9$, even when $X_4$ is also losing a parent.

Despite the previous limitation, the global update method was still significantly more time-consuming than the local counterpart in the presence of abrupt concept drifts, an interesting advantage when processing data streams. However, it is noteworthy that the difference in update times between these methods varies with the type of concept drift. When using detection thresholds of 1.2 and 1.4 for the local and global update methods, respectively, the differences in total update time were not as significant in the case of abrupt concept drifts (mean time: 22.346s for local vs. 28.329s for global) as with gradual concept drifts (mean time: 12.524s for local vs. 20.059s for global). This observation is supported by the more significant p-value obtained with gradual concept drifts (0.00195) than with abrupt changes.
Figure 8.4: Results of the different updating strategies over a synthetic data stream with gradual concept drifts. The detection threshold is set to 1.2 for the local update method and to 1.6 for the global update method.

(0.01367). This fact leads us to expect that the larger the part of a Multi-CTBNC is affected by concept drifts, the more appropriate it could become to use the global update method.

8.4.3.2 Feature streams

The results obtained from processing the feature streams are summarized in Table 8.4, providing an overview of the performance of each method under varying feature stream sizes.

Overall, the batch training and model retraining methods demonstrate statistically significant improvements in the accuracy of the reported models compared to methods using online FSS. However, it is essential to note that there is still room for improvement in both the measures used to quantify the relevance and redundancy of a feature and in the approach to deal with multiple class variables. The conclusions in Section 8.5 will further discuss these issues in detail.

Despite the detriment to model accuracy, online FSS algorithms can still be highly beneficial in different scenarios. First, performing online FSS with batch training and model retraining demonstrates a statistically significant improvement in learning times compared to their counterparts without FSS (except for batch training with 25 features). Furthermore, if the size of the feature stream is very large, even unknown, waiting for all features to perform offline training may be infeasible, and real-time prediction may be necessary while the feature
stream is being processed. A simple solution is to retrain the model whenever a new feature is received. However, it is evident that the learning times would become prohibitive if no prior FSS is performed to remove irrelevant and redundant features, as can be seen in our experiments. In addition, the model retraining with online FSS shows no statistical differences in learning time compared to batch training for feature streams of 50 and 75 features while offering the possibility of real-time prediction.

As we commented in Chapter 7, online FSS is not only useful for scenarios where features are received over time but also when dealing with static datasets of high dimensionality. In these cases, the application of traditional batch algorithms may encounter memory problems, as reflected in our experiments with feature streams of 100 and 125 variables. Both the batch and model retraining methods led to memory problems with the available computational resources. Thus, only results from methods employing the online FSS algorithm could be reported in Table 8.4.

Incorporating online FSS into the feature stream processing also shows a significant improvement in classification times. This fact could make these approaches even more attractive for real-time applications. However, it is essential to acknowledge that this improvement could be influenced by the fact that the FSS algorithm may incorrectly exclude certain strongly relevant features. This is what would lead to the decreased classification power of our results (at least with feature streams of 25 to 75 variables). Despite this limitation, this trade-off between speed and accuracy could benefit some real-time applications.

In conclusion, as we continue to refine and enhance the online FSS algorithm and better incorporate the information given by the dependencies between class variables, the model

### Table 8.3: Estimated performance measures (mean ± std. deviation) over ten synthetic data streams with abrupt concept drifts.

