EINDHOVEN UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Generalized Bayesian Network Classifiers

Author:
Yang Yang

Supervisors:
Cassio de Campos
Vu-Linh Nguyen

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Engineering

in the

Uncertainty in Artificial Intelligence Group
Department of Mathematics and Computer Science

EINDHOVEN UNIVERSITY OF TECHNOLOGY

November 5, 2022

Abstract

A fundamental goal of probabilistic classification is the ability to predict a probabilistic distribution over the output space given the input space. Rather than only predicting the most likely class that an input observation should belong to, a probabilistic classifier is able to provide reliable uncertainty information associated with the predictions. In many real-world applications with strict transparency and interpretability requirements, such as medical image classification, facial recognition and automated recruitment, understanding the uncertainties of the predictions is as important as improving the predictive performance. It is therefore no wonder that much attention has been devoted to learning probabilistic classifiers.

In this thesis, we approach the Multi-Dimensional Classification (MDC) problem, an extension of both the traditional Multi-Class Classification (MCC) problem and the well-known Multi-Label Classification (MLC) problem, in which an observation of input is characterized by multiple class variables. In probabilistic MDC, the main challenge is that one needs to predict multi-variate distributions instead of univariate distributions. Besides, there are common situations in which complex types of data coexist, such as numeric values, discrete signals, images, etc. Such type of data is also referred to as multi-modal data.

We propose Generalized Bayesian Network Classifier (GBNC), a generalized framework for solving probabilistic MDC problems with complex types of input. Unlike the existing Multi-dimensional Bayesian Network Classifier (MBNC) framework, GBNC imposes a new structural constraint to learn a discrete BN to model the class distribution by maximizing the Conditional Log-Likelihood (CLL). We present a theoretical analysis of the decomposability of the learning problem and leverage them to devise efficient learning algorithms. Experimental results on different types of benchmark data sets validate the superiority of GBNCs in comparison to existing probabilistic MDC baselines.

Acknowledgements

This thesis marks the end of my student life at TU/e. I did learn a lot in this project. I would like to use this opportunity to express my sincerest gratitude to the people who helped me a lot in this hard and fruitful journey.

First of all, I would like to thank my supervisors Cassio and Linh. Cassio is a great supervisor and brilliant researcher. He continuously provided ideas and suggestions, and always puts his trust in me. I enjoyed our inspiring discussions which often helped me to overcome difficulties. Linh is the most responsible advisor I have ever met. He guided me into the world of probabilistic classification and other related topics. He helped me with much patience in understanding the theory, designing the experiments and writing the thesis. He has always made time to help me when I needed it. I cannot finish this thesis without them. Many thanks to them for supporting my academic exploration and development.

I also wish to thank Renata Medeiros de Carvalho for being the third member of the assessment committee for my thesis and for taking the time to read my thesis on such short notice.

My final thanks go to my parents for their never-ending support and encouragement.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence	1, 2
AIC	Akaike Information Criterion	16
AUC	Area Under the ROC Curve	64
BIC	Bayesian Information Criterion	16, 34
BN	Bayesian Network	ii, v, viii, ix, 2–5, 11–15, 17–34, 36, 39, 42, 49, 57–64, 66–68, 70
BNC	Bayesian Network Classifier	2–4, 19–22
BOP	Bayes-Optimal Prediction	10, 21, 25, 36–41, 46–48, 63, 64
BR	Binary Relevance	v, 19, 20, 45, 47, 48, 50–53, 55–58
CC	Classifier Chain	v, 20, 45–48, 50–53, 55–58
CDF	Cumulative Distribution Function	viii, 8, 9
CLL	Conditional Log-Likelihood	ii, x, 3, 4, 10, 19, 24–26, 28–34, 40, 49, 62, 63, 66, 69
CNN	Convolutional Neural Network	46
CP	Class Powerset	20, 45, 47, 48, 50–53, 55–57
CPD	Conditional Probability Distribution	ix, 13, 14, 18
CPT	Conditional Probability Table	37–39, 41
CV	Class Variable	42–44
DAG	Directed Acyclic Graph	viii, 12, 13, 20, 22, 23, 27, 33, 40, 57, 65, 66
EMS	Exact Match Score	44, 46, 51, 53, 55, 56
GBNC	Generalized Bayesian Network Classifier	ii, v, vi, 3, 4, 25–27, 31, 32, 34, 36, 39, 40, 42, 45–64
HS	Hamming Score	44, 46, 50, 52, 55, 56
i.i.d.	independently and identically distributed	9, 14
IP	Integer Programming	33, 34
LL	Log-Likelihood	3, 10, 14–16, 24, 26
MAP	Maximum A-Posterior	17, 18, 38, 57, 63, 64
MBNC	Multi-dimensional Bayesian Network Classifier	ii, 3, 22, 23, 25, 27, 45
MCC	Multi-Class Classification	ii, 1, 2, 9–11, 20, 45, 48
MDC	Multi-Dimensional Classification	ii, vi, 2–5, 9–11, 19, 22, 25, 26, 37, 43–46, 48, 50–53, 55–57, 62, 63
MDL	Minimum Description Length	16
ML	Machine Learning	1, 2
MLC	Multi-Label Classification	ii, v, 1, 2, 9, 10, 19, 20, 37, 43, 45, 63, 64
MLE	Maximum Likelihood Estimation	14–16, 22, 23
MPE	Most Probable Explanation	17, 18, 21, 37, 38, 41, 49, 57, 63, 64
PDF	Probability Density Function	viii, 8, 10, 25
PGM	Probabilistic Graphical Model	2, 11, 20
PMF	Probability Mass Function	viii, 8, 10, 25, 30
RV	Random Variable	viii, ix, 4, 7–9, 13–15, 17, 18, 20, 21, 34
TAN	Tree-Augmented Network	21
w.r.t.	with respect to	10, 25, 37–39, 41, 45, 47, 48, 62, 63

Notation

Probability Theory

Ω	Sample space.
ω	Elementary event.
E	Event.
Σ	Event space.
P	Probability measure, probability distribution.
$P(A \mid B)$	Conditional probability of A given B.
$P(A, B), P(A \cap B)$	Joint probability of A and B.
X, Y, Z	Random Variables (RVs); throughout this thesis we use uppercase letters to denote RVs.
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Set of RVs, also called *random vectors*.
x, y, z	Realizations (values) of RVs X, Y and Z, respectively; throughout this thesis we use lowercase letters to denote values of RVs.
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	State spaces of RVs X, Y and Z, respectively; throughout this thesis we use caligraphic letters to denote state spaces of RVs.
$p_X(x)$	Probability Mass Function (PMF) of discrete X; the subscript X is omitted when the RV is clear from the context.
$f_X(x)$	Probability Density Function (PDF) of continuous X; the subscript X is omitted when the RV is clear from the context.
$F_X(x)$	Cumulative Distribution Function (CDF) of RV X; the subscript X is omitted when the RV is clear from the context.

Graph Theory and Graphical Models

$\mathcal{B}$	Bayesian Network (BN).
$\mathcal{G}$	Graph; Directed Acyclic Graph (DAG) or BN structure in the context of BNs.
$\boldsymbol{\Theta}$	Parameter set of BN $\mathcal{B}$.

Θ_i	Parameter set of RV X_i.
$\mathbf{V}$	Set of nodes in a graph.
$\mathbf{E}$	Set of edges in a graph.
$V_i \to V_j$	Directed edge from V_i to V_j.
$V_i - V_j$	Undirected edge between V_i and V_j.
$\mathbf{Pa}_{\mathcal{G}}(V), \mathbf{Pa}_{\mathcal{G}}(X)$	Set of parent nodes of node V in a directed graph $\mathcal{G}$, and set of parent variables of RV X in a BN structure $\mathcal{G}$, respectively; the subscript $\mathcal{G}$ is omitted when the structure is clear from the context.
$\mathbf{Ch}_{\mathcal{G}}(V), \mathbf{Ch}_{\mathcal{G}}(X)$	Set of child nodes of node V in a directed graph $\mathcal{G}$, and set of child variables of RV X in a BN structure $\mathcal{G}$, respectively; the subscript $\mathcal{G}$ is omitted when the structure is clear from the context.
$\mathbf{pa}(x)$	Configuration of $\mathbf{Pa}(X)$.
$P(X \mid \mathbf{Pa}(X))$	Conditional Probability Distribution (CPD) of RV X.
$P_{\mathcal{B}}$	Probability distribution represented by $\mathcal{B}$.
$\mathcal{I}(\mathcal{G})$	Set of conditional independencies hold in $\mathcal{G}$.
$\mathcal{G}_{\mathbf{Y}}$	Class subgraph, i.e., the graph consisting of all class variables $\mathbf{Y}$.

Probabilistic Classification

X	Feature (random) variable.
Y	Class (random) variable.
x	Realization of X, feature value.
y	Realization of Y, class value.
m	The number of feature variables.
d	The number of class variables.
$\mathbf{X}$	Feature vector.
$\mathbf{Y}$	Class vector.
$\boldsymbol{\mathcal{X}}$	Feature space, i.e., state space of the feature vector $\mathbf{X}$.
i	Index used in notations related to feature variables, $1 \leq i \leq m$.

$\mathcal{X}_i$	State space of X_i.
$\boldsymbol{\mathcal{Y}}$	Class space, i.e., state space of the class vector $\mathbf{Y}$.
j	Index used in notations related to class variables, $1 \leq j \leq d$.
$\mathcal{Y}_j$	State space of Y_j.
r_i	Number of values of X_i if X_i is discrete, i.e., $r_i = \lvert\mathcal{X}_i\rvert$.
r_j	Number of values of Y_j, i.e., $r_j = \lvert\mathcal{Y}_j\rvert$.
$\mathbf{x}$	Configuration of $\mathbf{X}$.
$\mathbf{y}$	Configuration of $\mathbf{Y}$.
$\mathcal{D}$	Training set.
$\mathcal{T}$	Test set.
l	Index used in notations related to data sets.
ℓ	Loss function; ℓ_{HL} is the *Hamming loss* and $\ell_{0/1}$ is the *subset 0/1 loss*.
$\mathrm{CLL}(\cdot \mid \mathcal{D})$	Conditional Log-Likelihood (CLL) of some model given $\mathcal{D}$.
$\mathbf{\Pi}_j$	Set of parent variables of Y_j within $\mathbf{Y}$, i.e., $\mathbf{\Pi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{Y}$.
$\boldsymbol{\pi}_j$	Configuration of $\mathbf{\Pi}_j$.
q_j	Number of configurations of $\mathbf{\Pi}_j$, i.e., $q_j = \prod_{k=1:Y_k \in \mathbf{\Pi}_j}^{d} r_k$.
$\mathbf{\Phi}_j$	Set of parent variables of Y_j within continuous feature variables, i.e., $\mathbf{\Phi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{X}_c$.
$\boldsymbol{\phi}_j$	Configuration of $\mathbf{\Phi}_j$.
$\mathbf{\Lambda}_j$	Set of parent variables of Y_j within discrete feature variables, i.e., $\mathbf{\Lambda}_j = \mathbf{Pa}(Y_j) \cap \mathbf{X}_d$.
$\boldsymbol{\lambda}_j$	Configuration of $\mathbf{\Lambda}_j$.
e_j	Number of configurations of $\mathbf{\Lambda}_j$, i.e., $e_j = \prod_{i=1:X_i \in \mathbf{\Lambda}_j}^{m} r_i$.
$\mathbf{MB}(Y)$	Markov blanket of Y.
$\mathcal{G}_j$	Subgraph consisting of Y_j and its parent set $\mathbf{Pa}(Y_j)$.
$\mathcal{G}_{\mathbf{\Pi}_j}$	Subgraph consisting of Y_j and the parent set $\mathbf{\Pi}_j$.

$\mathcal{G}_{\mathbf{\Phi}_j}$	Subgraph consisting of Y_j and the parent set $\mathbf{\Phi}_j$.
$\mathscr{G}$	Structure space, i.e., set of all possible $\mathcal{G}$.
$\mathbf{\Psi}$	Parameter space, i.e., set of all possible $\mathbf{\Theta}$.
$\mathscr{G}^X$	Structure space under Constraint 4.
$P_j(Y_j \mid \mathbf{Pa}(Y_j))$	Conditional distribution of Y_j given $\mathbf{Pa}(Y_j)$ specified by Θ_j.
$P_j(y_j \mid \mathbf{pa}(y_j))$	Conditional probability of y_j given $\mathbf{pa}(y_j)$ computed using Θ_j.
$P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \mathbf{\Phi}_j)$	Conditional distribution of Y_j given $\mathbf{\Phi}_j$ under the configuration $\boldsymbol{\pi}_{jk}$ of $\mathbf{\Pi}_j$.
$P_j^{(\boldsymbol{\lambda}_{jn},\boldsymbol{\pi}_{jk})}(Y_j \mid \mathbf{\Phi}_j)$	Conditional distribution of Y_j given $\mathbf{\Phi}_j$ under the configurations $\boldsymbol{\lambda}_{jn}$ of $\mathbf{\Lambda}_j$ and $\boldsymbol{\pi}_{jk}$ of $\mathbf{\Pi}_j$.

Introduction 1

Classification is at the very core of Artificial Intelligence (AI) and Machine Learning (ML). Conventional Multi-Class Classification (MCC) problems deal with only one class variable, i.e., an input instance will be characterized by one output variable, where the possible values of the variable are also called *classes*. For example, to classify a set of animal images, we may define a class variable *animal*, where the class value can either be *no animals*, *cat*, *dog* or *pig*. MCC assumes that a sample can be only classified into one class, i.e., an animal image cannot be both a cat and a dog.

In many real-world scenarios, however, an instance cannot be simply depicted with only one class variable. There is a growing need to deal with multiple class variables in modern ML. Multi-Label Classification (MLC) (Tsoumakas and Katakis, 2007; Zhang and Zhou, 2014) can be seen as the simplest approach to handle multiple class variables, where all the class variables are binary and are also called *labels*. Moreover, MLC can be seen in a variety of practical applications, e.g., a document may contain content from several topics such as politics, economics and history at the same time, a movie may belong to more than one genre, a piece of music may be performed by several instruments, etc.

Although MLC tries to characterize instances from multiple dimensions, the binary class variables are not expressive enough to represent complex information in many other practical problems, where each output dimension is non-binary:

- In bioinformatics (Fernandez-Gonzalez et al., 2015), the neuron can be identified by species (with classes *rat*, *human*, *mouse* or *elephant*), gender (with classes *female* or *male*), development stage (with classes *neonate*, *young*, *adult* or *old*)), etc.
- In sentiment analysis (Ortigosa-Hernández et al., 2012), the attitude of the customer can be characterized from his/her post with three related class variables: subjectivity (with classes *objective* or *subjective*), the sentiment polarity (with classes *very negative*, *negative*, *neutral*, *positive* or *very positive*, will to influence (with classes *declarative text*, *soft*, *medium* or *high*).
- In scene image classification (Read et al., 2014), an image can be annotated from the season dimension (with classes *spring*, *summer*, *autumn* or *winter*) and the landscape dimension (with classes *countryside*, *mountain*, *lake*, etc).

To represent the rich semantics of such instances, Multi-Dimensional Classification (MDC) has been proposed as a generalization of both MCC and MLC in a series of previous studies (Bielza et al., 2011; Zaragoza et al., 2011; Read et al., 2014; Ma and Chen, 2018). Concretely, in MDC, each instance is associated with multiple class variables instead of one, where each class variable can take two or more discrete values.

MDC is a much more difficult problem than its counterparts MCC and MLC (Bielza et al., 2011). First of all, the number of class value configurations in MDC grows exponentially with the number of class variables (usually a higher exponential base than MLC), which makes it intractable to estimate the parameters in modeling the joint distribution. For example, for d class variables with r class values for each[1], the number of class value configurations is r^d. While regarding its counterparts, the number of class values in MCC is r (with r classes) and the number of label combinations of MLC is 2^d (with d labels). Moreover, MDC (or MLC) usually needs to take the probabilistic dependencies among class variables into consideration (Zaragoza et al., 2011; Read et al., 2014), which is a key challenge and has a great impact on the predictive performance.

1: We assume that all class variables take the same number of values for simplicity. In practice, each class variable may have different number of values, e.g., for Y_1, Y_2, Y_3, the number of class values can be r_1, r_2, r_3, where $r_1 \neq r_2 \neq r_3$ and all of them can be larger than two.

Nowadays, interpretability and trustworthiness play an increasing role in modern ML and AI (Wing, 2021). Indeed, probabilistic models provide a principled way for interpretable reasoning under uncertainty with the help of probability theory (Pearl, 1989). It is therefore no wonder significant attention in ML research has been attracted by probabilistic learning. Although MDC is not necessarily conducted in a probabilistic setting (Jia and Zhang, 2022a; Jia and Zhang, 2022b), we devote this thesis to develop probabilistic methods to solve MDC problems and aim to propose a generalized probabilistic MDC framework.

Specifically, a probabilistic classifier estimates the probabilistic distribution of the output space, instead of just predicting the most likely output of an input instance. In probabilistic MDC, classifiers need to estimate multi-variate distributions instead of univariate distributions. MLC can be seen as a special case of MDC in this setting, where classifiers estimate multi-variate discrete distributions of binary variables.

Out of many probabilistic approaches, Bayesian Networks (BNs) (Pearl, 1989), a representative of Probabilistic Graphical Models (PGMs), have been widely used in generative and discriminative prediction tasks due to its advantages on interpretability and transparency (Larrañaga et al., 2012; Kyrimi et al., 2021; Mihaljevic et al., 2021). As a special type of BNs, Bayesian Network Classifiers (BNCs) (Friedman et al., 1997) are designed to deal with one-dimensional MCC problems. BNCs offer a graphical, intuitive and explainable way to represent conditional dependencies and uncertainties supported by probabilistic theory and Bayes' theorem. Furthermore, van der Gaag and de Waal (2006) proposed an algorithmic extension of BNCs called Multi-dimensional Bayesian Network Classifiers (MBNCs) to directly tackle MDC problems. MBNCs inherit many advantages from BNs and BNCs, which effectively model the probabilistic relationships between feature and class variables and reduce the number of parameters used to estimate the joint distribution by exploiting conditional (in)dependencies between variables.

Traditionally, parameters of BNs and BNCs are computed using unsupervised methods such as maximizing the joint likelihood (Friedman et al., 1997). In classification tasks, however, it has been recognized, both theoretically and experimentally, that using supervised methods such as maximizing the Conditional Log-Likelihood (CLL) usually yields a better predictive performance than maximizing the unsupervised joint Log-Likelihood (LL) (Friedman et al., 1997; Greiner et al., 1997; Ng and Jordan, 2001; Kontkanen et al., 2001). Moreover, Roos et al. (2005) observed that in the *M-open* case (Bernardo and Smith, 2000), i.e., the case in which the data generating distribution cannot be represented with a BN class, maximizing the joint likelihood may not converge to the best possible distribution as maximizing the CLL.

Based on these observations, we propose Generalized Bayesian Network Classifier (GBNC), a generalized framework to solve probabilistic MDC problems by maximizing the CLL. Overall, our contributions of this work are summarized as follows:

- We show that the structural constraints made in existing MBNCs prevent one from decomposing the CLL function over the BN structure. Thus, in MBNCs, there is no closed-form solution for finding the optimal parameters for the CLL function. In GBNCs, we propose another set of structural constraints which reserve the possibility of finding the optimal BN structure as well as enable the CLL decomposability.
- We propose partitioning-based algorithms for learning the structure and parameters of GBNCs, in which the CLL is maximized. This is realized by doing score-based BN structure learning on a set of local conditional probabilities computed on partitions from the original input space. The parameters are computed by training base classifiers on local partitions by maximizing the CLL. Combining with our structural constraints, GBNC theoretically guarantees to optimize the CLL in both structure and parameter learning in an effective way. On a series of MDC tabular benchmark data sets, GBNCs

outperform a couple of probabilistic MDC counterparts.

- Although the proposed structure constraints enable decomposing the CLL function, score-based BN structure learning is an NP-hard task and approaches that aim for finding the optimal structure can hardly handle more than ten RVs. We discuss how pruning rules from de Campos et al. (2018) can help simplify learning GBNCs without loss of optimality. This further guarantees the scalability of GBNCs.
- While probabilistic MDC is already a challenging problem, in many real-world applications, this problem becomes more complex as the input data can be *mixed*, i.e., the input data contains more than one type of features, such as numeric values, ordinal values, discrete signals, images, etc. GBNCs are capable of accepting mixed data and achieve competitive performance in these scenarios.

The remainder of this thesis is organized as follows:

- In Chapter 2, we first introduce the mathematical tools required throughout this thesis. We then formulate the MDC problem in both general and probabilistic settings. The foundations of BNs will also be discussed to the extent as required in this thesis.
- In Chapter 3, we give an overview of existing work on probabilistic classification and Bayesian Network Classifiers (BNCs), and discuss their strengths and weaknesses.
- The main contributions are developed in Chapter 4: We first present the theoretical results for supporting additional structural constraints in GBNCs, which enable the CLL decomposability. Then we introduce the formal framework of GBNCs and partitioning-based structure and parameter learning algorithms, in which the CLL is maximized. Furthermore, we discuss how pruning rules from the literature can help reduce the structure search space and improve the scalability. Finally, we go one step further and introduce learning and inference approaches for GBNCs under mixed data.
- Experimental results are presented and discussed in Chapter 5. We demonstrate the applicability and superiority of GBNCs on tabular and image data sets. We also test the performance of GBNCs on mixed data which contains both numeric values and discrete signals.
- In Chapter 6, we conclude this thesis and give some possible future research directions.
- For better readability, lengthy proofs of lemmas, propositions or other formal results are shifted to the appendix.

Background 2

In this chapter, we elaborate on the theoretical background required throughout this thesis. We begin by providing an overview of *probability theory* in Section 2.1. Based on it, in Section 2.2, we introduce the Multi-Dimensional Classification (MDC) problem and formalize it in a probabilistic setting. In Section 2.3, we first review the required concepts and notations from *graph theory*. Then we introduce mathematical foundations of Bayesian Networks (BNs) and discuss learning and inference methods for BNs.

2.1 Probability Theory

In this section, we introduce the basic mathematical formalisms of probability theory. Probabilities are used to represent the *uncertainty* or *randomness* of unknown quantities. In the following, we illustrate some important concepts for reasoning uncertainties in classification tasks.

2.1.1 Probability Space

To model *uncertainty*, we first need to introduce the notion of *universe* or *sample space*, denoted by Ω, which is composed of all *elementary events* ω. All elementary events are mutually exclusive. In probability theory, we assign a probability to each elementary event $\omega \in \Omega$ to represent a degree of *certainty*, i.e., how certain an elementary event ω will happen.

An *event* E is defined as a set of elementary events, where $E \subseteq \Omega$ and its probability is the sum of probabilities of all $\omega \in E$. Furthermore, the set of all possible events Σ is called the *event space*. Formally, we require an event space Σ should be a *sigma-algebra* over Ω.

Definition 2.1.1 (Sigma-algebra (σ-algebra)) *Let Ω be a set. A sigma-algebra (σ-algebra) Σ over Ω is a set of subsets of Ω with the following properties:*

- $\emptyset \in \Sigma$
- *Σ is closed under complement[1], i.e., if $A \in \Sigma$, then $\overline{A} \in \Sigma$.*
- *Σ is closed under union, i.e., if $A_1, A_2, \dots \in \Sigma$, then their union is in Σ: $\bigcup_i A_i \in \Sigma$.*

1: The complement of a set A is defined as $\overline{A} := \Omega \backslash A$

Based on the definition of the event space, now we introduce the notion of *probability distribution* and *probability space*.

Definition 2.1.2 (Probability Distribution) *Let Ω be the sample space and Σ be the event space. A probability distribution $P : \Sigma \mapsto \mathbb{R}$ over (Ω, Σ) is defined as a function that maps events from Σ to real numbers with the following properties:*

- $P(\emptyset) = 0$.
- $P(\Omega) = 1$.
- $\forall A \in \Sigma$, $P(A) \geq 0$.
- *For any disjoint $A_1, A_2, \dots \in \Sigma$ with $A = \bigcup_i A_i$, i.e., $\forall i \neq j$, $A_i \cap A_j = \emptyset$:*
$$P(A) = \sum_i P(A_i)$$

Definition 2.1.3 (Probability Space) *A probability space is defined as a triple (Ω, Σ, P), where*

- *Ω is the sample space, which is a non-empty set.*
- *Σ is the event space, which is a σ-algebra over Ω.*
- *P is a probability distribution over (Ω, Σ), which is a mapping from the event space to real numbers $P : \Sigma \mapsto \mathbb{R}$ that satisfies the properties in Definition 2.1.2.*

2.1.2 Conditioning and Independence

To reason about uncertainty, we need to introduce an important concept in probability theory: *conditional probability*. The conditional probability is the probability of an event given some evidence. Formally, we have the following definition:

Definition 2.1.4 (Conditional Probability) *Let (Ω, Σ, P) be a probability space, and $A, B \in \Sigma$ be two events. The conditional probability of A given B is*

$$P(A \mid B) := \frac{P(A \cap B)}{P(B)} \tag{2.1}$$

where $P(B) > 0$. When $P(B) = 0$, $P(A \mid B)$ is not defined.

Based on Definition 2.1.4, we introduce the *Bayes' rule*, which is the backbone for complex probabilistic reasoning.

Definition 2.1.5 (Bayes' Rule) *Let (Ω, Σ, P) be a probability space, and $A, B \in \Sigma$ be two events. Bayes' rule is mathematically stated as* *follows:*

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.2}$$

where $P(B) > 0$.

In addition to Bayes' rule, by extending Definition 2.1.4 we can also derive the so-called *chain rule* of probability:

Definition 2.1.6 (Chain Rule) *Let (Ω, Σ, P) be a probability space, and $A_1, A_2, \ldots A_k \in \Sigma$ be any number of events from Σ, the chain rule is written as:*

$$\begin{aligned} P(A_1 \cap A_2 \cap \cdots \cap A_k) = & P(A_1)P(A_2 \mid A_1) \cdots \\ & P(A_k \mid A_1 \cap A_2 \cap \cdots \cap A_{k-1}) \end{aligned} \tag{2.3}$$

where for any ordering of the events, the chain rule always holds.

The concept of conditioning is crucial for reasoning about uncertainty. However, for an event, not any other event in the event space can affect its probability. For such case, we introduce the notion of *independence*:

Definition 2.1.7 (Independence) *Let (Ω, Σ, P) be a probability space, and $A, B \in \Sigma$ be two events. We say A and B are independent, denoted by $A \perp\!\!\!\perp B$, when*

$$P(A \cap B) = P(A)P(B) \tag{2.4}$$

2.1.3 Random Variable

We now introduce the concept of Random Variables (RVs), which model unknown quantities as a function of *randomness* or *uncertainty*. More formally, we have the following definition:

Definition 2.1.8 (Random Variable) *Let (Ω, Σ, P) be a probability space and $\mathcal{O}$ be an output space, e.g., real numbers. A random variable X is defined as a function that maps the probability space to an output space: $X : \Omega \mapsto \mathcal{O}$.*

Throughout this thesis, we use uppercase letters to denote RVs, e.g., X, Y, Z, etc. The realization or outcome of an RV is denoted by a lowercase letter, e.g., x is a realization of X. Moreover, the state space of an RV is denoted by a calligraphic letter, e.g., $\mathcal{X}$ is the state space of X.

