Title
Probabilistic Reasoning for Fair and Robust Decision Making

Author
Choi, YooJung

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Probabilistic Reasoning for Fair and Robust Decision Making

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

YooJung Choi

2022

ABSTRACT OF THE DISSERTATION

Probabilistic Reasoning for Fair and Robust Decision Making

by

YooJung Choi
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2022
Professor Guy Van den Broeck, Chair

Automated decision-making systems are increasingly being deployed in areas with high personal and societal impact. This naturally led to growing interest in trustworthy artificial intelligence (AI) and machine learning (ML), encompassing many fields of research including algorithmic fairness, robustness, explainability, privacy, and more. These works share a common theme of questioning and moderating the behavior of automated tools in various real-world settings, which inherently exhibit different uncertainties.

This dissertation explores how probabilistic modeling and reasoning as a framework offer a principled way to handle uncertainties when addressing trustworthy AI issues, in particular by explicitly modeling the underlying distribution of the world. The main contributions are as follows. First, it demonstrates that many problems in trustworthy AI can be cast as *probabilistic reasoning* tasks of varying complexities. Secondly, it proposes algorithms to learn fair and robust decision-making systems, while handling many sources of uncertainties such as missing or biased labels at training time and missing features at prediction time. The proposed approach relies heavily on probabilistic models that are *expressive* enough to describe the world underlying the system, whilst being *tractable* enough to answer various probabilistic queries. The final contribution of this thesis

is showing that *probabilistic circuits* are an effective model for this framework and expanding their reasoning capabilities even further.

The dissertation of YooJung Choi is approved.

Kai-Wei Chang

Stefano Soatto

Rina Dechter

Guy Van den Broeck, Committee Chair

University of California, Los Angeles

2022

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

There are many who helped me along the way on this exciting but challenging journey.

First and foremost I am extremely grateful to my advisor Guy Van den Broeck for his support and guidance. I appreciate the constructive feedback and encouragement I have gotten over the years, as well as the interesting research discussions and freedom to explore on my own when I needed them. Looking back to when I started my graduate studies, I realize I was ignorant to what PhD or academia is really like. Guy has not only taught me to be an independent researcher but also inspired me to be an effective communicator and a caring mentor.

I am also honored to have Profs. Stefano Soatto, Rina Dechter, and Kai-Wei Chang on my dissertation committee. I would like to express my gratitude for their valuable feedback on this thesis and for all the insightful questions and discussions.

I consider myself extremely lucky to have shared this journey with the amazing members of the StarAI Lab and friends at UCLA: Tal Friedman, Yitao Liang, Steven Holtzen, Pasha Khosravi, Zhe Zeng, Honghua Zhang, Kareem Ahmed, Antonio Vergari, Anji Liu, Meihua Dang, Poorva Garg, Yujia Shen, and Arthur Choi. All of you shaped my PhD life, from the whiteboard discussions and late-night writing sessions before deadlines to the coffee breaks and board game nights.

To DJ, my biggest supporter: you helped me through the stressful times and were always there to share the highlights. Thank you for being patient when I am distracted and for always making me laugh.

Finally, I want to thank my family for understanding and putting up with me when I missed many events. I could not have undertaken this journey without their unconditional love and support.

VITA

2012–2016 B.S. in Computer Science, *summa cum laude*. UCLA

PUBLICATIONS

YooJung Choi, Tal Friedman, and Guy Van den Broeck. Solving Marginal MAP Exactly by Probabilistic Circuit Transformations. *In Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS), 2022.*

Antonio Vergari, **YooJung Choi**, Anji Liu, Stefano Teso, and Guy Van den Broeck. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. *In Advances in Neural Information Processing Systems 35 (NeurIPS), 2021.*

YooJung Choi, Meihua Dang, and Guy Van den Broeck. Group Fairness by Probabilistic Modeling with Latent Fair Decisions. *In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI), 2021.*

YooJung Choi*, Golnoosh Farnadi*, Behrouz Babaki*, and Guy Van den Broeck. Learning Fair Naive Bayes Classifiers by Discovering and Eliminating Discrimination Patterns. *In Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI), 2020.*

Alicia Solow-Niederman, **YooJung Choi**, and Guy Van den Broeck. The Institutional Life of Algorithmic Risk Assessment. *In Berkeley Technology Law Journal (BTLJ), 2019.*

YooJung Choi and Guy Van den Broeck. On Robust Trimming of Bayesian Network Classifiers. *In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), 2018.*

YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal Feature Selection for Decision Robustness in Bayesian Networks. *In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), 2017.*

CHAPTER 1

Introduction

Machine learning systems are increasingly being used for critical decision making in a wide range of areas: from personalized ads and financial lending, to healthcare and criminal justice settings [Chouldechova, 2017; Berk et al., 2018; Datta et al., 2015; Henderson et al., 2015]. Despite their significant impact, these systems are often used without much reasoning about their behaviors, as black-box functions only comparable by metrics such as classification accuracy. As a result, there has been growing interest and need for methods that provide explanations and guarantees, especially relating to building trustworthy AI/ML systems that are robust, fair, accountable, and explainable [Choi et al., 2012; Dwork et al., 2012; Barocas et al., 2019; Gunning, 2017].

Questions about model behaviors such as robustness and fairness must be answered with respect to the world in which the model will operate. For instance, an important consideration for algorithmic fairness is the existence of *proxy* variables. These are variables that are correlated with sensitive attributes, such as race and gender which are protected by law, and may leak information and introduce bias even when the sensitive attributes are not directly used to make decisions (e.g. zip code as a proxy to race). The degree to which a variable is correlated with a sensitive attribute depends on the underlying population; in fact, a seemingly innocuous variable in one population may be a problematic proxy in another. While it is practically impossible to capture a perfect description of the world in all its details, we can use probabilistic models to represent the underlying distribution with inherent uncertainties.

Given such a model of the world, various questions in the field of trustworthy AI can be cast as probabilistic inference tasks on the model. For example, one can provide explanations

for a certain instance of image classification by asking which subset of the pixels lead to the same classification with the highest probability. In addition, a simple notion of fairness checks whether the average decision differs significantly between protected groups (e.g. between males and females). This corresponds to comparing the expectation of a model output, computed with respect to the underlying distribution of each subpopulation. Therefore, a probabilistic model with flexible inference capabilities would allows us to reason about different trustworthy AI behaviors.

Furthermore, there are additional sources of uncertainty when AI/ML systems are deployed in the real world. While models are defined over a set of features, observing a feature is often associated with a cost in practical settings. Consider a medical diagnosis setting: a patient is diagnosed without running all possible tests, as that would be costly and unrealistic. Thus, different subsets of features may be observed for different individuals, or the set of features may need to be reduced, in which case one may wonder how robust the decision is against potential outcomes of unobserved features. Moreover, there may be noise or bias in the training labels. This certainly makes it challenging not only to learn fair classifiers but even to measure fairness. As we will show, probabilistic modeling and reasoning provide a clear language and tool to reason about these model behaviors while handling aforementioned types of uncertainties.

This thesis seeks to develop probabilistic reasoning algorithms for quantifying and verifying robustness and fairness, and in turn, to use those to learn decision-making systems that can give guarantees. In particular, as these reasoning tasks are often computationally hard, we especially focus on tractable probabilistic models (TPMs), and in particular probabilistic circuits, which are expressive models that allow for reliable and efficient inference. More concretely, the main contributions of this thesis are: (1) showing how different robustness and fairness notions can be written as probabilistic queries of varying complexities; (2) for each of those queries, identifying a corresponding family of probabilistic circuits for tractable inference; (3) proposing algorithms to learn probabilistic models that satisfy previously defined robustness and fairness guarantees; and lastly, (4) introducing new techniques for tractable inference on probabilistic circuits.

1.1 Structure of the Thesis

Chapter 2 provides key background in probabilistic modeling and reasoning. It discusses the trade-off between expressiveness and tractability of probabilistic models, and reviews some common inference tasks. The remainder of the chapter studies tractable circuit representations, their syntax and semantics as well as how they support efficient inference of a large class of probabilistic queries.

The next two chapters demonstrate how tractable probabilistic models can be leveraged for fairness-aware machine learning. Chapter 3, based on Choi et al. [2021], addresses the common problem of biased class labels in the training data. Specifically, the biased labels are treated as noisy versions of some hidden fair label, which we explicitly model as a latent variable. This approach is enabled by the tractability of probabilistic circuits in handling latent variables during learning and inference, in addition to their expressiveness in modeling real-world distributions. Next, Chapter 4 identifies how existing notions of fairness often overlooks the common setting in which predictions or decisions are made with missing information, and proposes a new notion called *discrimination patterns* to tackle this scenario. Focusing on probabilistic classifiers which naturally handle predictions with missing values, it then presents algorithms to mine discrimination patterns from a given model and to learn naive Bayes classifiers while eliminating discrimination patterns. This chapter is largely based on Choi et al. [2020a] and includes part of a work under review [Selvam et al., 2022].

Chapter 5 is concerned with developing decision-making systems that are robust to future observations. As noted previously, observing features are often expensive, and thus decisions may need to be made only with partial information. Then it is valuable to know whether such decisions are likely to stay the same even with further observations. This chapter proposes a *robust trimming* of probabilistic classifiers as a way to derive classifiers that use a smaller number of features while making robust decisions. It shows that the feature selection criterion is a computationally hard probabilistic query and proposes the first exact inference algorithm for it, using probabilistic circuits. The work discussed in this chapter was previously published in Choi and Van den Broeck [2018]

and Choi et al. [2017].

In Chapter 6 we turn our attention to further advancing the tractable probabilistic inference framework to answer more complex queries. The work on algorithmic fairness and robustness presented in this thesis—as well as other methods in trustworthy AI such as explainability—is made possible by the tractable inference capabilities of probabilistic circuits. This chapter thus studies how we can push this framework and answer harder queries, which would in turn expand the problems in trustworthy AI that we can tackle. In particular, it presents a new framework of exact inference through circuit transformations, focusing on the marginal MAP query [Choi et al., 2022].

Other works that are relevant but not necessarily a central chapter to this dissertation are discussed in short intermezzos throughout the thesis. They include *expected predictions* [Khosravi et al., 2019b,a, 2020], *algorithmic fairness in law* [Solow-Niederman et al., 2019], and *inference by composition of circuit transformations* [Vergari et al., 2021].

Finally, Chapter 7 concludes by summarizing the thesis and discussing future research.

CHAPTER 2

Foundations

This chapter will provide the background on probabilistic modeling and inference using tractable circuit representations, which is integral to our proposed approach for robust and fair decision making.

Notation We use uppercase letters (e.g., X) for discrete random variables and lowercase letters (x) for their assignments. Negation of a binary assignment x is denoted by $\bar{x}$. Sets of variables are denoted by bold uppercase letters ($\mathbf{X}$), and their joint assignments by bold lowercase ($\mathbf{x}$). The set of all possible values or assignments to variables $\mathbf{X}$ is written as $\mathsf{val}(\mathbf{X})$. Concatenations of sets ($\mathbf{XY}$) represent their union ($\mathbf{X} \cup \mathbf{Y}$).

2.1 Probabilistic Models and Queries

A probabilistic model is a particular representation of a probability distribution. The simplest form of a model for a distribution over discrete variables would be a joint probability table. Each row of the table corresponds to a joint assignment to the variables and is associated with a probability mass. Table 2.1 shows some example rows of a joint probability table over 4 Boolean variables D, R_1, R_2, AC. For a distribution over n Boolean variables, the table would require 2^n rows which is clearly not feasible in practice. Therefore, most probabilistic models are concerned with more *concisely* representing distributions.

Probabilistic graphical models (PGMs), such as Bayesian networks and factor graphs, represent

Table 2.1: Joint Probability Table

D	R_1	R_2	AC	$\Pr(D, R_1, R_2, AC)$
$+$	$+$	$+$	$+$	0.0756
$+$	$+$	$+$	$-$	0.0084
$+$	$+$	$-$	$+$	0.0112
$+$	$+$	$-$	$-$	0.0448
		$\vdots$		$\vdots$
$-$	$-$	$-$	$+$	0
$-$	$-$	$-$	$-$	0.576

a probability distribution with a graph structure and a set of small tables associated with nodes in the graph [Koller and Friedman, 2009; Darwiche, 2009a]. For example, a Bayesian network (BN) consists of a directed acyclic graph (DAG) whose nodes represent random variables and are each associated with a conditional probability table (CPT). Figure 2.1 shows an example Bayesian network representing a paper review scenario. The variable D denotes the true quality of a submitted paper; two reviewers, R_1 and R_2, independently evaluate the paper, and their assessments are summarized by the area chair AC. We can see from the CPTs that, for example, R_1 will positively rate a paper of high quality with 0.7 probability, and AC will agree with two positive reviews with 0.9 probability. Note that this distribution can be described by a Bayesian network with 9 parameters, as opposed to $2^4 - 1 = 15$ free parameters in the case of a joint probability table. Moreover, the DAG structure of a Bayesian network can concisely represent conditional dependencies in the probability distribution. For instance, in Figure 2.1, the random variables R_1 and R_2 are conditionally independent given D.

Given such representation of the world as a probability distribution, we would naturally want to ask questions, or *queries*, about various quantities of interest from the distribution. One of the most basic queries asks about the probability of some complete assignment to the random variables. We call this a *complete evidence* query or a likelihood. Bayesian networks can compute a complete evidence query in linear time in the number of variables. For example, the BN in Figure 2.1

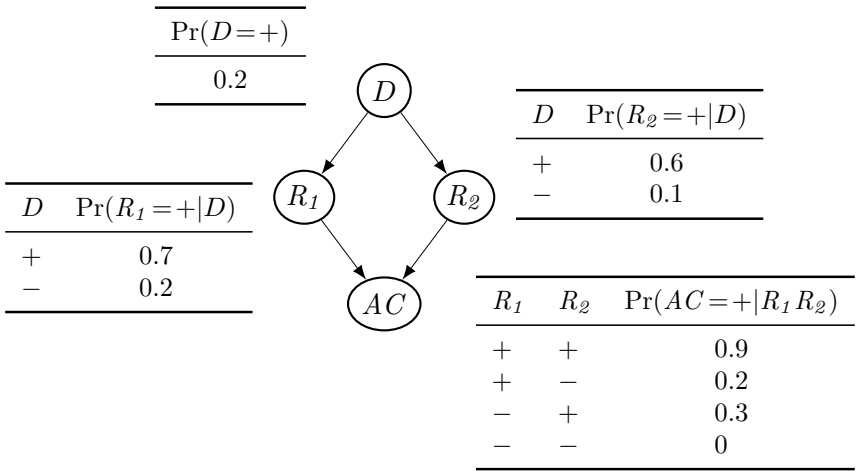


Figure 2.1: Bayesian network for review decisions

factorizes the distribution as:

$$\Pr(D, R_1, R_2, AC) = \Pr(D) \cdot \Pr(R_1 \mid D) \cdot \Pr(R_2 \mid D) \cdot \Pr(AC \mid R_1, R_2).$$

Hence, a complete evidence query $\Pr(D = +, R_1 = +, R_2 = -, AC = +)$ can be computed as $0.2 \cdot 0.7 \cdot 0.4 \cdot 0.2 = 0.0112$.

On the other hand, we may also want to query the probability of an event that is given by a partial assignment, rather than a complete assignment to all variables; computing queries of such form is called the *marginal inference*. For instance, the probability that the area chair will recommend accept is given by the sum of probabilities of all complete evidence that set $AC = +$.

$$\Pr(AC = +) = \sum_{d, r_1, r_2} \Pr(D = d, R_1 = r_1, R_2 = r_2, AC = +)$$

In addition, we could also query the probability of this event conditioned on another event, such as the paper having high quality ($D = +$):

$$\Pr(AC = + \mid D = +) = \frac{\Pr(AC = +, D = +)}{\Pr(D = +)}.$$

This is called a *conditional inference*. In general, computing marginal and conditional probabilities of PGMs is a #P-hard task.

Nevertheless, there exist subclasses of PGMs that support *tractable* marginal inference—that is, computing marginal probabilities in polynomial time. For example, naive Bayes and other tree-structured PGMs allow linear-time marginal inference. The cost of this tractability is their *expressivity*: these models make assumptions about (conditional) independence between variables that prohibit them from representing certain distributions. For example, a naive Bayes network with D as the root will not be able to represent the distribution given by the BN in Figure 2.1.