<table>
<thead>
<tr>
<th>Detection threshold</th>
<th>Update method</th>
<th>Global accuracy</th>
<th>Mean accuracy</th>
<th>Macro $F_1$ score</th>
<th>Global Brier score</th>
<th>Updating time(^a)</th>
<th>Total updating time(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>No update</td>
<td>0.1283 ± 0.1903</td>
<td>0.4783 ± 0.1872</td>
<td>0.4084 ± 0.1718</td>
<td>1.0692 ± 0.2294</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.1759 ± 0.1864</td>
<td>0.5748 ± 0.1588</td>
<td>0.4800 ± 0.1663</td>
<td>0.9304 ± 0.1679</td>
<td>0.3492 ± 0.1479*</td>
<td>22.3460*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2556 ± 0.1916*</td>
<td>0.6399 ± 0.1668*</td>
<td>0.5045 ± 0.1772*</td>
<td>0.8733 ± 0.1906*</td>
<td>0.6688 ± 0.1025</td>
<td>48.8900</td>
</tr>
<tr>
<td>1.4</td>
<td>No update</td>
<td>0.1283 ± 0.1903</td>
<td>0.4783 ± 0.1872</td>
<td>0.4084 ± 0.1718</td>
<td>1.0692 ± 0.2294</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.1455 ± 0.1877</td>
<td>0.5244 ± 0.1792</td>
<td>0.4551 ± 0.1780</td>
<td>0.9836 ± 0.1834</td>
<td>0.3445 ± 0.1268*</td>
<td>17.2270*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2839 ± 0.1911*</td>
<td>0.6666 ± 0.1568*</td>
<td>0.5350 ± 0.1725*</td>
<td>0.8403 ± 0.1872*</td>
<td>0.6438 ± 0.0952</td>
<td>28.3200</td>
</tr>
<tr>
<td>1.6</td>
<td>No update</td>
<td>0.1283 ± 0.1903</td>
<td>0.4783 ± 0.1872</td>
<td>0.4084 ± 0.1718</td>
<td>1.0692 ± 0.2294</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.1444 ± 0.1867</td>
<td>0.5228 ± 0.1773</td>
<td>0.4388 ± 0.1789</td>
<td>0.9881 ± 0.1837</td>
<td>0.3333 ± 0.1131*</td>
<td>15.0000*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2681 ± 0.1918*</td>
<td>0.6482 ± 0.1712*</td>
<td>0.5132 ± 0.1809*</td>
<td>0.8549 ± 0.1809*</td>
<td>0.6549 ± 0.0909</td>
<td>24.1900</td>
</tr>
<tr>
<td>1.8</td>
<td>No update</td>
<td>0.1283 ± 0.1903</td>
<td>0.4783 ± 0.1872</td>
<td>0.4084 ± 0.1718</td>
<td>1.0692 ± 0.2294</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.1411 ± 0.1876</td>
<td>0.5161 ± 0.1795</td>
<td>0.4324 ± 0.1753</td>
<td>0.9955 ± 0.1840</td>
<td>0.3351 ± 0.1140*</td>
<td>14.0730*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2519 ± 0.1876*</td>
<td>0.6325 ± 0.1749*</td>
<td>0.5019 ± 0.1823*</td>
<td>0.8709 ± 0.1799*</td>
<td>0.6362 ± 0.1010</td>
<td>20.7950</td>
</tr>
<tr>
<td>2.0</td>
<td>No update</td>
<td>0.1283 ± 0.1903</td>
<td>0.4783 ± 0.1872</td>
<td>0.4084 ± 0.1718</td>
<td>1.0692 ± 0.2294</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Local update</td>
<td>0.1401 ± 0.1883</td>
<td>0.5072 ± 0.1855</td>
<td>0.4260 ± 0.1793</td>
<td>0.9965 ± 0.1872</td>
<td>0.3145 ± 0.1137*</td>
<td>12.8950*</td>
</tr>
<tr>
<td></td>
<td>Global update</td>
<td>0.2389 ± 0.1893*</td>
<td>0.6222 ± 0.1761*</td>
<td>0.4954 ± 0.1812*</td>
<td>0.8824 ± 0.1833*</td>
<td>0.6383 ± 0.0797*</td>
<td>17.8720*</td>
</tr>
</tbody>
</table>

\(^a\) The \(\ast\) symbol indicates that there is no associated updating time, as the model is not updated.

\(^b\) The mean updating time is computed excluding batches where no updates occurred (in seconds).

\(^c\) Total updating time to process the ten data streams (in seconds).
retraining with online FSS demonstrates a promising balance between speed (learning and classification times) and prediction power, while keeping an anytime model that can deliver predictions at any point. Furthermore, the ability of this framework to handle feature streams of high dimensionality, coupled with the advantage of real-time processing, positions it as a valuable option for the processing of feature streams when employing Multi-CTBNCs.