There are two types of RVs: *discrete*, which are variables with countably many values, and *continuous*, which are variables with uncountably infinite values. Generally, an RV is associated with a probability distribution that assigns probabilities to realizations of the RV. For discrete RVs, we can enumerate all the possible realizations and assigns a probability value for each possible realization. We introduce the Probability Mass Function (PMF) to define the probability distribution for discrete RVs.

Definition 2.1.9 (Probability Mass Function) *Let X be a discrete RV. The Probability Mass Function (PMF) of X is defined as a function $p_X(\cdot)$ that*

$$p_X(x) := P(X = x), \forall x \in \mathcal{X} \tag{2.5}$$

where $\mathcal{X}$ represents the state space of X, which is essentially a finite set of all possible values of X when X is a discrete RV. The subscript X of $p_X(x)$ is omitted if it is clear from the context, i.e., $p(x)$.

In contrast, for continuous RVs, we have uncountably infinite realizations, so it is impossible to assign a probability value for each realization. Instead of the PMF, we use the Probability Density Function (PDF) to define the probability distribution for continuous RVs. Before defining the PDF, we first introduce the Cumulative Distribution Function (CDF) that is important for further discussion.

Definition 2.1.10 (Cumulative Distribution Function) *Let (Ω, Σ, P) be a probability space and X be an RV over it. The Cumulative Distribution Function (CDF) of X is defined as a function $F_X(\cdot)$ that*

$$F_X(x) := P(X \leq x), \forall x \in \mathcal{X} \tag{2.6}$$

where the subscript X of $F_X(x)$ is omitted if it is clear from the context, i.e., $F(x)$.

Now we introduce the definition of the PDF.

Definition 2.1.11 (Probability Density Function) *Let X be a continuous RV and $F_X(\cdot)$ be the CDF of X. A Probability Density Function (PDF) of X is defined as a function $f_X(\cdot)$ that*

$$F_X(x) = \int_{-\infty}^{x} f_X(z)dz \tag{2.7}$$

When $F_X(x)$ is differentiable, we obtain the PDF $f_X(x)$ by computing *the derivative of $F_X(x)$ over x:*

$$f_X(x) = \frac{\partial F_X(x)}{\partial x} \tag{2.8}$$

where the subscript X is omitted if it is clear from the context, i.e., $f(x)$.

2.2 Multi-Dimensional Classification

In this section, we discuss the Multi-Dimensional Classification (MDC) problem in detail. We first introduce the classical MDC problem in a general setting and illustrate its relations with Multi-Class Classification (MCC) and Multi-Label Classification (MLC). Next, based on concepts from probability theory reviewed in the last section, we formalize the MDC problem in a probabilistic setting.

2.2.1 General Setting

Let $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ be a random vector of m feature variables and $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_d\}$ be a random vector of d class variables. We define the feature space as $\boldsymbol{\mathcal{X}} := \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m$ and the class space as $\boldsymbol{\mathcal{Y}} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_d$, where each feature space $\mathcal{X}_i$ $(1 \leq i \leq m)$ can be either continuous or discrete, and each class space $\mathcal{Y}_j$ $(1 \leq j \leq d)$ consists of r_j possible class values, i.e., $|\mathcal{Y}_j| = r_j$.

We assume observations to be i.i.d. realizations sampled from an unknown probability distribution P^* on space $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$, i.e., an observation of features $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ is a realization of $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ and an observation of class values $\mathbf{y} = \{y_1, y_2, \ldots, y_d\}$ is a realization of $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_d\}$.

Given a training set $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\} \subset \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ of N i.i.d. observations, MDC aims for learning a predictive model $h : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ that assigns predicted class values $\hat{\mathbf{y}} \in \boldsymbol{\mathcal{Y}}$ to each observation of features $\mathbf{x} \in \boldsymbol{\mathcal{X}}$. Thus, the output of h is a vector

$$\hat{\mathbf{y}} := h(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_d(\mathbf{x})) \subset \boldsymbol{\mathcal{Y}} \tag{2.9}$$

MDC can be seen as a generalization of Multi-Class Classification (MCC) and Multi-Label Classification (MLC). Specifically, when all the class variables are binary, i.e., $|\mathcal{Y}_j| = 2$ $(1 \leq j \leq d)$, the MDC problem turns into the Multi-Label Classification (MLC) problem. On the other hand, when there is only one class variable, i.e., $d = 1$, the problem becomes Multi-Class Classification (MCC).

2.2.2 Probabilistic Setting

In a probabilistic setting, the predictive model $h : \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{Y}}$ is learned in an indirected way and the MDC problem is typically solved in two stages: 1) learning an effective probabilistic model $s : \boldsymbol{\mathcal{X}} \rightarrow P(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\mathcal{X}})$; 2) constructing an efficient inference function $g : P(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\mathcal{X}}) \rightarrow \boldsymbol{\mathcal{Y}}$. Here, the model s is learned directly from the training set $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$. For each input $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, s outputs a conditional distribution $P(\boldsymbol{\mathcal{Y}} \mid \mathbf{x})$ which serves as the input of g for finding the Bayes-Optimal Prediction (BOP) $\hat{\mathbf{y}}$ w.r.t. a predefined loss function $\ell : \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \rightarrow \mathbb{R}_+$.

To achieve these, we parameterize a function $s = s_{\boldsymbol{\Theta}}$ by a probabilistic model with parameters $\boldsymbol{\Theta}$. Moreover, we mainly focus on probabilistic models that learn the conditional distribution $P(\boldsymbol{\mathcal{Y}} \mid \mathbf{x})$ by maximizing the Conditional Log-Likelihood (CLL) function:

$$
\begin{aligned}
\mathrm{CLL}(s_{\boldsymbol{\Theta}} \mid \mathcal{D}) &:= \log \prod_{l=1}^{N} p_{\boldsymbol{\Theta}}(\mathbf{y}^l \mid \mathbf{x}^l) \\
&:= \log \prod_{l=1}^{N} \frac{f_{\boldsymbol{\Theta}}(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_{\boldsymbol{\Theta}}(\mathbf{x}^l, \mathbf{y})}
\end{aligned}
\tag{2.10}
$$

where $f_{\boldsymbol{\Theta}}(\mathbf{x}, \mathbf{y})$ is the joint Probability Density Function (PDF) of $\mathbf{X}$ and $\mathbf{Y}$ represented by $s_{\boldsymbol{\Theta}}$, and $p_{\boldsymbol{\Theta}}(\mathbf{y} \mid \mathbf{x})$ is the conditional Probability Mass Function (PMF) of $\mathbf{Y}$ given $\mathbf{X}$ represented by $s_{\boldsymbol{\Theta}}$.

Optimizing the CLL function can be much more difficult than optimizing the Log-Likelihood (LL) function (Friedman et al., 1997). However, we consider it is worthy as in the *M-open* case (Bernardo and Smith, 2000), optimizing the LL may not converge to the best possible distribution as maximizing the CLL (Roos et al., 2005).

Regarding the inference function g, the BOP output $\hat{\mathbf{y}}$ is defined in a point-wise way, i.e., it is computed for each $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ individually:

$$
\hat{\mathbf{y}} = g(P(\boldsymbol{\mathcal{Y}} \mid \mathbf{x})) = \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \ell(\mathbf{y}, \mathbf{y}') P(\mathbf{y} \mid \mathbf{x}) \tag{2.11}
$$

Finding $\hat{\mathbf{y}}$ requires solving an optimization problem with different loss functions, which may lead to different BOPs (Dembczynski et al., 2012; Waegeman et al., 2014; Gil-Begue et al., 2021; Nguyen and Hüllermeier, 2021). In this thesis, we will mainly focus on two most commonly used loss functions that evaluate the predictive performance from different perspectives: the *Hamming loss* and the *subset zero-one loss*.

Concretely, the Hamming loss measures the fraction of wrongly predicted class values to the total number of class variables. Let $\mathbf{y} = (y_1, y_2, \ldots, y_d)$ be the ground truth and $\hat{\mathbf{y}} = (\hat{y}_2, \hat{y}_2, \ldots, \hat{y}_d)$ be the prediction of h, the Hamming loss function ℓ_{HL} is defined as[2]

$$\ell_{\mathrm{HL}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{d} \mathbb{1}_{y_i \neq \hat{y}_i} \tag{2.12}$$

2: For a predicate A, the expression $\mathbb{1}_A$ evaluates to 1 if A is true and to 0 if A is false.

Subset zero-one loss generalizes the well-know $0/1$ loss from MCC to MDC. This measure is more strict than the Hamming loss. It punishes a mistake on one class variable as hardly as mistakes on all class variables. Although it does not discriminate well between "partially correct", "almost correct" and "completely correct", this measure is obviously interesting and challenging with respect to class dependencies. The subset $0/1$ loss function is defined as

$$\ell_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbb{1}_{\mathbf{y} \neq \hat{\mathbf{y}}} \tag{2.13}$$

2.3 Bayesian Networks

In this section, we review the mathematical foundations and important theorems of Bayesian Networks (BNs). In Probabilistic Graphical Models (PGMs), especially BNs, graphs are used as a compact representation for modeling joint probability distributions. Before going deep into BNs, we first introduce some important concepts from the graph theory.

2.3.1 Graph Theory

Definition 2.3.1 (Graph) *A graph is defined as a pair $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of nodes or vertices, and $\mathbf{E}$ is a set of edges. An edge $E \in \mathbf{E}$ is defined as a tuple (V_i, V_j), $\forall V_i, V_j \in \mathbf{V}$.*

Definition 2.3.2 (Subgraph) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a graph. A subgraph of $\mathcal{G}$ is defined as a pair $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$, where $\mathbf{V}' \subseteq \mathbf{V}$ and $\mathbf{E}' \subseteq \mathbf{E}'$.*

There are two types of graphs: *directed*, which are graphs with directed edges, and *undirected*, which are graphs with undirected edges. For example, let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a graph, we say $\mathcal{G}$ is directed if all edges $E \in \mathbf{E}$ are directed, i.e., $\forall E = (V_i, V_j) \in \mathbf{E}$, the order of V_i and V_j in E matters. We denote a directed edge $E = (V_i, V_j)$ as $V_i \rightarrow V_j$. On the other hand, we call a graph undirected if all edges in the graph are undirected, i.e., the order of the nodes in any edge does not matter. We denote an undirected edge $E = (V_i, V_j)$ as $V_i - V_j$.

Throughout this thesis, we mainly focus on directed graphs. In the following, we introduce some notions to describe the node relationships in directed graphs.

Definition 2.3.3 (Parent and Child) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a directed graph and $V_i, V_j \in \mathbf{V}$ be two nodes from $\mathcal{G}$. If an edge $E = (V_i, V_j) \in \mathbf{E}$, i.e., $(V_i \to V_j) \in \mathbf{E}$, then V_i is a parent of V_j and V_j is a child of V_i.*

For each $V \in \mathbf{V}$, we use the symbol $\mathbf{Pa}(V)$ to denote the set of parents of V, and the symbol $\mathbf{Ch}(V)$ to denote the set of children of V.

The relationships of *parents* and *children* are defined based on a single edge. For relationships among multiple edges, we introduce the concept of *paths*.

Definition 2.3.4 (Path) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a directed graph and $V_1, V_2, \ldots, V_k$ ($k \geq 1$) be nodes from $\mathcal{G}$. A path from V_1 to V_k is defined as a sequence of nodes $\Pi = (V_1, V_2, \ldots, V_k)$ that*

$$\forall i \in \{1, 2, \ldots, k-1\} : (V_i \to V_{i+1}) \in \mathbf{E} \tag{2.14}$$

where we denote the length k of the path Π as $|\Pi| = k$.

Based on the definition of paths, we now define *cycles* and Directed Acyclic Graphs (DAGs) that are the basis of BNs.

Definition 2.3.5 (Cycle) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a directed graph and $V \in \mathbf{V}$ be a node from $\mathcal{G}$. A cycle is a path Π from V to itself if $|\Pi| \geq 2$.*

Definition 2.3.6 (Directed Acyclic Graph) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a directed graph. $\mathcal{G}$ is a Directed Acyclic Graph (DAG) if there is no cycle in $\mathcal{G}$.*

In DAGs, we can also naturally extend the direct node relationships (*parents* and *children*) to indirect node relationships (*ancestors* and *descendants*).

Definition 2.3.7 (Ancestor and Descendant) *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $V_i, V_j \in \mathbf{V}$ be two nodes from $\mathcal{G}$. If there exists a path from V_i to V_j, then V_i is an ancestor of V_j, and V_j is a descendant of V_i.*

For each $V \in \mathbf{V}$, we use the symbol $\mathbf{Anc}(V)$ to denote the set of ancestors of V, and the symbol $\mathbf{Desc}(V)$ to denote the set of descendants of V.

2.3.2 Definitions of Bayesian Networks

Definition 2.3.8 (Bayesian Network) *A Bayesian Network (BN) $\mathcal{B}$ is defined as a pair $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$, where $\mathcal{G}$ is a Directed Acyclic Graph (DAG) with a set of Random Variables (RVs) $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ as nodes. $\mathbf{\Theta}$ is a set of parameters $\mathbf{\Theta} = \{\Theta_i \mid 1 \leq i \leq m\}$ that encode local Conditional Probability Distributions (CPDs) $\{P(X_i \mid \mathbf{Pa}(X_i)) \mid 1 \leq i \leq m\}$, where $\mathbf{Pa}(X)$ denotes parent variables of X in $\mathcal{G}$. We also call the DAG $\mathcal{G}$ as a BN structure. The BN $\mathcal{B}$ defines a joint probability distribution $P_{\mathcal{B}}(\mathbf{X})$ for $\mathbf{X}$ which is factorized as*

$$P_{\mathcal{B}}(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X \mid \mathbf{Pa}(X)) \tag{2.15}$$

Equation 2.15 can be interpreted as the *chain rule* of probability theory, when the conditional (in)dependency assumptions guided by the structure $\mathcal{G}$ were made.

Exploiting the conditional (in)dependencies is key for efficient learning and inference in BNs, which directly relate to the factorization property in a BN structure.

Theorem 2.3.1 (Local Conditional Independence) *Let $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ be a Bayesian Network (BN) over $\mathbf{X}$ and $P_{\mathcal{B}}$ be the joint probability distribution defined by $\mathcal{B}$. Then we have the following local independencies[3]:*

$$X \perp\!\!\!\perp (\mathbf{Ndesc}(X) \backslash \mathbf{Pa}(X)) \mid \mathbf{Pa}(X), X \in \mathbf{X} \tag{2.16}$$

where $\mathbf{Ndesc}(X)$ represents the non-descendants of X in the BN structure $\mathcal{G}$. In words, each $X \in \mathbf{X}$ is conditionally independent of its non-descendants given its parents in the BN structure $\mathcal{G}$.

3: The operator $\perp\!\!\!\perp$ means "is independent of", e.g., for an expression $X \perp\!\!\!\perp Y \mid Z$, we say X is independent of Y given Z.

To further unveil (in)dependencies hold in BNs, we can employ the notion of *d-separation* (Pearl, 1989; Koller and Friedman, 2009). Concretely, an RV d-separates two other RVs if it blocks all the paths between them. We denote by $\mathcal{I}(\mathcal{G})$ the set of conditional independencies hold in a DAG $\mathcal{G}$ by d-separation. Moreover, a probability distribution P is faithful to a DAG $\mathcal{G}$ if $\mathcal{I}(P) = \mathcal{I}(\mathcal{G})$. In this setting, $\mathcal{G}$ is said to be a P-map for P. On the other hand, a DAG $\mathcal{G}$ is an I-map for a distribution P if $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$, and a DAG $\mathcal{G}'$ is an I-map for a DAG $\mathcal{G}$ if $\mathcal{I}(\mathcal{G}') \subseteq \mathcal{I}(\mathcal{G})$.

2.3.3 Learning Bayesian Networks

After specifying properties of BNs for modeling probability distributions, now we discuss how to learn BNs from data.

Formally, a BN is defined as a pair $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$, which contains two components: the BN structure $\mathcal{G}$ and the set $\mathbf{\Theta}$ of parameters of local CPDs induced by $\mathcal{G}$. Therefore, learning BNs can be naturally decomposed into two subtasks: *structure learning*, which corresponds to learning $\mathcal{G}$, i.e., determining the dependencies between variables from data, and *parameter learning*, which corresponds to estimating $\mathbf{\Theta}$ from data when $\mathcal{G}$ is already known.

Parameter Learning

For parameter learning, we assume an appropriate BN structure is already given, which can be either learned from a training set or predefined based on expert knowledge. Formally, let $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ be a BN over RVs $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$, and $\mathcal{D} = \{\mathbf{x}^l \mid 1 \leq l \leq N\}$ be a training set of N i.i.d. data points, the task of parameter learning is to estimate $\mathbf{\Theta}$ from $\mathcal{D}$.

The first common strategy in parameter learning of BNs is the Maximum Likelihood Estimation (MLE). The basic idea of MLE is to find a model, under which the observed data is most probable. This is achieved by maximizing the likelihood function. Concretely, given i.i.d. data $\mathcal{D}$, the likelihood function for $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ is given as

$$\begin{aligned}
\mathcal{L}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = p_{\mathbf{\Theta}}(\mathcal{D}) &= \prod_{l}^{N} p_{\mathbf{\Theta}}(\mathbf{x}^l) \\
&= \prod_{l} P(\mathbf{x}^l \mid \mathbf{\Theta}) \\
&= \prod_{l}^{N} \prod_{i}^{m} P(x_i^l \mid \mathbf{pa}(x_i)^l, \Theta_i) \\
&= \prod_{i}^{m} \mathcal{L}(\mathcal{G}_i, \Theta_i \mid \mathcal{D})
\end{aligned} \tag{2.17}$$

where $\mathcal{G}_i \subseteq \mathcal{G}$ denotes the subgraph consisting of only X_i and its parent set $\mathbf{Pa}(X_i)$.

Note that for continuous RVs $\mathbf{X}$, we need to use the joint density function $f_{\mathbf{\Theta}}$ to replace $p_{\mathbf{\Theta}}$. For notational convenience, we only discuss discrete RVs $\mathbf{X}$ in the following.

In practice, we often handle Equation 2.17 in the log space for computational simplicity, which gives rise to the Log-Likelihood

(LL) function:

$$\begin{aligned}\mathrm{LL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = \log \mathcal{L}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) &= \sum_{i}^{m} \sum_{l}^{N} \log P(x_i^l \mid \mathbf{pa}(x_i)^l, \Theta_i) \\ &= \sum_{i}^{m} \mathrm{LL}(\mathcal{G}_i, \Theta_i \mid \mathcal{D})\end{aligned} \tag{2.18}$$

It is clear to see the LL can be decomposed into local terms with respect to the BN structure. Therefore, we can obtain the MLE parameters $\hat{\Theta}_i$ for each X_i $(1 \leq i \leq m)$ as follows:

$$\hat{\Theta}_i = \arg\max_{\Theta_i} \mathrm{LL}(\mathcal{G}_i, \Theta_i \mid \mathcal{D}) \tag{2.19}$$

Another common approach in parameter learning is Bayesian parameter estimation. The basic idea is to treat the parameters as RVs, rather than some unknown constants. Concretely, the Bayesian approach assigns a prior distribution $P(\mathbf{\Theta})$ to $\mathbf{\Theta}$, then it updates $P(\mathbf{\Theta})$ with new observed data $\mathcal{D}$ via a likelihood function $\mathcal{L}(\mathbf{\Theta} \mid \mathcal{D})$, which finally results in a posterior parameter distribution $P(\mathbf{\Theta} \mid \mathcal{D})$, i.e.

$$P(\mathbf{\Theta} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{\Theta})P(\mathbf{\Theta})}{P(D)} = \frac{P(\mathcal{D} \mid \mathbf{\Theta})P(\mathbf{\Theta})}{\int P(\mathcal{D} \mid \mathbf{\Theta})P(\mathbf{\Theta})d\mathbf{\Theta}} \tag{2.20}$$

where $P(\mathcal{D})$ is also known as *marginal likelihood*, which means $\mathbf{\Theta}$ has been marginalized out.

In practice, a *conjugate* parameter prior for the likelihood function is often applied, i.e., the prior and the posterior belong to the same probability distribution family, such that the posterior $P(\mathbf{\Theta} \mid \mathcal{D})$ can be directly solved in closed-form. For example, for data that is assumed to be generated from a multinomial distribution, one could assign a *Dirichlet* prior to parameters, such that the parameter posterior will also be a Dirichlet distribution (Heckerman et al., 1995). By such assumptions, we can avoid computing the complex integral of $P(\mathcal{D})$ and solve Equation 2.20 efficiently.

Structure Learning

There are two main approaches for structure learning: *score-based* and *constraint-based*.

Score-based The score-based approach is also known as the "score-and-search" approach. This approach first defines a scoring function to evaluate how well a candidate BN structure fits the training data. Then, in the space of candidate BN structures, the score-based approach employs some search algorithms to find the optimal structure that maximizes the score. In this setting, the score-based approach is essentially a search procedure consisting of two parts: the definition of an appropriate scoring function and the development of a search algorithm.

Classical scoring functions can be classified into two families: *information-theoretic scores* and *Bayesian scores*. Concretely, given a candidate structure $\mathcal{G}$ parameterized by $\mathbf{\Theta}$, and a data set $\mathcal{D}$, we can write the general form of information-theoretic scoring functions as

$$\text{Score}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = \text{LL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) + \sigma(|\mathcal{D}|)\|\mathcal{G}\| \tag{2.21}$$

Note that to compute the LL function we first need to estimate the parameters for $\mathcal{G}$ using MLE.

It is clear to see that the LL function is also a valid scoring function. However, we cannot directly use it as it can be proved that a fully-connected graph will always give an optimal LL, which is probably overfitting the data. Therefore, we introduce the second term $\sigma(|\mathcal{D}|)\|\mathcal{G}\|$ in Equation 2.21 as a regularization term, which takes the model complexity into account and penalizes it. Here, $|\mathcal{D}|$ is the number of data points in $\mathcal{D}$, and $\|\mathcal{G}\|$ denotes the structure complexity, which is the number of parameters used in $\mathcal{G}$.

When $\sigma(|\mathcal{D}|) = 1$, the scoring function is known as the Akaike Information Criterion (AIC) (Akaike, 1974).

$$\text{AIC}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = \text{LL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) + \|\mathcal{G}\| \tag{2.22}$$

When $\sigma(|\mathcal{D}|) = \frac{\log|\mathcal{D}|}{2}$, the scoring function is known as the Bayesian Information Criterion (BIC) (Schwarz, 1978) or equivalently the Minimum Description Length (MDL) (Rissanen, 1978).

$$\text{BIC}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = \text{LL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) + \frac{\log|\mathcal{D}|}{2}\|\mathcal{G}\| \tag{2.23}$$

The Bayesian score is another family of scoring functions. In contrast to evaluating the model with only the MLE parameters in the LL function, the Bayesian score considers the whole parameter space through marginalization and implicitly penalizes the model complexity. Concretely, it defines a parameter prior $P(\mathbf{\Theta} \mid \mathcal{G})$ and then marginalizes over $\mathbf{\Theta}$:

$$\text{Bayes}(\mathcal{G} \mid \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) = \log \int P(\mathcal{D}, \mathbf{\Theta} \mid \mathcal{G}) P(\mathbf{\Theta} \mid \mathcal{G}) d\mathbf{\Theta} \tag{2.24}$$

After the scoring functions have been defined, the remaining problem is to develop a search algorithm to find the optimal structure. For learning large BN structures, general approaches are local search algorithms and approximative. *Greedy hill climbing* (Heckerman et al., 1995) is a common local search algorithm, it starts with an empty graph. At each step, it tests if a single operation, e.g., adding an edge, deleting an edge or reversing an edge, can improve the overall graph score. If so, it updates the graph, otherwise it makes other tests. Other approximative approaches include the *K2* algorithm (Cooper and Herskovits, 1992), *simulated annealing* (Heckerman et al., 1995), *tabu-search* (Glover, 1977), *genetic programming* (Larrañaga et al., 1996), etc.

There are also works investigating on exact approaches, i.e., finding a globally optimized structure. Previous studies include *dynamic programming* (Koivisto and Sood, 2004; Silander and Myllymäki, 2006), *integer linear programming* (Jaakkola et al., 2010; Cussens, 2011; Cussens and Bartlett, 2013; Cussens, 2020), *branch-and-bound* (Suzuki, 1996; de Campos et al., 2009; de Campos and Ji, 2011), etc. Currently, applying exact approaches to large-scale BN structures is still computationally expensive, which restricts their applications in practical cases.

Constraint-based Constraint-based approaches employ statistical independence tests to find a BN structure $\mathcal{G}$ that represents the same conditional (in)dependencies as in the underlying data distribution P^*, i.e., $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P^*)$. These approaches are theoretically sound and work well in large data sets. However, when the available data is relatively small, the conditional independence tests are usually inaccurate and make these approaches less reliable in practice.

2.3.4 Inference in Bayesian Networks

In this section, we turn our attention to the *inference* problem in BNs. Concretely, the inference problem is to perform reasoning tasks with a BN, i.e., answering queries using a BN. There are different types of inference tasks: *marginal inference*, *conditional inference*, *Maximum A-Posterior (MAP) inference*, etc. For classification problems, we are mainly interested in MAP and a special case of it: Most Probable Explanation (MPE).

Formally, assume that $P_{\mathcal{B}}$ is a BN distribution over RVs $\mathbf{X}$, where $\mathbf{X}$ is divided into three sets: *observed variables* ($\mathbf{E}$), *variables of interest* ($\mathbf{Q}$) and *hidden variables* ($\mathbf{H} = \mathbf{X} \backslash (\mathbf{E} \cup \mathbf{Q})$). The MAP output is the configuration $\mathbf{q}^*$ of $\mathbf{Q}$ that maximizes the posterior probability, given $\mathbf{E}$ and ignore $\mathbf{H}$:

$$\begin{aligned}\mathbf{q}^* &= \arg\max_{\mathbf{q}\in\mathcal{Q}} P_{\mathcal{B}}(\mathbf{q} \mid \mathbf{e}) \\ &= \arg\max_{\mathbf{q}\in\mathcal{Q}} P_{\mathcal{B}}(\mathbf{q}, \mathbf{e}) \\ &= \arg\max_{\mathbf{q}\in\mathcal{Q}} \int_{\mathcal{H}} P_{\mathcal{B}}(\mathbf{q}, \mathbf{e}, \mathbf{h}) d\mathbf{h}\end{aligned} \tag{2.25}$$

MPE is a special case of MAP, in which $\mathbf{H} = \emptyset$, i.e., all the variables that are not observed are variables of interest. Due to this, MAP inference is generally much harder than MPE inference (Park, 2002). The MPE configuration $\mathbf{q}^*$ of $\mathbf{Q}$ is

$$\begin{aligned}\mathbf{q}^* &= \arg\max_{\mathbf{q}\in\mathcal{Q}} P_{\mathcal{B}}(\mathbf{q} \mid \mathbf{e}) \\ &= \arg\max_{\mathbf{q}\in\mathcal{Q}} P_{\mathcal{B}}(\mathbf{q}, \mathbf{e})\end{aligned} \tag{2.26}$$

MAP inference is a challenging task, which has been proved as an NP-hard problem in general (Koller and Friedman, 2009). *Variable elimination* (Zhang and Poole, 1994) is a common exact inference algorithm for MAP. This algorithm proceeds by selecting an RV at a time according to a predefined order and replacing its CPDs with a new factor that summarizes the effect of all the selected RVs so far. Although the algorithm has exponential time complexity, it can be reasonably efficient in practice for BNs with low-treewidth structures if an appropriate elimination order was used. For complex BN structures, there are also methods that resort to approximate inference algorithms, such as *variational methods*, *Markov chain Monte Carlo* and *expectation propagation* (Beal, 2003; Koller and Friedman, 2009; Minka, 2001). These algorithms allow us to obtain useful solutions in large data sets.