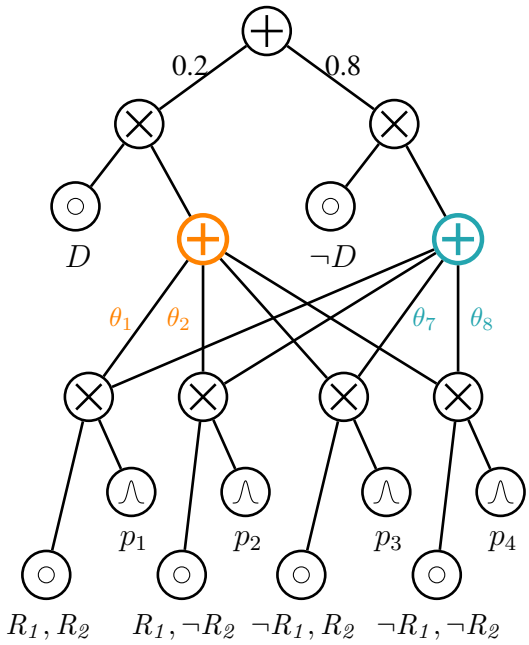
Hence, choosing a specific family of probabilistic models often involve making a tradeoff between how tractable the models are for certain queries and how expressive they are in concisely representing a wide range of distributions. The probabilistic reasoning framework for fair and robust decision making developed by this thesis employs a family of models called probabilistic circuits, which aims to strike a balance and achieve expressivity as well as tractability for many queries of interest. The remainder of this chapter provides relevant background on probabilistic circuits.

2.2 Probabilistic Circuits

A large family of tractable probabilistic models—including arithmetic circuits [Darwiche, 2002, 2003], and-or search spaces [Dechter and Mateescu, 2007] and multi-valued decision diagrams [Mateescu et al., 2008], probabilistic sentential decision diagrams [Kisa et al., 2014a], cutset networks [Rahman et al., 2014], and sum-product networks [Poon and Domingos, 2011]—can collectively be understood using the framework of *probabilistic circuits* (PCs) [Vergari et al., 2020].

2.2.1 Syntax and Semantics

A probabilistic circuit (PC) $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ over variables $\mathbf{X}$, characterized by its structure $\mathcal{G}$ and parameters $\boldsymbol{\theta}$, defines a (possibly unnormalized) probability distribution over $\mathbf{X}$ in a recursive



$\theta_1 = 0.7 \cdot 0.6$
$\theta_2 = 0.7 \cdot 0.4$
$\theta_3 = 0.3 \cdot 0.6$
$\theta_4 = 0.3 \cdot 0.4$
$\theta_5 = 0.2 \cdot 0.1$
$\theta_6 = 0.2 \cdot 0.9$
$\theta_7 = 0.8 \cdot 0.1$
$\theta_8 = 0.8 \cdot 0.9$

$p_1(AC = +) = 0.9$
$p_2(AC = +) = 0.2$
$p_3(AC = +) = 0.3$
$p_4(AC = +) = 0$

Figure 2.2: A probabilistic circuit over variables D, R_1, R_2, AC. For graphical conciseness, a node labeled R_1, R_2 denotes a product node with literals R_1 and R_2 as inputs. The edge parameters of the orange (resp. blue) sum node are $\theta_1, \ldots, \theta_4$ (resp. $\theta_5, \ldots \theta_8$) from left to right.

manner. The circuit structure $\mathcal{G}$ is a directed acyclic graph (DAG) such that each inner node is either a *sum* node or a *product* node, and each *leaf* (input) node is associated with a univariate input distribution. We denote the distribution associated with leaf n by $f_n(.)$. This may be any probability mass function, a special case being an indicator function such as $[X = 1]$. Every edge (n, c) between a sum unit n and its child c is also associated with a parameter $\theta_{n,c} > 0$. A subcircuit rooted at a PC node is itself a valid probabilistic circuit.

Let $\mathsf{ch}(n)$ denote the set of children, or inputs, of an inner node n. A *scope* of a node n, denoted by $\phi(n)$, refers to all variables that appear in the subcircuit rooted at n. In other words, the scope of a leaf node is simply the variable associated with the univariate distribution $f_n(.)$, and the scope of an inner unit is the union of the scopes of its inputs: $\phi(n) = \bigcup_{c \in \mathsf{ch}(n)} \phi(c)$. A PC node n whose

scope is $\mathbf{X}$ then recursively defines a distribution as the following: for every $\boldsymbol{x} \in \mathsf{val}(\mathbf{X})$,

$$n(\boldsymbol{x}) = \begin{cases} f_n(\boldsymbol{x}) & \text{if } n \text{ is a leaf node} \\ \prod_{c \in \mathsf{ch}(n)} c(\boldsymbol{x}) & \text{if } n \text{ is a product node} \\ \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \cdot c(\boldsymbol{x}) & \text{if } n \text{ is a sum node} \end{cases}$$

We write $\mathcal{C}(\boldsymbol{x})$ to refer to $n(\boldsymbol{x})$—the output of node n—where n is the root of the PC $\mathcal{C}$. If the PC defines a normalized distribution, we equivalently use $\Pr_n(\boldsymbol{x})$. Lastly, the *support* of n is the set of all complete assignments to $\mathbf{X}$ for which the output of n is non-zero: $\mathsf{supp}(n) = \{\boldsymbol{x} \in \mathsf{val}(\mathbf{X}) \mid n(\boldsymbol{x}) \neq 0\}$. Figure 2.2 illustrates an example probabilistic circuit, representing the same distribution as the Bayesian network in Figure 2.1. Note that the literals D and $\neg D$ refer to the positive $[D = +]$ and negative $[D = -]$ assignments to D, respectively.

2.2.2 Tractable Inference

A key strength of probabilistic circuits is that, based on the structural properties they satisfy, certain queries can be answered in polynomial time in the size of the circuit. In other words, they support *tractable* inference of different classes of queries. We now define the structural properties needed for some key query classes.

2.2.2.1 Marginal and Conditional Probability

Probabilistic circuits support efficient marginal inference if they are smooth and decomposable.

Definition 2.1 (Smoothness)**.** *A circuit is* smooth *if for every sum node n, its inputs depend on the same variables:* $\forall\, c_1, c_2 \in \mathsf{ch}(n),\ \phi(c_1) = \phi(c_2)$.

Smooth PCs generalize shallow mixture models [McLachlan et al., 2019] to deep and hierarchical models. Smoothness is sometimes referred to as *completeness* [Poon and Domingos, 2011].

Definition 2.2 (Decomposability)**.** *A circuit is* decomposable *if the inputs of every product node n*

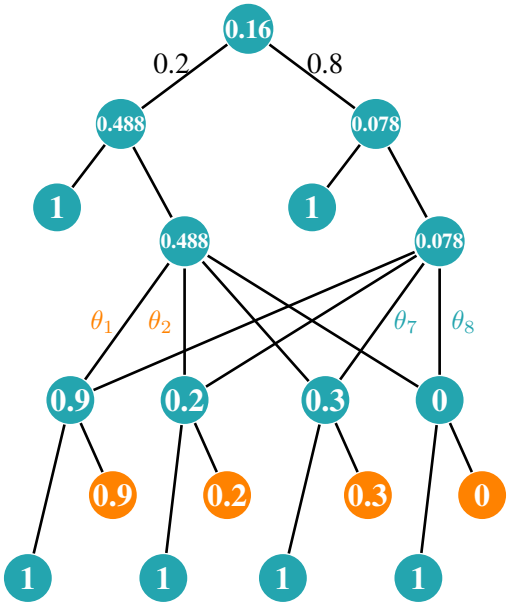


Figure 2.3: Computing the marginal probability $\Pr(AC = +)$ on the PC in Figure 2.2.

depend on disjoint sets of variables: $\forall\, c_1, c_2 \in \mathsf{ch}(n),\ \phi(c_1) \cap \phi(c_2) = \emptyset$.

Decomposable product nodes encode local factorizations. That is, a decomposable product n over variables $\mathbf{X}$ encodes $n(\mathbf{X}) = c_1(\mathbf{X}_1) \cdot c_2(\mathbf{X}_2) \cdots c_m(\mathbf{X}_m)$, where $m = |\mathsf{ch}(n)|$, and $\mathbf{X}_1, \ldots, \mathbf{X}_m$ form a partition of $\mathbf{X}$.

Taken together, decomposability and smoothness are sufficient conditions[1] for performing tractable marginalization over arbitrary sets of variables in a single feedforward pass, as they enable larger sums to be efficiently decomposed into smaller ones [Darwiche and Marquis, 2002; Choi et al., 2020b]. Next proposition formalizes it.

Proposition 2.1 (Tractable marginals)**.** *Let $\mathcal{C}$ be a smooth and decomposable circuit defining a distribution $\Pr(.)$ over discrete variables $\mathbf{X}$. Then for any variables $\mathbf{Y} \subseteq \mathbf{X}$ and their assignment $\boldsymbol{y}$, the marginalization $\sum_{\boldsymbol{z} \in \mathsf{val}(\mathbf{Z})} \Pr(\boldsymbol{y}, \boldsymbol{z})$ can be computed exactly in $\Theta(|\mathcal{C}|)$ time, where $\mathbf{Z}$ denotes $\mathbf{X} \setminus \mathbf{Y}$.*

[1]They are also the necessary conditions if the variables in the PC may be continuous or discrete. If the PC is defined over Boolean variables, smoothness and *consistency* [Poon and Domingos, 2011]—a relaxed notion of decomposability—suffice.

To answer a marginal query in a PC, we traverse the circuit in a feedforward manner as follows. Leaf node n is evaluated as 1 if it does not depend on a variable in $\mathbf{Y}$, and as $f_n(\boldsymbol{y})$ otherwise. Then we simply evaluate the circuit, taking (weighted) sums and products accordingly. Moreover, any conditional query can be written as a ratio of marginal probabilities, and can thus be computed by two feedforward evaluations of the PC.

For instance, we can easily see that the PC in Figure 2.2 is smooth and decomposable. Then to compute the marginal $\Pr(AC = +)$, we evaluate the leaf nodes whose scope is AC (i.e. those marked as $p_1, \ldots, p_4$) with the positive assignment and replace all other leaf nodes with 1. Evaluating the circuit bottom up, we get the marginal probability $\Pr(AC = +) = 0.16$. This is illustrated in Figure 2.3.

2.2.2.2 Maximum A Posteriori

Another useful structural property is *determinism*; a PC over variables $\mathbf{X}$ is deterministic if for every complete input $\mathbf{x}$, at most one child of every sum node has a non-zero output. In addition to enabling tractable inference for more queries [Choi and Darwiche, 2017], it leads to closed-form parameter estimation of probabilistic circuits given complete data.

Definition 1 (Determinism)**.** *A circuit is* deterministic *if the inputs of every sum node n have disjoint supports:* $\forall\, c_1, c_2 \in \mathsf{ch}(n),\ c_1 \neq c_2 \implies \mathsf{supp}(c_1) \cap \mathsf{supp}(c_2) = \emptyset$.

Analogously to decomposability, determinism induces a recursive partitioning, but this time over the support of a circuit. Determinism together with decomposability allows for tractable maximization of circuits [Darwiche, 2009b]. The maximum a posteriori (MAP) queries—also called the most probable explanation (MPE)—ask to find the complete assignment that maximizes the probability, possibly given some evidence. This can be done in linear time for decomposable and deterministic PCs, again by a feedforward evaluation after replacing each sum node with a maximization node. For instance, consider the following MAP query on the example PC in

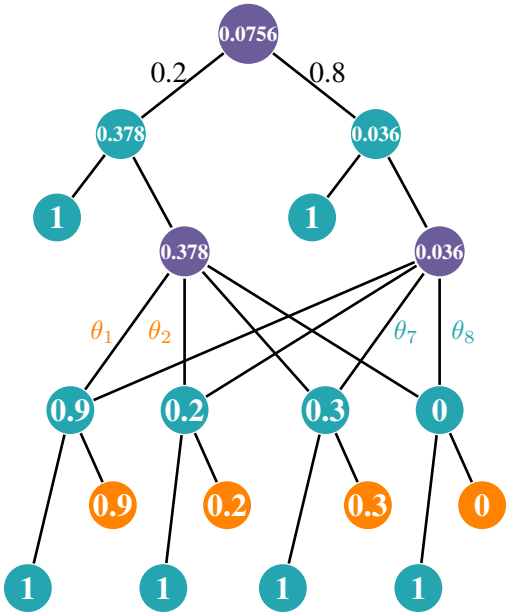


Figure 2.4: Computing the MAP query $\max_{d,r_1,r_2} \Pr(D = d, R_1 = r_1, R_2 = r_2, AC = +)$ on the PC in Figure 2.2. The max nodes are highlighted in purple.

Figure 2.2:

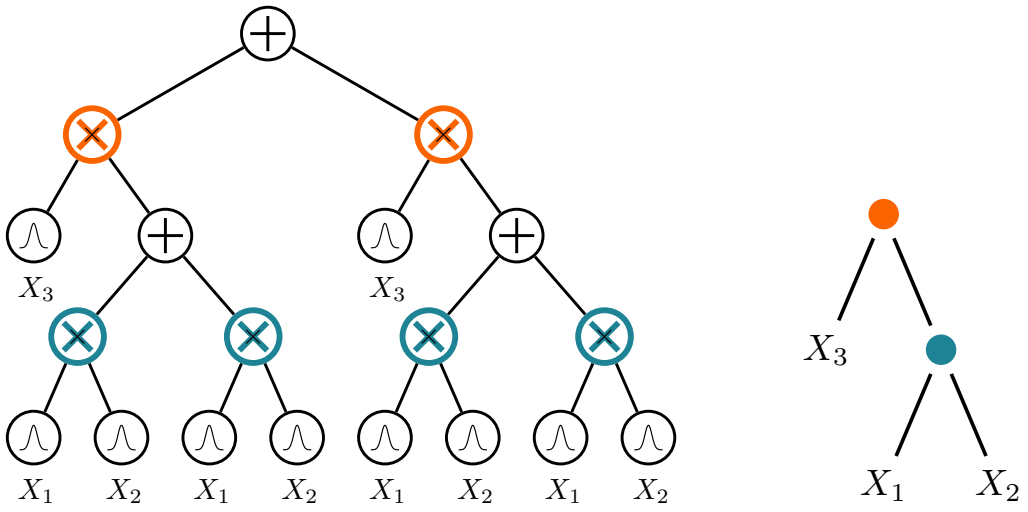
$$\max_{d,r_1,r_2} \Pr(D = d, R_1 = r_1, R_2 = r_2, AC = +).$$

Because the PC is also deterministic, we can perform MAP inference as illustrated in Figure 2.4.

2.2.2.3 Beyond Marginals and MAP

Many types of PCs support tractable inference of harder queries by additionally inducing a form of hierarchical scope partitioning [Pipatsrisawat and Darwiche, 2008; Mateescu et al., 2008]. In particular, we study the property called *compatibility* [Vergari et al., 2021], which is defined over pairs of PCs and asks whether they share the same scope decompositions. Intuitively, two decomposable circuits are compatible if they can be rearranged in polynomial time[2] such that their respective product units, once matched by scope, decompose in the same way. We formalize this

[2]By changing the order in which n-ary product units are turned into a series of binary product units.



(a) A structured decomposable circuit (b) A vtree

Figure 2.5: A structured decomposable PC over $\mathbf{X} = \{X_1, X_2, X_3\}$ and its corresponding vtree.

with the following inductive definition.

Definition 2 (Compatibility)**.** *Two circuits $\mathcal{C}$ and $\mathcal{C}'$ over variables $\mathbf{X}$ are* compatible *if (1) they are smooth and decomposable and (2) any pair of product units $n \in \mathcal{C}$ and $m \in \mathcal{C}'$ with the same scope can be rearranged into products that are mutually compatible and decompose in the same way: $(\phi(n) = \phi(m)) \implies (\phi(n_i) = \phi(m_i),\ n_i$ and m_i are compatible$)$ for some ordering of the inputs of n and m.*

We can derive from compatibility the following property pertaining to a single circuit, which will be useful in our analysis later.

Definition 3. *A circuit is* structured-decomposable *if it is compatible with itself.*

Figure 2.5a shows an example of a structured decomposable PC. The decomposition of variables can be described using a *vtree*: a full binary tree structure, with each of its leaves corresponding to a variable in the PC [Pipatsrisawat and Darwiche, 2008]. Two structured-decomposable PCs are compatible if they share the same vtree.