8.5 Conclusions and future work

In this chapter, we explored the application of Multi-CTBNCs for processing data streams and feature streams, presenting two frameworks for their respective processing. The analysis conducted on the processing of data streams under different update methods and in the presence of different types of concept drifts revealed some significant insights. The local and global update methods outperformed the no-update method in the presence of both gradual and abrupt concept drifts, validating the effectiveness of the concept drift detection and model update methods for Multi-CTBNCs. The local update method, which employs an online extension of the MB-CTPC algorithm, reported important advantages regarding updating times with respect to the global update approach, which is particularly significant in a data stream setting. Moreover, in the case of gradual concept drifts or stationary data streams, the global update method was more likely to significantly deteriorate the model’s
Figure 8.6: Comparison of the structures of a Multi-CTBNC before (a) and after (b) a concept drift. The graphs at the bottom showcase the evolution of the average local log-likelihood (c) and Page-Hinkley statistic (d) for each feature node across different data batches, with a black dashed vertical line indicating the onset of the concept drift.

performance, making the local update method an approach to consider in those scenarios. Nevertheless, the local update method encountered challenges in detecting certain concept drifts and exhibited a worse prediction performance than the global update method in a scenario where significant parts of the model were changing.

Regarding experiments on feature streams, these demonstrated that the inclusion of online FSS yielded significant improvements in learning times compared to simple batch training or model retraining approaches. For this learning scenario, the model retraining method with online FSS is of special interest since, in addition to providing an anytime model, it achieved competitive learning times. This solution is particularly relevant for large feature streams, where memory constraints make it impossible to apply offline solutions, and waiting for all the features to arrive to perform predictions is infeasible. Nonetheless, it is essential to consider that the employed online FSS algorithm might incorrectly exclude strongly relevant features, leading to a trade-off between learning speed and accuracy.
The research conducted in this chapter shows promising results. However, there is still a lot of pending work that we expect to address in future studies. From improving the concept drift detection methods or the FSS algorithm to finding efficient ways to update a Multi-CTBNC when a variable is received from a feature stream. Furthermore, we plan to conduct a more extensive experimental study with a larger set of synthetic and real-world datasets of more diverse characteristics to better identify where each algorithm performs best and why. In addition, experimental results may vary in other conditions, such as if there is a higher percentage of relevant features, underlying models have different structure complexities, or more noise is present. Among the areas of open research that were encountered while conducting this work, we can highlight the following:

- **Model updating without true class values.** Assuming that true class values will be available shortly after a new data batch is received from a data stream may not correspond with most real-world scenarios. Thus, it would be of great practical interest to be able to decide when and how to update the model in the absence of actual class configurations. The work of Souza et al. [2015a] provides interesting insights into this aspect and could serve as a starting point for further exploration.

- **Optimizing local concept drift detection for parent removals.** The local concept...
drifts detector was found more likely to fail when concept drifts occurred due to removing parents of a node instead of adding new ones. In future work, we would like to study the reasons behind this and whether the inclusion of a penalty, such as the one used in BIC, to the average local log-likelihood can have a positive influence in these cases.

- **Dynamic threshold for concept drift detection.** The concept drift detector depends on a threshold parameter whose constant value significantly affects the detector’s performance, and it is not trivial to set. However, due to the dynamic nature of data streams, it could be interesting to evaluate a dynamic detection threshold that could evolve based on specific statistics.

- **Online update of node parameters.** Even if no local concept drifts are detected at certain nodes, there could exist small variations in their CPDs that go undetected. Therefore, it might be interesting to include an online update of the nodes’ parameters, following similar approaches to those presented by Bauer et al. [1997] and Shi and You [2006] for CPT and CIM nodes, respectively.

- **Online FSS for data streams.** Including an online FSS algorithm for the context with data streams is a pending study. Applying the LP method to deal with multi-dimensional classification is not straightforward in that setting since this method only allows generalization within the previously received class configurations. Therefore, it would be necessary to assess other approaches to solve this problem and to experimentally evaluate if including FSS could provide additional advantages in Multi-CTBNC learning, mainly in the face of high dimensional data streams.