Related Work 3

In the last chapter, we reviewed the theory behind probabilistic Multi-Dimensional Classification (MDC) and the mathematical tools used in modeling conditional distributions with Bayesian Networks (BNs).

In this chapter, we give an overview of existing work on probabilistic classification and Bayesian Network Classifiers (BNCs). In spite of its appealing advantages in uncertainty modeling and interpretability, probabilistic MDC has been rarely studied. To the best of our knowledge, there is no existing work that tries to optimize the Conditional Log-Likelihood (CLL) function in MDC tasks. In the existing literature, there are studies that optimize the CLL function in Multi-Label Classification (MLC) problems, which we consider are most related and will be discussed in Section 3.1.

Moreover, although current BNCs do not optimize the CLL function, we believe it is essential to review them as many of their theoretical foundations have guided the advancement in the field of probabilistic classification as well as inspired the work of this thesis. In Section 3.2 and Section 3.3, we will discuss them in detail for one-dimensional and multi-dimensional classification, respectively.

3.1 Probabilistic Multi-Label Classifiers

Binary Relevance (BR) is the most straightforward and intuitive approach to MLC, which decomposes an MLC problem with d labels into d independent binary classification problems by assuming all labels are independent, i.e., training a separate binary classifier $h_j(\cdot)$ for each label:

$$\begin{gathered}\hat{\mathbf{y}} = \{h_j(\mathbf{x}) \mid 1 \leq j \leq d\} \\ \text{where, } h_j(\mathbf{x}) = \underset{y_j \in \{0,1\}}{\arg\max}\, P(y_j \mid \mathbf{x})\end{gathered} \tag{3.1}$$

Due to its strong independence assumption, BR has often been criticized for being unable to yield satisfactory performance, especially in small data sets. Even though, BR has advantages on training and inference time. On the other hand, from a BN perspective, BR can be understood as a structure learner that aims for constructing an "empty" BN to model the conditional distribution $P(\mathbf{Y} \mid \mathbf{X})$. Here, "empty" means that the BN has no edges between all the class variables. Learning edges between any class variable and feature variables is essentially a binary classification problem shown in Equation 3.1. We show an intuitive example of this in Figure 3.1.

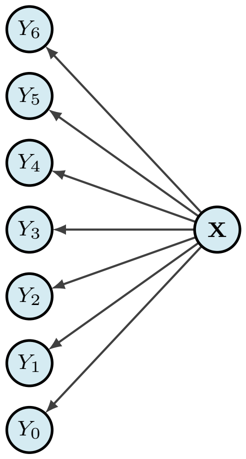


Figure 3.1: Implicit BN structure in BR for an MLC problem with 7 labels.

Classifier Chains (CCs) (Read et al., 2009; Dembczynski et al., 2010; Nam et al., 2017; Gerych et al., 2021; Read et al., 2021) are a family of multi-label classifiers based on chain-like directed PGMs. CC assumes that all the (in)dependencies among feature and class variables can be modeled with a BN $\mathcal{B} = (\mathcal{G}, \Theta)$ with the following structural constraints: 1) class variables cannot be parents of feature variables; 2) the parent set of any class variables contains all the feature variables. A graphical formulation of CC is shown in Figure 3.2. With this formulation, CC employs the chain rule to factorize $P_{\mathcal{B}}(\mathbf{y} \mid \mathbf{x})$ and train a separate classifier $h_j(\cdot)$ for each class variable Y_j $(1 \leq j \leq d)$, with the following inference procedure:

$$\begin{aligned} \hat{\mathbf{y}} &= \{h_j(\mathbf{x}) \mid 1 \leq j \leq d\} \\ \text{where, } h_j(\mathbf{x}) &= \underset{y_j \in \{0,1\}}{\arg\max}\, P(y_j \mid \mathbf{x}, \boldsymbol{\pi}_j) \end{aligned} \tag{3.2}$$

where $\mathbf{\Pi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{Y}$ contains all the parents of Y_j within $\mathbf{Y}$, and $\boldsymbol{\pi}_j$ is its configuration. Due to its chain-like directed structure, CC can be seen as a feature space augmentation approach. This means that all the previous labels in the chain will be considered as augmented features for the prediction of the current label.

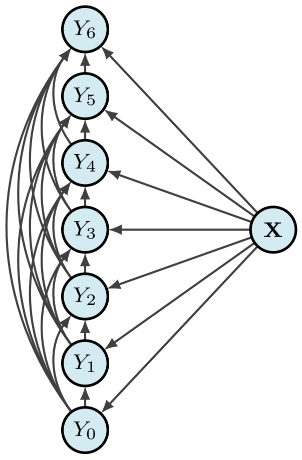


Figure 3.2: Implicit BN structure in CC for an MLC problem with 7 labels. The chain order is the natural order of the labels.

Class Powerset (CP) (also known as *label powerset* in the MLC literature) (Boutell et al., 2004; Tsoumakas and Katakis, 2007) is another simple but effective MLC approach. Different from the two multi-label classifiers we discussed above, CP does not try to decompose the MLC problem into smaller problems. Instead, it treats the labels as a whole and transforms each unique combination of labels existing in the training set into one of the classes of a new MCC task. Hence, CP obviously takes the label dependencies into account and reliably models $P(\mathbf{Y} \mid \mathbf{X})$ when the training data is sufficiently large. However, the difficulty of collecting sufficient data and handling the large number of new classes in the new MCC problem become the main challenges of CP.

3.2 Bayesian Network Classifiers

Bayesian Network Classifiers (BNCs) are specific BNs proposed by Friedman et al. (1997) to deal with one-dimensional classification problems. Formally, a BNC is defined as a pair $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$. RVs in the DAG $\mathcal{G}$ can be partitioned into two parts: a set of m feature variables $\mathbf{X} = \{X_1, X_2, \dots, X_m\}$, and a class variable Y.

For an input instance $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, a BNC outputs the Most Probable Explanation (MPE) of the class variable Y as the prediction $\hat{y}$

$$\hat{y} = \arg\max_{y} P(y \mid \mathbf{x}) = \arg\max_{y} P(\mathbf{x}, y) \tag{3.3}$$

which is also the Bayes-Optimal Prediction (BOP) under the $0/1$ loss function (Duda et al., 2001).

The factorization property in BNs also holds in BNCs

$$P_{\mathcal{B}}(\mathbf{X}, Y) = P(Y \mid \mathbf{Pa}(Y)) \prod_{i=1}^{m} P(X_i \mid \mathbf{Pa}(X_i)) \tag{3.4}$$

When a special structural assumption that $\mathbf{Pa}(Y) = \emptyset$ is made, the problem becomes to maximize

$$P(Y \mid \mathbf{X}) \propto P(\mathbf{X}, Y) = P(Y)P(\mathbf{X} \mid Y) \tag{3.5}$$

This assumption gives rise to the naive Bayes classifier (Maron and Kuhns, 1960; Minsky, 1961) and its augmented variants, which correspond to different factorizations of $P(\mathbf{X} \mid Y)$. Specially, the naive Bayes classifier is the simplest BNC, where the class variable Y is the parent of all feature variables $\mathbf{X}$ and there are no dependencies among $\mathbf{X}$. One can also introduce different levels of dependencies to $\mathbf{X}$, which lead to a family of augmented naive Bayes classifiers, such as selective naive Bayes (Langley and Sage, 1999), Tree-Augmented Networks (TANs) (Friedman et al., 1997) and k-dependence Bayesian classifiers (Sahami, 1996).

In a general setting, the class variable Y can also have parents. In such case, we have to search the Markov blanket of Y to solve the MPE inference problem in Equation 3.3. For a BN structure $\mathcal{G}$ over RVs $\mathbf{Z} = \mathbf{X} \cup \{Y\}$, the Markov blanket $\mathbf{MB}(Y)$ of Y is a set of RVs that make Y conditionally independent of all the other RVs in $\mathcal{G}$ given $\mathbf{MB}(Y)$, i.e.,

$$Y \perp\!\!\!\perp \mathbf{Z} \backslash \mathbf{MB}(Y) \mid \mathbf{MB}(Y) \tag{3.6}$$

Therefore, for predicting the MPE of Y, we only need to know the Markov blanket $\mathbf{MB}_Y$ of Y, i.e., $P(Y \mid \mathbf{X}) = P(Y \mid \mathbf{X} \cap \mathbf{MB}(Y))$.

Similar to learning approaches in general BNs, for learning the Markov blanket of a class variable and the structure of a BNC, major approaches include *score-based* methods (Cooper and Herskovits, 1992; Heckerman et al., 1995) and *constraint-based* methods (Spirtes et al., 2000). Standard approaches for learning parameters of BNCs include Maximum Likelihood Estimation (MLE) and Bayesian estimation.

3.3 Multi-Dimensional Bayesian Network Classifiers

The Multi-dimensional Bayesian Network Classifier (MBNC) was first proposed by van der Gaag and de Waal (2006), which extends the standard one-dimensional Bayesian Network Classifier (BNC) to deal with Multi-Dimensional Classification (MDC) problems. Bielza et al. (2011) further extended the initial MBNC definition of van der Gaag and de Waal (2006), which releases some structure constraints and can accept a greater number of BN structures. In this section, we adapt the definitions from Bielza et al. (2011) and Gil-Begue et al. (2021) to formally describe MBNCs.

Definition 3.3.1 (Multi-dimensional Bayesian Network Classifier) *A Multi-dimensional Bayesian Network Classifier (MBNC) is defined as a pair $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$, where $\mathcal{G}$ is a Directed Acyclic Graph (DAG) and $\mathbf{\Theta}$ is a set of parameters that encode local conditional distributions. In MBNCs, the DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is pair consisting of a set of nodes or vertices $\mathbf{V}$ and a set of edges $\mathbf{E}$.*

Note that under this definition standard BNCs are a special case of MBNCs where the number of class variables $d = 1$.

Definition 3.3.2 (Nodes in MBNCs) *The node set $\mathbf{V}$ can be partitioned into two parts: $\mathbf{V_Y} = \{Y_1, Y_2, \ldots, Y_d\}$ corresponds to d class variables $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_d\}$, and $\mathbf{V_X} = \{X_1, X_2, \ldots, X_m\}$ corresponds to m feature variables $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$.*

MBNCs (van der Gaag and de Waal, 2006) are essentially BNs with specific structural properties, which impose the following constraints on the edges in an MBNC.

Constraint 1 *There is no directed edge from feature variables to class variables.*

Constraint 2 *For any class variable $Y \in \mathbf{Y}$, there is a feature variable $X \in \mathbf{X}$ with a directed edge $(Y, X) \in \mathbf{E_{YX}}$ in the DAG $\mathcal{G}$.*

Constraint 3 *For any feature variable $X \in \mathbf{X}$ covered by $\mathcal{G}$, there is a class variable $Y \in \mathbf{Y}$ with a directed edge $(Y, X) \in \mathbf{E_{YX}}$ in the DAG $\mathcal{G}$.*

Then, the definition of edges in MBNCs can be formalized.

Definition 3.3.3 (Edges in MBNCs) *The edge set* $\mathbf{E}$ *in an MBNC can be partitioned into three subsets:* $\mathbf{E_X}$, $\mathbf{E_Y}$ *and* $\mathbf{E_{YX}}$, *which are defined as*

- *The set* $\mathbf{E_X} \subseteq \mathbf{V_X} \times \mathbf{V_X}$ *is composed of the edges between the feature variables, which creates a subgraph* $\mathcal{G}_\mathbf{X} = (\mathbf{V_X}, \mathbf{E_X})$, *called the feature subgraph, of* $\mathcal{G}$ *induced by* $\mathbf{V_X}$.
- *The set* $\mathbf{E_Y} \subseteq \mathbf{V_Y} \times \mathbf{V_Y}$ *is composed of the edges between the class variables, which creates a subgraph* $\mathcal{G}_\mathbf{Y} = (\mathbf{V_Y}, \mathbf{E_Y})$, *called the class subgraph, of* $\mathcal{G}$ *induced by* $\mathbf{V_Y}$.
- *The set* $\mathbf{E_{YX}} \subseteq \mathbf{V_Y} \times \mathbf{V_X}$ *is composed of the edges from class variables to feature variables, which creates a subgraph* $\mathcal{G}_\mathbf{YX} = (\mathbf{V}, \mathbf{E_{YX}})$, *called the bridge subgraph, of* $\mathcal{G}$ *connecting the class and feature variables.*

Learning MBNCs is essentially learning BNs under the structure constraints. In the following, we first discuss the structure learning problem in MBNCs, where the cardinality of a MBNC structure space is key to the complexity of the learning problem.

Proposition 3.3.1 (Bielza et al., 2011) *For an MBNC* $\mathcal{B} = (\mathcal{G}, \boldsymbol{\Theta})$, *the number of all possible DAGs* $\mathcal{G}$ *with* d *class variables and* m *feature variables,* $|\mathcal{G}|$, *is*

$$|\mathcal{G}| = S(d) \cdot 2^{dm} \cdot S(m) \tag{3.7}$$

where the function

$$S(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} S(n-i) \tag{3.8}$$

is Robinson's formula (Robinson, 1977) that counts the number of possible DAG structures of n *nodes, which is initialized as* $S(0) = S(1) = 1$. *Therefore, in Equation 3.7,* $S(d)$ *and* $S(m)$ *count the number of possible DAG structures in the class subgraph and feature subgraph, respectively.* 2^{dm} *is the number of possible DAG structures in the bridge subgraph.*

The number of all possible DAG structures in a standard BN over $d + m$ variables is $S(d + m)$, which is larger than the number in Equation 3.7 by a few orders of magnitude. Such gap can be explained by the structure constraint in Definition 3.3.1 that there is no edge from feature variables to class variables.

To date, several studies including score-based and constraint-based have investigated the structure learning problem of MBNCs (Gil-Begue et al., 2021). However, none of them focus on the parameter learning problem, as it has been done in general BNs using standard methods such as MLE and Bayesian estimation.

As we indicated earlier in Chapter 1, regarding parameter learning, optimizing the Conditional Log-Likelihood (CLL) function in supervised classification tasks is more favorable than optimizing the joint LL. However, it can also be much harder. First, as pointed by Friedman et al. (1997), the CLL function does not decompose with respect to the BN structure. Therefore, we cannot maximize it by maximizing local conditional probabilities independently. Second, conditional density functions is typically modeled by using Gaussian distributions (Gil-Begue et al., 2021). This would require a further structure constraint: continuous feature variables should not be the parent of discrete feature variables.

Generalized Bayesian Network Classifiers 4

In this chapter, we propose a general framework to solve Multi-Dimensional Classification (MDC) problems in a probabilistic setting, which we call Generalized Bayesian Network Classifiers (GBNCs).

GBNC takes inspiration from Multi-dimensional Bayesian Network Classifiers (MBNCs), trying to model the data distribution with a Bayesian Network (BN) under specific structural constraints. Moreover, GBNC has a flexible and versatile formalism, being able to deal with data containing both continuous and discrete features, and providing Bayes-Optimal Predictions (BOPs) under different loss functions.

In Section 4.1, we illustrate the research objectives of GBNCs. Then, in Section 4.2, we present the structural constraints made in GBNCs and theoretically prove that imposing these constraints does affect the possibility of finding the optimal structure. In Section 4.3 and Section 4.4, we discuss parameter and structure learning methods for GBNCs. Pruning rules that simplify the structure learning problem are introduced in Section 4.4.1. After that, in Section 4.5, we show how Bayes-Optimal Predictions (BOPs) w.r.t. the Hamming loss and the subset 0/1 loss can be realized by reducing them into different inference tasks in BNs. Finally, in Section 4.6, we discuss how to adapt GBNCs to handle mixed data.

4.1 Objectives

The main objective of GBNCs is to learn the conditional probability distribution $P(\mathbf{Y} \mid \mathbf{X})$ by maximizing the Conditional Log-Likelihood (CLL) in MDC problems. Recall that for a probabilistic model $s : \boldsymbol{\mathcal{X}} \to P(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\mathcal{X}})$, assume we have a training set $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$, the CLL is defined as

$$\begin{aligned} \mathrm{CLL}(s \mid \mathcal{D}) &:= \log \prod_{l=1}^{N} p_s(\mathbf{y}^l \mid \mathbf{x}^l) \\ &:= \log \prod_{l=1}^{N} \frac{f_s(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_s(\mathbf{x}^l, \mathbf{y}^l)} \end{aligned} \tag{4.1}$$

where $p_s(\mathbf{y} \mid \mathbf{x})$ is the PMF of the conditional distribution $P(\mathbf{Y} \mid \mathbf{X}$ represented by s, and $f_s(\mathbf{x}, \mathbf{y})$ is the PDF of the joint distribution $P(\mathbf{X}, \mathbf{Y})$ represented by s.

GBNC parameterizes s with a BN $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$. Moreover, it aims to solve MDC problems with mixed data, i.e., the data may contain both continuous and discrete features.

Formally, let $\mathbf{X}_c$ and $\mathbf{X}_d$ be the set of continuous features and the set of discrete features, respectively. Let $\mathbf{Y}$ be the set of class variables, we can write the joint density function as

$$\begin{aligned} f_{\mathcal{B}}(\mathbf{x}, \mathbf{y}) = & \prod_{X \in \mathbf{X}_c} f_{\mathcal{B}}(x \mid \mathbf{pa}(x)) \prod_{X \in \mathbf{X}_d} p_{\mathcal{B}}(x \mid \mathbf{pa}(x)) \\ & \prod_{Y \in \mathbf{Y}} p_{\mathcal{B}}(y \mid \mathbf{pa}(y)) \end{aligned} \tag{4.2}$$

Substituting Equation 4.2 to Equation 4.1, we can get the CLL expression for GBNCs:

$$\mathrm{CLL}(\mathcal{B} \mid \mathcal{D}) = \mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) := \log \prod_{l=1}^{N} \frac{f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)} \tag{4.3}$$

Learning GBNCs is essentially learning the underlying BN structures and parameters. We denote the structure space as $\mathscr{G}$ and the parameter space with $\mathbf{\Psi}$. The objectives of this study are to solve the structure and parameter learning problems of GBNCs, which can be written as

$$(\mathcal{G}^*, \mathbf{\Theta}^*) = \underset{(\mathcal{G}, \mathbf{\Theta}) \in \mathscr{G} \times \mathbf{\Psi}}{\arg\max} \ \mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) \tag{4.4}$$

$$\text{where, } \mathcal{G}^* = \underset{\mathcal{G} \in \mathscr{G}}{\arg\max}\, \mathrm{CLL}(\mathcal{G}, \mathbf{\Theta}^* \mid \mathcal{D}) \tag{4.5}$$

$$\mathbf{\Theta}^* = \underset{\mathbf{\Theta} \in \mathbf{\Psi}}{\arg\max}\, \mathrm{CLL}(\mathbf{\Theta} \mid \mathcal{G}, \mathcal{D}) \tag{4.6}$$

However, achieving the above objectives is far from obvious. As the CLL does not decompose as the LL does, there is no closed-form solution to learn parameters so that the CLL is maximized (Pernkopf and Bilmes, 2010; Roos et al., 2005), i.e., the optimization problem in Equation 4.6 can be computationally expensive. Moreover, without any structural constraint, the number of all possible $\mathcal{G}$ in $\mathscr{G}$ can be huge (Gil-Begue et al., 2021), which makes it impractical to optimally solve Equation 4.5.

4.2 Structural Constraints

In order to simplify the learning problem (Equation 4.4), we need to introduce structural constraints to reduce the search space of BN structures.

Recall the structural constraints made in Multi-dimensional Bayesian Network Classifiers (MBNCs):

Constraint 1 *There is no directed edge from feature variables to class variables.*

Constraint 2 *For any class variable $Y \in \mathbf{Y}$, there is a feature variable $X \in \mathbf{X}$ with a directed edge $(Y, X) \in \mathbf{E_{YX}}$ in the DAG $\mathcal{G}$.*

Constraint 3 *For any feature variable $X \in \mathbf{X}$ covered by $\mathcal{G}$, there is a class variable $Y \in \mathbf{Y}$ with a directed edge $(Y, X) \in \mathbf{E_{YX}}$ in the DAG $\mathcal{G}$.*

In GBNCs, we proceed with a new structural constraint:

Constraint 4 *There is no edge from class variables to feature variables.*

This is actually the opposite of Constraint 1. Note that we also do not make Constraint 2 and Constraint 3. This is because making these assumptions may adversely affect the assessment of the data-generating distribution. For example, if there are class variables whose parent sets contain no feature variables in the true distribution, assigning parent feature variables to these class variables may impair the predictive performance and lead to low-quality classifiers.

Denote by $\mathscr{G}^X$ all the BN structures satisfying Constraint 4. We have following properties.

Proposition 4.2.1 *The number of possible DAGs in $\mathscr{G}^X$ with m feature variables and d class variables is*

$$|\mathscr{G}^X| = S(m)2^{dm}S(d) \tag{4.7}$$

where $S(n) = \sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}2^{i(n-1)}S(n-i)$ is the Robinson's formula (Robinson, 1977) that counts the number of possible DAGs with n nodes, which is initialized as $S(0) = S(1) = 1$.

Proof. $S(m)$ and $S(d)$ count the number of possible DAGs for the feature subgraph and the class subgraph, respectively. 2^{dm} is the number of possible bridge subgraphs. Thus, the number of possible DAGs within $\mathscr{G}^X$ is the product of them. □

Proposition 4.2.2 *Let $\mathscr{G}$ be the set of all possible DAGs. For any $\mathcal{G} \in \mathscr{G}$, there is a $\mathcal{G}' \in \mathscr{G}^X$ such that $\mathcal{G}'$ is an I-map of $\mathcal{G}$.*

Based on these two propositions, it is clear that Constraint 4 effectively reduces the search space of possible BN structures, as well as theoretically guarantees the possibility of finding I-maps of true distributions.

Under Constraint 4, a structure $\mathcal{G} \in \mathscr{G}^X$ does not contain any edge $Y \to X$. Thus, for any $X \in \mathbf{X}$, $\mathbf{Pa}(X) \cap \mathbf{Y} = \emptyset$. Then, we can simplify the CLL function as

$$\begin{aligned}
\mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) &= \log \prod_{l=1}^{N} \frac{f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)} \\
&= \log \prod_{l=1}^{N} \frac{\prod_{X \in \mathbf{X}_c} f_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{X \in \mathbf{X}_d} P_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{X \in \mathbf{X}_c} f_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{X \in \mathbf{X}_d} P_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)} \\
&= \log \prod_{l=1}^{N} \frac{\prod_{X \in \mathbf{X}_c} f_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{X \in \mathbf{X}_d} P_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l)}{\prod_{X \in \mathbf{X}_c} f_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l) \prod_{X \in \mathbf{X}_d} P_{\mathcal{B}}(x^l \mid \mathbf{pa}(x)^l)} \frac{\prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)} \\
&= \log \prod_{l=1}^{N} \frac{\prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{Y \in \mathbf{Y}} P_{\mathcal{B}}(y^l \mid \mathbf{pa}(y)^l)} \\
&= \log \prod_{l=1}^{N} \frac{\prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)}
\end{aligned} \tag{4.8}$$

Moreover, we have the following proposition to further simplify the CLL function.

Proposition 4.2.3 *Assume a BN structure $\mathcal{G}$ satisfies Constraint 4, the CLL function can be simplified as*

$$\begin{aligned}
\mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) &= \log \prod_{l=1}^{N} \frac{f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)} \\
&= \log \prod_{l=1}^{N} \frac{\prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)} \\
&= \log \prod_{l=1}^{N} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l) \\
&= \sum_{l=1}^{N} \sum_{j=1}^{d} \log P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)
\end{aligned} \tag{4.9}$$

By using Proposition 4.2.3, for BN structures satisfying Constraint 4, we get a CLL function that can decompose into local CLLs over the class variables $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_d\}$, i.e., for each $j \in \{1, 2, \ldots, d\}$

$$\mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D}) = \sum_{l=1}^{N} \log P_j(y_j^l \mid \mathbf{pa}(y_j)^l) \tag{4.10}$$

where $\mathcal{G}_j \subseteq \mathcal{G}$ denotes the subgraph consisting of only Y_j and its parent set $\mathbf{Pa}(Y_j)$. $P_j(y_j^l \mid \mathbf{pa}(y_j)^l)$ denotes the conditional probability of y_j^l given $\mathbf{pa}(y_j)^l$ computed using the parameter set Θ_j, and $P_j(Y_j \mid \mathbf{Pa}(Y_j))$ is the conditional distribution of Y_j given $\mathbf{Pa}(Y_j)$ specified by Θ_j.

Based on this, optimizing the CLL is essentially optimizing the local CLLs for each class variable. Therefore, the learning problem in Equation 4.4 can be divided into two interrelated parts: 1) learning a BN structure $\mathcal{G}$ that maximizes the overall CLL function: $\mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D})$; 2) learning the parameter set Θ_j for each class conditional distribution $P_j(Y_j \mid \mathbf{Pa}(Y_j))$ ($1 \leq j \leq d$) by maximizing each local CLL function: $\mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D})$.

In the next two sections, we develop efficient algorithms to simultaneously solve the structure learning and parameter learning problems. For simplicity, we first focus on the case that there are only continuous feature variables. The case in which there are continuous and discrete features coexisting in data will be discussed in Section 4.6.

4.3 Parameter Learning

As indicated by Equation 4.10, learning the parameter set $\mathbf{\Theta}$ to best approximate the underlying $P(\mathbf{Y} \mid \mathbf{X})$ can be decomposed into a sequence of sub-tasks over class variables, i.e., learning Θ_j for each $P(Y_j \mid \mathbf{Pa}(Y_j))$ ($1 \leq j \leq d$).

Formally, assume that all feature variables are continuous, let $\mathcal{G}$ be a BN structure over feature variables $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ and class variables $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_d\}$. For each class variable Y_j ($1 \leq j \leq d$), the parent set $\mathbf{Pa}(Y_j)$ is composed of two parts: parent feature variables, which is denoted as $\mathbf{\Phi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{X}$, and parent class variables, which is denoted as $\mathbf{\Pi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{Y}$. Their realizations are denoted as $\boldsymbol{\phi}_j$ and $\boldsymbol{\pi}_j$, respectively. Then, we can write the CLL function as

$$
\begin{aligned}
\mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) &= \sum_{l=1}^{N} \sum_{j=1}^{d} \log P_j(y_j^l \mid \boldsymbol{\phi}_j^l, \boldsymbol{\pi}_j^l) \\
\mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D}) &= \sum_{l=1}^{N} \log P_j(y_j^l \mid \boldsymbol{\phi}_j^l, \boldsymbol{\pi}_j^l)
\end{aligned}
\tag{4.11}
$$

For any $1 \leq j \leq d$, we denote the number of values of the discrete class variable Y_j as r_j, and denote the number of configurations of $\mathbf{\Pi}_j$ as q_j, where $q_j = \prod_{k=1:Y_k \in \mathbf{\Pi}_j}^{d} r_k$. Given any possible configuration of $\boldsymbol{\pi}_{jk}$ ($1 \leq k \leq q_j$) for any Y_j ($1 \leq j \leq d$), we can write the conditional PMF of Y_j as

$$\begin{aligned} p_j^{\boldsymbol{\pi}_{jk}}(y_j \mid \boldsymbol{\phi}_j) &= P_j(Y_j = y_j \mid \boldsymbol{\Phi}_j = \boldsymbol{\phi}_j, \boldsymbol{\Pi}_j = \boldsymbol{\pi}_{jk}) \\ &= P_j^{\boldsymbol{\pi}_{jk}}(y_j \mid \boldsymbol{\phi}_j) \end{aligned} \tag{4.12}$$

With this expression, we can rewrite the CLL function as follows:

$$\begin{aligned} \mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) &= \sum_{l=1}^{N} \sum_{j=1}^{d} \log P_j(y_j^l \mid \boldsymbol{\phi}_j^l, \boldsymbol{\pi}_j^l) \\ &= \sum_{l=1}^{N} \sum_{j=1}^{d} \log \sum_{k=1}^{q_j} \mathbb{1}_{\boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk}} P_j^{\boldsymbol{\pi}_{jk}}(y_j^l \mid \boldsymbol{\phi}_j^l) \\ &= \sum_{l=1}^{N} \sum_{j=1}^{d} \sum_{\substack{k=1: \\ \mathbb{1}_{\boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk}}}}^{q_j} \log P_j^{\boldsymbol{\pi}_{jk}}(y_j^l \mid \boldsymbol{\phi}_j^l) \end{aligned} \tag{4.13}$$

In words, we will need $q = q_1 + q_2 + \cdots + q_d$ probabilistic models, one for each $P_j^{\boldsymbol{\pi}_{jk}}(y_j \mid \boldsymbol{\phi}_j)$ ($1 \leq j \leq d$, $1 \leq k \leq q_j$), to represent the joint conditional distribution $P_{\mathcal{B}}(\mathbf{Y} \mid \mathbf{X})$.