In addition, a structured decomposable PC is *strongly deterministic* if every sum node is deterministic in the children nodes corresponding to its left vtree node. Consider the example

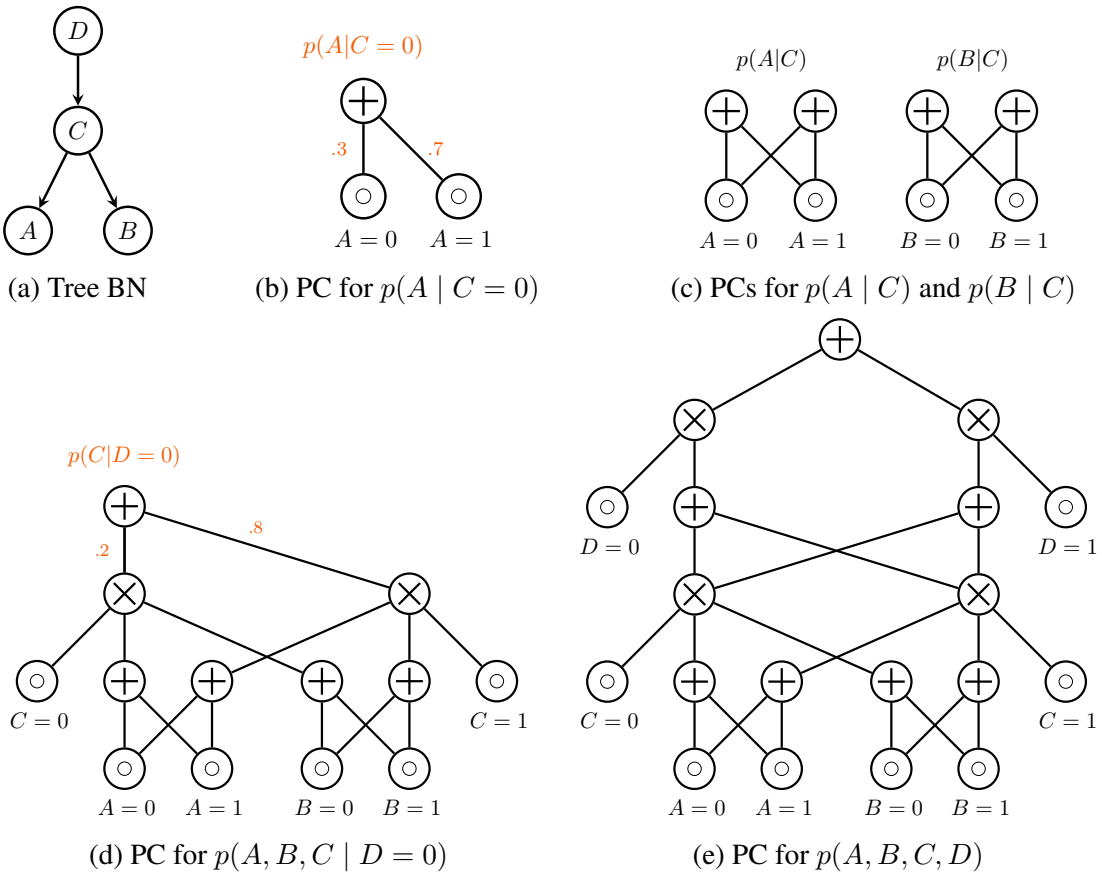
PC in Figure 2.5a. The root sum node is strongly deterministic if the two leaf nodes for X_3 have disjoint supports; i.e., at most one outputs non-zero for every input with an assignment to X_3. PCs satisfying structured decomposability and strong determinism are known to be tractable, for certain variable ordering in the PCs, for hard queries such as marginal MAP and the same-decision probability [Marinescu et al., 2018; Oztok et al., 2016], which are $\mathsf{NP}^{\mathsf{PP}}$- and $\mathsf{PP}^{\mathsf{PP}}$-complete, respectively, for PGMs in general. We will more closely study these queries and the class of PCs tractable for them in Chapters 5 and 6.

2.2.3 Learning and Compiling PCs

Probabilistic circuits are expressive models, shown to achieve competitive likelihoods in various density estimation tasks [Dang et al., 2022; Liu et al., 2022; Peharz et al., 2020]. There exist various probabilistic circuit structure and parameter learning methods [Peharz et al., 2020; Rahman and Gogate, 2016; Liang et al., 2017; Dang et al., 2020] to directly fit the data. Alternatively, PCs can also be compiled from other types of probabilistic models.

Bounded-treewidth graphical models—including naive Bayes networks, Chow-Liu trees [Chow and Liu, 1968], polytrees [Dasgupta, 2013], and thin junction trees [Bach and Jordan, 2001]—can be translated into smooth, decomposable, and deterministic probabilistic circuits in polynomial time. Examples of such translation can be found in Darwiche [2009b]; Vergari et al. [2020]; Liu et al. [2022].

At a high level, we can compile a Bayesian network to a PC by compiling its CPTs in a bottom-up fashion, i.e., children before parents. For instance, consider the tree structured Bayesian network in Figure 2.6a. To compile it into a smooth, decomposable, and deterministic PC, we first compile the conditional probability $p(A \mid C = 0)$ using a sum node with indicator leaf nodes as inputs, illustrated in Figure 2.6b. Note that the edge parameters correspond to $p(A = 0 \mid C = 0) = .3$ and $p(A = 1 \mid C = 0) = .7$. Similarly, we can compile the CPTs for both leaf nodes of the BN $p(A \mid C)$ and $p(B \mid C)$ as Figure 2.6c. Continuing to its parent, we connect the compiled



(a) Tree BN (b) PC for $p(A \mid C = 0)$ (c) PCs for $p(A \mid C)$ and $p(B \mid C)$

(d) PC for $p(A, B, C \mid D = 0)$ (e) PC for $p(A, B, C, D)$

Figure 2.6: A tree Bayesian network for $p(A, B, C, D)$ compiled into a PC

conditional probability distributions to the corresponding indicators for C, repeating the process until we reach the root node and obtain the PC in Figure 2.6e.

This has also been studied extensively in the field of *knowledge compilation* [Darwiche and Marquis, 2002], where Bayesian networks are compiled into circuit representations called weighted model counting (WMC) circuits to perform probabilistic inference. It proceeds by first encoding the BN into a weighted Boolean formula, which is then compiled into a logical circuit with some of its leaf nodes associated with parameters. Then a marginal inference task on the original BN can be performed by an analogous task called weighted model counting on the compiled circuit. WMC is tractable if the circuit is smooth and decomposable, where the properties are concerned with OR and AND gates in a logical circuit instead of sum and product nodes, respectively, in a probabilistic circuit. For such WMC circuit, there exists an equivalent PC such that the parameters associated with leaf nodes in the WMC circuit appear as edge parameters on sum nodes of the PC. We refer to Darwiche [2003] for more details.

CHAPTER 3

Fairness-aware Learning from Biased Labels

Ensuring fairness of machine learning models is often made challenging by the fact that labels in the data are biased. This chapter studies learning fair probability distributions from biased data by explicitly modeling a latent variable that represents a hidden, unbiased label. We also show that group fairness guarantees are meaningful only if the distribution used to provide those guarantees indeed captures the real-world data. In order to closely model the data distribution, we employ probabilistic circuits and propose an algorithm to learn them from incomplete data. We evaluate our approach on a synthetic dataset in which observed labels indeed come from fair labels but with added bias, and demonstrate that the fair labels are successfully retrieved. Moreover, we show on real-world datasets that our approach not only is a better model than existing methods of how the data was generated but also achieves competitive accuracy. The results described in this chapter were previously published in [Choi et al., 2021].

3.1 Introduction

As discussed previously, there is growing concern that machine learning algorithms may produce decisions that discriminate against particular groups of people. To address these concerns, various methods have been proposed to quantify and ensure fairness in automated decision making systems [Chouldechova, 2017; Dwork et al., 2012; Feldman et al., 2015; Kusner et al., 2017; Kamishima et al., 2012; Zemel et al., 2013]. A widely used notion of fairness is demographic parity, which states that sensitive attributes such as gender or race must be statistically independent of the class predictions.

This chapter studies the problem of enforcing demographic parity in probabilistic classifiers. In particular, focusing on the fact that class labels in the data are often biased, it proposes a latent variable approach that treats the observed labels as biased proxies of hidden, fair labels that satisfy demographic parity. The process that generated bias is modeled by a probability distribution over the fair label, observed label, and other features including the sensitive attributes. Moreover, this chapter shows that group fairness guarantees for a probabilistic model hold in the real world only if the model accurately captures the real-world data. Therefore, the goal of learning a fair probabilistic classifier also entails learning a distribution that achieves high likelihood.

The first contribution is to systematically derive the assumptions of a fair probabilistic model in terms of independence constraints. Each constraint serves the purpose of explaining how the observed, biased labels come from hidden fair labels and/or ensuring that the model closely represents the data distribution. The second contribution is an algorithm to learn probabilistic circuits so that the fairness constraints are satisfied. Specifically, this involves encoding independence assumptions into the circuits and developing an algorithm to learn PCs from incomplete data, as the model has a latent variable. Finally, the proposed approach is evaluated empirically on synthetic and real-world datasets, comparing against existing fair learning methods as well as a baseline that does not include a latent variable. The experiments demonstrate that our method achieves high likelihoods that indeed translate to more trustworthy fairness guarantees. It also has high accuracy for predicting the true fair labels in the synthetic data, and the predicted fair decisions can still be close to unfair labels in real-world data.

3.2 Latent Fair Decisions

Let S denote a *sensitive attribute*, such as gender or race, and let $\mathbf{X}$ be the *non-sensitive attributes* or features. Here we assume S to be binary for simplicity, but our method can be easily generalized to multiple multi-valued sensitive attributes. We have a dataset $\mathcal{D}$ in which each individual is characterized by variables S and $\mathbf{X}$ and labeled with a binary decision/class variable D.

One of the most popular and yet simple fairness notions is demographic (or statistical) parity. It requires that the classification is independent of the sensitive attributes; i.e., the rate of positive classification is the same across groups defined by the sensitive attributes. Since we focus on probabilistic classifiers, we consider a generalized version introduced by Pleiss et al. [2017], sometimes also called *strong demographic parity* [Jiang et al., 2019]:

Definition 3.1 (Generalized demographic parity)**.** *Suppose f is a probabilistic classifier and p is a distribution over variables $\mathbf{X}$ and S. Then f satisfies demographic parity w.r.t. p if:*

$$\mathbb{E}_p[f(\mathbf{X}, S) \mid S = 1) = \mathbb{E}_p[f(\mathbf{X}, S) \mid S = 0].$$

Probabilistic classifiers are often obtained from joint distributions $\Pr(.)$ over $D, \mathbf{X}, S$ by computing $\Pr(D|\mathbf{X}, S)$. Then we say the distribution satisfies demographic parity if $\Pr(D|S=1) = \Pr(D|S=0)$, i.e., D is independent of S.

3.2.1 Motivation

A common fairness concern when learning decision making systems is that the dataset used is often biased. In particular, observed labels may not be the true target variable but only its proxy. For example, re-arrest is generally used as a label for recidivism prediction, but it is not equivalent to re-offense and may be biased. We will later show how the relationship between observed label and true target can be modeled probabilistically using a latent variable.

Moreover, probabilistic group fairness guarantees hold in the real world only if the model accurately captures the real world distribution. In other words, using a model that only achieves low likelihood w.r.t the data, it is easy to give false guarantees. For instance, consider a probabilistic classifier $f(X, S)$ over a binary sensitive attribute S and non-sensitive attribute X shown below.

S, X	$f(X,S)$	$P_{\text{data}}(X\|S)$	$\mathbb{E}_{P_{\text{data}}}[f\|S]$	$Q(X\|S)$	$\mathbb{E}_Q[f\|S]$
1,1	0.8	0.7	0.65	0.5	0.55
1,0	0.3	0.3		0.5	
0,1	0.7	0.4	0.52	0.5	0.55
0,0	0.4	0.6		0.5	

Suppose in the data, the probability of $X = 1$ given $S = 1$ (resp. $S = 0$) is 0.7 (resp. 0.4). Then this classifier does not satisfy demographic parity, as the expected prediction for group $S = 1$ is $0.8 \cdot 0.7 + 0.3 \cdot 0.3 = 0.65$ while for group $S = 0$ it is 0.52. On the other hand, suppose you have a distribution Q that incorrectly assumes the feature X to be uniform and independent of S. Then you would conclude, incorrectly, that the prediction is indeed fair, with the average prediction for both protected groups being 0.55. Therefore, to provide meaningful fairness guarantees, we need to model the data distribution closely, i.e., with high likelihood.

3.2.2 Modeling with a latent fair decision

We now describe our proposed latent variable approach to address the aforementioned issues. We suppose there is a hidden variable that represents the true label without discrimination. This latent variable is denoted as D_f and used for prediction instead of D; i.e., decisions for future instances can be made by inferring the conditional probability $\Pr(D_f|\mathbf{e})$ given some feature observations $\mathbf{e}$ for $\mathbf{E} \subseteq \mathbf{X} \cup S$. We assume that the latent variable D_f is independent of S, thereby satisfying demographic parity. Moreover, the observed label D is modeled as being generated from the fair label by altering its values with different probabilities depending on the sensitive attribute. That is, the probability of D depends on D_f and S.

In addition, our model also assumes that the observed label D and non-sensitive features $\mathbf{X}$ are conditionally independent given the fair label and sensitive attributes, i.e., $D \perp\!\!\!\perp \mathbf{X}|D_f, S$. This is a crucial assumption to learn the model from data where D_f is hidden. To illustrate why, suppose

Figure 3.1: Bayesian network structures that represent the proposed fair latent variable approach (left) and model without a latent variable (right). Abusing notation, the set of features $\mathbf{X}$ is represented as a single node, but refers to some local Bayesian network over $\mathbf{X}$.

there is no such independence. Then the induced model allows variables $S, \mathbf{X}, D$ to depend on one another freely. Thus, such model can represent any marginal distribution over these variables, regardless of the parameters for D_f. We can quickly see this from the fact that for all $s, \mathbf{x}, d$,

$$\begin{aligned}\Pr(s\mathbf{x}d) &= \Pr(s)\Pr(\mathbf{x}d|s) \\ &= \Pr(s)\big(\Pr(\mathbf{x}d|s, D_f{=}1)\Pr(D_f{=}1) + \Pr(\mathbf{x}d|s, D_f{=}0)\Pr(D_f{=}0)\big).\end{aligned}$$

That is, multiple conditional distributions involving the latent fair decision D_f will result in the same marginal distribution over $S, \mathbf{X}, D$, and thus the real joint distribution is not identifiable when learning from data where D_f is completely hidden. For instance, the learner will not be incentivized to represent any dependence between D_f and other features, and may return a model in which the latent decision variable is completely independent of the observed variables. This is clearly undesirable because we want to use the latent variable to make decisions based on feature observations.

The independence assumptions of our proposed model are summarized as a Bayesian network structure in Figure 3.1a. Note that the set of features $\mathbf{X}$ is represented as a single node, as we do not make any independence assumptions among the features. In practice, we learn the statistical relationships between these variables from data. This is in contrast to the latent variable model in Calders and Verwer [2010] which had a naive Bayes assumption among the non-sensitive features. As we will later show empirically, such strong assumption not only affects the prediction quality

but also limits the fairness guarantee, as it will hold only if the naive Bayes assumption is indeed true in the data distribution.

Lastly, we emphasize that Bayesian network structures were used in this section only to illustrate the independence assumptions of our model. In practice, other probabilistic models can be used to represent the distribution as long as they satisfy our independence assumptions; we use probabilistic circuits as discussed in the next section.

3.3 Learning Fair Probabilistic Circuits

There are several challenges in modeling a fair probability distribution. First, as shown previously, fairness guarantees hold with respect to the modeled distribution, and thus we want to closely model the data distribution. A possible approach is to learn a deep generative model such as a generative adversarial networks (GANs) [Goodfellow et al., 2014]. However, then we must resort to approximate inference, or deal with models that have no explicit likelihood, and the fairness guarantees no longer hold. An alternative is to use models that allow exact inference such as Bayesian networks. Unfortunately, marginal inference, which is needed to make predictions $\Pr(D_f|\mathbf{e})$, is #P-hard for general BNs [Roth, 1996]. Tree-like BNs such as naive Bayes allow polytime inference, but they are not expressive enough to accurately capture the real world distribution. Hence, the second challenge is to also support tractable exact inference without sacrificing expressiveness. Lastly, the probabilistic modeling method we choose must be able to encode the independencies outlined in the previous section, to satisfy demographic parity and to learn a meaningful relationship between the latent fair decision and other variables. Probabilistic circuits (PCs) are tractable and expressive as shown in Chapter 2; we will now show how we encode the independence assumptions in PCs and learn fair PCs from data.