- **Enhancing FSS for categorical time series.** Identifying relevant and redundant features using conditional independence tests may not be the most appropriate form to perform FSS for learning classification models. In our experiments, the employed FSS algorithm had difficulties identifying spouses of class variables, as they are likely identified as irrelevant or redundant. Further research on FSS methods specifically designed for categorical time series is necessary, and efforts in this direction can significantly improve the efficacy of the FSS and lead to more accurate classification models. We considered other approaches to define the relevance and redundancy of the variables, such as using the non-parametric mutual information estimation method proposed by Ircio et al. [2020]. However, this solution was discarded for two reasons. First, it is too computationally expensive for our streaming context, as it relies on the kNN algorithm with DTW. Second, this method requires encoding the categorical states of the features into numerical values, which can be difficult, or even pointless, to do, as no obvious order may exist between the states of the variables.

- **Multi-dimensional FSS beyond the LP method.** Explore other transformation methods or algorithm adaptation solutions to apply FSS to a multi-dimensional clas-
sification problem. Although the LP is a relatively fast solution for a streaming data context, it can also lead to important problems, such as a significant loss of information or exponential growth of class configurations when including new class variables, which could severely impact the FSS results.

- **Local model update in feature streams.** In our experiments, the entire classification model is relearned when a new variable is received from a feature stream. However, updating the model locally based on this new feature would be more convenient. As it was done for data streams, we plan to adapt the MB-CTPC algorithm to this scenario.

- **Handling of relevant feature removal in feature streams.** Exploring the possibility of a relevant feature being removed due to, for example, erroneous data collection when processing a feature stream. One approach to address this concern could involve preserving a set of weakly relevant and non-redundant variables. These variables could then be retrieved to mitigate a possible deterioration in model accuracy without incurring severe memory consumption.
Part IV

CONCLUSIONS
Chapter 9

Conclusions and future work

This concluding chapter summarizes the most significant contributions of the dissertation and outlines a selection of future research areas discovered throughout our studies. Included in this chapter is also the list of publications and software produced during our research.

Chapter outline

This chapter is organized as follows. Section 9.1 summarizes the main contributions of this dissertation, organized by each contributing chapter. Section 9.2 enumerates the scientific journal articles and conference papers published to support the findings of this dissertation. Section 9.3 describes the software developed to implement the models and algorithms introduced in this dissertation and employed to perform the different numerical experiments. Finally, Section 9.4 discusses potential future research areas and open issues.

9.1 Summary of contributions

Chapters 5 through 8 present the novel contributions of this dissertation.

- Chapter 5 studies the problem of multi-dimensional classification in categorical multivariate time series data. It introduces a novel PGM classifier known as Multi-CTBNC, which extends the one-dimensional CTBNCS, and describes different methods for learning both the parameters and the structure of this model directly from data. The proposed model explicitly represents temporal dynamics in continuous time while modeling the probabilistic dependencies between class variables. Extensive empirical studies, including experiments on synthetic datasets and a real-world Industry 4.0 dataset,
serve to validate the effectiveness and applicability of the introduced model, which outperforms the application of individual CTBNCs both in predictive accuracy and learning time. The experiments showed that not only does the fact of modeling class variable dependencies improve the prediction results, but using a single model that knows the simultaneous dependencies of the features on different class variables allows the Multi-CTBNC to obtain more accurate results. Additionally, this chapter introduces a software tool that allows the learning, application, evaluation and visualization of Multi-CTBNCs. The results presented in this chapter align with objectives O1, O7, O8, and O9 of this dissertation and offer compelling evidence supporting hypotheses H1 and H2.