As each probabilistic model has its own parameter set $\Theta_j^{\boldsymbol{\pi}_{jk}}$, maximizing the CLL function is equivalent to independently maximizing $q = q_1 + q_2 + \cdots + q_d$ small CLL functions, i.e.,

$$\mathrm{CLL}(\mathcal{G}_{\boldsymbol{\Phi}_j}, \Theta_j^{\boldsymbol{\pi}_{jk}} \mid \mathcal{D}) = \sum_{\substack{l=1: \\ \mathbb{1}_{\boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk}}}}^{N} \log P_j^{\boldsymbol{\pi}_{jk}}(y_j^l \mid \boldsymbol{\phi}_j^l) \tag{4.14}$$

where $\mathcal{G}_{\boldsymbol{\Phi}_j} \subseteq \mathcal{G}$ denotes the subgraph consisting of Y_j and the parent set $\boldsymbol{\Phi}_j$.

For modeling any conditional distribution $P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \boldsymbol{\Phi}_j)$ ($1 \leq j \leq d$, $1 \leq k \leq q_j$), we need to specify the parent set $\boldsymbol{\Pi}_j$ and $\boldsymbol{\Phi}_j$, which can be extracted from the BN structure $\mathcal{G}$, and a training set $\mathcal{D}_j^{\boldsymbol{\pi}_{jk}}$, which can be extracted from $\mathcal{D}$ using Algorithm 1.

Algorithm 1: Extract training data

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$, class variable
    $Y_j$ ($1 \leq j \leq d$), parent set $\boldsymbol{\Pi}_j$, parent configuration $\boldsymbol{\pi}_{jk}$,
    parent set $\boldsymbol{\Phi}_j$
Ensure: $\mathcal{D}_j^{\boldsymbol{\pi}_{jk}}$
1: Initialize $\mathcal{D}_j^{\boldsymbol{\pi}_{jk}} \longleftarrow \emptyset$
2: for $l = 1, 2, \ldots, N$ do
3:    if $\boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk}$ then
4:       Update $\mathcal{D}_j^{\boldsymbol{\pi}_{jk}} \longleftarrow \mathcal{D}_j^{\boldsymbol{\pi}_{jk}} \cup \{(\boldsymbol{\phi}_j^l, y_j)\}$
5:    end if
6: end for
```

The local parameter learning problem is independent of what probabilistic models we use. In fact, we can employ any kind of probabilistic models to represent the conditional distributions for each class variable and its parent configurations. In this case, we can take the local CLL as the objective function and use gradient descent to train the model by maximizing the CLL. This essentially makes the modeling of any conditional distribution an one-dimensional classification problem, i.e., a well-trained probabilistic model for $P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \boldsymbol{\Phi}_j)$ should be capable of predicting the outcome of Y_j given an observation $\boldsymbol{\phi}_j$ of $\boldsymbol{\Phi}_j$. Note that a local probabilistic model for $P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \boldsymbol{\Phi}_j)$ only gives a prediction of Y_j for a particular configuration $\boldsymbol{\pi}_{jk}$, we still need to design a special inference mechanism to integrate information from all configurations of $\boldsymbol{\Pi}_j$. We will discuss the inference problem more thoroughly in Section 4.5.

Furthermore, except for Constraint 4, we do not set any additional structural constraint on the connections between class variables and feature variables. In GBNCs, we assume that any class variable is dependent on all feature variables. We also do not need to consider the dependencies between feature variables as they do not affect the optimization of the CLL (cf. Equation 4.9). Although in high-dimensional data some features can be redundant for a particular class variable, we believe that many optimization algorithms can be capable of identifying relevant features as we only need to handle one class variable at each time.

To sum up, given any parent configuration $\boldsymbol{\pi}_{jk}$ for Y_j $(1 \le j \le d)$, we can train a probabilistic classifier for $P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \boldsymbol{\Phi}_j)$, where $\boldsymbol{\Phi}_j = \mathbf{X}$. We describe this procedure in Algorithm 2.

Algorithm 2: Train local probabilistic model

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \le l \le N\}$, class variable
    $Y_j$ $(1 \le j \le d)$, parent set $\boldsymbol{\Pi}$, parent configuration $\boldsymbol{\pi}$, parent
    set $\boldsymbol{\Phi}$ ($\boldsymbol{\Phi} = \mathbf{X}$ by default if not specified)
Ensure: Local classifier $\mathsf{C}_j^{\boldsymbol{\pi}}$
 1: Extract $\mathcal{D}_j^{\boldsymbol{\pi}}$ using Algorithm 1 with input $\mathcal{D}$, $Y_j$, $\boldsymbol{\Pi}$, $\boldsymbol{\pi}$ and $\boldsymbol{\Phi}$
 2: Train $\mathsf{C}_j^{\boldsymbol{\pi}}$ using $\mathcal{D}_j^{\boldsymbol{\pi}}$ by maximizing the CLL
```

Once all local probabilistic models are trained based on a given BN structure $\mathcal{G}$, we gather them together to form the whole parameter set $\boldsymbol{\Theta}$ for $\mathcal{G}$. The whole parameter learning procedure with a given BN structure is summarized in Algorithm 3.

Algorithm 3: Parameter learning in $\mathcal{G}$

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$, BN structure $\mathcal{G}$
Ensure: Local classifiers $\mathsf{C}$
1: Initialize $\mathsf{C} \longleftarrow \emptyset$
2: for $j = 1, 2, \ldots, d$ do
3:     Extract $\mathbf{\Pi}_j$ and $\mathbf{\Phi}_j$ from $\mathcal{G}$
4:     for $k = 1, 2, \ldots, q_j$ do
5:         Train local classifier $\mathsf{C}_j^{\boldsymbol{\pi}_{jk}}$ using Algorithm 2 with input
           $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j$, $\boldsymbol{\pi}_{jk}$ and $\mathbf{\Phi}_j$
6:         Update $\mathsf{C} \longleftarrow \mathsf{C} \cup \mathsf{C}_j^{\boldsymbol{\pi}_{jk}}$
7:     end for
8: end for
```

4.4 Structure Learning

We now introduce an algorithm for learning the BN structure $\mathcal{G}$ by maximizing the CLL. Under Constraint 4 in GBNCs, we exploit an important property of the CLL: its function is decomposable and can be written in terms of the local class variables in $\mathcal{G}$, i.e.,

$$\mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) = \sum_{j=1}^{d} \mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D}) \tag{4.15}$$

Based on this, we treat the CLL as a decomposable scoring function. Note that to compute the local CLL scores $\mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D})$ $(1 \leq j \leq d)$, one needs to quantify the probabilistic relationships between Y_j and its parent sets $\mathbf{\Pi}_j$ and $\mathbf{\Phi}_j$.

In the last section, we demonstrate that any differentiable probabilistic model can be used to model the conditional distribution $P_j^{\boldsymbol{\pi}_{jk}}(Y_j \mid \mathbf{\Phi}_j)$ between Y_j and $\mathbf{\Phi}_j$ with respect to any configuration $\boldsymbol{\pi}_{jk}$ of $\mathbf{\Pi}_j$. Moreover, we do not consider the probabilistic relationships between feature variables and any class variable as well as the dependencies between feature variables. The feature selection is actually done in training local probabilistic classifiers.

In this setting, the quality of a probabilistic model that represent any local conditional distribution $P_j(Y_j \mid \mathbf{\Phi}_j)$ $(1 \leq j \leq d)$ is only related to determining the parent set $\mathbf{\Pi}_j$. Therefore, learning $\mathcal{G}^*$ that maximizes the CLL is equivalent to learning the best class subgraph $\mathcal{G}^*_{\mathbf{Y}}$ that maximizes the CLL.

From now on, we restrict ourselves to the structure learning of the class subgraph $\mathcal{G}_{\mathbf{Y}}$. We can rewrite the CLL for $\mathcal{G}_{\mathbf{Y}}$ as

$$\begin{aligned} \mathrm{CLL}(\mathcal{G}_{\mathbf{Y}}, \boldsymbol{\Theta} \mid \mathcal{D}) &= \sum_{j=1}^{d} \mathrm{CLL}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D}) \\ &= \sum_{j=1}^{d} \sum_{l=1}^{N} \sum_{k=1}^{q_j} P_j^{\boldsymbol{\pi}_{jk}}(y_j^l \mid \boldsymbol{\phi}_j^l) \end{aligned} \tag{4.16}$$

where $\mathcal{G}_{\mathbf{\Pi}_j} \subseteq \mathcal{G}_{\mathbf{Y}}$ denotes the subgraph consisting of Y_j and the parent set $\mathbf{\Pi}_j$. In the case that a class variable Y_j $(1 \leq j \leq d)$ has no parent class variables (i.e., $\mathbf{\Pi}_j = \emptyset$), $\mathrm{CLL}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D}) = \sum_{l=1}^{N} \log P_j(y_j^l \mid \boldsymbol{\phi}_j^l)$.

Once the scoring criterion has been defined, a common approach for solving the structure learning problem can be decomposed into two stages:

1. **Candidate Parent Set Identification** For each variable, construct a collection of potential candidate parent sets and their corresponding local scores.
2. **Structure Search** For each variable, find the best parent set from the collection of candidate parent sets in order to maximize the overall structure score as well as always keep the overall structure a DAG.

In our case, for the subproblem of candidate parent set identification, we do not set any structural constraint in the class subgraph $\mathcal{G}_{\mathbf{Y}}$. Thus, we set the candidate parent set for each class variable $Y \in \mathbf{Y}$ as the powerset of $\mathbf{Y} \backslash \{Y\}$ and compute their scores using Equation 4.16.

For the structure search problem, we use the formulation provided by the GOBNILP system (Cussens, 2011; Cussens, 2020), which is an exact score-based structure learning approach for BNs. GOBNILP formulates the score-based structure learning problem into an Integer Programming (IP) problem and uses state-of-the-art IP techniques to solve the optimization problem.

In our setting, for each class variable $Y_j \in \mathbf{Y}$ $(1 \leq j \leq d)$, any $\mathbf{\Pi} \subseteq \mathbf{Y} \backslash \{Y_j\}$ is allowed to be a potential candidate parent set for Y_j. We denote the collection of all candidate parent sets for Y_j as

$\mathcal{P}(j)$. Then, the IP formulation can be written as follows:

$$\begin{aligned}
&\textsc{Maximize} \sum_{j=1}^{d} \sum_{\mathbf{\Pi} \in \mathcal{P}(j)} \mathrm{CLL}(\mathcal{G}_{\mathbf{\Pi}}, \Theta_j \mid \mathcal{D}) I(Y_j, \mathbf{\Pi}) \\
&\textsc{Subject to} \sum_{\mathbf{\Pi} \in \mathcal{P}(j)} I(Y_j, \mathbf{\Pi}) = 1 \quad \forall j \in \{1, 2, \ldots, d\} \\
&\sum_{j \in \mathbf{v}} \sum_{\substack{\mathbf{\Pi} \in \mathcal{P}(j): \\ \mathbf{\Pi} \cap \mathbf{V} = \emptyset}} I(Y_j, \mathbf{\Pi}) = 1 \quad {\scriptstyle \substack{\forall \mathbf{v} \subseteq \{1,2,\ldots,d\}: \\ |\mathbf{v}|>1, \mathbf{V}=\{Y_k | k \in \mathbf{v}\}}} \\
&I(Y_j, \mathbf{\Pi}) \in \{0, 1\} \quad \forall j \in \{1, 2, \ldots, d\}, \mathbf{\Pi} \in \mathcal{P}(j)
\end{aligned} \tag{4.17}$$

where $I(Y_j, \mathbf{\Pi})$ is a binary indicator variable: $I(Y_j, \mathbf{\Pi}) = 1$ if $\mathbf{\Pi}$ is the parent set of Y_j in $\mathcal{G}_{\mathbf{Y}}$ and $I(Y_j, \mathbf{\Pi}) = 0$ otherwise.

4.4.1 Pruning Rules

The structure learning of BNs is known to be an NP-hard problem (Chickering et al., 2004). The subproblem of candidate parent sets identification is also unlikely to admit a polynomial-time (in the number of RVs) algorithm. It has been proved to be LOGSNP-hard (Koivisto, 2006).

A generic approach to simplify the structure learning problem is to set a maximum in-degree γ (the number of parents for each variable) and only compute the scores for candidate parent sets with at most γ variables. This approach indeed reduces the search space of BN structures. However, it may also lose many potentially optimal structures.

In GBNCs, we employ pruning rules on candidate parent sets identification from de Campos et al. (2018), which can greatly reduce the search space without loss of optimality.

Before we introduce the concrete pruning rules, let us define a penalized scoring metric based on the CLL:

$$\begin{aligned}
S(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) &= \mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) + \mathrm{PEN}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) \\
&= \sum_{j=1}^{d} (\mathrm{CLL}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D}) + \mathrm{PEN}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D})) \\
&= \sum_{j=1}^{d} S(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D})
\end{aligned} \tag{4.18}$$

where $\mathrm{PEN}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D})$ is a complexity penalization originating from the Bayesian Information Criterion (BIC) score (Schwarz, 1978), which is defined for Y_j and the parent set $\mathbf{\Pi}_j$:

$$\mathrm{PEN}(\mathcal{G}_{\mathbf{\Pi}_j}, \Theta_j \mid \mathcal{D}) = -\frac{\log N}{2}(r_j - 1)q_j \tag{4.19}$$

Based on this penalized scoring metric, the following pruning rules can be formalized:

Lemma 4.4.1 (de Campos and Ji, 2011) *Let $Y \in \mathbf{Y}$ be any class variable in $\mathcal{G}_{\mathbf{Y}}$, Θ be its parameter set, and $\mathbf{\Pi}'$ be a candidate parent set for Y. Suppose these exists a parent set $\mathbf{\Pi}$ such that $\mathbf{\Pi} \subset \mathbf{\Pi}'$ and $S(\mathcal{G}_{\mathbf{\Pi}}, \Theta \mid \mathcal{D}) \geq S(\mathcal{G}_{\mathbf{\Pi}'}, \Theta \mid \mathcal{D})$. Then $\mathbf{\Pi}'$ can be safely removed from the collection of candidate parent sets of Y.*

Lemma 4.4.2 (de Campos and Ji, 2011) *Let $Y \in \mathbf{Y}$ be any class variable in $\mathcal{G}_{\mathbf{Y}}$, Θ be its parameter set, and $\mathbf{\Pi}'$ be a candidate parent set for Y. Suppose these exists a parent set $\mathbf{\Pi}$ such that $\mathbf{\Pi} \subset \mathbf{\Pi}'$ and $S(\mathcal{G}_{\mathbf{\Pi}}, \Theta \mid \mathcal{D}) \geq \mathrm{PEN}(\mathcal{G}_{\mathbf{\Pi}'}, \Theta \mid \mathcal{D})$. Then $\mathbf{\Pi}'$ and all its supersets can be safely removed from the collection of candidate parent sets of Y.*

Here we say safely removing a candidate parent set means that we can still find the optimal structure even if we never use that parent set.

Based on the above pruning rules, we summarize a structure learning algorithm in Algorithm 4. Note that we focus on the class subgraph, the parent set of any Y_j ($1 \leq j \leq d$) is referred to the parent set within class variables $\mathbf{\Pi}_j$ only.

Algorithm 4: Structure learning of $\mathcal{G}_{\mathbf{Y}}$

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$
Ensure: Class subgraph $\mathcal{G}_{\mathbf{Y}}$
 1: Initialize a score dict $\mathsf{S}$    ▷ where $\mathsf{S}[j][\mathbf{\Pi}]$ stores the local score for $Y_j$ given the parent set $\mathbf{\Pi}$
 2: for $j = 1, 2, \ldots, d$ do
 3:   Initialize candidate parent sets $\mathsf{K}_j$ of $Y_j$ as the powerset of $\mathbf{Y} \backslash Y_j$
 4:   for $u = 1, 2, \ldots, |\mathsf{K}_j|$ do
 5:     Compute $\mathrm{PEN}(\mathcal{G}_{\mathsf{K}_{ju}}) \longleftarrow -\frac{\log N}{2}(r_j - 1)q_j$    ▷ here $q_j = \prod_{c=1:Y_c \in \mathsf{K}_{ju}}^{d} r_c$
 6:     if $\mathsf{K}_{ju}$ can be pruned then    ▷ using Lemma 4.4.2
 7:       Remove all supersets of $\mathsf{K}_{ju}$ from $\mathsf{K}$
 8:       continue
 9:     end if
10:     Initialize $\mathrm{S} \longleftarrow 0$
11:     for $k = 1, 2, \ldots, q_j$ do
12:       Extract $\mathcal{D}_j^{\mathsf{k}_{juk}}$ using Algorithm 1 with input $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j = \mathsf{K}_{ju}$, $\boldsymbol{\pi}_j = \mathsf{k}_{juk}$ and $\mathbf{\Phi} = \mathbf{X}$
13:       Train local classifier $\mathsf{C}_j^{\mathsf{k}_{juk}}$ using Algorithm 2 with input $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j = \mathsf{K}_{ju}$, $\boldsymbol{\pi}_j = \mathsf{k}_{juk}$ and $\mathbf{\Phi}_j = \mathbf{X}$
14:       Update $\mathrm{S} \longleftarrow \mathrm{S} + \mathsf{C}_j^{\mathsf{k}_{juk}}(\mathcal{D}_j^{\mathsf{k}_{juk}})$
15:     end for
16:     if $\mathsf{K}_{ju}$ can be pruned then    ▷ using Lemma 4.4.1
17:       continue
18:     end if
19:     $\mathsf{S}[j][\mathsf{K}_{ju}] \longleftarrow \mathrm{S}$
20:   end for
21: end for
22: Learn $\mathcal{G}_{\mathbf{Y}}$ using GOBNILP with input $\mathsf{S}$
```

4.5 Inference in GBNCs

In the previous sections, we discussed learning approaches to fit a BN structure to given data in GBNCs. To perform classification with GBNCs, i.e., inference with the underlying BN, we introduce an instance-based inference algorithm to find the Bayes-Optimal Prediction (BOP) in this section.

Recall that for a given loss function $\ell : \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}_+$ and an input instance $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, the BOP $\hat{\mathbf{y}}$ is defined as:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \ell(\mathbf{y}, \mathbf{y}') P(\mathbf{y} \mid \mathbf{x}) \tag{4.20}$$

In general, performing any inference approach on the underlying BN can produce valid predictions. However, to ensure the prediction is Bayes-optimal, we need to employ different inference approaches for different loss functions. In the following, we study two loss functions that are considered the most representative in the MLC and MDC literature.

4.5.1 Subset Zero-One Loss

We first consider the *subset zero-one loss*, which is a natural generalization of the $0/1$ loss function in binary classification to MLC and MDC scenarios. The definition is as follows:

$$\ell_{0/1}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbb{1}_{\mathbf{y} \neq \hat{\mathbf{y}}} \tag{4.21}$$

Then, we can rewrite the BOP with respect to the subset $0/1$ loss as:

$$\begin{aligned}
\hat{\mathbf{y}} &= \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \mathbb{1}_{\mathbf{y} \neq \mathbf{y}'} P(\mathbf{y} \mid \mathbf{x}) \\
&= \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} (1 - \mathbb{1}_{\mathbf{y} = \mathbf{y}'} P(\mathbf{y} \mid \mathbf{x})) \\
&= \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} (1 - \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \mathbb{1}_{\mathbf{y} = \mathbf{y}'} P(\mathbf{y} \mid \mathbf{x})) \\
&= \arg\min_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} (1 - P(\mathbf{y}' \mid \mathbf{x})) \\
&= \arg\max_{\mathbf{y}' \in \boldsymbol{\mathcal{Y}}} P(\mathbf{y}' \mid \mathbf{x})
\end{aligned} \tag{4.22}$$

This expression indeed aligns with the Most Probable Explanation (MPE) in Equation 2.26. In words, we treat $\mathbf{X}$ as observed variables and $\mathbf{Y}$ as query variables and then perform MPE inference on the structure $\mathcal{G}$ to find the BOP.

For any input instance $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, we can compute the conditional probabilities $P_j^{\boldsymbol{\pi}_{jk}}(y_j \mid \mathbf{x})$ using the local probabilistic classifiers learned using Algorithm 3 for each class variable Y_j $(1 \leq j \leq d)$. These conditional probabilities are collected and stored, constituting the Conditional Probability Tables (CPTs) of the class subgraph $\mathcal{G}_{\mathbf{Y}}$, i.e.,

$$P(y_{jt} \mid \boldsymbol{\pi}_{jk}) = P_j^{\boldsymbol{\pi}_{jk}}(y_j \mid \mathbf{x})$$
$$\text{where } j \in \{1,2,\ldots,d\}, t \in \{1,2,\ldots,r_j\} \text{ and } k \in \{1,2,\ldots,q_j\} \tag{4.23}$$

To this end, the MPE inference on $\mathcal{G}$ is equivalent to the MPE inference on $\mathcal{G}_{\mathbf{Y}}$. We summarize the procedure to perform the MPE inference as well as compute the BOP w.r.t. the subset 0/1 loss in Algorithm 5.

Algorithm 5: BOP w.r.t. the subset 0/1 loss

```
Require: Test set $\mathcal{T} = \{\mathbf{x}^l \mid 1 \leq l \leq L\}$, local probabilistic
    classifiers C, class subgraph $\mathcal{G}_{\mathbf{Y}}$
Ensure: The set of predictions $\mathcal{T}_{\mathbf{Y}} = \{\hat{\mathbf{y}}^l \mid 1 \leq l \leq L\}$
 1: Initialize $\mathcal{T}_{\mathbf{Y}} \longleftarrow \emptyset$
 2: Initialize CPTs of $\mathcal{G}_{\mathbf{Y}}$: $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow 0$, with all
    $j \in \{1,2,\ldots,d\}, t \in \{1,2,\ldots,r_j\}$ and $k \in \{1,2,\ldots,q_j\}$
 3: for $l = 1,2,\ldots,L$ do
 4:   for $j = 1,2,\ldots,d$ do
 5:     for $t = 1,2,\ldots,r_j$ do
 6:       for $k = 1,2,\ldots,q_j$ do
 7:         Update $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow \mathsf{C}_j^{\boldsymbol{\pi}_{jk}}(\mathbf{x}^l)$
 8:       end for
 9:     end for
10:   end for
11:   Perform MPE inference on $\mathcal{G}_{\mathbf{Y}}$ to compute
      $\hat{\mathbf{y}}^l \longleftarrow \arg\max_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} P(\mathbf{y} \mid \mathbf{x}^l)$
12:   Update $\mathcal{T}_{\mathbf{Y}} \longleftarrow \mathcal{T}_{\mathbf{Y}} \cup \{\hat{\mathbf{y}}^l\}$
13: end for
```

4.5.2 Hamming Loss

The second representative loss function we consider is the *Hamming loss*, which is formally defined as:

$$\ell_{\mathrm{HL}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^{d} \mathbb{1}_{y_j \neq \hat{y}_j} \tag{4.24}$$

It is clear to see that the Hamming loss is indeed a sum of the traditional 0/1 loss functions in terms of each class variable. Therefore, we can easily obtain its BOP as follows:

$$\begin{aligned}
\hat{\mathbf{y}} &= \{\hat{y}_j \mid 1 \leq j \leq d\} \\
\text{where, } \hat{y}_j &= \arg\min_{y' \in \mathcal{Y}_j} \sum_{y \in \mathcal{Y}_j} \mathbb{1}_{y \neq y'} P(y \mid \mathbf{x}) \\
&= \arg\min_{y' \in \mathcal{Y}_j} \sum_{y \in \mathcal{Y}_j} (1 - \mathbb{1}_{y = y'} P(y \mid \mathbf{x})) \\
&= \arg\min_{y' \in \mathcal{Y}_j} (1 - \sum_{y \in \mathcal{Y}_j} \mathbb{1}_{y = y'} P(y \mid \mathbf{x}))) \\
&= \arg\min_{y' \in \mathcal{Y}_j} (1 - P(y' \mid \mathbf{x})) \\
&= \arg\max_{y' \in \mathcal{Y}_j} P(y' \mid \mathbf{x})
\end{aligned} \tag{4.25}$$

In words, computing the BOP w.r.t. the Hamming loss is equivalent to performing Maximum A-Posterior (MAP) inference for each class variable $Y \in \mathbf{Y}$ in $\mathcal{G}_{\mathbf{Y}}$. Hence, we can exactly solve this problem by using the *variable elimination* algorithm (Zhang and Poole, 1994). We summarize this procedure in Algorithm 6.

Algorithm 6: BOP w.r.t. the Hamming loss

```
Require: Test set $\mathcal{T} = \{\mathbf{x}^l \mid 1 \leq l \leq L\}$, local probabilistic
    classifiers $\mathsf{C}$, class subgraph $\mathcal{G}_{\mathbf{Y}}$
Ensure: The set of predictions $\mathcal{T}_{\mathbf{Y}} = \{\hat{\mathbf{y}}^l \mid 1 \leq l \leq L\}$
 1: Initialize $\mathcal{T}_{\mathbf{Y}} \longleftarrow \emptyset$
 2: Initialize CPTs of $\mathcal{G}_{\mathbf{Y}}$: $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow 0$, with all
    $j \in \{1, 2, \ldots, d\}$, $t \in \{1, 2, \ldots, r_j\}$ and $k \in \{1, 2, \ldots, q_j\}$
 3: for $l = 1, 2, \ldots, L$ do
 4:   for $j = 1, 2, \ldots, d$ do
 5:     for $t = 1, 2, \ldots, r_j$ do
 6:       for $k = 1, 2, \ldots, q_j$ do
 7:         Update $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow \mathsf{C}_j^{\boldsymbol{\pi}_{jk}}(\mathbf{x}^l)$
 8:       end for
 9:     end for
10:   end for
11:   Initialize $\hat{\mathbf{y}}^l \longleftarrow \emptyset$
12:   for $j = 1, 2, \ldots, d$ do
13:     Perform variable elimination on $\mathcal{G}_{\mathbf{Y}}$ to compute
        $\hat{y}_j^l \longleftarrow \arg\max_{y \in \mathcal{Y}_j} P(y \mid \mathbf{x}^l)$
14:     Update $\hat{\mathbf{y}}^l \longleftarrow \hat{\mathbf{y}}^l \cup \hat{y}_j^l$
15:   end for
16:   Update $\mathcal{T}_{\mathbf{Y}} \longleftarrow \mathcal{T}_{\mathbf{Y}} \cup \{\hat{\mathbf{y}}^l\}$
17: end for
```

4.6 Mixed Data

In the previous sections, we discussed GBNCs in the case of only continuous features. In this section, we go one step further and introduce learning and inference approaches for GBNCs under *mixed data*, i.e., continuous and discrete features coexist in the data.