Encoding independence assumptions Consider the example PC in Figure 3.2: regardless of parameterization, this circuit structure always encodes a distribution where D is independent of

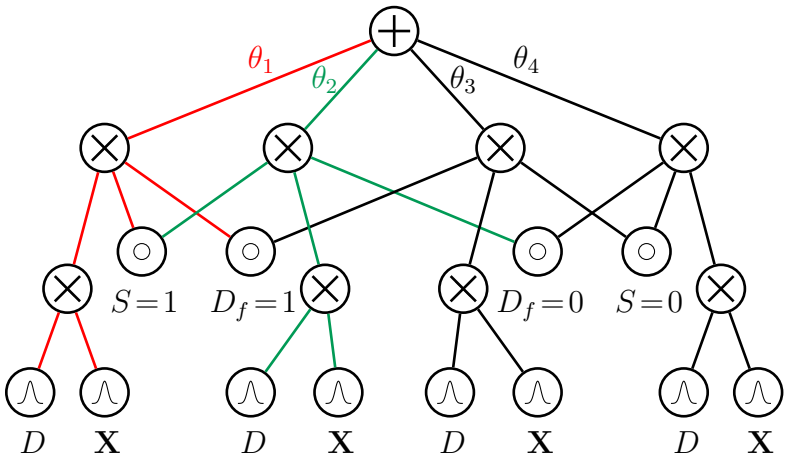


Figure 3.2: A probabilistic circuit over variables $S, \mathbf{X}, D, D_f$

$\mathbf{X}$ given S and D_f. Observe that the four product nodes in the second layer each correspond to four possible assignments to S and D_f. Effectively, the sub-circuits rooted at these nodes represent conditional distributions $\Pr(D, \mathbf{X}|s, d_f)$ for assignments s, d_f. Because the distributions for D and $\mathbf{X}$ factorize, we have $\Pr(D, \mathbf{X}|s, d_f) = \Pr(D|s, d_f) \cdot \Pr(\mathbf{X}|s, d_f)$, thereby satisfying the conditional independence $D \perp\!\!\!\perp \mathbf{X}|D_f, S$. We also need to encode the independence between D_f and S. In the example circuit, each edge parameter θ_i corresponds to $\Pr(s, d_f)$ for a joint assignment to S, D_f; e.g. $\theta_1 = \Pr(S=1, D_f=1)$. With no restriction on these parameters, the circuit does not necessarily imply $D_f \perp\!\!\!\perp S$. Thus, we introduce auxiliary parameters ϕ_s and ϕ_{d_f} representing $\Pr(S=1)$ and $\Pr(D_f=1)$, respectively, and enforce that the circuit parameters for $\Pr(S, D_f)$ factorize as follows:

$$\phi_s = \theta_1 + \theta_2, \quad \phi_{d_f} = \theta_1 + \theta_2,$$
$$\theta_1 = \phi_s \cdot \phi_{d_f},\ \theta_2 = \phi_s \cdot (1 - \phi_{d_f}),\ \theta_3 = (1 - \phi_s) \cdot \phi_{d_f},\ \theta_4 = (1 - \phi_s) \cdot (1 - \phi_{d_f}).$$

Hence, when learning these parameters, we limit the degree of freedom such that the four edge parameters are given by two free variables ϕ_s and ϕ_{d_f} instead of the four θ_i variables.

Next, we discuss how to learn a fair probabilistic circuit with latent variable from data. This consists of two parts: learning the circuit structure and estimating the parameters of a given structure.

3.3.1 Parameter Learning

Given a complete data set, maximum-likelihood parameters of a smooth, decomposable, and deterministic PC can be computed in closed-form [Kisa et al., 2014a]. However, our proposed approach for fair distribution includes a latent variable, and thus must be learned from incomplete data. One of the most common methods to learn parameters of a probabilistic model from incomplete data is the Expectation Maximization (EM) algorithm [Koller and Friedman, 2009; Darwiche, 2009a]. EM iteratively completes the data by computing the probability of unobserved values (E-step) and estimates the maximum-likelihood parameters from the expected dataset (M-step). We propose an EM parameter learning algorithm for PCs that does not explicitly complete the data, but rather utilizes a notion called *circuit flows*.

3.3.1.1 EM using expected flows

Definition 3.2 (Context)**.** *Let $\mathcal{C}$ be a PC over RVs $\mathbf{Z}$ and n be one of its nodes. The* context γ_n *of node n denotes all joint assignments that return a nonzero value for all nodes in a path between the root of $\mathcal{C}$ and n.*

$$\gamma_n := \bigcup_{p \in \mathsf{pa}(n)} \gamma_p \cap \mathsf{supp}(n),$$

where $\mathsf{pa}(n)$ refers to the parent nodes of n, and $\mathsf{supp}(n) := \{\mathbf{z} : \mathcal{C}_n(\mathbf{z}) > 0\}$ the support of node n.

Note that the context of a node is different from its support. Even if the node returns a non-zero value for some input, its output may be multiplied by 0 at its ancestor nodes; i.e., such node does not contribute to the circuit output of that assignment.

Then the circuit flow is defined as the following.

Definition 3.3 (Circuit flow)**.** *Let $\mathcal{C}$ be a PC over variables $\mathbf{Z}$, (n, c) its edge, and $\mathbf{z}$ a joint assignment to $\mathbf{Z}$. The* circuit flow *of (n, c) given $\mathbf{z}$ is*

$$F_{\mathbf{z}}(n, c) = [\mathbf{z} \in \gamma_n \cap \gamma_c]. \tag{3.1}$$

Intuitively, the context of a circuit node is the set of all complete inputs that "activate" the node. Hence, an edge is "activated" by an input if it is in the contexts of both nodes for that edge. Then the flow given a dataset is simply the sum of flows given each data point. That is, given a complete data $\mathcal{D}$, the circuit flow of (n, c) is

$$F_{\mathcal{D}}(n, c) = \sum_{\mathcal{D}_i \in \mathcal{D}} F_{\mathcal{D}_i}(n, c),$$

where $\mathcal{D}_i$ is the i-th data point of $\mathcal{D}$, which must be a complete assignment $\mathbf{z}$. For example, in Figure 3.2, the edges activated by sample $\{D_f = 1, S = 1, d, \mathbf{x}\}$, for any assignments $d, \mathbf{x}$, are colored red.

For an edge between a sum node n and its child c, the associated maximum-likelihood parameter for a complete dataset $\mathcal{D}$ is given by:

$$\theta_{n,c} = F_{\mathcal{D}}(n, c) / \sum_{c \in \mathsf{ch}(n)} F_{\mathcal{D}}(n, c). \tag{3.2}$$

We now introduce the notion of *expected flow*, which is the core element of our proposed expectation-maximization approach.

Definition 3.4 (Expected flow)**.** *Let $\mathcal{C}$ be a PC over variables $\mathbf{Z}$, (n, c) its edge, and $\mathbf{e}$ a partial assignment to $\mathbf{E} \subseteq \mathbf{Z}$. The* expected flow *of (n, c) given $\mathbf{e}$ is given by*

$$\mathrm{EF}_{\mathbf{e},\theta}(n, c) := \mathbb{E}_{\mathbf{z} \sim \Pr_{\mathcal{C}}(\cdot|\mathbf{e})}[F_{\mathbf{z}}(n, c)] = \sum_{\mathbf{z} \models \mathbf{e}} \Pr_{\mathcal{C}}(\mathbf{z}|\mathbf{e}) \cdot F_{\mathbf{z}}(n, c), \tag{3.3}$$

where $\mathbf{z} \models \mathbf{e}$ are the possible completions of partial assignment $\mathbf{e}$.

Again, given an incomplete data $\mathcal{D}$, the expected flow of (n, c) w.r.t. parameters θ is

$$\mathrm{EF}_{\mathcal{D},\theta}(n, c) = \sum_{\mathcal{D}_i \in \mathcal{D}} \mathrm{EF}_{\mathcal{D}_i,\theta}(n, c),$$

where $\mathcal{D}_i$ may be a partial assignment $\mathbf{e}$ for some $\mathbf{E} \subseteq \mathbf{Z}$. For example, in Figure 3.2, the

expected flows of the edges highlighted in red and green, given a sample $\{S = 1, d, \mathbf{x}\}$, are $\Pr_{\mathcal{C}}(D_f = 1 \mid S = 1, d, \mathbf{x})$ and $\Pr_{\mathcal{C}}(D_f = 0 \mid S = 1, d, \mathbf{x})$, respectively. Similar to circuit flows, the expected flows for all edges can be computed with a single bottom-up and top-down evaluation of the circuit, as illustrated in the next section. Then, we can perform both the E- and M-step by the following closed-form solution.

Proposition 3.1. *Given a smooth, decomposable, and deterministic circuit with parameters $\boldsymbol{\theta}$ and an incomplete data $\mathcal{D}$, the parameters for the next EM iteration are given by:*

$$\theta_{n,c}^{(new)} = \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c) / \sum_{c \in \mathsf{ch}(n)} \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c).$$

Note that this is very similar to the ML estimate from complete data in Eq.3.2, except using expected flows instead of circuit flows. Moreover, the expected flow can be computed even if each data sample has different variables missing; thus, the EM method can naturally handle missing values for other features as well.

Proof of Proposition 3.1. Completing a dataset $\mathcal{D}$ with missing values, given a distribution $\Pr_{\boldsymbol{\theta}}(.)$, amounts to constructing an auxiliary dataset $\mathcal{D}'$ as follows: for each data sample $\mathcal{D}_i \in \mathcal{D}$, there are samples $\mathcal{D}'_{i,k} \in \mathcal{D}'$ for $k = 1, \ldots, m_i$ with weights $\alpha_{i,k}$ such that each $\mathcal{D}'_{i,k}$ is a full assignment that agrees with $\mathcal{D}_i$. Moreover, the weights are defined by the given distribution as: $\alpha_{i,k} = \Pr_{\boldsymbol{\theta}}(\mathcal{D}'_{i,k} | \mathcal{D}_i)$. Then the max-likelihood parameters of a circuit given this completed dataset $\mathcal{D}'$ can be computed as: $\theta_{n,c} = F_{\mathcal{D}'}(n, c) / \sum_{c \in \mathsf{ch}(n)} F_{\mathcal{D}'}(n, c)$. Note that since $\mathcal{D}'$ is an expected/weighted dataset, the flows $F_{\mathcal{D}'}$ are real numbers as opposed to integers, which is the case when every sample has weight 1. Specifically,

$$\begin{aligned} F_{\mathcal{D}'}(n, c) &= \sum_{\mathcal{D}'_{i,k} \in \mathcal{D}'} \alpha_{i,k} F_{\mathcal{D}'_{i,k}}(n, c) = \sum_{\mathcal{D}_i \in \mathcal{D}} \sum_{k=1}^{m_i} \Pr_{\boldsymbol{\theta}}(\mathcal{D}'_{i,k} | \mathcal{D}_i) F_{\mathcal{D}'_{i,k}}(n, c) \\ &= \sum_{\mathcal{D}_i \in \mathcal{D}} \sum_{\mathbf{z} \models \mathcal{D}_i} \Pr_{\boldsymbol{\theta}}(\mathbf{z} | \mathcal{D}_i) F_{\mathbf{z}}(n, c) = \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c). \end{aligned}$$

□

3.3.1.2 Computing Expected Flow

We now describe how to compute expected flows, in particular for smooth, decomposable, and deterministic PCs. First, focusing on the expected flow given a single partial assignment $\mathbf{e}$, we can express the expected flow as the following using Equations 3.1 and 3.3.

$$\mathrm{EF}_{\mathbf{e},\theta}(n,c) = \mathbb{E}_{\mathbf{z}\sim\mathrm{Pr}_{\mathcal{C}}(\cdot|\mathbf{e})}[\mathbf{z} \in \gamma_n \cap \gamma_c] = \mathrm{Pr}_{\mathcal{C}}(\gamma_n \cap \gamma_c|\mathbf{e}) \tag{3.4}$$

Furthermore, with determinism, the sub-circuit formed by "activated" edges for any complete input forms a tree [Choi and Darwiche, 2017]. Thus, for a complete evidence $\mathbf{z}$, a node n has exactly one parent p such that $F_{\mathbf{z}}(p,n) = 1$, or equivalently, $\mathbf{z} \in \gamma_p \cap \gamma_n$. Thus,

$$\sum_{p\in\mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p,n) = \sum_{p\in\mathsf{pa}(n)} \mathrm{Pr}_{\mathcal{C}}(\gamma_p \cap \gamma_n|\mathbf{e}) = \mathrm{Pr}_{\mathcal{C}}(\gamma_n|\mathbf{e}). \tag{3.5}$$

We can observe from Equations 3.4 and 3.5 that if $\sum_{p\in\mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p,n) = 0$, then we also have $\mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(n,c) = 0$.

For an edge (n,c) where n is a sum node,

$$\begin{aligned}\mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(n,c) = \mathrm{Pr}_{\mathcal{C}}(\gamma_n \cap \gamma_c|\mathbf{e}) &= \frac{\mathrm{Pr}_{\mathcal{C}}(\mathbf{e},\gamma_c|\gamma_n)\,\mathrm{Pr}_{\mathcal{C}}(\gamma_n)}{\mathrm{Pr}_{\mathcal{C}}(\mathbf{e})} = \frac{\mathrm{Pr}_{\mathcal{C}}(\gamma_n|\mathbf{e})\,\mathrm{Pr}_{\mathcal{C}}(\mathbf{e},\gamma_c|\gamma_n)\,\mathrm{Pr}_{\mathcal{C}}(\gamma_n)}{\mathrm{Pr}_{\mathcal{C}}(\gamma_n|\mathbf{e})\,\mathrm{Pr}_{\mathcal{C}}(\mathbf{e})} \\ &= \frac{\mathrm{Pr}_{\mathcal{C}}(\gamma_n|\mathbf{e})\,\mathrm{Pr}_{\mathcal{C}}(\mathbf{e},\gamma_c|\gamma_n)}{\mathrm{Pr}_{\mathcal{C}}(\mathbf{e}|\gamma_n)} = \left(\sum_{p\in\mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p,n)\right)\frac{\theta_{n,c}\,\mathrm{Pr}_c(\mathbf{e})}{\mathrm{Pr}_n(\mathbf{e})}.\end{aligned} \tag{3.6}$$

Here, Pr_n and Pr_c refer to the distribution defined by the sub-circuits rooted at nodes n and c, respectively. Because $\mathbf{e}$ can be partial observations, these probability corresponds to marginal queries. For a smooth and decomposable probabilistic circuit, the marginals given a partial input for all circuit nodes can be computed by a single bottom-up evaluation of the circuit [Darwiche

and Marquis, 2002]. This amounts to marginalizing the leaf nodes according to the partial input (i.e., plugging in 1 for unobserved variables) and evaluating the circuit according to its recursive definition.

For an edge (n, c) where n is a product node, we have $\gamma_n \subseteq \gamma_c$ as follows:

$$\gamma_n = \gamma_n \cap \mathsf{supp}(n) \subseteq \gamma_n \cap \mathsf{supp}(c) \subseteq \bigcup_{p \in \mathsf{pa}(c)} \gamma_p \cap \mathsf{supp}(c) = \gamma_c \tag{3.7}$$

where Equation 3.7 follows from Definition 3.2 and the fact that any assignment that leads to a non-zero output for n must also output non-zero for c (i.e. $\mathsf{supp}(n) \subseteq \mathsf{supp}(c)$). Then we can write the expected flow of (n, c) as the following:

$$\mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(n, c) = \Pr_{\mathcal{C}}(\gamma_n \cap \gamma_c | \mathbf{e}) = \Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}) = \sum_{p \in \mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p, n). \tag{3.8}$$

Therefore, Equations 3.6 and 3.8 describe how expected flow on edge (n, c) can be computed using the expected flows from parents of n and the marginal probabilities at nodes n and c. We can thus compute the the expected flow via a bottom-up evaluation (to compute the marginals) followed by a top-down pass as shown in Algorithm 1. We cache intermediate results to avoid redundant computations and to ensure a linear-time evaluation.

To compute the expected flow on a dataset, we can compute the expected flow of each data sample in parallel via vectorization, and then simply sum the results per edge.