- Chapter 6 extends the previous chapter, which employs a traditional score-based approach for structure learning, by introducing novel constraint-based and hybrid algorithms for Multi-CTBNCs. These are the first algorithms of their kind for the learning of CTBNCs. The new constraint-based algorithm, MB-CTPC, focuses on optimizing model learning times by taking advantage of the unique structural characteristics of Multi-CTBNCs. Meanwhile, the novel hybrid algorithm, an approach not previously explored even for traditional CTBNs, aims to combine the advantages of both score-based and constraint-based methods. A comprehensive experimental study on synthetic datasets revealed that the MB-CTPC algorithm considerably enhances both the learning and classification times of Multi-CTBNCs while maintaining a competitive classification performance. The improved computational time of MB-CTPC suggests it could be well-suited for real-time environments with dynamic data. In addition, the algorithm proves particularly convenient for high-dimensional datasets, although it also offers significant improvements irrespective of the number of variables, their cardinality or structure complexity. In a real-world dataset experiment, MB-CTPC was statistically found to be the most effective option, achieving the highest predictive performance while significantly reducing the learning time compared to the second-best algorithm. Finally, the newly introduced hybrid algorithm emerges as an interesting solution for datasets with low cardinality features, where it outperforms the results of the CTPC algorithm while significantly reducing the learning and classification times of score-based solutions. The results presented in this chapter align with objectives O2, O3, O7, O8, and O9 of this dissertation and offer compelling evidence supporting the hypothesis H3.

- Chapter 7 presents a comprehensive review of FSS algorithms designed for dynamic data environments, classifying these data into data streams, feature streams, and a combination of both. This chapter provides a comparative analysis of the studied algorithms, investigating their theoretical foundations and offering new alternatives for some of their shortcomings. The review focuses on a wide set of algorithms, each specifically designed for particular learning tasks, which include, among others,
Chapter 9. Conclusions and future work

supervised learning (binary, multi-class and multi-label), unsupervised learning, multi-
task learning, handling discrete and continuous features or processing new instances or
features either individually or in groups. This chapter also identifies the limited research in
areas such as semi-supervised learning, ensemble learning, and distributed online
FSS solutions, highlighting opportunities for future investigations and developments in
the incremental FSS field. The review concludes by highlighting the growing interest
in incremental FSS. This area is particularly relevant given the current developments
in big data and real-time systems. Notably, 47 out of the 62 algorithms evaluated for
this review have been developed in the last decade alone. The results presented in this
chapter align with objectives O4 and O5 of this dissertation and provide the foundation
for addressing hypothesis H5 in a subsequent chapter.

• Chapter 8 explores the application of Multi-CTBNCs in scenarios involving data and
feature streams, presenting tailored frameworks for each setting. For data streams, it
introduces a local update method for Multi-CTBNCs, offering a time-efficient alternative
to traditional global updates when adapting the model to concept drifts. For this
purpose, a novel online MB-CTPC algorithm was designed to facilitate local updates.
Experimental findings indicated that global updates could significantly undermine
performance in cases of gradual concept drift or stationary data streams. Nonetheless,
the local update method can occasionally be less robust, particularly amid abrupt
concept drifts. In the context of feature streams, the framework incorporates an online
FSS algorithm, improving model learning times and enabling the processing of large
feature streams that are infeasible to handle using traditional batch methods. However,
a trade-off exists between learning time and accuracy due to the eventual exclusion of
relevant features by the employed FSS algorithm. Overall, this chapter provides the
first application of Multi-CTBNCs to dynamic data and opens areas for future research
with the potential to enhance the performance of the proposed frameworks. The results
presented in this chapter align with objectives O5, O6, O7 and O9 of this dissertation
and offer compelling evidence supporting the hypotheses H4 and H5.

9.2 List of publications

Peer-reviewed JCR journals

• C. Villa-Blanco, P. Larrañaga, and C. Bielza. Multidimensional continuous time Bayesian

• C. Villa-Blanco, C. Bielza, and P. Larrañaga. Feature subset selection for data and


**Peer-reviewed conferences**


### 9.3 Software

The research presented in this dissertation was experimentally supported with the following software that we have developed:

- A Java software suite called CTBNLab that allows the learning, application, evaluation and visualization of (multi-dimensional) continuous-time Bayesian network classifiers. [https://github.com/carlvilla/Multi-CTBNCs](https://github.com/carlvilla/Multi-CTBNCs).

### 9.4 Future work

This section summarizes several lines of future research that were identified during the course of this dissertation.

- **Optimizing classification time of Multi-CTBNCs.** The main limitation of Multi-CTBNCs lies in their exponential classification time complexity, which scales with respect to the number of class variables and their cardinality. To mitigate this computational constraint, especially in high-dimensional problems, future work could explore the proposal of class-bridge decomposable [Bielza et al., 2011] Multi-CTBNCs to enhance efficiency.