In general BNs, a common approach to handle mixed data is to discretize all the continuous variables and then all the existing learning and inference methods for discrete BNs can be applied. However, the learned model will be just an approximation to the real data distribution. Conversely, in GBNCs, we do not specially treat the continuous features. Instead, we treat the discrete feature variables as "special" class variables. Then, we can design learning and inference approaches based on the algorithms introduced in previous sections.

Formally, we divide the feature variables $\mathbf{X} = \{X_1, X_2, \dots, X_m\}$ into two disjoint sets: $\mathbf{X}_c$, which contains all continuous feature variables, and $\mathbf{X}_d$, which contains all discrete feature variables. Moreover, under Constraint 4 in GBNCs, any class variable $Y \in \mathbf{Y}$ can have parents from either $\mathbf{X}_c$, $\mathbf{X}_d$ or $\mathbf{Y} \backslash \{Y\}$. To simplify notations, for any Y_j $(1 \leq j \leq d)$, we denote the three parent sets as $\mathbf{\Phi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{X}_c$, $\mathbf{\Lambda}_j = \mathbf{Pa}(Y_j) \cap \mathbf{X}_d$ and $\mathbf{\Pi}_j = \mathbf{Pa}(Y_j) \cap \mathbf{Y}$, respectively. Possible configurations of the three parent sets are denoted as $\boldsymbol{\phi}_j$, $\boldsymbol{\lambda}_j$ and $\boldsymbol{\pi}_j$, respectively. In addition to q_j which represents the number of configurations of $\mathbf{\Pi}_j$, we use e_j to denote the number of configurations of $\mathbf{\Lambda}_j$.

Then, we can rewrite the CLL function as follows:

$$\begin{aligned}
\mathrm{CLL}(\mathcal{G}, \mathbf{\Theta} \mid \mathcal{D}) &= \sum_{j=1}^{d} \mathrm{CLL}(\mathcal{G}_j, \Theta_j \mid \mathcal{D}) \\
&= \sum_{j=1}^{d} \sum_{l=1}^{N} \log P_j(y_j^l \mid \boldsymbol{\phi}_j^l, \boldsymbol{\lambda}_j^l, \boldsymbol{\pi}_j^l) \\
&= \sum_{j=1}^{d} \sum_{l=1}^{N} \log \sum_{k=1}^{q_j} \sum_{n=1}^{e_j} \mathbb{1}_{\boldsymbol{\lambda}_j^l = \boldsymbol{\lambda}_{jn} \wedge \boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk}} P_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})}(y_j^l \mid \boldsymbol{\phi}_j^l)
\end{aligned} \tag{4.26}$$

It is clear to see that the difference between Equation 4.13 and Equation 4.26 is due to the introduction of discrete features $\mathbf{X}_d$. Essentially, by treating them as "special" class variables, we will only need $e_j \cdot q_j$ probabilistic classifiers $\{P_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})}(Y_j \mid \mathbf{\Phi}_j) \mid 1 \leq n \leq e_j, 1 \leq k \leq q_j\}$ to model the conditional distribution $P(Y_j \mid \mathbf{X})$ for any Y_j $(1 \leq j \leq d)$. Hence, training these probabilistic classifiers by maximizing the decomposed CLL will maximize the overall

CLL. For any new instance $\mathbf{x} \in \boldsymbol{\mathcal{X}}$, the conditional probabilities computed using these local classifiers will be summarized in the subgraph $\mathcal{G}_{\mathbf{DY}}$ (the DAG that removes all continuous feature variables $\mathbf{X}_c$ and all edges connected to them from $\mathcal{G}$), which can then be used to perform various inference tasks.

To this end, the parameter and structure learning of $\mathcal{G}_{\mathbf{DY}}$ is essentially similar to the ones of $\mathcal{G}_{\mathbf{Y}}$ with only continuous features. For better readability, we shift the detailed algorithms for them to the appendix. Next, we study how the BOP computation can be realized with mixed data.

Concretely, for any new instance $\mathbf{x} = \{x_1, x_2, \ldots, x_m\} \in \boldsymbol{\mathcal{X}}$, we partition the values into continuous values $\mathbf{x}_c$ and discrete values $\mathbf{x}_d$. Then, for each class variable Y_j ($\leq j \leq d$), we extract the corresponding local classifiers based on $\mathbf{x}_d$, i.e., $\{P_j^{(\boldsymbol{\lambda}_j^l, \boldsymbol{\pi}_{jk})}(Y_j \mid \boldsymbol{\phi}_j) \mid 1 \leq k \leq q_j\}$, where $\boldsymbol{\lambda}_j^l = \{x_b^l \mid 1 \leq b \leq m \wedge X_b \in \boldsymbol{\Lambda}_j\}$. In other words, this procedure can be understood as that we are extracting information from the discrete feature variables. After that, we can safely ignore $\mathbf{X}_d$ as their effects have been taken into account. That is, after all conditional probabilities have been computed and collected, we compute the BOP by performing a specific inference task on the class subgraph $\mathcal{G}_{\mathbf{Y}}$ based on the selected loss function. In Algorithm 7, we summarize the inference procedure under mixed data w.r.t. the subset 0/1 loss.

Algorithm 7: BOP w.r.t. the subset 0/1 loss under mixed data

```
Require: Test set $\mathcal{T} = \{\mathbf{x}^l = (\mathbf{x}_c^l, \mathbf{x}_d^l) \mid 1 \leq l \leq L\}$, local
    probabilistic classifiers $\mathsf{C}$, the subgraph $\mathcal{G}_{\mathbf{DY}}$
Ensure: The set of predictions $\mathcal{T}_{\mathbf{Y}} = \{\hat{\mathbf{y}}^l \mid 1 \leq l \leq L\}$
 1: Initialize $\mathcal{T}_{\mathbf{Y}} \longleftarrow \emptyset$
 2: Initialize CPTs of $\mathcal{G}_{\mathbf{Y}}$: $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow 0$, with all
    $j \in \{1, 2, \ldots, d\}$, $t \in \{1, 2, \ldots, r_j\}$ and $k \in \{1, 2, \ldots, q_j\}$
 3: for $l = 1, 2, \ldots, L$ do
 4:     for $j = 1, 2, \ldots, d$ do
 5:         Extract $\boldsymbol{\lambda}_j^l \longleftarrow \{x_b^l \mid 1 \leq b \leq m \wedge X_b \in \boldsymbol{\Lambda}_j\}$
 6:         for $t = 1, 2, \ldots, r_j$ do
 7:             for $k = 1, 2, \ldots, q_j$ do
 8:                 Update $P(y_{jt} \mid \boldsymbol{\pi}_{jk}) \longleftarrow \mathsf{C}_j^{(\boldsymbol{\lambda}_j^l, \boldsymbol{\pi}_{jk})}(\mathbf{x}_c^l)$
 9:             end for
10:         end for
11:     end for
12:     Perform MPE inference on $\mathcal{G}_{\mathbf{Y}}$ to compute
        $\hat{\mathbf{y}}^l \longleftarrow \arg\max_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} P(\mathbf{y} \mid \mathbf{x}^l)$
13:     Update $\mathcal{T}_{\mathbf{Y}} \longleftarrow \mathcal{T}_{\mathbf{Y}} \cup \{\hat{\mathbf{y}}^l\}$
14: end for
```

Experiments & Discussion 5

In this chapter, we provide key implementation details of the algorithms for GBNCs described in the last chapter. Moreover, we demonstrate the applicability of GBNCs in benchmark data sets of different characteristics, including *tabular data* and *image data*. We evaluate the performance of GBNCs against baseline probabilistic multi-dimensional classifiers and validate the superiority and versatility of GBNCs in different scenarios.

In Section 5.1, we introduce the benchmark data sets and present evaluation metrics used in our experiments. Then we briefly introduce three baselines to which we compare GBNCs. Last, we present the implementation details and hyper-parameter settings for all approaches. In Section 5.2, we test GBNCs on tabular data, image data and mixed data, respectively. We compare GBNCs to the three baseline approaches based on different evaluation metrics. Finally, in Section 5.3, we perform case studies on the learned BN structures of GBNCs and qualitatively discuss their validity on different data sets.

5.1 Experimental Setup

In this section, we start by introducing the detailed information of the benchmark data sets, including *tabular data* and *image data*. Some of them may contain both continuous and discrete signals, which we specially treat them and call them *mixed data*. In Section 5.1.2, we present evaluation metrics used in our experiments. The configuration and hyper-parameter setting of GBNCs and the baseline methods are discussed in Section 5.1.3.

5.1.1 Benchmark Data Sets

Tabular Data Sets

We first consider the application of GBNCs on data that is organized in a table with rows and columns. We exploit the benchmark data sets[1] that are commonly used in the MDC literature and collected by Jia and Zhang (2022a).

1: The data sets can be downloaded from: `http://palm.seu.edu.cn/zhangml/Resources.htm`

Table 5.1 summarizes the detailed statistics of the tabular data sets. The number of class variables in each data set ranges from two to sixteen. The meaning of each column is

- #CVs: the number of class variables in each data set.

Data Set	#CVs	#Samples	#States/CV	#Features
Edm	2	154	3	$16n$
Jura	2	359	4,5	$9n$
Enb	2	768	2,4	$6n$
Voice	2	3136	4,2	$19n$
Song	3	785	3	$98n$
Flickr	5	12198	$3, 4, 3, 4, 4$	$1536n$
Fera	5	14052	6	$136n$
WQplants	7	1060	4	$16n$
WQanimals	7	1060	4	$16n$
Rf1	8	8987	$4, 4, 3, 4, 4, 3, 4, 3$	$64n$
Pain	10	9734	$2, 5, 4, 2, 2, 5, 2, 5, 2, 2$	$136n$
Disfa	12	13095	$5, 5, 6, 3, 4, 4, 5, 4, 4, 4, 6, 4$	$136n$
WaterQuality	14	1060	4	$16n$
Oes97	16	334	3	$263n$
Oes10	16	403	3	$298n$
Scm20d	16	8966	4	$61n$
Scm1d	16	9803	4	$280n$
Adult	4	18419	$7, 7, 5, 2$	$5n, 5x$
Default	4	28779	$2, 7, 4, 2$	$14n, 6x$
Thyroid	7	9172	$5, 5, 3, 2, 4, 4, 3$	$7n, 22x$

Table 5.1: Statistics of the tabular benchmark data sets.

- #Samples: the number of samples in each data set.
- #States/CV: the number of states for each class variable. For example, the Flickr data set has five class variables, which have $3, 4, 3, 4, 4$ states respectively. If all class variables contain the same number of states, only this number is reported.
- #Features: the number of feature variables in each data set, where n means a feature is numeric and x means a feature is discrete.

Among all the tabular data sets in Table 5.1, the last three data sets, Thyroid, Adult and Default, contain both continuous and discrete features, which are used as *mixed data* for specific analysis.

Image Data Sets

For image data, there is no commonly accepted benchmark data sets in MDC research. Therefore, we propose to convert some MLC image data sets into MDC.

We first consider the PASCAL VOC 2007[2] data set proposed by Everingham et al. (2007). This is a standard MLC image data set, consisting of 9,963 natural images with 20 labels. The data set is

2: The data set can be downloaded from: `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/`

CV	States
Person	no person, person
Animal	no animals, bird, cat, cow, dog, horse, sheep
Vehicle	no vehicles, aeroplane, bicycle, boat, bus, car, motorbike, train
Indoor	no indoor objects, bottle, chair, dining table, potted plant, sofa, tv/monitor

Table 5.2: Characteristics of the PASCAL VOC 2007 data set in an MDC setting.

equally split into two parts: 50% for training and validation, and 50% for testing.

We pre-process PASCAL VOC 2007 to make it applicable to MDC. First, we introduce four CVs based on the 20 labels in the original data set. Second, we add an extra state to indicate there is no such object for each CV. Third, we remove the images that may take multiple states for a class variable (e.g. an image that contains both birds and cats). Table 5.2 summarizes the information of the class variables of the processed PASCAL VOC 2007 data set.

5.1.2 Evaluation Metrics

In general, the *Hamming loss* and the *subset 0/1 loss* introduced in Chapter 2 can be directly used to evaluate the learned classifiers. In order to be consistent with the literature, we introduce two accuracy scores based on the two loss functions: the Hamming Score (HS) and the Exact Match Score (EMS). They have been used in a series of previous studies (Zhu et al., 2005; Bielza et al., 2011; Jia and Zhang, 2022a; Jia and Zhang, 2022b).

HS and EMS are essentially equivalent to the Hamming loss and the subset 0/1 loss, respectively. Concretely, given a test set $\mathcal{T} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq L\}$ and an MDC classifier as $h : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$. Let $\{\hat{\mathbf{y}}^l \mid 1 \leq l \leq L\}$ be the prediction given by h, the HS and EMS are defined as

$$\begin{aligned} \mathrm{HS}(h) &= \frac{1}{L}\sum_{l=1}^{L}\frac{1}{d}\sum_{j=1}^{d}\mathbb{1}_{y_j^l=\hat{y}_j^l} \\ &= \frac{1}{L}\sum_{l=1}^{L}(1-\ell_{\mathrm{HL}}(\mathbf{y}^l,\hat{\mathbf{y}}^l)) \end{aligned} \tag{5.1}$$

$$\begin{aligned} \mathrm{EMS}(h) &= \frac{1}{L}\sum_{l=1}^{L}\mathbb{1}_{\mathbf{y}^l=\hat{\mathbf{y}}^l} \\ &= \frac{1}{L}\sum_{l=1}^{L}(1-\ell_{0/1}(\mathbf{y}^l,\hat{\mathbf{y}}^l)) \end{aligned} \tag{5.2}$$

5.1.3 Probabilistic MDC Baselines

Due to a lack of work on probabilistic MDC, there are no probabilistic classifiers to which GBNCs can directly compare. Moreover, as we consider MDC data sets that contain both continuous and discrete features, MBNCs also cannot be directly applied.

For this reason, we adapt three probabilistic multi-label classifiers[3] introduced in Chapter 3 to MDC problems: Binary Relevance (BR), Classifier Chain (CC), Class Powerset (CP). These methods provide us probabilistic MDC baselines to which we compare GBNCs.

3: For an implementation in Python, please see: `http://scikit.ml/api/skmultilearn.html`.

The three classifiers can be easily extended from MLC to MDC. Specifically, for an MDC problem with d class variables, BR assumes that all class variables are independent and decomposes the MDC problem into d MCC problems w.r.t. each class variable. CC takes the probabilistic relationships between class variables into account. It trains d MCC base classifiers, one for each class variable, in a predefined order so that each base classifier incorporates the prediction of previous class variables in the chain as additional features. Last, CP transforms an MDC problem into a new MCC problem. It considers each unique combination of class values existing in the training set as a new meta-class and trains an MCC classifier over the meta-classes. Such transformation naturally takes the dependencies between class variables into account.

Note that all three methods can be instantiated with different base MCC classifiers. To ensure a fair comparison, we equip BR, CC and GBNCs with the same base classifier. For CP, there is only one MCC classifier being trained, in which we use the same MCC learner as in other approaches.

5.1.4 Implementation Details

For tabular data sets, we perform experiments on all methods with two base classifiers[4]: *logistic regression* and *naive Bayes*. For each method, we conduct 10-fold cross-validation in each data set and record the mean values and standard deviations of all evaluation metrics. We also conduct the *Wilcoxon signed-rank test* (Demsar, 2006) with a significance level of 0.05 to test the statistical significance of the superiority or inferiority of GBNCs over other methods.

4: We use the default setting of logistic regression classifiers with SGD training and Gaussian naive Bayes classifiers. Further details can be found in: (`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html`) and (`https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html`), respectively.

As BR, CC and CP cannot directly handle discrete features, for mixed tabular data sets, when employing logistic regression as the base classifier, we use one-hot encoding to encode discrete features, and train local classifiers of the three baselines on the combination of continuous features and one-hot encodings. When employing naive Bayes as the base classifier, we use a mixed naive Bayes[5] as the base classifier for the three baselines, implementing categorical naive Bayes for discrete feature variables, and Gaussian naive Bayes for continuous feature variables. For additional features from previous predictions in CC, we also implement categorical naive Bayes for them.

5: For an implementation based on scikit-learn, please see: `https://github.com/remykarem/mixed-naive-bayes`.

For image data sets, we employ the Convolutional Neural Network (CNN) as the base classifier. Specifically, ResNet-18[6] (He et al., 2016) with weights pretrained on the ImageNet data set (Deng et al., 2009) is used in the experiments on all the image data sets. We freeze all layers of ResNet-18 except for the last fully-connected layer, and retrain it on the PASCAL VOC 2007 data set. The model is trained for 10 epochs. We use stochastic gradient descent to train the model with a batch size of 128 and a learning rate of 0.01. Note that for CC, there is no way to use predicted class values as additional features in image data. Therefore, we remove CC from the comparison of all approaches under image data. Since training deep CNNs is much more complex and time-consuming, we do not perform cross-validation for experiments on image data.

6: For an implementation in PyTorch, please see `https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html`.

For implementing the baselines, we randomly generated 10 chain orders for CC, and pick the best order in terms of the exact match score. In experiments, we often find the original order of the class variables (i.e., $Y_1, Y_2, \ldots, Y_d$) often achieves better performance than randomly generated orders in many data sets. This may be caused by some pre-processing from the data source. Therefore, we also add the original order into the comparison (i.e., 11 chain orders in total) and report the best performance as the result of CC.

We implement all methods in Python and use the PyTorch (Paszke et al., 2019) framework to implement neural networks.

5.2 Experimental Results

In this section, we evaluate the performance of each method in terms of HS and EMS presented in Section 5.1.2. We show the versatility of GBNCs by employing different inference methods to compute BOPs, and compare GBNC to other probabilistic MDC baselines.

5.2.1 Tabular Data

We first present experimental results for tabular data sets that contain only continuous features. Table 5.3-5.10 summarize the detailed experimental results in terms of Hamming scores and exact match scores for all probabilistic approaches with the two base classifiers: *logisitic regression* and *naive Bayes*. Besides, we also report the performance of GBNCs with the two different inference algorithms (Algorithm 5 and Algorithm 6) that compute BOPs w.r.t. the subset 0/1 loss and the Hamming loss, respectively. In Table 5.3-5.10, the best performance is highlighted by showing in boldface. Moreover, we conduct the *Wilcoxon signed-rank test* (Demsar, 2006) with a significance level of 0.05 to test the statistical significance of the superiority or inferiority of GBNC over other methods.

Quantitative Comparison of Probabilistic MDC Methods

As shown in Table 5.3-5.10, GBNC achieves the best performance in terms of both Hamming scores and exact match scores on most data sets.

- ▶ When employing logisitc regression as the base classifier and performing inference in GBNCs to optimize the subset 0/1 loss (Algorithm 5):
  - In terms of Hamming scores, GBNC ranks first on 15 data sets and never ranks last. Moreover, it significantly outperforms BR on 5 data sets, CC on 9 data sets and CP on 16 data sets. It is only significantly inferior to BR and CC on the Flickr data set (with 5 class variables, 12,198 instances).
  - In terms of exact match scores, GBNC ranks first on 11 data sets and ranks last on 1 data set (WaterQuality, with 14 class variables and 1,060 instances). Moreover, it significantly outperforms BR on 6 data sets, CC on 8 data sets and CP on 16 data sets. It is only significantly inferior to BR and CC on the Flickr data set.
- ▶ When employing logistic regression as the base classifier and performing inference in GBNCs to optimize the Hamming loss (Algorithm 6):
  - In terms of Hamming scores, GBNC ranks first on 15 data sets among the 17 data sets and never ranks last. Moreover, it significantly outperforms BR on 7 data sets, CC on 11 data sets and CP on 16 data sets. It is only significantly inferior to BR and CC on the Flickr data set (with 5 class variables, 12,198 instances).
  - In terms of exact match scores, GBNC ranks first on 11 data sets and ranks last on 1 data set (WaterQuality, with 14 class variables and 1,060 instances). Moreover, it significantly outperforms BR on 8 data sets, CC on 5

data sets and CP on 14 data sets. It is only significantly inferior to BR and CC on the Flickr data set.

In general, the three baselines have different optimization objectives for probabilistic MDC. Specifically, BR tries to optimize the Hamming loss. It solves an MDC problem by solving MCC problems one for each class variable independently. On the other hand, CC and CP can be regarded as methods optimizing the more challenging subset 0/1 loss. CC tries to model the joint distribution of class variables via a chain-like structure and CP tries to predict the mode of the joint distribution of class variables by reducing any unique class value combination to a new meta-class in a new MCC problem. From the reported results, it is shown that BR often achieves better Hamming scores against CC and CP, while CC often wins the competition in terms of exact match scores, especially on large data sets. This validates the effectiveness of their optimization strategies. Besides, we find that CP is constantly the worst method, even in terms of exact match scores. This is probably due to a large number of different class value combinations. Moreover, given limited data, CP cannot generalize to test instances where the class value combination does not occur in the training set.

Through comparison between GBNCs and other baselines, it is shown that GBNC often achieves superior Hamming scores against BR as well as outperforms CC and CP in terms of exact match scores. When employing naive Bayes as the base classifier, similar observations can also be made. The discrepancy between GBNC and other baselines becomes more significant on large data sets where the class dependencies are more complex. These findings further validate the superiority of GBNCs to other baselines.

Effectiveness of Inference Algorithms

Following the reported results in Table 5.3-5.10, we analyze the effectiveness of the two inference algorithms in GBNCs.

As introduced in Chapter 4, we use two inference algorithms (Algorithm 5 and 6) in GBNCs to compute Bayes-Optimal Predictions (BOPs) w.r.t. the subset 0/1 loss and the Hamming loss, respectively. By comparing the Hamming scores in Table 5.3 and 5.4, it is shown that GBNCs (optimizing ℓ_{HL}) often achieves better performance than GBNCs (optimizing $\ell_{0/1}$). For the more challenging exact match scores (cf. Table 5.5 and 5.6), we can also see GBNCs optimizing the subset 0/1 loss have a significant improvement.

Moreover, we observe that even using an "inappropriate" inference algorithm (e.g. evaluating GBNCs (ℓ_{HL}) in terms of exact match scores or evaluating GBNCs ($\ell_{0/1}$) in terms of Hamming scores), GBNCs can still achieve superior performance against other baselines. This indeed aligns with our previous theoretical analysis, since GBNCs always try to learn a joint class distribution in the training procedure. When base classifiers have been trained well by maximizing the local CLL functions, they are incapable of modeling conditional distributions of all class variables, thus giving reliable predictions for each class variable independently as well as optimizing the Hamming scores. Furthermore, based on the decomposability of the overall CLL under the structural constraint of GBNCs, these local classifiers can together optimize the overall CLL and form the parameter set of the learned BN structure. This will finally produce a reliable representation for the joint class distribution as well as perform MPE inference to improve the exact match score.

Finally, we also would like to emphasize the effect of different inference algorithms in GBNCs. Although the training algorithms of GBNCs can provide theoretical guarantees for learning an applicable probabilistic model, performing appropriate inference algorithms will have a major effect on optimizing different evaluation metrics.

Data Set	Hamming Score (HS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Edm	$\mathbf{0.736 \pm 0.093}$	0.696 ± 0.093	0.685 ± 0.099	0.725 ± 0.071
Jura	$\mathbf{0.634 \pm 0.029}$	0.625 ± 0.043	0.607 ± 0.075	$0.317 \pm 0.051 \uparrow$
Enb	0.781 ± 0.031	$\mathbf{0.787 \pm 0.045}$	0.762 ± 0.020	$0.685 \pm 0.033 \uparrow$
Voice	$\mathbf{0.926 \pm 0.012}$	$0.915 \pm 0.015 \uparrow$	$0.916 \pm 0.017 \uparrow$	$0.584 \pm 0.013 \uparrow$
Song	$\mathbf{0.760 \pm 0.029}$	0.752 ± 0.023	$0.738 \pm 0.039 \uparrow$	$0.507 \pm 0.040 \uparrow$
Flickr	0.770 ± 0.007	$\mathbf{0.797 \pm 0.006} \downarrow$	$0.796 \pm 0.006 \downarrow$	$0.506 \pm 0.004 \uparrow$
Fera	$\mathbf{0.617 \pm 0.008}$	0.616 ± 0.007	$0.605 \pm 0.009 \uparrow$	$0.475 \pm 0.018 \uparrow$
WQplants	$\mathbf{0.655 \pm 0.012}$	$\mathbf{0.655 \pm 0.010}$	0.650 ± 0.014	$0.611 \pm 0.024 \uparrow$
WQanimals	$\mathbf{0.627 \pm 0.017}$	0.626 ± 0.019	0.624 ± 0.021	$0.579 \pm 0.024 \uparrow$
Rf1	$\mathbf{0.900 \pm 0.008}$	$0.836 \pm 0.004 \uparrow$	$0.835 \pm 0.006 \uparrow$	$0.635 \pm 0.007 \uparrow$
Pain	$\mathbf{0.953 \pm 0.003}$	$\mathbf{0.953 \pm 0.003}$	$0.951 \pm 0.003 \uparrow$	$0.948 \pm 0.003 \uparrow$
Disfa	$\mathbf{0.896 \pm 0.003}$	$0.894 \pm 0.003 \uparrow$	$0.894 \pm 0.002 \uparrow$	$0.871 \pm 0.003 \uparrow$
WaterQuality	$\mathbf{0.641 \pm 0.019}$	0.637 ± 0.011	0.639 ± 0.017	$0.597 \pm 0.018 \uparrow$
Oes97	$\mathbf{0.724 \pm 0.023}$	0.716 ± 0.018	$0.706 \pm 0.030 \uparrow$	$0.521 \pm 0.032 \uparrow$
Oes10	$\mathbf{0.806 \pm 0.013}$	0.801 ± 0.019	$0.791 \pm 0.021 \uparrow$	$0.605 \pm 0.046 \uparrow$
Scm20d	$\mathbf{0.662 \pm 0.009}$	$0.640 \pm 0.008 \uparrow$	$0.613 \pm 0.011 \uparrow$	$0.424 \pm 0.012 \uparrow$
Scm1d	$\mathbf{0.808 \pm 0.003}$	$0.763 \pm 0.006 \uparrow$	$0.748 \pm 0.005 \uparrow$	$0.444 \pm 0.011 \uparrow$

Table 5.3: Hamming scores (mean $\pm$ std.) of each MDC approach (**base classifier:** ***logisitic regression***). GBNC performs inference by **optimizing** $\ell_{0/1}$. The best performance is highlighted in bold, and $\uparrow$ / $\downarrow$ indicates whether GBNC is significantly superior/inferior to other approaches on each data set by using a *Wilcoxon signed-rank test* with a significance level of 0.05.