3.3.1.3 Initial parameters using prior knowledge

Typically the EM algorithm is run starting from randomly initialized parameters. While the algorithm is guaranteed to improve the likelihood at each iteration until convergence, it still has the problem of multiple local maxima and identifiability, especially when there is a latent variable involved [Koller and Friedman, 2009]. Namely, we can converge to different learned models with similar likelihoods but different parameters for the latent fair variable, thus resulting in different

Algorithm 1 Computing the expected flow

Input: PSDD $\mathcal{C}$, one data sample d, marginal likelihood $\Pr_{\mathcal{C}}$ cached from bottom-up pass
Output: Expected flow of sample d for each node and edge, cached in EF

for n in PSDD $\mathcal{C}$ **do** ▷ parents before children
  if n is root **then**
    $\mathrm{EF}(n) \leftarrow 1$
  else
    $\mathrm{EF}(n) \leftarrow \sum_{p \in \mathsf{pa}(n)} \mathrm{EF}(p, n)$
  if n is a sum node **then**
    for c in $\mathsf{ch}(n)$ **do**
      $\mathrm{EF}(n, c) \leftarrow \mathrm{EF}(n) \cdot \frac{\theta_{n,c} \cdot \Pr_c(d)}{\Pr_n(d)}$
  else if n is a product node **then**
    for c in $\mathsf{ch}(n)$ **do**
      $\mathrm{EF}(n, c) \leftarrow \mathrm{EF}(n)$

behaviors in the prediction task. For example, for a given fair distribution, we can flip the value of D_f and the parameters accordingly such that the marginal distribution over $S, \mathbf{X}, D$, as well as the likelihood on the dataset, is unchanged. However, this clearly has a significant impact on the predictions which will be completely opposite.

Therefore, instead of random initialization, we encode prior knowledge in the initial parameters that determine $\Pr(D|S, D_f)$. In particular, it is obvious that D_f should be equal to D if the observed labels are already fair. Furthermore, for individual predictions, we would want D_f to be close to D as much as possible while ensuring fairness. Thus, we start the EM algorithm from a conditional probability $\Pr(d|s, d_f) = [d = d_f]$.

3.3.2 Structure Learning

Lastly, we describe how a fair probabilistic circuit structure is learned from data. As described previously, top layers of the circuit are fixed in order to encode the independence assumptions of our latent variable approach. On the other hand, the sub-circuits over features $\mathbf{X}$ can be learned

to best fit the data. We adopt the STRUDEL algorithm to learn the structures [Dang et al., 2020].[1] Starting from a Chow-Liu tree initial distribution [Chow and Liu, 1968], STRUDEL performs a heuristic-based greedy search over possible candidate structures. At each iteration, it first selects the edge with the highest circuit flow and the variable with the strongest dependencies on other variables, estimated by the sum of pairwise mutual informations. Then it applies the *split* operation – a simple structural transformation that "splits" the selected edge by introducing new sub-circuits conditioned on the selected variable. Intuitively, this operation aims to model the data more closely by capturing the dependence among variables (variable heuristic) appearing in many data samples (edge heuristic). After learning the structure, we update the parameters of the learned circuit using EM as described previously.

3.4 Empirical Evaluation

We now empirically evaluate our proposed model FAIRPC on real-world benchmark datasets as well as synthetic data.

Baselines We first compare FAIRPC to three other probabilistic methods: fair naive Bayes models (2NB and LATNB) [Calders and Verwer, 2010] and PCs without latent variable (NLATPC). We also compare against methods that learn discriminative classifiers satisfying group fairness: (1) FAIRLR [Zafar et al., 2017], which learns a classifier subject to co-variance constraints; (2) REDUCTION [Agarwal et al., 2018], which reduces the fair learning problem to cost-sensitive classification problems and learns a randomized classifier; and (3) REWEIGHT [Jiang and Nachum, 2020] which corrects bias by re-weighting the data points. All three methods learn logistic regression classifiers, either with constraints or modified objective functions.

[1] It learns PCs that also satisfy properties such as structured decomposability that are not necessary in our case.

Evaluation criteria For predictive performance, we use accuracy and F1 score. Note that models with latent variables use the latent fair decision D_f to make predictions, while other models directly use D. Moreover, in the real-world datasets, we do not have access to the fair labels and instead evaluate using the observed labels which may be "noisy" and biased. We emphasize that the accuracy w.r.t unfair labels is not the goal of our method, as we want to predict the true target, not its biased proxy. Rather, it measures how similar the latent variable is to the observed labels, thereby justifying its use as fair decision. To address this, we also evaluate on synthetic data where fair labels can be generated. For fairness performance, we define the discrimination score as the difference in average prediction probability between the majority and minority groups, i.e., $\Pr(D_f=1|S=0) - \Pr(D_f=1|S=1)$ estimated on the test set.

3.4.1 Real-World Data

Data We use three datasets: COMPAS [Propublica, 2016], Adult, and German [Dua and Graff, 2017], which are commonly studied benchmarks for fair ML. They contain both numerical and categorical features and are used for predicting recidivism, income level, and credit risk, respectively. We wish to make predictions that are fair with respect to a protected attribute: "sex" for Adult and German, and "ethnicity" for COMPAS. As pre-processing, we discretize numerical features, remove unique or duplicate features (e.g. names of individuals), and remove low frequency counts.

Probabilistic methods We first compare against probabilistic methods to illustrate the effects of using latent variables and learning more expressive distributions. Figure 3.3 summarizes the result. In terms of log-likelihoods, both PC-based methods outperform NB models, which aligns with our motivation for relaxing the naive Bayes assumption—to better fit the data distribution. Furthermore, models with latent variables outperform their corresponding non-latent models, i.e., LATNB outperforms 2NB and FAIRPC outperforms NLATPC. This validates our earlier argument that the latent variable approach can achieve higher likelihood than enforcing fairness directly in the observed label. Next, we compare the methods using F1-score as there is class imbalance in

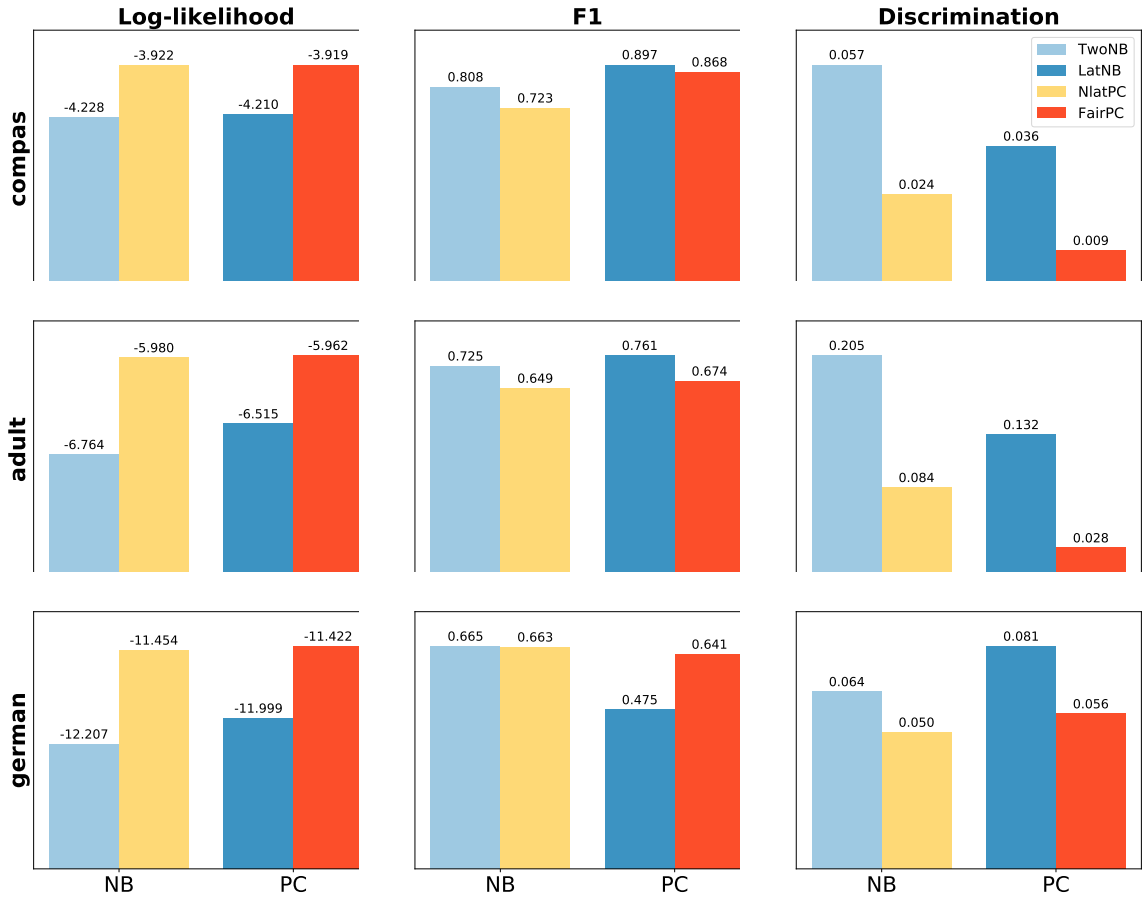


Figure 3.3: Comparison of fair probability distributions. **Columns:** log-likelihood, F1-score, discrimination score (higher is better for the first two; lower is better for last). **Rows:** COMPAS, Adult, German datasets. The four bars in each graph from left to right are: 1) 2NB, 2) LATNB, 3) NLATPC, 4) FAIRPC.

these datasets. Although it is measured with respect to possibly biased labels, FAIRPC achieves competitive performance, demonstrating that the latent fair decision variable still exhibits high similarity with the observed labels. Lastly, FAIRPC achieves the lowest discrimination scores in COMPAS and Adult datasets by a significant margin. As expected, PCs also achieve lower discrimination scores than their counterpart NB models, as they fit the data distribution better.

Discriminative classifiers Next we compare FAIRPC to existing fair classification methods. Figure 3.4 shows the trade-off between predictive performance and fairness. We add two other baselines to the plot: RAND, which makes random predictions, and LR, which is an unconstrained

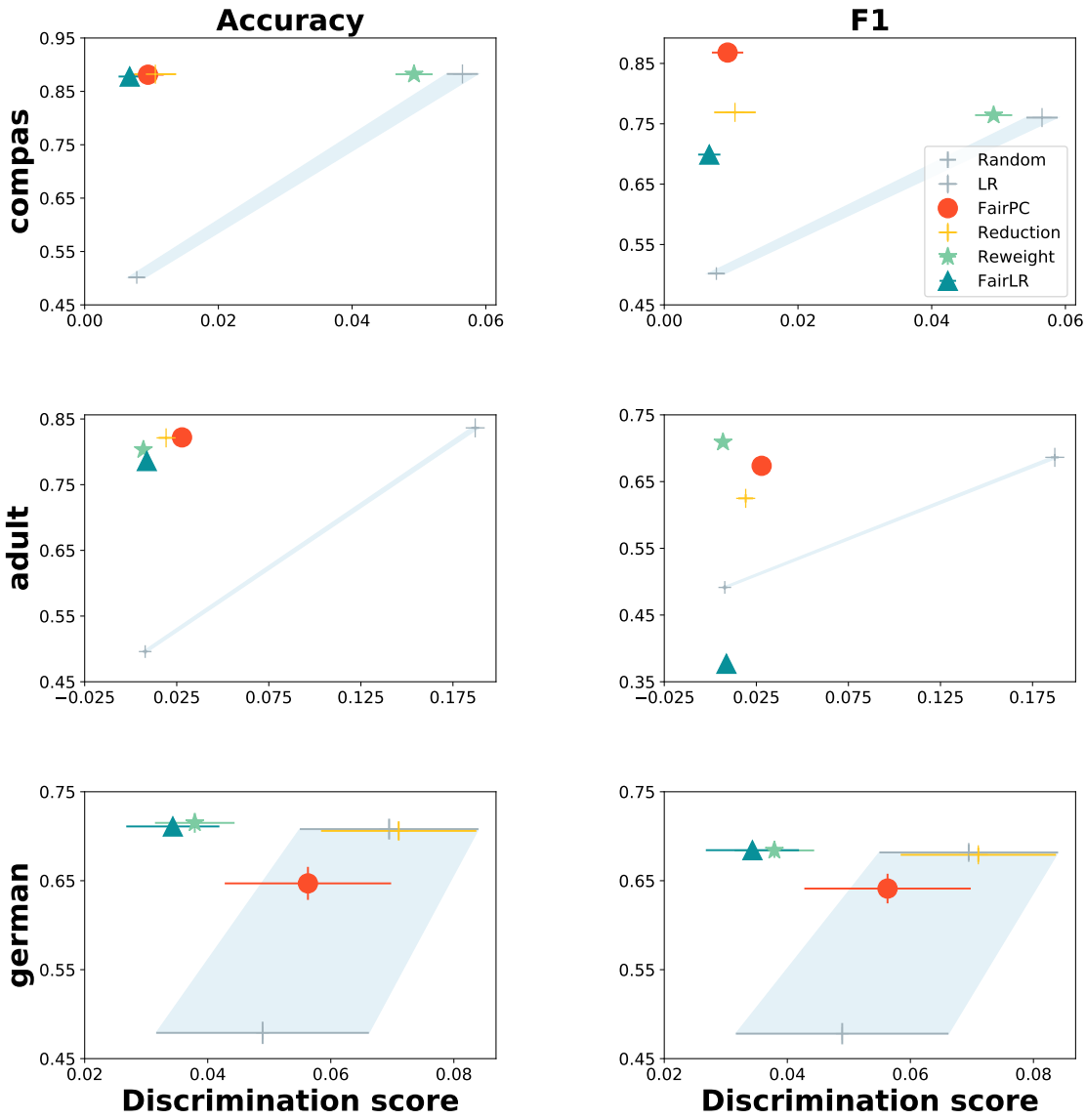


Figure 3.4: Predictive performance (y-axis) vs. discrimination score (x-axis) for FAIRPC and fair classification methods (FAIRLR, REDUCTION, REWEIGHT), in addition with two trivial baselines (RAND and LR). **Columns:** accuracy, F1-score. **Rows:** COMPAS, Adult, German datasets.

logistic regression classifier. They represent the two ends of the fairness-accuracy tradeoff. RAND has no predictive power but low discrimination, while LR has high accuracy but unfair. Informally, the further above the line between these baselines, the better the method optimizes this tradeoff.

On COMPAS and Adult datasets, our approach achieves a good balance between predictive performance and fairness guarantees. In fact, it achieves the best or close to best accuracy and F1-score, again showing that the latent decision variable is highly similar to the observed labels even though the explicit objective is not to predict the unfair labels. However, on German dataset, while FAIRLR and REWEIGHT achieve the best performance on average, the estimates for all models including the trivial baselines are too noisy to draw a statistically significant conclusion. This may be explained by the fact that the dataset is relatively small with 1000 samples.

3.4.2 Synthetic Data

As discussed previously, ideally we want to evaluate against the true target labels, but they are generally unknown in real-world data. Therefore, we also evaluate on synthetic data with fair ground-truth labels in order to evaluate whether our model indeed captures the hidden process of bias and makes accurate predictions.

Generating Data We generate data by constructing a fair PC $\mathcal{C}_{true}$ to represent the "true distribution" and sampling from it. The process that generates biased labels d is represented by the following (conditional) probability table:

$\cdot$	D_f	S	d_f, s	1,1	1,0	0,1	0,0
$\Pr(\cdot = 1)$	0.5	0.3	$\Pr(D=1 \mid D_f=d_f, S=s)$	0.8	0.9	0.1	0.4

Here, $S = 1$ is the minority group, and the unfair label D is in favor of the majority group: D is more likely to be positive for the majority group $S=0$ than for $S=1$, for both values of fair label D_f but at different rates. To evaluate on a wide range of datasets, we randomly generate the sub-circuits of $\mathcal{C}_{true}$ over features $\mathbf{X}$ as tree distributions, randomly initializing the parameters with

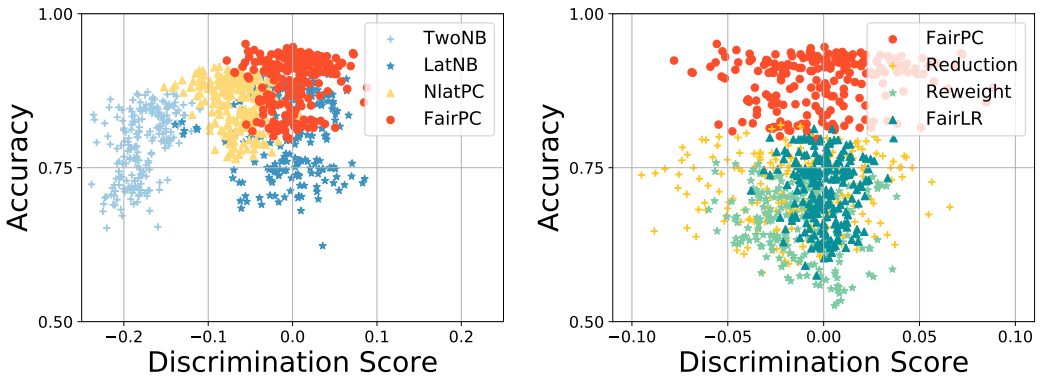


Figure 3.5: Accuracy (y-axis) vs. discrimination score (x-axis) on synthetic datasets. We compare FAIRPC with 2NB, LATNB, NLATPC (left) and with REDUCTION, REWEIGHT, FAIRLR (right). Each dot is a single run on a generated dataset using the method indicated by its color.