- **Continuous variables in CTBNs.** The conventional CTBN framework is designed to handle discrete variables constrained to take on a finite set of states. As a result, one of its main limitations is that it cannot be directly applied to continuous variables
without prior discretization. Embedding an automated, data-adaptive discretization process within the CTBN learning could be an interesting first step to improve the versatility of these models. Nonetheless, incorporating continuous variables directly into CTBNs could expand their applicability to a wider range of real-world problems and may improve model accuracy by avoiding the information loss associated with discretization. However, this extension would require re-examining how transition rates are represented and estimated or even completely re-evaluating the CTBN paradigm.

- **Improving Multi-CTBNC versatility for real-world settings.** The proposed Multi-CTBNC was applied under certain limitations that may not always hold in practical applications. First, the model assumes no missing data or hidden variables, a condition that may not be met in real-world datasets. Developing methods for learning the model under these circumstances is, therefore, an important direction for future research. Furthermore, the Multi-CTBNC currently employs exponential distributions to model waiting times. However, alternative distributions, such as hypoexponential distributions [Liu et al., 2018c], may be more suitable for certain applications. Future research could address these specific limitations, thus extending the model’s applicability in real-world settings.

- **Improving descendant identification of MB-CTPC.** The MB-CTPC algorithm can sometimes struggle to accurately identify descendants of class variables. Implementing phase distributions, such as the Erlang [Liu et al., 2018c], for modeling transition times of features conditioned on ancestor class variables could potentially enhance this identification process and improve the overall accuracy of learned models. Future research could also explore whether such modification may help mitigate the influence of noise in the data.

- **Adapting the HITON algorithm for Multi-CTBNC structure learning.** The HITON algorithm [Aliferis et al., 2003], which has been successfully employed in MBC structure learning [Borchani et al., 2012], offers potential for adaptation to the context of Multi-CTBNCs. We intend to investigate how to perform this adaptation and assess its performance relative to other structure learning approaches discussed in this dissertation.

- **Improving the hybrid structure learning algorithm.** The proposed hybrid structure learning algorithm has demonstrated advantages in the learning of Multi-CTBNCs. Nevertheless, to date, these improvements have been only observed when handling features of low cardinality. Future research should focus on assessing the utility of these algorithms in real-world settings and optimizing their performance across a wider range of situations. As a preliminary step, it may be beneficial to incorporate ideas from established hybrid algorithms for other PGMs. For example, adapting the max-min parents and children algorithm for the restriction phase, which
is employed in the max-min hill-climbing hybrid algorithm [Tsamardinos et al., 2006] (see an adaptation for DBNs in Trabelsi et al. [2013]). Similarly, the HPC algorithm could also be evaluated for this phase [Gasse et al., 2014]. The HPC employs multiple constraint-based algorithms, each designed to identify specific relationships within the graph, attempting to obtain a more robust collective framework. Its implementation within the H2PC hybrid structure learning algorithm yielded promising results when solving multi-label classification problems [Gasse et al., 2014], which highlights its potential for learning Multi-CTBNCs.

- **Online ensemble FSS.** Ensemble-based FSS methods have received interest in batch processing scenarios [Bolón-Canedo and Alonso-Betanzos, 2019]. However, their development for dynamic settings remains limited, especially in feature stream settings. The complexity inherent in some ensemble algorithms could be one reason for this limitation, as such methods may not be practical for real-time data analysis. Nevertheless, adopting an ensemble approach in these settings could offer more robust solutions, so it is an area of interest for further research.

- **Handling of relevant feature removal in feature streams.** Future research could investigate the likelihood and impact of removing strongly relevant features for reasons such as erroneous data collection when processing a real-world feature stream. The idea would be to explore robust strategies to handle such situations that could try to avoid compromising model accuracy without incurring excessive memory consumption. A potential solution could involve preserving a second fixed-size subset of features formed by previously removed weakly relevant variables, which are non-redundant between them. These variables could then be retrieved to mitigate a possible model performance deterioration due to unexpected removals.