Data Set	Hamming Score (HS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Edm	$\mathbf{0.736 \pm 0.091}$	0.696 ± 0.093	0.685 ± 0.099	0.725 ± 0.071
Jura	$\mathbf{0.634 \pm 0.029}$	0.625 ± 0.043	0.607 ± 0.075	$0.317 \pm 0.051 \uparrow$
Enb	0.781 ± 0.031	$\mathbf{0.787 \pm 0.045}$	0.762 ± 0.020	$0.685 \pm 0.033 \uparrow$
Voice	$\mathbf{0.927 \pm 0.013}$	$0.915 \pm 0.015 \uparrow$	$0.916 \pm 0.017 \uparrow$	$0.584 \pm 0.013 \uparrow$
Song	$\mathbf{0.766 \pm 0.028}$	0.752 ± 0.023	$0.738 \pm 0.039 \uparrow$	$0.507 \pm 0.040 \uparrow$
Flickr	0.784 ± 0.006	$\mathbf{0.797 \pm 0.006} \downarrow$	$0.796 \pm 0.006 \downarrow$	$0.506 \pm 0.004 \uparrow$
Fera	$\mathbf{0.624 \pm 0.010}$	$0.616 \pm 0.007 \uparrow$	$0.605 \pm 0.009 \uparrow$	$0.475 \pm 0.018 \uparrow$
WQplants	$\mathbf{0.658 \pm 0.013}$	0.655 ± 0.010	0.650 ± 0.014	$0.611 \pm 0.024 \uparrow$
WQanimals	$\mathbf{0.630 \pm 0.018}$	0.626 ± 0.019	$0.624 \pm 0.021 \uparrow$	$0.579 \pm 0.024 \uparrow$
Rf1	$\mathbf{0.902 \pm 0.008}$	$0.836 \pm 0.004 \uparrow$	$0.835 \pm 0.006 \uparrow$	$0.635 \pm 0.007 \uparrow$
Pain	$\mathbf{0.953 \pm 0.003}$	$\mathbf{0.953 \pm 0.003}$	$0.951 \pm 0.003 \uparrow$	$0.948 \pm 0.003 \uparrow$
Disfa	$\mathbf{0.897 \pm 0.002}$	$0.894 \pm 0.003 \uparrow$	$0.894 \pm 0.002 \uparrow$	$0.871 \pm 0.003 \uparrow$
WaterQuality	$\mathbf{0.642 \pm 0.018}$	0.637 ± 0.011	0.639 ± 0.017	$0.597 \pm 0.018 \uparrow$
Oes97	$\mathbf{0.731 \pm 0.023}$	$0.716 \pm 0.018 \uparrow$	$0.706 \pm 0.030 \uparrow$	$0.521 \pm 0.032 \uparrow$
Oes10	$\mathbf{0.809 \pm 0.014}$	0.801 ± 0.019	$0.791 \pm 0.021 \uparrow$	$0.605 \pm 0.046 \uparrow$
Scm20d	$\mathbf{0.685 \pm 0.007}$	$0.640 \pm 0.008 \uparrow$	$0.613 \pm 0.011 \uparrow$	$0.424 \pm 0.012 \uparrow$
Scm1d	$\mathbf{0.815 \pm 0.003}$	$0.763 \pm 0.006 \uparrow$	$0.748 \pm 0.005 \uparrow$	$0.444 \pm 0.011 \uparrow$

Table 5.4: Hamming scores (mean $\pm$ std.) of each MDC approach (**base classifier:** ***logisitic regression***). GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Exact Match Score (EMS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Edm	**0.570 ± 0.126**	0.475 ± 0.129	0.469 ± 0.126	0.450 ± 0.141 ↑
Jura	**0.415 ± 0.054**	0.409 ± 0.066	0.368 ± 0.114	0.014 ± 0.042 ↑
Enb	0.561 ± 0.061	**0.574 ± 0.090**	0.525 ± 0.041	0.371 ± 0.065 ↑
Voice	**0.858 ± 0.024**	0.836 ± 0.031 ↑	0.837 ± 0.033 ↑	0.187 ± 0.024 ↑
Song	**0.438 ± 0.049**	0.422 ± 0.041	0.392 ± 0.061 ↑	0.068 ± 0.035 ↑
Flickr	0.289 ± 0.013	0.324 ± 0.012 ↓	**0.325 ± 0.013** ↓	0.042 ± 0.006 ↑
Fera	**0.196 ± 0.012**	0.187 ± 0.014	0.193 ± 0.011	0.164 ± 0.015 ↑
WQplants	0.093 ± 0.034	0.093 ± 0.026	**0.095 ± 0.029**	0.075 ± 0.033 ↑
WQanimals	0.046 ± 0.008	0.047 ± 0.020	**0.057 ± 0.015** ↓	0.023 ± 0.016 ↑
Rf1	**0.532 ± 0.018**	0.283 ± 0.020 ↑	0.291 ± 0.018 ↑	0.062 ± 0.009 ↑
Pain	**0.758 ± 0.018**	0.751 ± 0.015 ↑	0.754 ± 0.017 ↑	0.751 ± 0.015 ↑
Disfa	**0.401 ± 0.011**	0.393 ± 0.012 ↑	0.394 ± 0.013 ↑	0.371 ± 0.011 ↑
WaterQuality	0.005 ± 0.005	**0.007 ± 0.006**	**0.007 ± 0.006**	0.006 ± 0.006
Oes97	**0.057 ± 0.044**	0.051 ± 0.046	0.030 ± 0.014 ↑	0.000 ↑
Oes10	0.087 ± 0.041	0.089 ± 0.032	**0.092 ± 0.034**	0.005 ± 0.010 ↑
Scm20d	**0.124 ± 0.014**	0.045 ± 0.008 ↑	0.062 ± 0.011 ↑	0.076 ± 0.012 ↑
Scm1d	**0.189 ± 0.010**	0.098 ± 0.009 ↑	0.109 ± 0.008 ↑	0.091 ± 0.009 ↑

Table 5.5: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***logisitic regression***). GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Exact Match Score (EMS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Edm	**0.571 ± 0.119**	0.475 ± 0.129	0.469 ± 0.126	0.450 ± 0.141
Jura	**0.415 ± 0.054**	0.409 ± 0.066	0.368 ± 0.114	0.014 ± 0.042 ↑
Enb	0.561 ± 0.061	**0.574 ± 0.090**	0.525 ± 0.041	0.371 ± 0.065 ↑
Voice	**0.860 ± 0.026**	0.836 ± 0.031 ↑	0.837 ± 0.033 ↑	0.187 ± 0.024 ↑
Song	**0.452 ± 0.047**	0.422 ± 0.041 ↑	0.392 ± 0.061 ↑	0.068 ± 0.035 ↑
Flickr	0.305 ± 0.011	**0.324 ± 0.012** ↓	**0.325 ± 0.013** ↓	0.042 ± 0.006 ↑
Fera	**0.197 ± 0.012**	0.187 ± 0.014 ↑	0.193 ± 0.011	0.164 ± 0.015 ↑
WQplants	**0.098 ± 0.032**	0.093 ± 0.026	0.095 ± 0.029	0.075 ± 0.033 ↑
WQanimals	0.050 ± 0.014	0.047 ± 0.020	**0.057 ± 0.015**	0.023 ± 0.016 ↑
Rf1	**0.497 ± 0.027**	0.283 ± 0.020 ↑	0.291 ± 0.018 ↑	0.062 ± 0.009 ↑
Pain	**0.757 ± 0.018**	0.751 ± 0.015 ↑	0.754 ± 0.017	0.751 ± 0.015 ↑
Disfa	**0.397 ± 0.010**	0.393 ± 0.012 ↑	0.394 ± 0.013	0.371 ± 0.011 ↑
WaterQuality	0.004 ± 0.005	**0.007 ± 0.006**	**0.007 ± 0.006**	0.006 ± 0.006
Oes97	**0.054 ± 0.045**	0.051 ± 0.046	0.030 ± 0.014	0.000 ↑
Oes10	0.089 ± 0.036	0.089 ± 0.032	**0.092 ± 0.034**	0.005 ± 0.010 ↑
Scm20d	0.072 ± 0.009	0.045 ± 0.008 ↑	0.062 ± 0.011 ↑	**0.076 ± 0.012**
Scm1d	**0.161 ± 0.013**	0.098 ± 0.009 ↑	0.109 ± 0.008 ↑	0.091 ± 0.009 ↑

Table 5.6: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***logisitic regression***). GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Hamming Score (HS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Edm	0.696 ± 0.124	0.679 ± 0.065	0.667 ± 0.107 ↑	**0.717 ± 0.043**
Jura	**0.570 ± 0.061**	0.550 ± 0.054 ↑	0.557 ± 0.049	0.308 ± 0.031 ↑
Enb	**0.771 ± 0.025**	0.702 ± 0.041 ↑	0.700 ± 0.041 ↑	0.687 ± 0.021 ↑
Voice	**0.905 ± 0.012**	0.883 ± 0.009 ↑	0.875 ± 0.009 ↑	0.545 ± 0.011 ↑
Song	**0.648 ± 0.020**	0.618 ± 0.031 ↑	0.618 ± 0.030 ↑	0.427 ± 0.035 ↑
Flickr	**0.690 ± 0.007**	0.648 ± 0.008 ↑	0.649 ± 0.008 ↑	0.481 ± 0.005 ↑
Fera	**0.445 ± 0.005**	0.424 ± 0.008 ↑	0.422 ± 0.008 ↑	0.409 ± 0.005 ↑
WQplants	0.453 ± 0.028	0.395 ± 0.022 ↑	0.342 ± 0.023 ↑	**0.506 ± 0.027** ↓
WQanimals	0.473 ± 0.020	0.383 ± 0.018 ↑	0.382 ± 0.018 ↑	**0.508 ± 0.017** ↓
Rf1	**0.827 ± 0.007**	0.766 ± 0.005 ↑	0.764 ± 0.005 ↑	0.623 ± 0.008 ↑
Pain	0.772 ± 0.007	0.654 ± 0.010 ↑	0.562 ± 0.011 ↑	**0.814 ± 0.004** ↓
Disfa	0.782 ± 0.008	0.666 ± 0.004 ↑	0.630 ± 0.004 ↑	**0.799 ± 0.003** ↓
WaterQuality	0.502 ± 0.032	0.392 ± 0.017 ↑	0.365 ± 0.019 ↑	**0.593 ± 0.018** ↓
Oes97	**0.703 ± 0.026**	0.665 ± 0.026 ↑	0.666 ± 0.027 ↑	0.299 ± 0.034 ↑
Oes10	**0.774 ± 0.022**	0.747 ± 0.019 ↑	0.745 ± 0.021 ↑	0.431 ± 0.055 ↑
Scm20d	**0.505 ± 0.006**	0.467 ± 0.010 ↑	0.428 ± 0.013 ↑	0.408 ± 0.005 ↑
Scm1d	**0.666 ± 0.007**	**0.666 ± 0.007**	0.659 ± 0.008 ↑	0.425 ± 0.013 ↑

Table 5.7: Hamming scores (mean ± std.) of each MDC approach (**base classifier: *naive Bayes***). GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Hamming Score (HS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Edm	0.696 ± 0.124	0.679 ± 0.065	0.667 ± 0.107 ↑	**0.717 ± 0.043**
Jura	**0.580 ± 0.058**	0.550 ± 0.054 ↑	0.557 ± 0.049 ↑	0.308 ± 0.031 ↑
Enb	**0.771 ± 0.025**	0.702 ± 0.041 ↑	0.700 ± 0.041 ↑	0.687 ± 0.021 ↑
Voice	**0.906 ± 0.011**	0.883 ± 0.009 ↑	0.875 ± 0.009 ↑	0.545 ± 0.011 ↑
Song	**0.649 ± 0.020**	0.618 ± 0.031 ↑	0.618 ± 0.030 ↑	0.427 ± 0.035 ↑
Flickr	**0.690 ± 0.007**	0.648 ± 0.008 ↑	0.649 ± 0.008 ↑	0.481 ± 0.005 ↑
Fera	**0.446 ± 0.005**	0.424 ± 0.008 ↑	0.422 ± 0.008 ↑	0.409 ± 0.005 ↑
WQplants	0.474 ± 0.036	0.395 ± 0.022 ↑	0.342 ± 0.023 ↑	**0.506 ± 0.027**
WQanimals	0.488 ± 0.023	0.383 ± 0.018 ↑	0.382 ± 0.018 ↑	**0.508 ± 0.017** ↓
Rf1	**0.827 ± 0.007**	0.766 ± 0.005 ↑	0.764 ± 0.005 ↑	0.623 ± 0.008 ↑
Pain	0.775 ± 0.007	0.654 ± 0.010 ↑	0.562 ± 0.011 ↑	**0.814 ± 0.004** ↓
Disfa	0.784 ± 0.008	0.666 ± 0.004 ↑	0.630 ± 0.004 ↑	**0.799 ± 0.003** ↓
WaterQuality	0.540 ± 0.031	0.392 ± 0.017 ↑	0.365 ± 0.019 ↑	**0.593 ± 0.018** ↓
Oes97	**0.702 ± 0.027**	0.665 ± 0.026 ↑	0.666 ± 0.027 ↑	0.299 ± 0.034 ↑
Oes10	**0.774 ± 0.021**	0.747 ± 0.019 ↑	0.745 ± 0.021 ↑	0.431 ± 0.055 ↑
Scm20d	**0.510 ± 0.007**	0.467 ± 0.010 ↑	0.428 ± 0.013 ↑	0.408 ± 0.005 ↑
Scm1d	**0.667 ± 0.007**	0.666 ± 0.007	0.659 ± 0.008 ↑	0.425 ± 0.013 ↑

Table 5.8: Hamming scores (mean ± std.) of each MDC approach (**base classifier: *naive Bayes***). GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Exact Match Score (EMS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Edm	**0.527 ± 0.137**	0.435 ± 0.084 ↑	0.482 ± 0.109	0.440 ± 0.086
Jura	**0.357 ± 0.074**	0.321 ± 0.073 ↑	0.345 ± 0.058	0.011 ± 0.014 ↑
Enb	**0.542 ± 0.051**	0.403 ± 0.083 ↑	0.400 ± 0.082 ↑	0.375 ± 0.041 ↑
Voice	**0.819 ± 0.022**	0.783 ± 0.012 ↑	0.771 ± 0.012 ↑	0.153 ± 0.022 ↑
Song	**0.279 ± 0.039**	0.227 ± 0.049 ↑	0.224 ± 0.046 ↑	0.065 ± 0.025 ↑
Flickr	**0.170 ± 0.011**	0.139 ± 0.008 ↑	0.141 ± 0.009 ↑	0.034 ± 0.005 ↑
Fera	**0.066 ± 0.006**	0.020 ± 0.003 ↑	0.019 ± 0.002 ↑	0.044 ± 0.006 ↑
WQplants	0.002 ± 0.004	0.000	0.001 ± 0.003	**0.008 ± 0.008** ↓
WQanimals	**0.009 ± 0.009**	0.005 ± 0.005	0.008 ± 0.007	0.006 ± 0.006
Rf1	**0.305 ± 0.014**	0.163 ± 0.007 ↑	0.163 ± 0.007 ↑	0.068 ± 0.010 ↑
Pain	**0.112 ± 0.013**	0.068 ± 0.011 ↑	0.081 ± 0.016 ↑	0.085 ± 0.009 ↑
Disfa	**0.109 ± 0.011**	0.004 ± 0.002 ↑	0.004 ± 0.002 ↑	0.054 ± 0.006 ↑
WaterQuality	0.000	0.000	0.000	**0.003 ± 0.006**
Oes97	0.024 ± 0.029	0.054 ± 0.050 ↓	**0.057 ± 0.051** ↓	0.000 ↑
Oes10	**0.082 ± 0.038**	**0.082 ± 0.044**	**0.082 ± 0.044**	0.010 ± 0.017 ↑
Scm20d	0.015 ± 0.006	0.017 ± 0.005	0.024 ± 0.006 ↓	**0.040 ± 0.005** ↓
Scm1d	0.052 ± 0.010	0.082 ± 0.010 ↓	**0.089 ± 0.009** ↓	0.035 ± 0.006 ↑

Table 5.9: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***). GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Exact Match Score (EMS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Edm	**0.527 ± 0.137**	0.435 ± 0.084 ↑	0.482 ± 0.109	0.440 ± 0.086
Jura	**0.365 ± 0.073**	0.321 ± 0.073 ↑	0.345 ± 0.058	0.011 ± 0.014 ↑
Enb	**0.542 ± 0.051**	0.403 ± 0.083 ↑	0.400 ± 0.082 ↑	0.375 ± 0.041 ↑
Voice	**0.820 ± 0.021**	0.783 ± 0.012 ↑	0.771 ± 0.012 ↑	0.153 ± 0.022 ↑
Song	**0.279 ± 0.039**	0.227 ± 0.049 ↑	0.224 ± 0.046 ↑	0.065 ± 0.025 ↑
Flickr	**0.170 ± 0.011**	0.139 ± 0.008 ↑	0.141 ± 0.009 ↑	0.034 ± 0.005 ↑
Fera	**0.067 ± 0.007**	0.020 ± 0.003 ↑	0.019 ± 0.002 ↑	0.044 ± 0.006 ↑
WQplants	0.006 ± 0.005	0.000 ↑	0.001 ± 0.003 ↑	**0.008 ± 0.008**
WQanimals	**0.008 ± 0.009**	0.005 ± 0.005	**0.008 ± 0.007**	0.006 ± 0.006
Rf1	**0.304 ± 0.014**	0.163 ± 0.007 ↑	0.163 ± 0.007 ↑	0.068 ± 0.010 ↑
Pain	**0.112 ± 0.014**	0.068 ± 0.011 ↑	0.081 ± 0.016 ↑	0.085 ± 0.009 ↑
Disfa	**0.113 ± 0.012**	0.004 ± 0.002 ↑	0.004 ± 0.002 ↑	0.054 ± 0.006 ↑
WaterQuality	0.000	0.000	0.000	**0.003 ± 0.006**
Oes97	0.024 ± 0.029	0.054 ± 0.050 ↓	**0.057 ± 0.051** ↓	0.000 ↑
Oes10	**0.082 ± 0.038**	**0.082 ± 0.044**	**0.082 ± 0.044**	0.010 ± 0.017 ↑
Scm20d	0.015 ± 0.006	0.017 ± 0.005	0.024 ± 0.006 ↓	**0.040 ± 0.005** ↓
Scm1d	0.051 ± 0.010	0.082 ± 0.010 ↓	**0.089 ± 0.009** ↓	0.035 ± 0.006 ↑

Table 5.10: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***). GBNC performs inference by **optimizing** ℓ_{HL}.

5.2.2 Mixed Data

In this section, we analyze the experimental results for mixed data, i.e., data sets in which continuous and discrete features coexist.

Table 5.11-5.18 summarize the detailed experimental results for all methods. In comparison with the three baselines, GBNCs with appropriate inference algorithms achieves superior performance on two data sets among the three mixed data sets.

First of all, we observe that GBNCs have relatively poor performance on the Adult data set. We think it is reasonable as Adult has only 5 continuous feature variables. GBNCs works by using discrete variables to partition the input space to train fine-grained probabilistic models. In this setting, discrete feature variables are essentially treated as special "class variables" and are not used to train models. From the experimental results on tabular data sets containing only continuous features in the last section, we can see that such approach is more effective in exploiting continuous features than other baselines. However, in the case of the Adult data set, the information from the continuous features is limited. Therefore, other baselines that use discrete features to train models can often achieve better performance. On the other hand, when the dimension of continuous features increases, we can observe the superiority of GBNCs is more significant on the Default and Thyroid data sets. Moreover, it is not very surprising that BR gives the most promising results on the Adult data set, as it has been shown BR appears to be a competitive approach given a small number of class variables (Wu and Zhu, 2020).

Furthermore, GBNCs also have more flexibility than other methods. In the literature, there is no commonly accepted method to deal with both continuous and discrete features. In this experiment, when employing logistic regression as the base classifier, the baselines use the one-hot encoding method to pre-process the discrete features. Although such encoding method is generally applicable and effective on the three data sets, it results in increasing the dimensionality of features. The encoding dimensionality is strongly dependent on the value range of discrete features. This causes a "curse of dimensionality" and restricts its generalization on large data sets. Besides, when employing naive Bayes as the base classifier, the baselines have to make different distribution assumptions for different feature dimensions and train different naive Bayes classifiers. In contrast, GBNC is able to employ any probabilistic model as base classifiers without additional pre-processing or assumptions. This makes it more flexible and generalized in complex scenarios.

Data Set	Hamming Score (HS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Adult	0.663 ± 0.005	$\mathbf{0.718 \pm 0.005} \downarrow$	$0.715 \pm 0.005 \downarrow$	$0.585 \pm 0.006 \uparrow$
Default	$\mathbf{0.666 \pm 0.005}$	0.664 ± 0.007	0.663 ± 0.009	$0.561 \pm 0.004 \uparrow$
Thyroid	$\mathbf{0.966 \pm 0.003}$	0.965 ± 0.002	$0.964 \pm 0.003 \uparrow$	$0.962 \pm 0.003 \uparrow$

Table 5.11: Exact match scores (mean $\pm$ std.) of each MDC approach (**base classifier: *logisitic regression***) on **mixed data**. GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Hamming Score (HS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Adult	0.676 ± 0.004	$\mathbf{0.718 \pm 0.005} \downarrow$	$0.715 \pm 0.005 \downarrow$	$0.585 \pm 0.006 \uparrow$
Default	$\mathbf{0.666 \pm 0.004}$	0.664 ± 0.007	0.663 ± 0.009	$0.561 \pm 0.004 \uparrow$
Thyroid	$\mathbf{0.966 \pm 0.003}$	$0.965 \pm 0.002 \uparrow$	$0.964 \pm 0.003 \uparrow$	$0.962 \pm 0.003 \uparrow$

Table 5.12: Hamming scores (mean $\pm$ std.) of each MDC approach (**base classifier: *logistic regression***) on **mixed data**. GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Exact Match Score (EMS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Adult	0.245 ± 0.006	$\mathbf{0.274 \pm 0.011} \downarrow$	$\mathbf{0.274 \pm 0.013} \downarrow$	$0.134 \pm 0.007 \uparrow$
Default	$\mathbf{0.187 \pm 0.005}$	$0.177 \pm 0.009 \uparrow$	$0.177 \pm 0.012 \uparrow$	$0.060 \pm 0.004 \uparrow$
Thyroid	$\mathbf{0.784 \pm 0.018}$	$0.774 \pm 0.014 \uparrow$	$0.768 \pm 0.016 \uparrow$	$0.751 \pm 0.016 \uparrow$

Table 5.13: Exact match scores (mean $\pm$ std.) of each MDC approach (**base classifier: *logisitic regression***) on **mixed data**. GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Exact Match Score (EMS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Adult	0.202 ± 0.007	$\mathbf{0.274 \pm 0.011} \downarrow$	$\mathbf{0.274 \pm 0.013} \downarrow$	$0.134 \pm 0.007 \uparrow$
Default	0.176 ± 0.005	$\mathbf{0.177 \pm 0.009}$	$\mathbf{0.177 \pm 0.012}$	$0.060 \pm 0.004 \uparrow$
Thyroid	$\mathbf{0.784 \pm 0.018}$	$0.774 \pm 0.014 \uparrow$	$0.768 \pm 0.016 \uparrow$	$0.751 \pm 0.016 \uparrow$

Table 5.14: Exact match scores (mean $\pm$ std.) of each MDC approach (**base classifier: *logisitic regression***) on **mixed data**. GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Hamming Score (HS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Adult	0.547 ± 0.031	$\mathbf{0.677 \pm 0.015}$ ↓	0.651 ± 0.013 ↓	0.391 ± 0.010 ↑
Default	$\mathbf{0.583 \pm 0.019}$	0.526 ± 0.025 ↑	0.510 ± 0.025 ↑	0.362 ± 0.004 ↑
Thyroid	$\mathbf{0.964 \pm 0.003}$	0.760 ± 0.017 ↑	0.800 ± 0.008 ↑	0.922 ± 0.003 ↑

Table 5.15: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***). GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Hamming Score (HS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Adult	0.548 ± 0.031	$\mathbf{0.677 \pm 0.015}$ ↓	0.651 ± 0.013 ↓	0.391 ± 0.010 ↑
Default	$\mathbf{0.583 \pm 0.019}$	0.526 ± 0.025 ↑	0.510 ± 0.025 ↑	0.362 ± 0.004 ↑
Thyroid	$\mathbf{0.964 \pm 0.003}$	0.760 ± 0.017 ↑	0.800 ± 0.008 ↑	0.922 ± 0.003 ↑

Table 5.16: Hamming scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***) on **mixed data**. GBNC performs inference by **optimizing** ℓ_{HL}.

Data Set	Exact Match Score (EMS)			
	GBNC ($\ell_{0/1}$)	BR	CC	CP
Adult	0.067 ± 0.012	$\mathbf{0.230 \pm 0.027}$ ↓	0.220 ± 0.026 ↓	0.016 ± 0.002 ↓
Default	$\mathbf{0.078 \pm 0.017}$	0.042 ± 0.021 ↑	0.043 ± 0.021 ↑	0.001 ↑
Thyroid	$\mathbf{0.792 \pm 0.015}$	0.090 ± 0.011 ↑	0.110 ± 0.012 ↑	0.528 ± 0.014 ↑

Table 5.17: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***) on **mixed data**. GBNC performs inference by **optimizing** $\ell_{0/1}$.

Data Set	Exact Match Score (EMS)			
	GBNC (ℓ_{HL})	BR	CC	CP
Adult	0.067 ± 0.012	$\mathbf{0.230 \pm 0.027}$ ↓	0.220 ± 0.026 ↓	0.016 ± 0.002 ↑
Default	$\mathbf{0.078 \pm 0.018}$	0.042 ± 0.021 ↑	0.043 ± 0.021 ↑	0.001 ↑
Thyroid	$\mathbf{0.793 \pm 0.015}$	0.090 ± 0.011 ↑	0.110 ± 0.012 ↑	0.528 ± 0.014 ↑

Table 5.18: Exact match scores (mean ± std.) of each MDC approach (**base classifier:** ***naive Bayes***) on **mixed data**. GBNC performs inference by **optimizing** ℓ_{HL}.

Evaluation Metrics	Methods			
	GBNC ($\ell_{0/1}$)	GBNC (ℓ_{HL})	BR	CP
Hamming Score	**0.897**	**0.897**	0.891	0.887
Exact Match Score	**0.662**	**0.662**	0.604	0.655

Table 5.19: Results of each MDC approach (**base classifier: *ResNet-18***) for the PASCAL VOC 2007 data set.

5.2.3 Image Data

In this section, we analyze the applicability of GBNCs to the visual domain.

Table 5.19 reports the experimental results of all methods for the PASCAL VOC 2007 data set. It is shown that GBNC achieves superior performance in terms of exact match scores. This further supports the advantage of GBNCs on optimizing the subset 0/1 loss as well as learning a reliable representation of the joint class distribution.

Unsurprisingly, BR performs second to the best in terms of Hamming scores and CC performs second to the best in terms of exact match scores. Moreover, we do not observe a clear discrepancy in the two metrics when using different inference algorithms in GBNCs. As the constructed MDC PASCAL VOC 2007 data set has only four class variables, we think this behavior is reasonable. The learned BN structure of GBNCs is relatively simple, in which MPE and MAP inference may often give the same predictions.