Laplace smoothing. We generated different synthetic datasets with the number of non-sensitive features ranging from 10 to 30, using 10-fold CV for each.

Results We first test FAIRPC, LATNB, NLATPC and NLATPC on the generated datasets. Figure 3.5 (left) illustrates the accuracy and discrimination scores on separate test sets with fair decision labels. In terms of accuracy, PCs outperform NBs, and latent variable approaches outperform non-latent ones, which shows that adopting density estimation to fit the data and introducing a latent variable indeed help improve the performance. When comparing the average discrimination score for each method, 2NB and NLATPC always have negative scores, showing that the non-latent methods are more biased towards the majority group; while LATNB and FAIRPC are more equally distributed around zero on the x-axis, thus demonstrating that a latent fair decision variable helps to correct this bias. While both latent variable approaches achieve reasonably low discrimination on average, FAIRPC is more stable and has even lower average discrimination score than LATNB.

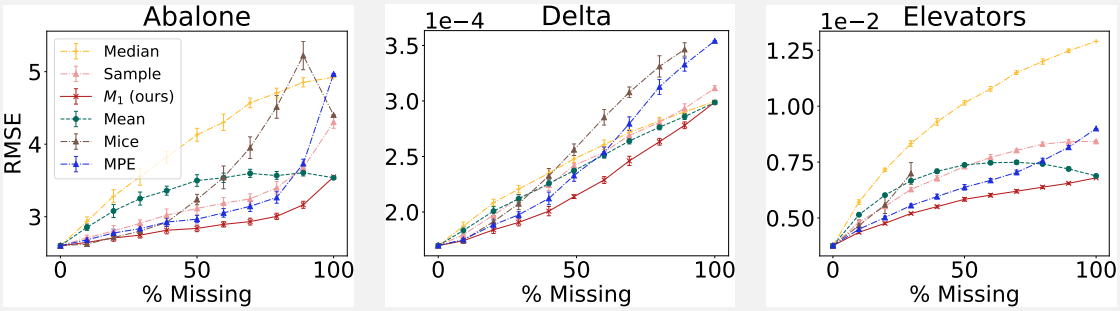
We also compare FAIRPC to FAIRLR, REDUCTION, and REWEIGHT, the results visualized in Figure 3.5 (right). Our method achieves a much higher accuracy w.r.t. the generated fair labels; for instance, the average accuracy of FAIRPC is around 0.17 higher than that of FAIRLR. Also, we are still being comparable in terms of discrimination score, illustrating the benefits of explicitly modeling the latent fair decision.

Intermezzo 1: Expected Predictions

Computing the *expected predictions* [Khosravi et al., 2019b] is a task that lets us reason about the outputs of a discriminative model with respect to a probability distribution. Formally, let p be a probability distribution over $\mathbf{X}$ and $f : \mathcal{X} \to \mathbb{R}$ be a discriminative model, e.g., a regressor, that assigns a real value (outcome) to each complete input configuration $\boldsymbol{x} \in \mathcal{X}$ (features). The task of computing the expected prediction of f with respect to the distribution p is defined as:

$$\mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})} \left[f(\boldsymbol{x}) \right].$$

This task appears in many interesting ML applications including missing value predictions, algorithmic fairness, and data analysis. For example, we can make predictions with missing features by computing the expected prediction with respect to the distribution conditioned on the partial observations, which outperformed various imputation techniques:



Moreover, suppose f is a regression model predicting the cost of healthcare for insurance purposes. We could ask: *"is the predictive model biased by gender?"* To answer this question, it would be interesting to compute:

$$\mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|Female)}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|Male)}[f(\boldsymbol{x})] = 14,170 - 13,196 = 974.$$

If the answer to this query is 0, the model exactly satisfies demographic parity with respect to p. Furthermore, we can also audit the model in terms of other group fairness notions: for instance by taking the expected predictions after conditioning the distributions also on the class variable, or the latent fair decision variable (Section 3.2). Thus, tractable computation of expected predictions would allow us to efficiently measure various fairness notions not just for generative models encoding distributions, but also for discriminative models.

Unfortunately, computing expectations of a discriminative model with respect to a probability distribution defined by an arbitrary generative model has been proven to be hard in general. In fact, the task is intractable even for simple models such as logistic regression and a naive Bayes distribution [Khosravi et al., 2019b]. Nevertheless, we identify pairs of generative and discriminative models that enable tractable computation of expectations—as well as moments of any order—of the latter with respect to the former. Specifically, this is the case for probabilistic circuits and compatible regression circuits [Khosravi et al., 2019a] as well as for decision trees [Khosravi et al., 2020].

3.4.3 Learning With Missing Values

As described in Section 3.3.1, if the training data has some missing values (in addition to the latent decision variable), FAIRPC parameter learning method still applies without change of the algorithm. Table 3.6 shows the test log-likelihoods given missing values at training time, with missing percentage ranging from 0% to 99%. We adopt missing completely at random (MCAR) missingness mechanism and fix the circuit structure to the ones learned previously. We only compare the density estimation performance here, comparing prediction performance as well as their fairness implications under different missingness is left as future work.

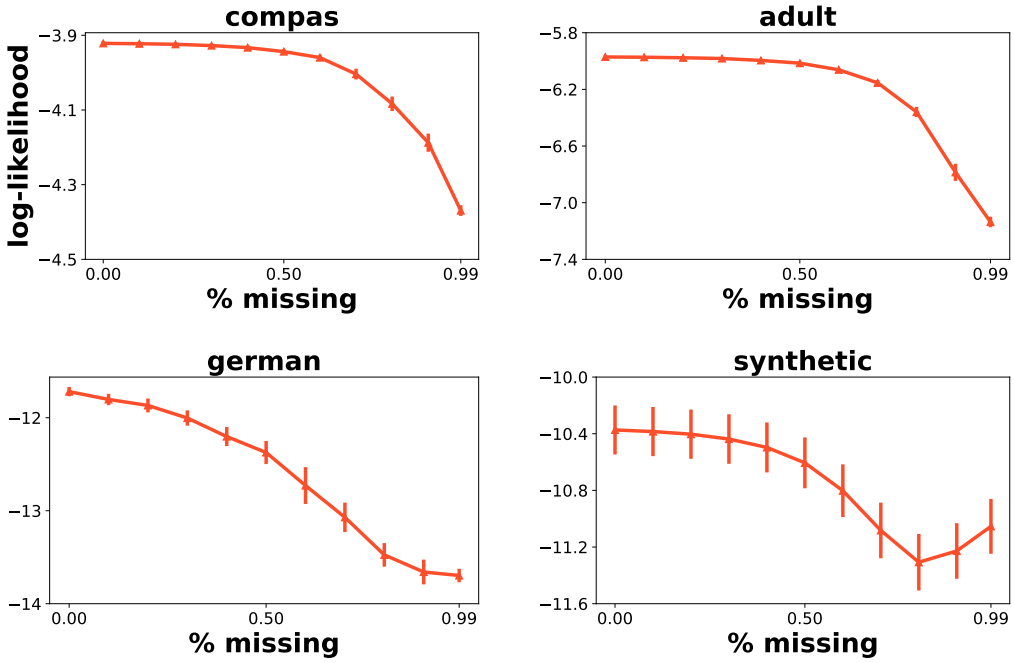


Figure 3.6: Test log-likelihood under different missingness percentages on real world and synthetic datasets.

3.5 Related Work

Several frameworks have been proposed to design fairness-aware systems. We discuss a few of them here and refer to Romei and Ruggieri [2014]; Barocas et al. [2019] for a more comprehensive

review.

Some of the most prominent fairness frameworks include individual fairness and group fairness. Individual fairness [Dwork et al., 2012] is based on the idea that similar individuals should receive similar treatments, although defining similarity between individuals can be challenging. On the other hand, group fairness aims to equalize some statistics across groups defined by sensitive attributes. These include equality of opportunity [Hardt et al., 2016] and demographic (statistical) parity [Calders and Verwer, 2010; Kamiran and Calders, 2009] as well as its relaxed notion of disparate impact [Feldman et al., 2015; Zafar et al., 2017].

There are several approaches to achieve group fairness, which can be broadly categorized into (1) pre-processing data to remove bias [Zemel et al., 2013; Kamiran and Calders, 2009; Calmon et al., 2017], (2) post-processing of model outputs such as calibration and threshold selection [Hardt et al., 2016; Pleiss et al., 2017], and (3) in-processing which incorporates fairness constraints directly in learning or optimization [Corbett-Davies et al., 2017; Agarwal et al., 2018; Kearns et al., 2018]. Some recent works on group fairness also consider bias in the observed labels, both for evaluation and learning [Fogliato et al., 2020; Blum and Stangl, 2020; Jiang and Nachum, 2020]. For instance, Blum and Stangl [2020] study empirical risk minimization (ERM) with various group fairness constraints and showed that ERM constrained by demographic parity does not recover the Bayes optimal classifier under one-sided, single-group label noise (this setting is subsumed by ours). In addition, Jiang and Nachum [2020] developed a pre-processing method to learn fair classifiers under noisy labels, by reweighting according to an unknown, fair labeling function. Here, the observed labels are assumed to come from a biased labeling function that is the "closest" to the fair one; on the other hand, we aim to find the bias mechanism that best explains the observed data.

We would like to point out that while pre-processing methods have the advantage of allowing any model to be learned on top of the processed data, it is also known that certain modeling assumptions can result in bias even when learning from fair data [Choi et al., 2020a]. Moreover, certain post-processing methods to achieve group fairness are shown to be suboptimal under some conditions [Woodworth et al., 2017]. Instead, we take the in-processing approach to explicitly

optimize the model's performance while enforcing fairness.

Many fair learning methods make use of probabilistic models such as Bayesian networks [Calders and Verwer, 2010; Mancuhan and Clifton, 2014]. Among those, perhaps the most related to our approach is the latent variable naive Bayes model by Calders and Verwer [2010], which also assumes a latent decision variable to make fair predictions. However, they make a naive Bayes assumption among features. We relax this assumption and will later demonstrate how this helps in more closely modeling the data distribution, as well as providing better fairness guarantees.

3.6 Discussion

This chapter proposed a latent variable approach to learning fair distributions that satisfy demographic parity, and developed an algorithm to learn fair probabilistic circuits from incomplete data. Experimental evaluation on simulated data shows that the proposed method consistently achieves high log-likelihoods and low discrimination. It also accurately predicts true fair decisions, and even on real-world data where fair labels are not available, our predictions remain close to the unfair ones.

While this work focused on demographic parity as the main fairness definition, in the future I hope to expand its applicability. In particular, a learned distribution over the latent fair decision can be used in a number of ways to audit and enforce other group fairness notions. For example, we can generate fair labels given a dataset with biased labels, by simply inferring the latent fair decision for each data point via conditional inference on the fair probabilistic circuit. The generated fair dataset can then be used by several downstream tasks. In addition, we can also measure other fairness notions such as equalized odds with respect to the hidden fair labels, not the biased ones.

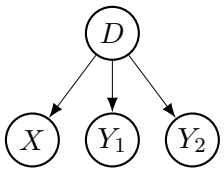
CHAPTER 4

Fairness of Predictions with Missing Features

This chapter studies fairness properties of probabilistic classifiers that represent joint distributions over the features and decision variable. In particular, Bayesian classifiers treat the classification or decision-making task as a probabilistic inference problem: given observed features, compute the probability of the decision variable. Such models have a unique ability that they can naturally handle missing features, by simply marginalizing them out of the distribution when they are not observed at prediction time. Hence, they effectively embed exponentially many classifiers, one for each subset of observable features. We ask whether such classifiers exhibit patterns of discrimination where similar individuals receive markedly different outcomes purely because they disclosed one or more sensitive attributes.

The first key contribution of this chapter is an algorithm to verify whether a Bayesian classifier is fair, or else to mine the classifier for discrimination patterns. We propose two alternative criteria for identifying the most important discrimination patterns that are present in the classifier. We specialize our pattern miner to efficiently discover discrimination patterns in naive Bayes models using branch-and-bound search. These classifiers are often used in practice because of their simplicity and tractability, and they allow for the development of effective bounds. Our empirical evaluation shows that naive Bayes models indeed exhibit vast numbers of discrimination patterns, and that our pattern mining algorithm is able to find them by traversing only a small fraction of the search space.

The second key contribution is a parameter learning algorithm for naive Bayes classifiers that ensures that no discrimination patterns exist in the the learned distribution. We propose a signomial programming approach to eliminate individual patterns of discrimination during maximum-



$P(d)$
0.2

D	$P(x\|D)$
$+$	0.8
$-$	0.5

D	$P(y_1\|D)$
$+$	0.7
$-$	0.1

D	$P(y_2\|D)$
$+$	0.8
$-$	0.3

Figure 4.1: Naive Bayes classifier with a sensitive attribute X and non-sensitive attributes Y_1, Y_2

likelihood learning. Moreover, to efficiently eliminate the exponential number of patterns that could exist in a naive Bayes classifier, we propose a cutting-plane approach that uses the discrimination pattern miner to find and iteratively eliminate discrimination patterns until the entire learned model is fair. Experiments show that this process converges in a small number of iteration, effectively removing millions of discrimination patterns. Moreover, the learned fair models are of high quality, achieving likelihoods that are close to the best likelihoods attained by models with no fairness constraints. Our method also achieves higher accuracy than other methods of learning fair naive Bayes models.

Lastly, we relax the naive Bayes assumption for the discrimination pattern miner. As previously discussed, fairness guarantees are only as meaningful as how closely the model captures the data distribution. Thus, we extend the search algorithm to a much more general class of models—probabilistic circuits—by proposing a bound that can be computed efficiently, exploiting the tractable circuit representations.

4.1 Problem Formalization

Each individual is characterized by an assignment to a set of discrete variables $\mathbf{Z}$, called attributes or features. Assignment d to a binary decision variable D represents a decision made in favor of the individual (e.g., a loan approval). A set of *sensitive attributes* $\mathbf{S} \subset \mathbf{Z}$ specifies a protected group of entities, such as gender and race. We now define the notion of a discrimination pattern.

Definition 4. *Let P be a distribution over $D \cup \mathbf{Z}$. Let $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$, respectively. The* degree of discrimination *of $\mathbf{xy}$ is: $\Delta_{P,d}(\mathbf{x}, \mathbf{y}) \triangleq P(d|\mathbf{xy}) - P(d|\mathbf{y})$.*

The assignment $\mathbf{y}$ identifies a group of similar individuals, and the degree of discrimination quantifies how disclosing sensitive information $\mathbf{x}$ affects the decision for this group. Note that sensitive attributes missing from $\mathbf{x}$ can still appear in $\mathbf{y}$. We drop the subscripts P, d when clear from context.

Definition 5. *Let P be a distribution over $D \cup \mathbf{Z}$, and $\delta \in [0, 1]$ a threshold. Joint assignments $\mathbf{x}$ and $\mathbf{y}$ form a* discrimination pattern *w.r.t. P and δ if: (1) $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \backslash \mathbf{X}$; and (2) $|\Delta_{P,d}(\mathbf{x}, \mathbf{y})| > \delta$.*

Intuitively, we do not want information about the sensitive attributes to significantly affect the probability of getting a favorable decision. Let us consider two special cases of discrimination patterns. First, if $\mathbf{Y} = \emptyset$, then a small discrimination score $|\Delta(\mathbf{x}, \emptyset)|$ can be interpreted as an approximation of statistical parity, which is achieved when $P(d|\mathbf{x}) = P(d)$. For example, the naive Bayes network in Figure 4.1 satisfies approximate parity for $\delta = 0.2$ as $|\Delta(x, \emptyset)| = 0.086 \leq \delta$ and $|\Delta(\bar{x}, \emptyset)| = 0.109 \leq \delta$. Second, suppose $\mathbf{X} = \mathbf{S}$ and $\mathbf{Y} = \mathbf{Z} \backslash \mathbf{S}$. Then bounding $|\Delta(\mathbf{x}, \mathbf{y})|$ for all joint states $\mathbf{x}$ and $\mathbf{y}$ is equivalent to enforcing individual fairness assuming two individuals are considered similar if their non-sensitive attributes $\mathbf{y}$ are equal. The network in Figure 4.1 is also individually fair for $\delta = 0.2$ because $\max_{xy_1y_2} |\Delta(x, y_1y_2)| = 0.167 \leq \delta$.[1]

Even though the example network is considered (approximately) fair at the group level and individual level with fully observed features, it may still produce a discrimination pattern. In particular, $|\Delta(\bar{x}, y_1)| = 0.225 > \delta$. That is, a person with $\bar{x}$ and y_1 observed and the value of Y_2 undisclosed would receive a much more favorable decision had they not disclosed X as well. Hence, naturally we wish to ensure that there exists no discrimination pattern across all subsets of observable features.