- **FSS and semi-supervised learning with infinitely delayed labels.** Several supervised and unsupervised incremental FSS algorithms were discussed in Chapter 7. However, no proposal that focuses on semi-supervised learning was found. One solution could be to use spectral feature selection incrementally, where the similarity measure between instances considers that class information may or may not exist in some pairwise comparisons. Furthermore, it could be useful to perform FSS on data streams with infinitely delayed labels, i.e., supervised problems where the availability of class variables is delayed. This context was already considered for classifying data streams [Souza et al., 2015a,b]. Unsupervised approaches could be used to perform FSS until the class variable is available when a supervised method could improve the current feature subset. A deeper study of these learning scenarios could also be interesting for the application of Multi-CTBNCs to data streams performed in Chapter 8, as the assumption that true class labels will be available shortly after a new data batch is received may not hold true in most real-world scenarios.
• **FSS for categorical time series.** There is a notable absence of research focused on FSS methods tailored for categorical time series, especially within the context of supervised learning. Given its practical implications, this research gap is particularly noteworthy. For instance, in Chapter 6, we handle real-world data that involves more than 1300 features. Given the complexity of this problem, the application of FSS algorithms becomes particularly beneficial in such a context. Furthermore, in the experiments conducted for Chapter 8, we had to rely on conditional independence tests to perform FSS. However, this approach may not be the most appropriate for learning classification models, as could be seen in the impact of the FSS on their accuracy. We also considered alternative approaches like using the non-parametric mutual information estimation proposed by Ircio et al. [2020]. Nonetheless, we ultimately dismissed this approach for a couple of reasons. Firstly, its reliance on the \( k \)NN algorithm combined with DTW would be too computationally intensive for our streaming context. Secondly, this method requires converting categorical states into numerical values, a challenging task when no clear ordinal relationships exist between the variable states. In summary, additional research on FSS algorithms designed for categorical time series could significantly enhance our ability to process high-dimensional datasets and feature streams more effectively.

• **Local model update in feature streams.** In the conducted experiments of Chapter 8, the entire classification model is relearned when a new relevant feature is received from a feature stream. However, a more efficient solution would be locally updating the model, taking into account this new variable and the potential redundancies that it could introduce. Based on the solution used for data streams, we aim to adapt the MB-CTPC algorithm to this scenario with feature streams.

• **Improvement of local concept drift detector.** Early findings suggested that the local concept drift detector is less reliable when concept drifts are caused by the removal of parents of a drifting node as opposed to when they are added. Further research is needed to understand the underlying causes of this issue. We aim to study whether incorporating penalties, such as the one used in BIC, into the average local log-likelihood score could mitigate this problem. Furthermore, the detector’s performance is currently influenced by a static threshold parameter that is not trivial to define. Given the dynamic nature of data streams, future research could consider developing methods for dynamically setting this threshold based on specific statistical measures.

• **Future industrial applications:** Although the proposed Multi-CTBNC has proven effective in one specific industrial project, its utility could potentially extend to broader applications. The industrial company that collaborated for the experimental section of Chapter 5 has provided us with multiple learning problems that can benefit from the work done in this dissertation. Here, we outline two examples of such problems.
The first example is a two-stage problem in which our model would be responsible for predicting the activation of multiple components of an industrial system. Then, in the second stage, an optimization algorithm would try to optimize these activations based on the predictions with the goal of reducing peak power consumption. The second problem is from the domain of laser-based manufacturing procedures, where a large number of variables are available. In this case, the objective is to use incremental FSS algorithms to identify the currently most relevant parameters that influence the quality or efficiency of the laser’s work. These selected features could then be used by other learning algorithms to optimize the process further, predict anomalies, or enhance quality control measures.
References


D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In


A. Boulesnane and S. Meshoul. WD2O: A novel wind driven dynamic optimization


Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an


Z. Meng, S. Han, and Y. Tong. Listen to your face: Inferring facial action units


C. Salperwyck, V. Lemaire, and C. Hue. Incremental weighted naïve Bayes classifiers for data


C. Villa-Blanco, P. Larrañaga, and C. Bielza. Multidimensional continuous time Bayesian


J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, and X. Wu. Online feature selection with