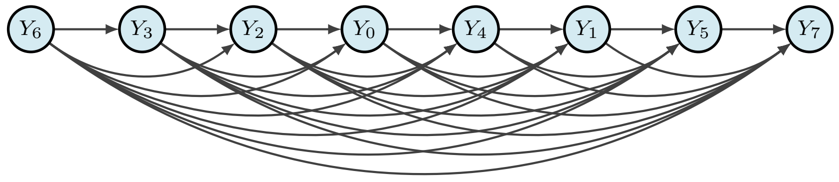
5.3 Analysis on BN Structures

From a BN perspective, BR and CC can also be regarded as methods trying to construct BN structures to model the joint probability distribution of the given data. Without special structural constraints, all feature variables (including both continuous and discrete) can be parents of any class variable. In this setting, BR assumes the class subgraph as an empty DAG and CC tries to model the class subgraph by learning a fully-connected BN according to some chain order.

The modeling strategies of BR and CC can be effective under some circumstances. However, they are unable to learn an unrestricted BN structure and do not provide guarantees to find an I-map of the real data distribution. In contrast, GBNC learns a BN structure for class variables in a flexible way. The only structural constraint (Constraint 4) does not prevent one from learning an I-map of the data distribution. In the following, we provide examples of learned class BN structures from both the tabular and image data sets to further analyze the effectiveness of GBNCs.

Y_0
Y_1
Y_2
Y_3
Y_4
Y_5
Y_6
Y_7

(a) BR



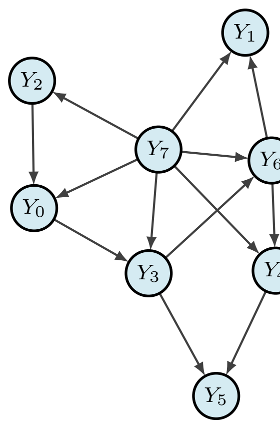
(b) CC

Figure 5.1: BN structures of BR and CC for the Rf1 data set. The chain order of CC is picked from 10 randomly generated orders with the best exact match score. The base classifier is *logisitic regression*.

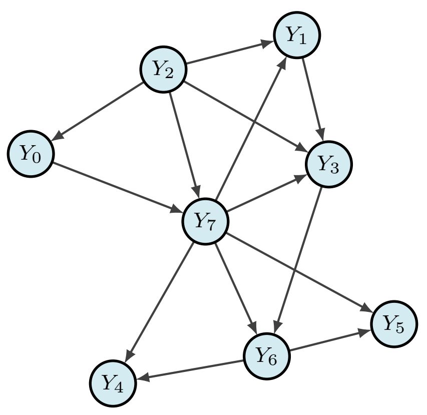
First of all, we choose the Rf1 data set (with 8 class variables), on which GBNCs achieved overwhelmingly better performance than the baselines. In Figure 5.1, we visualize the BN structures of BR and CC for the Rf1 data set, where the chain order of CC is picked from 10 randomly generated orders on 1 fold with the best exact match score.

Figure 5.2 shows four BN structures of GBNCs learned on four folds of the Rf1 data set. In comparison of BR and CC, it is shown that the learned BN structures of GBNCs are more varied and flexible. Moreover, we also observe some consistent behaviors in the four structures learned on different parts of the Rf1 data set. For example, Y_2 appears to be a root variable, and Y_7 seems to be a key variable that has probabilistic relationships with many other variables. Such special relationships are generally difficult to represent by constructed BNs of other methods. Together with the experimental results from Section 5.2.1, these findings demonstrate that GBNCs are more capable of learning flexible and effective class dependencies from data.

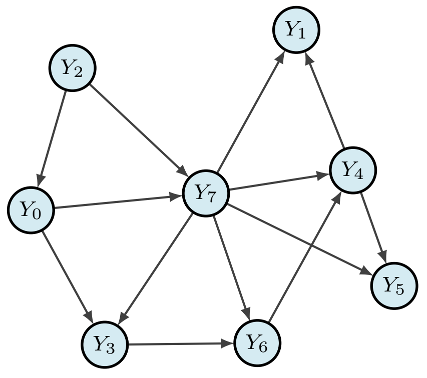
For mixed tabular data, in addition to learning the class subgraph, GBNCs are also employed as a feature selector for discrete feature variables. In such case, the discrete feature variables are not directly involved in learning BN structures or performing inference, but they implicitly affect the training of base classifiers by further partitioning the input space. In Figure 5.3-Figure 5.5, we visualize some learned BN structures of GBNCs on the three mixed tabular



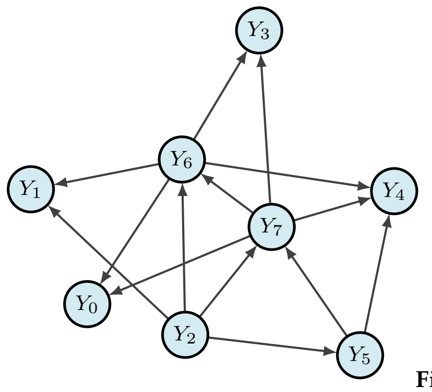
(a) Fold 0



(b) Fold 3



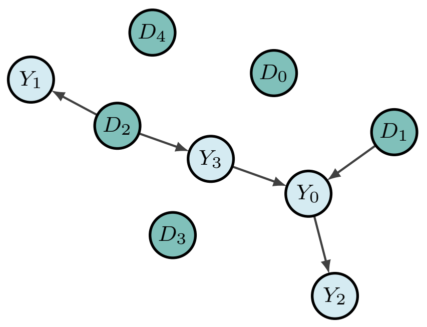
(c) Fold 5



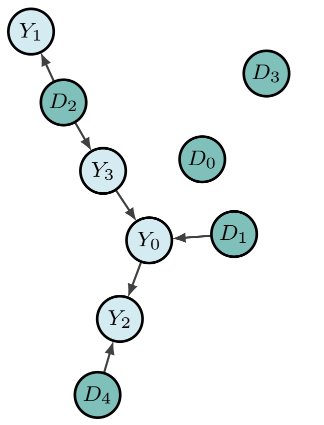
(d) Fold 9

Figure 5.2: Learned BN structures of GBNCs on the Rf1 data set. The base classifier is *logisitic regression*.

data sets, where the discrete feature variables are denoted as D_i for simplicity. By taking the extra structure complexity due to discrete feature variables into account, the pruning rules from Section 4.4.1 often prevent GBNCs from learning complex class BN structures or using too many discrete feature variables.

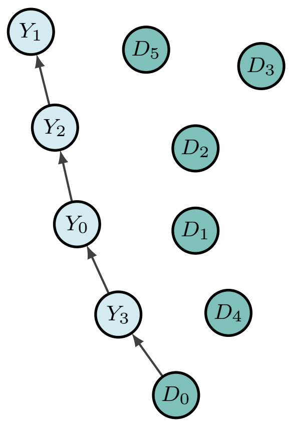


(a) Fold 2

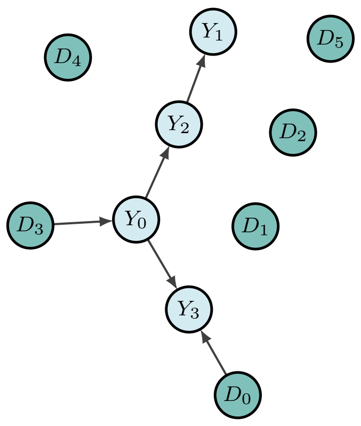


(b) Fold 9

Figure 5.3: Learned BN structure and selected discrete features of GBNCs on the Adult data set. The base classifier is *logisitic regression*.

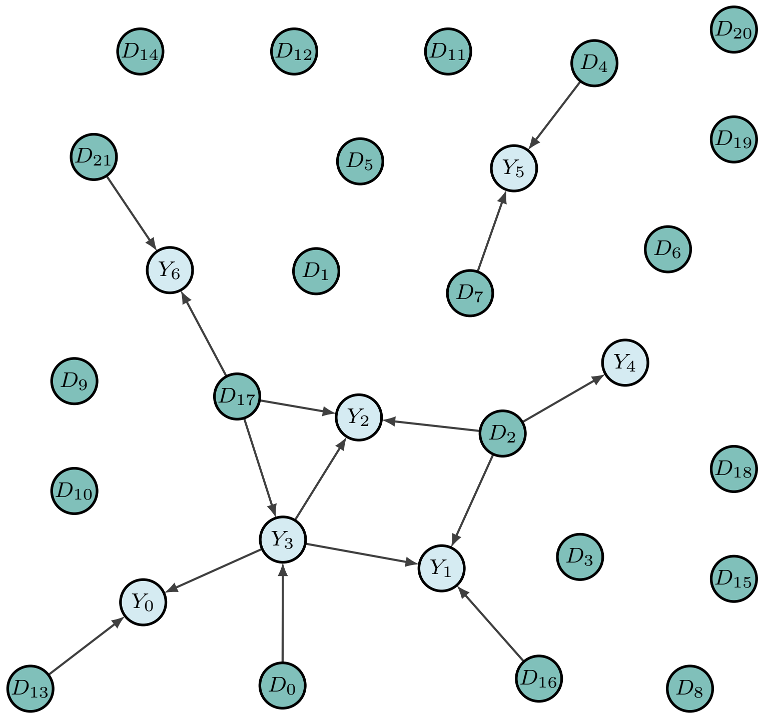


(a) Fold 3

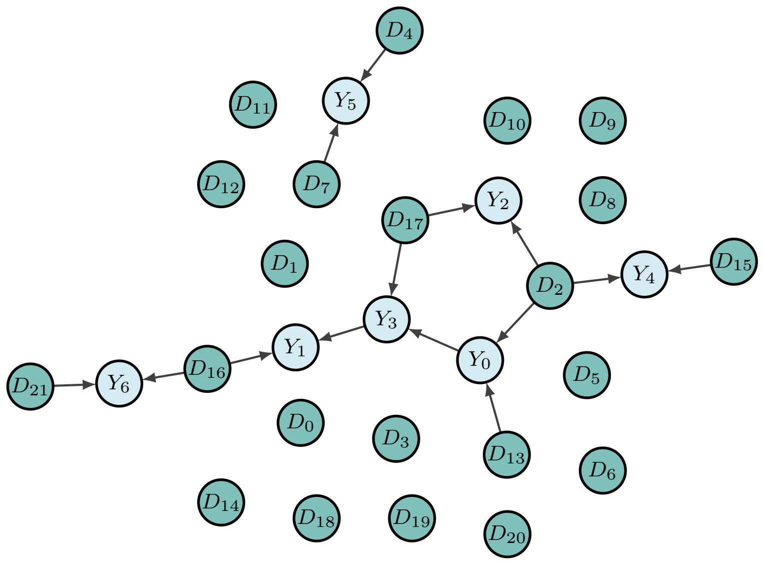


(b) Fold 7

Figure 5.4: Learned BN structure and selected discrete features of GBNCs on the Default data set. The base classifier is *logisitic regression*.



(a) Fold 4



(b) Fold 5

Figure 5.5: Learned BN structure and selected discrete features of GBNCs on the Thyroid data set. The base classifier is *logisitic regression*.

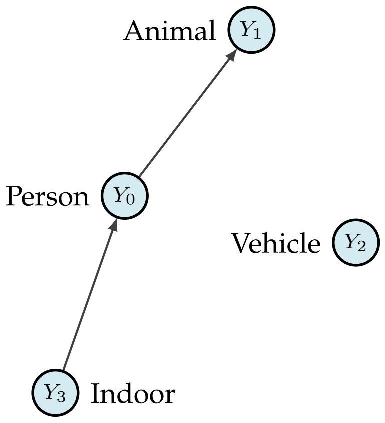


Figure 5.6: Learned BN structure of GBNCs on the PASCAL VOC 2007 data set. The base classifier is *ResNet-18* with weights pretrained on ImageNet.

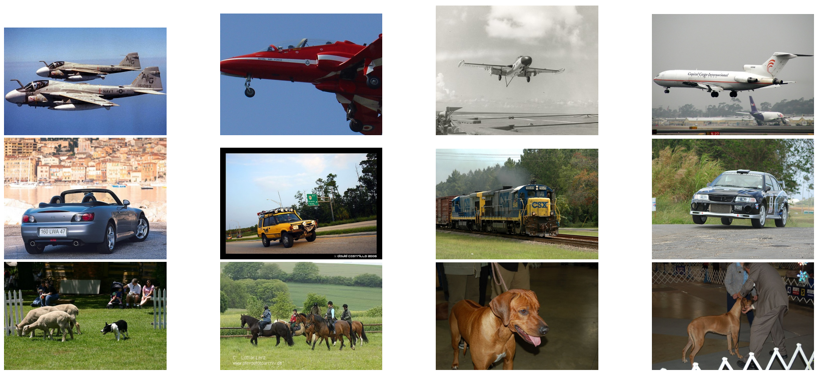


Figure 5.7: Example images from the PASCAL VOC 2007 data set.

Last but not least, the structure learning strength of GBNCs is also not affected by the base classifiers. On image data, GBNCs can extract probabilistic relationships from deep neural networks to learn effective class BN structures. Figure 5.6 shows the learned BN structure of GBNCs on the PASCAL VOC 2007 data set. It is shown that GBNCs tends to train an independent classifier for the vehicle class, and consider the mutual influence when training classifiers for the other three class variables. This behavior may be caused by the characteristics of the data set. As shown in Figure 5.7, an airplane image probably does not contain recognizable objects when it takes off or is flying; drivers or passengers may be inside vehicles, which are not identified in many images; people may appear indoors with their pets and appear outdoors with their livestock in many images. These co-occurring relationships probably have a significant effect on training classifiers.

Conclusion & Future Work 6

This chapter will outline the conclusion of this thesis and discuss some possible future research directions.

6.1 Conclusion

In this thesis, we proposed Generalized Bayesian Network Classifier (GBNC), a generalized framework for solving probabilistic Multi-Dimensional Classification (MDC) problems.

First of all, we introduced a new structural constraint for learning Bayesian Networks (BNs) by maximizing the conditional likelihood, in a way by decomposing the overall conditional likelihood into local items over the BN structure. The standard way to learn BN structures and parameters by maximizing the joint likelihood of all feature and class variables tends to perform poorly in classification problems. Due to this, our motivation is to learn BNs by maximizing the CLL of the class variables given the feature variables, which is the natural objective function in classification problems. Here, the major challenge is that the CLL does not decompose over the BN structure, which makes optimizing the CLL computationally very expensive and impractical on large data sets. To address this, we proposed a new structural constraint for learning BNs, by which the CLL can be decomposed into local conditional probabilities or conditional density functions w.r.t. each variable in the BN. We demonstrated the effectiveness of the new structural constraint and proved that introducing it does not prevent one from learning an I-map of the real data distribution.

Based on these theoretical results, we presented parameter and structure learning algorithms by leveraging the decomposability of the CLL. First, for parameter learning, we showed that learning parameters for each node by maximizing the local conditional probabilities or densities can be transformed into learning independent probabilistic classifiers by treating the decomposed CLL function as the objective function. The input features for training different local classifiers are partitioned from the original feature space by the corresponding class variable and its potential parents. Second, for structure learning, we proved that the probabilistic relationships between feature variables do not affect the optimization of the CLL. Based on this, we simplified the structure learning problem to structure learning of the class subgraph by assuming all feature variables are parents of any class variable. This does not mean that we do not consider the probabilistic relationships of different feature variables with different class variables, as they have been taken into account when learning each local probabilistic classifier. Based on these, we consider the CLL as a decomposable scoring function for the class subgraph and devised an exact score-based structure learning algorithm by using the GOBNILP system (Cussens, 2020). We employed the pruning rules from de Campos et al. (2018) to further simplify the structure learning problem. Last but not least, we showed that GBNCs can still work under the presence of mixed data, i.e., both continuous and discrete features coexist. The discrete feature variables are essentially used as special "class variables" to further partition the feature space to train the local classifiers.

Besides the learning algorithms, we also presented inference algorithms for GBNCs to find the Bayes-Optimal Prediction (BOP) under different loss functions. We discussed the two most representative loss functions in MLC and MDC research: the Hamming loss and the subset 0/1 loss. Among them, the subset 0/1 loss is more strict and challenging, which is also our focus. During training, GBNC aims for learning the joint distribution of class variables, thus naturally optimizing the subset 0/1 loss. For prediction, we showed that computing the BOP w.r.t. the Hamming loss and the subset 0/1 loss can be reduced to performing MAP inference and MPE inference in the learned BN, respectively. By using variable elimination, we can perform inference tasks as well as find the BOP efficiently.

Finally, we performed experiments to compare GBNCs against three probabilistic MDC baselines on a collection of tabular data sets and on the PASCAL VOC 2007 image data set. The experimental results validated the superiority of GBNCs in comparison to the baselines.

6.2 Future Work

This thesis tries to shed some light on probabilistic MDC and learning BN classifiers. The main advantages of the proposed GBNC framework are that it is principled and versatile, which can be extended and improved from several aspects. Much remains to be explored and here we conclude this thesis with some interesting topics for future work:

- For structure learning, we have only considered the structure complexity from the subgraphs of class variables and discrete feature variables. We assume that all continuous feature variables are parents of any class variable in the graph space and exclude them from computing the penalty of the structure complexity. However, as we said in Section 4.4, GBNCs essentially perform the feature selection, i.e., the structure learning for the edges between continuous feature variables and class variables, when training the local classifiers. After that, one would be able to evaluate the probabilistic relationship between any class variable and each continuous feature variable based on the trained base classifiers. Taking such information into account when computing the penalty of the structure complexity would benefit the structure learning of GBNCs.

- For performing inference in GBNCs, we have only considered two loss functions. However, it is also possible to extend other popular MLC measures to the multi-dimensional setting, and find the BOP under them. For example, the *Brier score* (Brier, 1950) and the *Area Under the ROC Curve (AUC) score* (Zhang and Zhou, 2014; Benjumeda et al., 2018).
- In our work so far, we use the discrete feature variables as special class variables to further partition the input space when training local classifiers. The main disadvantage of this is that GBNCs cannot learn effective local classifiers when the number of continuous feature variables is small. In addition to this, due to the exponentially increased penalty term, GBNCs usually prune most discrete feature variables and only use a small fraction of them. The information from the pruned discrete feature variables may be ignored. An avenue of further research in this direction would be developing methods to further leverage the discrete feature variables.
- Furthermore, a limitation of GBNCs is that training local classifiers can be computationally expensive, especially on data sets with a large number of class variables or when a complex base classifier (such as neural networks) has been employed. In addition to this, GBNC needs to equip the learned BN structure with different parameterizations for each new test instance. Together with that performing MAP or MPE inference is NP-hard, the instance-based inference algorithm is also computationally very expensive. The current implementation of the learning algorithms leaves a lot of room for parallelization. Further research is required to explore more effective pruning rules and to develop more efficient learning and inference algorithms.

A Appendix

A.1 Proofs for Chapter 4

Proposition 4.2.2 *Let $\mathscr{G}$ be the set of all possible DAGs. For any $\mathcal{G} \in \mathscr{G}$, there is a $\mathcal{G}' \in \mathscr{G}^X$ such that $\mathcal{G}'$ is an I-map of $\mathcal{G}$.*

Proof. To construct $\mathcal{G}'$, we make the following modifications based on $\mathcal{G}$:

1. Replacing all the subgraphs $Y \rightarrow X \leftarrow Y'$ with $(Y \leftarrow X \rightarrow Y', Y - Y')$.
2. Replacing all the subgraphs $X \rightarrow X' \leftarrow Y$ with $X \rightarrow Y \leftarrow X'$.
3. Replacing all the edges $X \leftarrow Y$ with edges $X \rightarrow Y$.

Note that the feature subgraph and the class subgraph will not be affected by these modifications. Moreover, these modifications will not introduce cycles and still make $\mathcal{G}'$ a DAG. This is because we can sequentially perform these modifications to ensure there is no cycle introduced. Concretely, we start by fixing the directions of paths $Y - Y'$ in (1) without introducing cycles, then safely perform replacements (2) and (3). For any existing path $Y - Y'$, the current class subgraph defined over $\mathbf{Y}$ is a DAG. We cannot have two paths $Y \rightarrow \cdots \rightarrow Y'$ and $Y \leftarrow \ldots, \leftarrow Y'$ simultaneously. Therefore, we can always safely assign directions to newly added paths $Y - Y'$ without introducing cycles. After performing all modifications, we have $\mathcal{G}' \in \mathscr{G}^X$.

Now we prove that $\mathcal{G}'$ is an I-map of $\mathcal{G}$. First, performing modifications (1) and (2) only introduces additional dependencies. For modification (1), replacing $Y \rightarrow X \leftarrow Y'$ with $(Y \leftarrow X \rightarrow Y', Y - Y')$ only introduces direct dependencies between Y and Y', which can be explained by conditional dependencies from $Y \rightarrow X \leftarrow Y'$. For modification (2), replacing $X \rightarrow X' \leftarrow Y$ with $X \rightarrow Y \leftarrow X'$ only introduces direct dependencies between X and Y. Next, performing modification (3) will not produce additional independencies. For modification (3), replacing $X \leftarrow Y$ with edges $X \rightarrow Y$ can contain three cases: $X \rightarrow Y$ to $X \leftarrow Y$, $X \rightarrow Y \rightarrow X'$ to $X \leftarrow Y \rightarrow X'$, and $Y \rightarrow X \rightarrow Y'$ to $Y \leftarrow X \rightarrow Y'$. All the three cases will not introduce additional independencies. Therefore, $\mathcal{I}(\mathcal{G}') \subseteq \mathcal{I}(\mathcal{G})$, i.e., $\mathcal{G}'$ is an I-map of $\mathcal{G}$. $\square$

Proposition 4.2.3 *Assume a BN structure $\mathcal{G}$ satisfies Constraint 4, the CLL function can be simplified as*

$$\begin{aligned}\mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) &= \log \prod_{l=1}^{N} \frac{f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} f_{\mathcal{B}}(\mathbf{x}^l, \mathbf{y}^l)} \\ &= \log \prod_{l=1}^{N} \frac{\prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)} \\ &= \log \prod_{l=1}^{N} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l) \\ &= \sum_{l=1}^{N} \sum_{j=1}^{d} \log P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)\end{aligned} \tag{4.9}$$

Proof. First of all, reproducing Equation 4.8 gives us that

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) = \log \prod_{l=1}^{N} \frac{\prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)}{\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l)} \tag{A.1}$$

Then, to show that

$$\mathrm{CLL}(\mathcal{G}, \boldsymbol{\Theta} \mid \mathcal{D}) = \log \prod_{l=1}^{N} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j^l \mid \mathbf{pa}(y_j)^l) \tag{A.2}$$

is equivalent to show that

$$\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j \mid \mathbf{pa}(y_j)) = 1 \tag{A.3}$$

Let $\tau(1), \tau(2), \ldots, \tau(d)$ be a topological ordering of the class variables $\mathbf{Y}$ induced by the DAG $\mathcal{G}$, where $\tau(j)$ is the index of the class value on position j of the order $\tau : \{1, 2, \ldots, d\} \mapsto \{1, 2, \ldots, d\}$. We have

$$\begin{aligned}&\sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_j \mid \mathbf{pa}(y_j)) \\ &= \sum_{\mathbf{y} \in \boldsymbol{\mathcal{Y}}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_{\tau(j)} \mid \mathbf{pa}(y_{\tau(j)})) \\ &= \sum_{y_{\tau(1)} \in \mathcal{Y}_{\tau(1)}} \sum_{y_{\tau(2)} \in \mathcal{Y}_{\tau(2)}} \cdots \sum_{y_{\tau(d)} \in \mathcal{Y}_{\tau(d)}} \prod_{j=1}^{d} P_{\mathcal{B}}(y_{\tau(j)} \mid \mathbf{pa}(y_{\tau(j)}))\end{aligned} \tag{A.4}$$

By performing variable elimination from $\tau(d)$ to $\tau(1)$, i.e., according to the reverse of the topological ordering $\tau(1), \tau(2), \ldots, \tau(d)$, we observe that at each step $j \in \{1, 2, \ldots, d\}$,

$$\sum_{y_{\tau(d-j+1)} \in \mathcal{Y}_{\tau(d-j+1)}} P_{\mathcal{B}}(y_{\tau(d-j+1)} \mid \mathbf{pa}(y_{\tau(d-j+1)})) = 1 \tag{A.5}$$

Two simple examples to further explain this observation are given in Example A.1.1 and Example A.1.2.

Therefore, we have

$$\begin{aligned}\sum_{y_{\tau(1)}\in\mathcal{Y}_{\tau(1)}}\sum_{y_{\tau(2)}\in\mathcal{Y}_{\tau(2)}}\cdots\sum_{y_{\tau(d)}\in\mathcal{Y}_{\tau(d)}}\prod_{j=1}^{d}P_{\mathcal{B}}(y_{\tau(j)}\mid\mathbf{pa}(y_{\tau(j)}))=1\\ \Rightarrow\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}\prod_{j=1}^{d}P_{\mathcal{B}}(y_j\mid\mathbf{pa}(y_j))=1\end{aligned}\tag{A.6}$$

□

Example A.1.1 Assume we have a BN $\mathcal{B}$ with two feature variables X_1 and X_2 and two class variables Y_1 and Y_2. The structure $\mathcal{G}$ is given below:

$$\begin{tikzcd} X_1 \arrow[r] \arrow[d] & X_2 \arrow[d] \\ Y_1 \arrow[r] & Y_2 \end{tikzcd}$$

The mixed joint density function is factorized as follows:

$$f_{\mathcal{B}}(\mathbf{x},\mathbf{y})=p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)\,f_{\mathcal{B}}(x_1)\,f_{\mathcal{B}}(x_2\mid x_1)$$

The joint conditional probabilities can be simplified as

$$\begin{aligned}p_{\mathcal{B}}(\mathbf{y}\mid\mathbf{x})&=\frac{f_{\mathcal{B}}(\mathbf{y},\mathbf{x})}{\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}f_{\mathcal{B}}(\mathbf{y},\mathbf{x})}\\&=\frac{p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)\,f_{\mathcal{B}}(x_1)\,f_{\mathcal{B}}(x_2\mid x_1)}{\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)\,f_{\mathcal{B}}(x_1)\,f_{\mathcal{B}}(x_2\mid x_1)}\\&=\frac{(p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2))\cdot(f_{\mathcal{B}}(x_1)\,f_{\mathcal{B}}(x_2\mid x_1))}{\left(\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)\right)\cdot(f_{\mathcal{B}}(x_1)\,f_{\mathcal{B}}(x_2\mid x_1))}\\&=\frac{p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)}{\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)}\end{aligned}$$

To show that

$$p_{\mathcal{B}}(\mathbf{y}\mid\mathbf{x})=p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)$$

is equivalent to show that

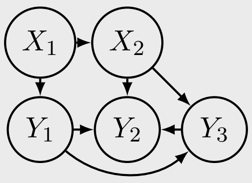
$$\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}}p_{\mathcal{B}}(y_1\mid x_1)\,p_{\mathcal{B}}(y_2\mid y_1,x_2)=1$$

Let us do it step by step:

$$\begin{aligned}\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, x_2) &= \sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, x_2)\\ &= \sum_{y_1\in\mathcal{Y}_1}\sum_{y_2\in\mathcal{Y}_2} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, x_2)\\ &= \sum_{y_1\in\mathcal{Y}_1} p_{\mathcal{B}}(y_1 \mid x_1) \sum_{y_2\in\mathcal{Y}_2} p_{\mathcal{B}}(y_2 \mid y_1, x_2)\\ &= \sum_{y_1\in\mathcal{Y}_1} p_{\mathcal{B}}(y_1 \mid x_1) = 1\end{aligned}$$

The above analysis is ensured $\mathbf{y}$ the fact that $\sum_{y_2\in\mathcal{Y}_2} p_{\mathcal{B}}(y_2 \mid y_1, x_2) = 1$ for any $(y_1, x_2) \in \mathcal{Y}_1 \times \mathcal{X}_2$ and $\sum_{y_1\in\mathcal{Y}_1} p_{\mathcal{B}}(y_1 \mid x_1) = 1$ for any $x_1 \in \mathcal{X}_1$.