Definition 6. *A distribution P is* δ-fair *if there exists no discrimination pattern w.r.t P and δ.*

Although our notion of fairness applies to any distribution, finding discrimination patterns can be computationally challenging: computing the degree of discrimination involves probabilistic inference, which is hard in general, and a given distribution may have exponentially many patterns.

[1] The highest discrimination score is observed at $\bar{x}$ and $y_1\bar{y_2}$, with $\Delta(\bar{x}, y_1\bar{y_2}) = -0.167$.

Algorithm 2 DISC-PATTERNS$(\mathbf{x}, \mathbf{y}, \mathbf{E})$

Input: P : Distribution over $D \cup \mathbf{Z}$, δ : discrimination threshold
Output: Discrimination patterns L
Data: $\mathbf{x} \leftarrow \emptyset$, $\mathbf{y} \leftarrow \emptyset$, $\mathbf{E} \leftarrow \emptyset$, $L \leftarrow []$

1: **for** all assignments z to some selected variable $Z \in \mathbf{Z} \setminus \mathbf{XYE}$ **do**
2: **if** $Z \in \mathbf{S}$ **then**
3: **if** $|\Delta(\mathbf{x}z, \mathbf{y})| > \delta$ **then** add $(\mathbf{x}z, \mathbf{y})$ to L
4: **if** $\text{UB}(\mathbf{x}z, \mathbf{y}, \mathbf{E}) > \delta$ **then** DISC-PATTERNS$(\mathbf{x}z, \mathbf{y}, \mathbf{E})$
5: **if** $|\Delta(\mathbf{x}, \mathbf{y}z)| > \delta$ **then** add $(\mathbf{x}, \mathbf{y}z)$ to L
6: **if** $\text{UB}(\mathbf{x}, \mathbf{y}z, \mathbf{E}) > \delta$ **then** DISC-PATTERNS$(\mathbf{x}, \mathbf{y}z, \mathbf{E})$
7: **if** $\text{UB}(\mathbf{x}, \mathbf{y}, \mathbf{E} \cup \{Z\}) > \delta$ **then** DISC-PATTERNS$(\mathbf{x}, \mathbf{y}, \mathbf{E} \cup \{Z\})$

Here we demonstrate how to discover and eliminate discrimination patterns of a naive Bayes classifier effectively by exploiting its independence assumptions. Concretely, we answer the following questions: (1) Can we certify that a classifier is δ-fair?; (2) If not, can we find the most important discrimination patterns?; (3) Can we learn a naive Bayes classifier that is entirely δ-fair?

4.2 Discovering Discrimination Patterns and Verifying δ-fairness

This section describes our approach to finding discrimination patterns or certifying that there are none.

4.2.1 Searching for Discrimination Patterns

One may naively enumerate all possible patterns and compute their degrees of discrimination. However, this would be very inefficient as there are exponentially many subsets and assignments to consider. We instead use branch-and-bound search to more efficiently decide if a model is fair.

Algorithm 2 finds discrimination patterns. It recursively adds variable instantiations and checks the discrimination score at each step. If the input distribution is δ-fair, the algorithm returns no pattern; otherwise, it returns the set of all discriminating patterns. Note that computing Δ requires probabilistic inference on distribution P. This can be done efficiently for large classes of graphical

models [Darwiche, 2009b; Poon and Domingos, 2011; Dechter, 2013; Rahman et al., 2014; Kisa et al., 2014b], and particularly for naive Bayes networks, which will be our main focus.

Furthermore, the algorithm relies on a good upper bound to prune the search tree and avoid enumerating all possible patterns. Here, $\mathrm{UB}(\mathbf{x}, \mathbf{y}, \mathbf{E})$ bounds the degree of discrimination achievable by observing more features after $\mathbf{xy}$ while excluding features $\mathbf{E}$.

Proposition 4.1. *Let P be a naive Bayes distribution over $D \cup \mathbf{Z}$, and let $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$. Let $\mathbf{x}'_u$ (resp. $\mathbf{x}'_l$) be an assignment to $\mathbf{X}' = \mathbf{S} \setminus \mathbf{X}$ that maximizes (resp. minimizes) $P(d|\mathbf{x}\mathbf{x}')$. Suppose $l, u \in [0, 1]$ such that $l \leq P(d|\mathbf{y}\mathbf{y}') \leq u$ for all possible assignments $\mathbf{y}'$ to $\mathbf{Y}' = \mathbf{Z} \setminus (\mathbf{X}\mathbf{Y})$. Then the degrees of discrimination for all patterns $\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}'$ that extend $\mathbf{xy}$ are bounded as follows:*

$$\min_{l \leq \gamma \leq u} \widetilde{\Delta}\left(P(\mathbf{x}\mathbf{x}'_l|d), P(\mathbf{x}\mathbf{x}'_l|\overline{d}), \gamma\right) \leq \Delta_{P,d}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') \leq \max_{l \leq \gamma \leq u} \widetilde{\Delta}\left(P(\mathbf{x}\mathbf{x}'_u|d), P(\mathbf{x}\mathbf{x}'_u|\overline{d}), \gamma\right),$$

where $\widetilde{\Delta}(\alpha, \beta, \gamma) \triangleq \frac{\alpha\gamma}{\alpha\gamma + \beta(1-\gamma)} - \gamma$.

Here, $\widetilde{\Delta} : [0, 1]^3 \to [0, 1]$ is introduced to relax the discrete problem of minimizing or maximizing the degree of discrimination into a continuous one. In particular, $\widetilde{\Delta}\left(P(\mathbf{x}|d), P(\mathbf{x}|\overline{d}), P(d|\mathbf{y})\right)$ equals the degree of discrimination $\Delta(\mathbf{x}, \mathbf{y})$. This relaxation allows us to compute bounds efficiently, as closed-form solutions. We refer to the Appendix for full proofs and details.

To apply above proposition, we need to find $\mathbf{x}'_u, \mathbf{x}'_l, l, u$ by maximizing/minimizing $P(d|\mathbf{x}\mathbf{x}')$ and $P(d|\mathbf{y}\mathbf{y}')$ for a given pattern $\mathbf{xy}$. Fortunately, this can be done efficiently for naive Bayes classifiers.

Lemma 1. *Given a naive Bayes distribution P over $D \cup \mathbf{Z}$, a subset $\mathbf{V} = \{V_i\}_{i=1}^n \subset \mathbf{Z}$, and an assignment $\mathbf{w}$ to $\mathbf{W} \subseteq \mathbf{Z} \setminus \mathbf{V}$, we have:* $\arg\max_{\mathbf{v}} P(d|\mathbf{v}\mathbf{w}) = \left\{\arg\max_{v_i} P(v_i|d)/P(v_i|\overline{d})\right\}_{i=1}^n$.

That is, the joint observation $\mathbf{v}$ that will maximize the probability of the decision can be found by optimizing each variable V_i independently; the same holds when minimizing. Hence, we can use Proposition 4.1 to upper-bound the discrimination scores of extended patterns in linear time.

Intermezzo 2: Fairness Considerations in Policy Making

Much of the research in fair machine learning, including the works described in this thesis, considers mathematical definitions of fairness and how to audit and enforce them. However, it is imperative that we ultimately ground these discussions to real-world settings. For instance, some technical notions or approaches to mitigate bias may not be compatible with certain legal framework of discrimination [Xiang, 2020; Corbett-Davies and Goel, 2018]. Moreover, algorithmic tools are developed and deployed within institutions and policies, and thus studying algorithmic fairness in isolation could miss important institutional considerations.

In particular, Solow-Niederman, Choi, and Van den Broeck [2019] used California's Money Bail Reform Act of 2017 (SB 10) as an example to demonstrate how risk assessment algorithms do not operate in isolation and that their fairness issues must be considered at the policy-making stage. Specifically, risk assessment statutes may create tension between a top-down, global understanding of fairness and accuracy and a tool that is well-tailored to local considerations.

This is also closely related to the fact that questions regarding the fairness of a model depends on the underlying distribution of the world, as argued throughout this thesis. Because the population distribution can differ, for instance, at the county and state level, we must pay close attention to the design of risk assessment policies, and specifically to who is granted authority and discretion over the tools.

4.2.2 Searching for Top-k Ranked Patterns

If a distribution is significantly unfair, Algorithm 2 may return exponentially many discrimination patterns. This is not only very expensive but makes it difficult to interpret the discrimination patterns. Instead, we would like to return a smaller set of "interesting" discrimination patterns.

An obvious choice is to return a small number of discrimination patterns with the highest absolute degree of discrimination. Searching for the k most discriminating patterns can be done with a small modification to Algorithm 2. First, the size of list L is limited to k. The conditions in Lines 3–7 are modified to check the current discrimination score and upper bounds against the smallest discrimination score of patterns in L, instead of the threshold δ.

Nevertheless, ranking patterns by their discrimination score may return patterns of very low probability. For example, the most discriminating pattern of a naive Bayes classifier learned on the COMPAS dataset [Propublica, 2016] has a high discrimination score of 0.42, but only has a 0.02%

probability of occurring.[2] The probability of a discrimination pattern denotes the proportion of the population (according to the distribution) that could be affected unfairly, and thus a pattern with extremely low probability could be of lesser interest. To address this concern, we propose a more sophisticated ranking of the discrimination patterns that also takes into account the probabilities of patterns.

Definition 7. *Let P be a distribution over $D \cup \mathbf{Z}$. Let $\mathbf{x}$ and $\mathbf{y}$ be joint instantiations to subsets $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$, respectively. The* divergence score *of $\mathbf{xy}$ is:*

$$\mathrm{Div}_{P,d,\delta}(\mathbf{x},\mathbf{y}) \triangleq \min_{Q} \mathrm{KL}\left(P \parallel Q\right) \;\; \textit{s.t.} \;\; |\Delta_{Q,d}(\mathbf{x},\mathbf{y})| \leq \delta,\; P(d\mathbf{z}) = Q(d\mathbf{z}),\; \forall\, d\mathbf{z} \not\models \mathbf{xy} \quad (4.1)$$

where $\mathrm{KL}\left(P \parallel Q\right) = \sum_{d,\mathbf{z}} P(d\mathbf{z}) \log(P(d\mathbf{z})/Q(d\mathbf{z}))$.

The divergence score assigns to a pattern $\mathbf{xy}$ the minimum Kullback-Leibler (KL) divergence between current distribution P and a hypothetical distribution Q that is fair on the pattern $\mathbf{xy}$ and differs from P only on the assignments that satisfy the pattern (namely $d\mathbf{xy}$ and $\overline{d}\mathbf{xy}$). Informally, the divergence score approximates how much the current distribution P needs to be changed in order for $\mathbf{xy}$ to no longer be a discrimination pattern. Hence, patterns with higher divergence score will tend to have not only higher discrimination score but also higher probabilities.

For instance, the pattern with the highest divergence score[3] on the COMPAS dataset has a discrimination score of 0.19 which is not insignificant, but also has a relatively high probability of 3.33% – more than two orders of magnitude larger than that of the most discriminating pattern. Therefore, such a general pattern could be more interesting for the user studying this classifier.

To find the top-k patterns with the divergence score, we need to be able to compute the score and its upper bound efficiently. The key insights are that KLD is convex and that Q, in Equation 4.1,

[2]The corresponding pattern is $\mathbf{x}=\{\text{White, Married, Female, } > 30 \text{ y/o}\}, \mathbf{y}=\{\text{Probation, Pretrial}\}$.

[3]$\mathbf{x} = \{\text{Married, } > 30 \text{ y/o}\}$, $\mathbf{y} = \{\}$.

Dataset Statistics						Proportion of search space explored					
						Divergence			Discrimination		
Dataset	Size	S	N	# Pat.	k	$\delta = 0.01$	$\delta = 0.05$	$\delta = 0.10$	$\delta = 0.01$	$\delta = 0.05$	$\delta = 0.10$
COMPAS	48,834	4	3	15K	1	6.387e-01	5.634e-01	3.874e-01	8.188e-03	8.188e-03	8.188e-03
					10	7.139e-01	5.996e-01	4.200e-01	3.464e-02	3.464e-02	3.464e-02
					100	8.222e-01	6.605e-01	4.335e-01	9.914e-02	9.914e-02	9.914e-02
Adult	32,561	4	9	11M	1	3.052e-06	7.260e-06	1.248e-05	2.451e-04	2.451e-04	2.451e-04
					10	7.030e-06	1.154e-05	1.809e-05	2.467e-04	2.467e-04	2.467e-04
					100	1.458e-05	1.969e-05	2.509e-05	2.600e-04	2.600e-04	2.597e-04
German	1,000	4	16	23B	1	5.075e-07	2.731e-06	2.374e-06	7.450e-08	7.450e-08	7.450e-08
					10	9.312e-07	3.398e-06	2.753e-06	1.592e-06	1.592e-06	1.592e-06
					100	1.454e-06	4.495e-06	3.407e-06	5.897e-06	5.897e-06	5.897e-06

Table 4.1: Data statistics (number of training instances, sensitive features S, non-sensitive features N, and potential patterns) and the proportion of patterns explored during the search, using the *Divergence* and *Discrimination* scores as rankings.

can freely differ from P only on one probability value (either that of $d\mathbf{xy}$ or $\overline{d}\mathbf{xy}$). Then:

$$\mathrm{Div}_{P,d,\delta}(\mathbf{x},\mathbf{y}) = P(d\mathbf{xy}) \log\left(\frac{P(d\mathbf{xy})}{P(d\mathbf{xy})+r}\right) + P(\overline{d}\mathbf{xy}) \log\left(\frac{P(\overline{d}\mathbf{xy})}{P(\overline{d}\mathbf{xy})-r}\right), \tag{4.2}$$

where $r = 0$ if $|\Delta_{P,d}(\mathbf{x},\mathbf{y})| \leq \delta$; $r = \frac{\delta - \Delta_{P,d}(\mathbf{x},\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}$ if $\Delta_{P,d}(\mathbf{x},\mathbf{y}) > \delta$; and $r = \frac{-\delta - \Delta_{P,d}(\mathbf{x},\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}$ if $\Delta_{P,d}(\mathbf{x},\mathbf{y}) < -\delta$. Intuitively, r represents the minimum necessary change to $P(d\mathbf{xy})$ for $\mathbf{xy}$ to be non-discriminating in the new distribution. Note that the smallest divergence score of 0 is attained when the pattern is already fair.

Lastly, we refer to the Appendix for two upper bounds of the divergence score, which utilize the bound on discrimination score of Proposition 4.1 and can be computed efficiently using Lemma 1.

4.2.3 Empirical Evaluation of Discrimination Pattern Miner

In this section, we report the experimental results on the performance of our pattern mining algorithms. All experiments were run on an AMD Opteron 275 processor (2.2GHz) and 4GB of RAM running Linux Centos 7. Execution time is limited to 1800 seconds.

Data and pre-processing. We use three datasets: The *Adult* dataset and *German* dataset are

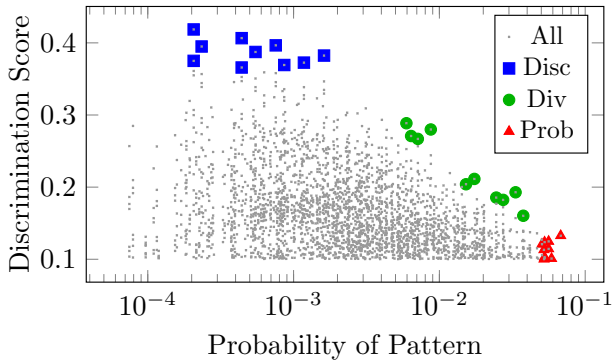


Figure 4.2: Discrimination patterns with $\delta = 0.1$ for the max-likelihood NB classifier on COMPAS.

used for predicting income level and credit risk, respectively, and are obtained from the UCI machine learning repository [Dua and Graff, 2017]; the *COMPAS* dataset is used for predicting recidivism. These datasets have been commonly studied regarding fairness and were shown to exhibit some form of discrimination by several previous works [Luong et al., 2011; Larson et al., 2016; Tramer et al., 2017; Salimi et al., 2019]. As pre-processing, we removed unique features (e.g. names of individuals) and duplicate features. See Table 4.1 for a summary.