Example A.1.2 Assume we have a BN $\mathcal{B}$ with two feature variables X_1 and X_2 and three class variables Y_1, Y_2 and Y_3. The structure $\mathcal{G}$ is given below:



$$\begin{tikzcd} X_1 \arrow[r] \arrow[d] & X_2 \arrow[d] \arrow[dr] & \\ Y_1 \arrow[r] \arrow[rr, bend right] & Y_2 & Y_3 \arrow[l] \end{tikzcd}$$

The mixed joint density function is factorized as follows:

$$f_{\mathcal{B}}(\mathbf{y}, \mathbf{x}) = p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\, f_{\mathcal{B}}(x_1)\, f_{\mathcal{B}}(x_2 \mid x_1)$$

The joint conditional probabilities can be simplified as

$$\begin{aligned} p_{\mathcal{B}}(\mathbf{y} \mid \mathbf{x}) &= \frac{f_{\mathcal{B}}(\mathbf{y}, \mathbf{x})}{\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} f_{\mathcal{B}}(\mathbf{y}, \mathbf{x})}\\ &= \frac{p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\, f_{\mathcal{B}}(x_1)\, f_{\mathcal{B}}(x_2 \mid x_1)}{p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\, f_{\mathcal{B}}(x_1)\, f_{\mathcal{B}}(x_2 \mid x_1)}\\ &= \frac{\left(p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\right) \cdot \left(f_{\mathcal{B}}(x_1)\, f_{\mathcal{B}}(x_2 \mid x_1)\right)}{\left(\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\right) \cdot \left(f_{\mathcal{B}}(x_1)\, f_{\mathcal{B}}(x_2 \mid x_1)\right)}\\ &= \frac{p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)}{\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)} \end{aligned}$$

To show that

$$p_{\mathcal{B}}(\mathbf{y} \mid \mathbf{x}) = p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)$$

is equivalent to show that

$$\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2) = 1$$

Again, we do it step by step:

$$\begin{aligned}
&\sum_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} \sum_{y_2\in\mathcal{Y}_2} \sum_{y_3\in\mathcal{Y}_3} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} \sum_{y_3\in\mathcal{Y}_3} \sum_{y_2\in\mathcal{Y}_2} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2)\, p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} \sum_{y_3\in\mathcal{Y}_3} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2) \sum_{y_2\in\mathcal{Y}_2} p_{\mathcal{B}}(y_2 \mid y_1, y_3, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} \sum_{y_3\in\mathcal{Y}_3} p_{\mathcal{B}}(y_1 \mid x_1)\, p_{\mathcal{B}}(y_3 \mid y_1, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} p_{\mathcal{B}}(y_1 \mid x_1) \sum_{y_3\in\mathcal{Y}_3} p_{\mathcal{B}}(y_3 \mid y_1, x_2) \\
&= \sum_{y_1\in\mathcal{Y}_1} p_{\mathcal{B}}(y_1 \mid x_1) = 1
\end{aligned}$$

A.2 Parameter and Structure Learning Algorithms under Mixed Data

Algorithm 8: Extract mixed training data

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \le l \le N\}$, class variable $Y_j$ ($1 \le j \le d$), parent set $\mathbf{\Pi}_j$, parent configuration $\boldsymbol{\pi}_{jk}$, parent set $\mathbf{\Lambda}_j$, parent configuration $\boldsymbol{\lambda}_{jn}$, parent set $\mathbf{\Phi}_j$
Ensure: $\mathcal{D}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})}$
1: Initialize $\mathcal{D}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})} \longleftarrow \emptyset$
2: for $l = 1, 2, \ldots, N$ do
3:     if $\boldsymbol{\pi}_j^l = \boldsymbol{\pi}_{jk} \wedge \boldsymbol{\lambda}_j^l = \boldsymbol{\lambda}_{jn}$ then
4:         Update $\mathcal{D}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})} \longleftarrow \mathcal{D}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})} \cup \{(\boldsymbol{\phi}_j^l, y_j)\}$
5:     end if
6: end for
```

Algorithm 9: Train local probabilistic model under mixed data

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}_c^l, \mathbf{x}_d^l, \mathbf{y}^l) \mid 1 \le l \le N\}$, class variable $Y_j$ ($1 \le j \le d$), parent set $\mathbf{\Pi}$, parent configuration $\boldsymbol{\pi}$, parent set $\mathbf{\Lambda}$, parent configuration $\boldsymbol{\lambda}$, parent set $\mathbf{\Phi}$ ($\mathbf{\Phi} = \mathbf{X}$ by default if not specified)
Ensure: Local classifier $\mathsf{C}_j^{(\boldsymbol{\lambda}, \boldsymbol{\pi})}$
1: Extract $\mathcal{D}_j^{(\boldsymbol{\lambda}, \boldsymbol{\pi})}$ using Algorithm 8 with input $\mathcal{D}$, $\mathbf{\Pi}$, $\boldsymbol{\pi}$, $\mathbf{\Lambda}$, $\boldsymbol{\lambda}$ and $\mathbf{\Phi}$
2: Train $\mathsf{C}_j^{(\boldsymbol{\lambda}, \boldsymbol{\pi})}$ using $\mathcal{D}_j^{(\boldsymbol{\lambda}, \boldsymbol{\pi})}$ by maximizing the CLL
```

Algorithm 10: Parameter learning in $\mathcal{G}$ under mixed data

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}^l, \mathbf{y}^l) \mid 1 \leq l \leq N\}$, BN structure $\mathcal{G}$
Ensure: Local classifiers $\mathsf{C}$
 1: Initialize $\mathsf{C} \longleftarrow \emptyset$
 2: for $j = 1, 2, \ldots, d$ do
 3:     Extract $\mathbf{\Pi}_j$, $\mathbf{\Lambda}_j$ and $\mathbf{\Phi}_j$ from $\mathcal{G}$
 4:     for $k = 1, 2, \ldots, q_j$ do
 5:         for $n = 1, 2, \ldots, e_j$ do
 6:             Train local classifier $\mathsf{C}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})}$ using Algorithm 9 with input $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j$, $\boldsymbol{\pi}_{jk}$, $\mathbf{\Lambda}_j$, $\boldsymbol{\lambda}_j$ and
                $\mathbf{\Phi}_j$
 7:             Update $\mathsf{C} \longleftarrow \mathsf{C} \cup \mathsf{C}_j^{(\boldsymbol{\lambda}_{jn}, \boldsymbol{\pi}_{jk})}$
 8:         end for
 9:     end for
10: end for
```

Algorithm 11: Structure learning of $\mathcal{G}_{\mathbf{DY}}$ under mixed data

```
Require: Training data $\mathcal{D} = \{(\mathbf{x}_c^l, \mathbf{x}_d^l, \mathbf{y}^l) \mid 1 \le l \le N\}$
Ensure: Class subgraph $\mathcal{G}_{\mathbf{Y}}$
 1: Initialize a score dictionary $\mathsf{S}$    ▷ where $\mathsf{S}[j][\mathbf{\Pi}]$ stores the local score for $Y_j$ given the parent set $\mathbf{\Pi}$
 2: Initialize a dictionary of selected feature variables $\mathsf{E}$    ▷ where $\mathsf{E}[j][\mathbf{\Pi}]$ stores the selected discrete feature variables $\mathbf{\Lambda}$ for the pair $(Y_j, \mathbf{\Pi})$
 3: for $j = 1, 2, \ldots, d$ do
 4:   Initialize candidate parent sets $\mathsf{K}_j$ of $Y_j$ as the powerset of $\mathbf{Y} \backslash Y_j$
 5:   Initialize candidate parent sets $\mathbf{F}_j$ of $Y_j$ as the powerset of $\mathbf{X}_d$
 6:   for $u = 1, 2, \ldots, |\mathsf{K}_j|$ do
 7:     Compute $\text{PEN}(\mathcal{G}_{\mathsf{K}_{ju}}) \longleftarrow -\frac{\log N}{2}(r_j - 1)q_j$    ▷ here $q_j = \prod_{c=1:Y_c \in \mathsf{K}_{ju}}^{d} r_c$
 8:     if $\mathsf{K}_{ju}$ can be pruned then    ▷ using Lemma 4.4.2
 9:       Remove all supersets of $\mathsf{K}_{ju}$ from $\mathsf{K}_j$
10:       continue
11:     end if
12:     Initialize $\mathsf{S}^* \longleftarrow -\infty$
13:     for $v = 1, 2, \ldots, |\mathsf{F}_j|$ do
14:       Compute $\text{PEN}(\mathcal{G}_{(\mathsf{K}_{ju}, \mathsf{F}_{jv})}) \longleftarrow -\frac{\log N}{2}(r_j - 1)(q_j \cdot e_j)$    ▷ here $q_j = \prod_{c=1:Y_c \in \mathsf{K}_{ju}}^{d} r_c$ and $e_j = \prod_{b=1:X_b \in \mathsf{F}_{jv}}^{m} |\mathcal{X}_b|$
15:       if $(\mathsf{K}_{ju}, \mathsf{F}_{jv})$ can be pruned then    ▷ using Lemma 4.4.2
16:         Remove all supersets of $\mathsf{F}_{jv}$ from $\mathsf{F}_j$
17:         continue
18:       end if
19:       Initialize $\mathsf{S} \longleftarrow 0$
20:       for $k = 1, 2, \ldots, q_j$ do
21:         for $n = 1, 2, \ldots, e_j$ do
22:           Extract $\mathcal{D}_j^{(\mathsf{f}_{jvn}, \mathsf{k}_{juk})}$ using Algorithm 8 with input $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j = \mathsf{K}_{ju}$, $\boldsymbol{\pi}_j = \mathsf{k}_{juk}$, $\mathbf{\Lambda}_j = \mathsf{F}_{jv}$, $\boldsymbol{\lambda}_j = \mathsf{f}_{jvn}$ and $\mathbf{\Phi} = \mathbf{X}$
23:           Train local classifier $\mathsf{C}_j^{(\mathsf{f}_{jvn}, \mathsf{k}_{juk})}$ using Algorithm 9 with input $\mathcal{D}$, $Y_j$, $\mathbf{\Pi}_j = \mathsf{K}_{ju}$, $\boldsymbol{\pi}_j = \mathsf{k}_{juk}$, $\mathbf{\Lambda}_j = \mathsf{F}_{jv}$, $\boldsymbol{\lambda}_j = \mathsf{f}_{jvn}$, $\mathbf{\Phi}_j = \mathbf{X}$
24:           Update $\mathsf{S} \longleftarrow \mathsf{S} + \mathsf{C}_j^{(\mathsf{f}_{jvn}, \mathsf{k}_{juk})}(\mathcal{D}_j^{(\mathsf{f}_{jvn}, \mathsf{k}_{juk})})$
25:         end for
26:       end for
27:       if $\mathsf{F}_{jv}$ can be pruned then    ▷ using Lemma 4.4.1
28:         continue
29:       end if
30:       if $\mathsf{S} > \mathsf{S}^*$ then
31:         Update $\mathsf{S}^* \longleftarrow \mathsf{S}$
32:         Update $\mathsf{E}[j][\mathsf{K}_{ju}] \longleftarrow \mathsf{F}_{jv}$
33:       end if
34:     end for
35:     if $\mathsf{K}_{ju}$ can be pruned then    ▷ using Lemma 4.4.1
36:       continue
37:     end if
38:     Update $\mathsf{S}[j][\mathsf{K}_{ju}] \longleftarrow \mathsf{S}$
39:   end for
40: end for
41: Learn $\mathcal{G}_{\mathbf{Y}}$ using GOBNILP with input $\mathsf{S}$
42: Form $\mathcal{G}_{\mathbf{DY}}$ using $\mathcal{G}_{\mathbf{Y}}$ and $\mathsf{E}$
```

References

Tsoumakas, G. and Katakis, I. (2007). Multi-Label Classification: An Overview. In: *Int. J. Data Warehous. Min.* 3.3, pp. 1–13. DOI: 10.4018/jdwm.2007070101 (cited on pages 1, 20).

Zhang, M. and Zhou, Z. (2014). A Review on Multi-Label Learning Algorithms. In: *IEEE Trans. Knowl. Data Eng.* 26.8, pp. 1819–1837. DOI: 10.1109/TKDE.2013.39 (cited on pages 1, 65).

Fernandez-Gonzalez, P., Larrañaga, P., and Bielza, C. (2015). Multidimensional classifiers for neuroanatomical data. In: *ICML Workshop on Statistics, Machine Learning and Neuroscience* (cited on page 1).

Ortigosa-Hernández, J., Rodríguez, J. D., Alzate, L., Lucania, M., Inza, I., and Lozano, J. A. (2012). Approaching Sentiment Analysis by using semi-supervised learning of multi-dimensional classifiers. In: *Neurocomputing* 92, pp. 98–115. DOI: 10.1016/j.neucom.2012.01.030 (cited on page 1).

Read, J., Bielza, C., and Larrañaga, P. (2014). Multi-Dimensional Classification with Super-Classes. In: *IEEE Trans. Knowl. Data Eng.* 26.7, pp. 1720–1733. DOI: 10.1109/TKDE.2013.167 (cited on pages 1, 2).

Bielza, C., Li, G., and Larrañaga, P. (2011). Multi-dimensional classification with Bayesian networks. In: *Int. J. Approx. Reason.* 52.6, pp. 705–727. DOI: 10.1016/j.ijar.2011.01.007 (cited on pages 2, 22, 23, 44).

Zaragoza, J. H., Sucar, L. E., Morales, E. F., Bielza, C., and Larrañaga, P. (2011). Bayesian Chain Classifiers for Multidimensional Classification. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. Ed. by T. Walsh. IJCAI/AAAI, pp. 2192–2197. DOI: 10.5591/978-1-57735-516-8/IJCAI11-365 (cited on page 2).

Ma, Z. and Chen, S. (2018). Multi-dimensional classification via a metric approach. In: *Neurocomputing* 275, pp. 1121–1131. DOI: 10.1016/j.neucom.2017.09.057 (cited on page 2).

Wing, J. M. (2021). Trustworthy AI. In: *Commun. ACM* 64.10, pp. 64–71. DOI: 10.1145/3448248 (cited on page 2).

Pearl, J. (1989). Probabilistic reasoning in intelligent systems - networks of plausible inference. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann (cited on pages 2, 13).

Jia, B. and Zhang, M. (2022a). Decomposition-Based Classifier Chains for Multi-Dimensional Classification. In: *IEEE Trans. Artif. Intell.* 3.2, pp. 176–191. DOI: 10.1109/TAI.2021.3110935 (cited on pages 2, 42, 44).

Jia, B. and Zhang, M. (2022b). Multi-dimensional Classification via Selective Feature Augmentation. In: *Int. J. Autom. Comput.* 19.1, pp. 38–51. DOI: 10.1007/s11633-022-1316-5 (cited on pages 2, 44).

Larrañaga, P., Karshenas, H., Bielza, C., and Santana, R. (2012). A review on probabilistic graphical models in evolutionary computation. In: *J. Heuristics* 18.5, pp. 795–819. DOI: 10.1007/s10732-012-9208-4 (cited on page 2).

Kyrimi, E., McLachlan, S., Dube, K., Neves, M. R., Fahmi, A., and Fenton, N. E. (2021). A comprehensive scoping review of Bayesian networks in healthcare: Past, present and future. In: *Artif. Intell. Medicine* 117, p. 102108. DOI: 10.1016/j.artmed.2021.102108 (cited on page 2).

Mihaljevic, B., Bielza, C., and Larrañaga, P. (2021). Bayesian networks for interpretable machine learning and optimization. In: *Neurocomputing* 456, pp. 648–665. DOI: 10.1016/j.neucom.2021.01.138 (cited on page 2).

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian Network Classifiers. In: *Mach. Learn.* 29.2-3, pp. 131–163. DOI: 10.1023/A:1007465528199 (cited on pages 2, 3, 10, 20, 21, 24).

van der Gaag, L. C. and de Waal, P. R. (2006). Multi-dimensional Bayesian Network Classifiers. In: *Third European Workshop on Probabilistic Graphical Models, 12-15 September 2006, Prague, Czech Republic. Electronic Proceedings*. Ed. by M. Studený and J. Vomlel, pp. 107–114 (cited on pages 3, 22).

Greiner, R., Grove, A. J., and Schuurmans, D. (1997). Learning Bayesian Nets that Perform Well. In: *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*. Ed. by D. Geiger and P. P. Shenoy. Morgan Kaufmann, pp. 198–207 (cited on page 3).

Ng, A. Y. and Jordan, M. I. (2001). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, pp. 841–848 (cited on page 3).

Kontkanen, P., Myllymäki, P., and Tirri, H. (2001). Classifier Learning with Supervised Marginal Likelihood. In: *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*. Ed. by J. S. Breese and D. Koller. Morgan Kaufmann, pp. 277–284 (cited on page 3).

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., and Tirri, H. (2005). On Discriminative Bayesian Network Classifiers and Logistic Regression. In: *Mach. Learn.* 59.3, pp. 267–296. DOI: 10.1007/s10994-005-0471-6 (cited on pages 3, 10, 26).

Bernardo, J. M. and Smith, A. (2000). Bayesian Theory. Wiley series in probability and statistics. Chichester: John Wiley & Sons Ltd. (cited on pages 3, 10).

de Campos, C. P., Scanagatta, M., Corani, G., and Zaffalon, M. (2018). Entropy-based pruning for learning Bayesian networks using BIC. In: *Artif. Intell.* 260, pp. 42–50. DOI: 10.1016/j.artint.2018.04.002 (cited on pages 4, 34, 64).

Dembczynski, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. In: *Mach. Learn.* 88.1-2, pp. 5–45. DOI: 10.1007/s10994-012-5285-8 (cited on page 10).

Waegeman, W., Dembczynski, K., Jachnik, A., Cheng, W., and Hüllermeier, E. (2014). On the bayes-optimality of F-measure maximizers. In: *J. Mach. Learn. Res.* 15.1, pp. 3333–3388. DOI: 10.5555/2627435.2750352 (cited on page 10).

Gil-Begue, S., Bielza, C., and Larrañaga, P. (2021). Multi-dimensional Bayesian network classifiers: A survey. In: *Artif. Intell. Rev.* 54.1, pp. 519–559. DOI: 10.1007/s10462-020-09858-x (cited on pages 10, 22–24, 26).

Nguyen, V. and Hüllermeier, E. (2021). Multilabel Classification with Partial Abstention: Bayes-Optimal Prediction under Label Independence. In: *J. Artif. Intell. Res.* 72, pp. 613–665. DOI: 10.1613/jair.1.12610 (cited on page 10).

Koller, D. and Friedman, N. (2009). Probabilistic Graphical Models - Principles and Techniques. MIT Press (cited on pages 13, 18).

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. In: *Mach. Learn.* 20.3, pp. 197–243. DOI: 10.1007/BF00994016 (cited on pages 15, 17, 21).

Akaike, H. (1974). A new look at the statistical model identification. In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723. DOI: 10.1109/TAC.1974.1100705 (cited on page 16).

Schwarz, G. (1978). Estimating the Dimension of a Model. In: *The Annals of Statistics* 6.2, pp. 461 –464. DOI: 10.1214/aos/1176344136 (cited on pages 16, 34).

Rissanen, J. (1978). Modeling by shortest data description. In: *Autom.* 14.5, pp. 465–471. DOI: 10.1016/0005-1098(78)90005-5 (cited on page 16).

Cooper, G. F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. In: *Mach. Learn.* 9, pp. 309–347. DOI: 10.1007/BF00994110 (cited on pages 17, 21).

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. In: *Decision Sciences* 8.1, pp. 156–166. DOI: https://doi.org/10.1111/j.1540-5915.1977.tb01074.x (cited on page 17).

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H. (1996). Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 18.9, pp. 912–926. DOI: 10.1109/34.537345 (cited on page 17).

Koivisto, M. and Sood, K. (2004). Exact Bayesian Structure Discovery in Bayesian Networks. In: *J. Mach. Learn. Res.* 5, pp. 549–573 (cited on page 17).

Silander, T. and Myllymäki, P. (2006). A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. In: *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence, Cambridge, MA, USA, July 13-16, 2006*. AUAI Press (cited on page 17).

Jaakkola, T. S., Sontag, D. A., Globerson, A., and Meila, M. (2010). Learning Bayesian Network Structure using LP Relaxations. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. Ed. by Y. W. Teh and D. M. Titterington. Vol. 9. JMLR Proceedings. JMLR.org, pp. 358–365 (cited on page 17).

Cussens, J. (2011). Bayesian network learning with cutting planes. In: *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*. Ed. by F. G. Cozman and A. Pfeffer. AUAI Press, pp. 153–160 (cited on pages 17, 33).

Cussens, J. and Bartlett, M. (2013). Advances in Bayesian Network Learning using Integer Programming. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. Ed. by A. E. Nicholson and P. Smyth. AUAI Press (cited on page 17).

Cussens, J. (2020). GOBNILP: Learning Bayesian network structure with integer programming. In: *International Conference on Probabilistic Graphical Models, PGM 2020, 23-25 September 2020, Aalborg, Hotel Comwell Rebild Bakker, Skørping, Denmark*. Ed. by M. Jaeger and T. D. Nielsen. Vol. 138. Proceedings of Machine Learning Research. PMLR, pp. 605–608 (cited on pages 17, 33, 64).

Suzuki, J. (1996). Learning Bayesian Belief Networks Based on the Minimum Description Length Principle: An Efficient Algorithm Using the B & B Technique. In: *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*. Ed. by L. Saitta. Morgan Kaufmann, pp. 462–470 (cited on page 17).

de Campos, C. P., Zeng, Z., and Ji, Q. (2009). Structure learning of Bayesian networks using constraints. In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by A. P. Danyluk, L. Bottou, and M. L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, pp. 113–120. DOI: 10.1145/1553374.1553389 (cited on page 17).

de Campos, C. P. and Ji, Q. (2011). Efficient Structure Learning of Bayesian Networks using Constraints. In: *J. Mach. Learn. Res.* 12, pp. 663–689. DOI: 10.5555/1953048.2021027 (cited on pages 17, 35).

Park, J. D. (2002). MAP Complexity Results and Approximation Methods. In: *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta,* *Canada, August 1-4, 2002*. Ed. by A. Darwiche and N. Friedman. Morgan Kaufmann, pp. 388–396 (cited on page 18).

Zhang, N. L. and Poole, D. (1994). A simple approach to Bayesian network computations. In: *Proc. of the Tenth Canadian Conference on Artificial Intelligence* (cited on pages 18, 39).

Beal, M. J. (2003). Variational algorithms for approximate Bayesian inference. PhD thesis. University College London, UK (cited on page 18).

Minka, T. P. (2001). Expectation Propagation for approximate Bayesian inference. In: *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*. Ed. by J. S. Breese and D. Koller. Morgan Kaufmann, pp. 362–369 (cited on page 18).

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier Chains for Multi-label Classification. In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II*. Ed. by W. L. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor. Vol. 5782. Lecture Notes in Computer Science. Springer, pp. 254–269. DOI: 10.1007/978-3-642-04174-7_17 (cited on page 20).

Dembczynski, K., Cheng, W., and Hüllermeier, E. (2010). Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Ed. by J. Fürnkranz and T. Joachims. Omnipress, pp. 279–286 (cited on page 20).

Nam, J., Mencía, E. L., Kim, H. J., and Fürnkranz, J. (2017). Maximizing Subset Accuracy with Recurrent Neural Networks in Multi-label Classification. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, pp. 5413–5423 (cited on page 20).

Gerych, W., Hartvigsen, T., Buquicchio, L., Agu, E., and Rundensteiner, E. A. (2021). Recurrent Bayesian Classifier Chains for Exact Multi-Label Classification. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, pp. 15981–15992 (cited on page 20).

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2021). Classifier Chains: A Review and Perspectives. In: *J. Artif. Intell. Res.* 70, pp. 683–718. DOI: 10.1613/jair.1.12376 (cited on page 20).

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. In: *Pattern Recognit.* 37.9, pp. 1757–1771. DOI: 10.1016/j.patcog.2004.03.009 (cited on page 20).

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). Pattern classification, 2nd Edition. Wiley (cited on page 21).

Maron, M. E. and Kuhns, J. L. (1960). On Relevance, Probabilistic Indexing and Information Retrieval. In: *J. ACM* 7.3, pp. 216–244. DOI: 10.1145/321033.321035 (cited on page 21).

Minsky, M. (1961). Steps toward artificial intelligence. In: *Proceedings of the IRE* 49.1, pp. 8–30. DOI: 10.1109/JRPROC.1961.287775 (cited on page 21).

Langley, P. and Sage, S. (1999). Tractable Average-Case Analysis of Naive Bayesian Classifiers. In: *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*. Ed. by I. Bratko and S. Dzeroski. Morgan Kaufmann, pp. 220–228 (cited on page 21).

Sahami, M. (1996). Learning Limited Dependence Bayesian Classifiers. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. Ed. by E. Simoudis, J. Han, and U. M. Fayyad. AAAI Press, pp. 335–338 (cited on page 21).

Spirtes, P., Glymour, C., and Scheines, R. (2000). Causation, Prediction, and Search, Second Edition. Adaptive computation and machine learning. MIT Press (cited on page 21).

Robinson, R. W. (1977). Counting unlabeled acyclic digraphs. In: *Combinatorial Mathematics V*. Ed. by C. H. C. Little. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 28–43. DOI: 10.1007/BFb0069178 (cited on pages 23, 27).

Pernkopf, F. and Bilmes, J. A. (2010). Efficient Heuristics for Discriminative Structure Learning of Bayesian Network Classifiers. In: *J. Mach. Learn. Res.* 11, pp. 2323–2360. DOI: 10.5555/1756006.1859932 (cited on page 26).

Chickering, D. M., Heckerman, D., and Meek, C. (2004). Large-Sample Learning of Bayesian Networks is NP-Hard. In: *J. Mach. Learn. Res.* 5, pp. 1287–1330 (cited on page 34).

Koivisto, M. (2006). Parent Assignment Is Hard for the MDL, AIC, and NML Costs. In: *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*. Ed. by G. Lugosi and H. U. Simon. Vol. 4005. Lecture Notes in Computer Science. Springer, pp. 289–303. DOI: 10.1007/11776420_23 (cited on page 34).

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. URL: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html (cited on page 43).

Zhu, S., Ji, X., Xu, W., and Gong, Y. (2005). Multi-labelled classification using maximum entropy method. In: *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. Ed. by R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait. ACM, pp. 274–281. DOI: 10.1145/1076034.1076082 (cited on page 44).

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. In: *J. Mach. Learn. Res.* 7, pp. 1–30 (cited on pages 45, 47).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cited on page 46).

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cited on page 46).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, pp. 8024–8035 (cited on page 46).

Wu, G. and Zhu, J. (2020). Multi-label classification: do Hamming loss and subset accuracy really conflict with each other? In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (cited on page 54).

Brier, G. W. (1950). Verification of Forecasts Expressed in Terms of Probability. In: *Monthly Weather Review* 78.1, pp. 1 –3. DOI: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2 (cited on page 65).

Benjumeda, M., Bielza, C., and Larrañaga, P. (2018). Tractability of most probable explanations in multidimensional Bayesian network classifiers. In: *Int. J. Approx. Reason.* 93, pp. 74–87. DOI: `10.1016/j.ijar.2017.10.024` (cited on page 65).