Q1. Does our pattern miner find discrimination patterns more efficiently than by enumerating all possible patterns? We answer this question by inspecting the fraction of all possible patterns that our pattern miner visits during the search. Table 4.1 shows the results on three datasets, using two rank heuristics (discrimination and divergence) and three threshold values (0.01, 0.05, and 0.1). The results are reported for mining the top-k patterns when k is 1, 10, and 100. A naive method has to enumerate all possible patterns to discover the discriminating ones, while our algorithm visits only a small fraction of patterns (e.g., one in every several millions on the German dataset).

Q2. Does the divergence score find discrimination patterns with both a high discrimination score and high probability? Figure 4.2 shows the *probability* and *discrimination score* of all patterns in the COMPAS dataset. The top-10 patterns according to three measures (discrimination score, divergence score, and probability) are highlighted in the figure. The observed trade-off between probability and discrimination score indicates that picking the top patterns according to each measure will yield low quality patterns according to the other measure. The divergence score,

however, balances the two measures and returns patterns that have high probability and discrimination scores. Also note that the patterns selected by the divergence score lie in the Pareto front. This in fact always holds by the definition of this heuristic; fixing the probability and increasing the discrimination score will also increase the divergence score, and vice versa.

4.3 Learning Fair Naive Bayes Classifiers

We now describe our approach to learning the maximum-likelihood parameters of a naive Bayes model from data while eliminating discrimination patterns. A common approach to learning naive Bayes models with certain properties is to formulate it as an optimization problem of certain form, for which efficient solvers are available [Khosravi et al., 2019b]. We formulate the learning subject to fairness constraints as a signomial program, which has the following form:

$$\text{minimize } f_0(x), \quad \text{s.t.} \quad f_i(x) \leq 1, \quad g_j(x) = 1 \quad \forall\, i, j$$

where each f_i is signomial while g_j is monomial. A *signomial* is a function of the form

$$\sum_k c_k x_1^{a_{1k}} \cdots x_n^{a_{1n}}$$

defined over real positive variables $x_1 \dots x_n$ where $c_k, a_{ij} \in \mathbb{R}$; a *monomial* is of the form $c x_1^{a_1} \cdots x_n^{a_n}$ where $c > 0$ and $a_i \in \mathbb{R}$. Signomial programs are not globally convex, but a locally optimal solution can be computed efficiently, unlike the closely related class of geometric programs, for which the globally optimum can be found efficiently [Ecker, 1980].

4.3.1 Parameter Learning with Fairness Constraints

The likelihood of a Bayesian network given data $\mathcal{D}$ is $P_\theta(\mathcal{D}) = \prod_i \theta_i^{n_i}$ where n_i is the number of examples in $\mathcal{D}$ that satisfy the assignment corresponding to parameter θ_i. To learn the

maximum-likelihood parameters, we minimize the inverse of likelihood which is a monomial: $\theta_{\text{ml}} = \arg\min_\theta \prod_i \theta_i^{-n_i}$. The parameters of a naive Bayes network with binary class consist of $\theta_d, \theta_{\bar{d}}$, and $\theta_{z|d}, \theta_{z|\bar{d}}$ for all z.

Next, we show the constraints for our optimization problem. To learn a valid distribution, we need to ensure that probabilities are non-negative and sum to one. The former assumption is inherent to signomial programs. To enforce the latter, for each instantiation d and feature Z, we need that $\sum_z \theta_{z|d} = 1$, or as signomial inequality constraints: $\sum_z \theta_{z|d} \leq 1$ and $2 - \sum_z \theta_{z|d} \leq 1$.

Finally, we derive the constraints to ensure that a given pattern $\mathbf{xy}$ is non-discriminating.

Proposition 4.2. *Let P_θ be a naive Bayes distribution over $D \cup \mathbf{Z}$, and let $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$. Then $|\Delta_{P_\theta,d}(\mathbf{x}, \mathbf{y})| \leq \delta$ for a threshold $\delta \in [0, 1]$ iff the following holds:*

$$r_{\mathbf{x}} = \frac{\prod_x \theta_{x|\bar{d}}}{\prod_x \theta_{x|d}}, \quad r_{\mathbf{y}} = \frac{\theta_{\bar{d}} \prod_y \theta_{y|\bar{d}}}{\theta_d \prod_y \theta_{y|d}},$$

$$\left(\frac{1-\delta}{\delta}\right) r_{\mathbf{x}} r_{\mathbf{y}} - \left(\frac{1+\delta}{\delta}\right) r_{\mathbf{y}} - r_{\mathbf{x}} r_{\mathbf{y}}^2 \leq 1, \quad -\left(\frac{1+\delta}{\delta}\right) r_{\mathbf{x}} r_{\mathbf{y}} + \left(\frac{1-\delta}{\delta}\right) r_{\mathbf{y}} - r_{\mathbf{x}} r_{\mathbf{y}}^2 \leq 1.$$

Note that above equalities and inequalities are valid signomial program constraints. Thus, we can learn the maximum-likelihood parameters of a naive Bayes network while ensuring a certain pattern is fair by solving a signomial program. Furthermore, we can eliminate multiple patterns by adding the constraints in Proposition 4.2 for each of them. However, learning a model that is entirely fair with this approach will introduce an exponential number of constraints. Not only does this make the optimization more challenging, but listing all patterns may simply be infeasible.

4.3.2 Learning δ-fair Parameters

To address the aforementioned challenge of removing an exponential number of discrimination patterns, we propose an approach based on the *cutting plane* method. That is, we iterate between *parameter learning* and *constraint extraction*, gradually adding fairness constraints to the optimiza-

tion. The parameter learning component is as described in the previous section, where we add the constraints of Proposition 4.2 for each discrimination pattern that has been extracted so far. For constraint extraction we use the top-k pattern miner from Section 4.2. At each iteration, we learn the maximum-likelihood parameters subject to fairness constraints, and find k more patterns using the updated parameters to add to the set of constraints in the next iteration. This process is repeated until the pattern miner finds no more discrimination pattern.

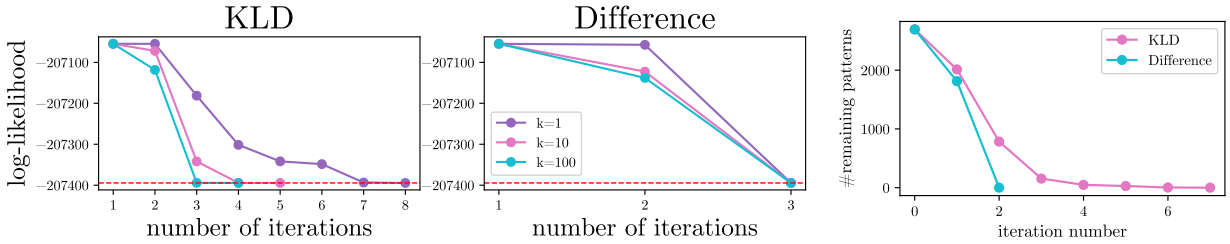
In the worst case, our algorithm may add exponentially many fairness constraints whilst solving multiple optimization problems. However, as we will later show empirically, we can learn a δ-fair model by explicitly enforcing only a small fraction of fairness constraints. The efficacy of our approach depends on strategically extracting patterns that are significant in the overall distribution. Here, we again use a ranking by discrimination or divergence score, which we also evaluate empirically.

4.3.3 Empirical Evaluation of δ-fair Learner

We will now evaluate our iterative algorithm for learning δ-fair naive Bayes models. We use the same datasets and hardware as in Section 4.2. To solve the signomial programs, we use *GPkit*, which finds local solutions to these problems using a convex optimization solver as its backend.[4] Throughout our experiments, Laplace smoothing was used to avoid learning zero probabilities.

Q1. Can we learn a δ-fair model in a small number of iterations while only asserting a small number of fairness constraints? We train a naive Bayes model on the COMPAS dataset subject to δ-fairness constraints. Fig. 4.3a shows how the iterative method converges to a δ-fair model, whose likelihood is indicated by the dotted line. Our approach converges to a fair model in a few iterations, including only a small fraction of the fairness constraints. In particular, adding only the most discriminating pattern as a constraint at each iteration learns an entirely δ-fair

[4]We use Mosek (`www.mosek.com`) as backend.



(a) Log-likelihood

(b) Number of remaining patterns

Figure 4.3: Log-likelihood and the number of remaining discrimination patterns after each iteration of learning on COMPAS dataset with $\delta = 0.1$.

Dataset	Unconstrained	δ-fair	Independent
COMPAS	-207,055	-207,395	-208,639
Adult	-226,375	-228,763	-232,180
German	-12,630	-12,635	-12,649

Table 4.2: Log-likelihood of models learned without fairness constraints, with the δ-fair learner ($\delta = 0.1$), and by making sensitive variables independent from the decision variable.

model with only three fairness constraints.[5] Moreover, Fig. 4.3b shows the number of remaining discrimination patterns after each iteration of learning with $k = 1$. Note that enforcing a single fairness constraint can eliminate a large number of remaining ones. Eventually, a few constraints subsume all discrimination patterns.

Q2. How does the quality of naive Bayes models from our fair learner compare to ones that make the sensitive attributes independent of the decision? and to the best model without fairness constraints? A simple method to guarantee that a naive Bayes model is δ-fair is to make all sensitive variables independent from the target value. An obvious downside is the negative effect on the predictive power of the model. We compare the models learned by our approach with: (1) a maximum-likelihood model with no fairness constraints (unconstrained) and (2) a model in which the sensitive variables are independent of the decision variable, and the remaining parameters are learned using the max-likelihood criterion (independent). These models lie at two opposite ends of the spectrum of the trade-off between fairness and accuracy. The δ-fair model falls between these

[5]There are 2695 discrimination patterns w.r.t. unconstrained naive Bayes on COMPAS and $\delta = 0.1$.

Dataset	λ =0.5	λ =0.9	λ =0.95	λ =0.99	λ =1.0
COMPAS	2,504	2,471	2,470	3,069	0
Adult	>1e6	661	652	605	0
German	>1e6	3	2	0	0

Table 4.3: Number of remaining patterns with $\delta = 0.1$ in naive Bayes models trained on discrimination-free data, where λ determines the trade-off between fairness and accuracy in the data repair step [Feldman et al., 2015].

dataset	Unconstrained	2NB	Repaired	δ-fair
COMPAS	0.880	0.875	0.878	0.879
Adult	0.811	0.759	0.325	0.827
German	0.690	0.679	0.688	0.696

Table 4.4: Comparing accuracy of our δ-fair models with two-naive-Bayes method and a naive Bayes model trained on repaired, discrimination-free data.

extremes, balancing approximate fairness and prediction power.

We compare the log-likelihood of these models, shown in Table 4.2, as it captures the overall quality of a probabilistic classifier which can make predictions with partial observations. The δ-fair models achieve likelihoods that are much closer to those of the unconstrained models than the independent ones. This shows that it is possible to enforce the fairness constraints without a major reduction in model quality.

Q3. Do discrimination patterns still occur when learning naive Bayes models from fair data? We first use the data repair algorithm proposed by Feldman et al. [2015] to remove discrimination from data, and learn a naive Bayes model from the repaired data. Table 4.3 shows the number of remaining discrimination patterns in such model. The results indicate that as long as preserving some degree of accuracy is in the objective, this method leaves lots of discrimination patterns, whereas our method removes all patterns.

Q4. How does the performance of δ-fair naive Bayes classifier compare to existing work? Table 4.4 reports the 10-fold CV accuracy of our method (δ-fair) compared to a max-likelihood naive Bayes model (unconstrained) and two other methods of learning fair classifiers: the two-naive-Bayes

method (2NB) [Calders and Verwer, 2010], and a naive Bayes model trained on discrimination-free data using the repair algorithm of Feldman et al. [2015] with $\lambda = 1$. Even though the notion of discrimination patterns was proposed for settings in which predictions are made with missing values, our method still outperforms other fair models in terms of accuracy, a measure better suited for predictions using fully-observed features. Moreover, our method also enforces a stronger definition of fairness than the two-naive-Bayes method which aims to achieve statistical parity, which is subsumed by the notion of discrimination patterns. It is also interesting to observe that our δ-fair NB models perform even better than unconstrained NB models for the Adult and German dataset. Hence, removing discrimination patterns does not necessarily impose an extra cost on the prediction task.

4.4 Finding Discrimination Patterns in Probabilistic Circuits

This section extends the search algorithm to find discrimination patterns in probabilistic circuits. Recall that the probability of a pattern corresponds to the proportion of the affected subpopulation, according to the probabilistic model. Therefore, a meaningful analysis of discrimination patterns depends on how well the model captures the population distribution. For instance, naive Bayes classifiers make strong independence assumptions and are generally too restrictive to fit real-world distributions. Thus, we consider a more expressive type of probabilistic models, in particular, probabilistic circuits.

Algorithm 2 can be applied to any probabilistic model that allows efficient computation of discrimination score and an upper bound for it. First, smooth and decomposable PCs support linear-time computation of conditional probabilities, and in turn, of discrimination score for any pattern. Next, we use the following as our bound $UB(\boldsymbol{x}, \boldsymbol{y}, \mathbf{E})$, which does not rely on the naive Bayes assumption:

$$\max \left\{ \left| \max_{\mathbf{u}} P(d \mid \boldsymbol{x}, \boldsymbol{y}, \mathbf{u}) - \min_{\mathbf{u}} P(d \mid \boldsymbol{y}, \mathbf{u}) \right|, \left| \min_{\mathbf{u}} P(d \mid \boldsymbol{x}, \boldsymbol{y}, \mathbf{u}) - \max_{\mathbf{u}} P(d \mid \boldsymbol{y}, \mathbf{u}) \right| \right\} \quad (4.3)$$

where $\mathbf{U}$ can be any subset of $\mathbf{Z} \setminus (\mathbf{X} \cup \mathbf{Y} \cup \mathbf{E})$—in other words, the remaining variables to extend the current pattern. The core component of above bound is maximizing or minimizing the conditional probability of the form $P(d \mid \boldsymbol{y}, \mathbf{u})$ over the values of some $\mathbf{U}$ for a given $\boldsymbol{y}$. We now show how such optimization can be done tractably for certain classes of probabilistic circuits.

We use two key observations, expressed by the following lemmas.

Lemma 1. *Let P be a distribution over $D \cup \mathbf{Z}$ and $\boldsymbol{x}$ a joint assignment to $\mathbf{X} \subseteq \mathbf{Z}$. Also denote $\mathbf{V} = \mathbf{Z} \setminus \mathbf{X}$. Then for any $\mathbf{U} \subseteq \mathbf{Z} \setminus \mathbf{X}$ the following holds:*

$$\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} P(d \mid \boldsymbol{x}, \mathbf{u}) \leq \max_{\mathbf{v} \in \mathsf{val}(\mathbf{V})} P(d \mid \boldsymbol{x}, \mathbf{v})$$

That is, to maximize a conditional probability given some (partial) assignments for a set of free variables, it suffices to consider only the complete assignments to those variables. Analogously, this statement holds for minimization as well, with the direction of inequality reversed.

Proof. Consider any $\mathbf{U} \subset \mathbf{Z} \setminus \mathbf{X}$ and $W \notin \mathbf{U}$. It suffices to show that

$$\forall\, \mathbf{u} \in \mathsf{val}(\mathbf{U}), \quad P(d \mid \boldsymbol{x}, \mathbf{u}) \leq \max_{w \in \mathsf{val}(W)} P(d \mid \boldsymbol{x}, \mathbf{u}, w),$$

as the lemma then follows via a simple inductive argument. Denote $\mathsf{val}(W) = \{w_1, w_2, \ldots, w_n\}$. To show that there is at least one $w \in \mathsf{val}(W)$ such that $P(d \mid \boldsymbol{x}, \mathbf{u}) \leq P(d \mid \boldsymbol{x}, \mathbf{u}, w)$ for any $\mathbf{u}$, we will show that $P(d \mid \boldsymbol{x}, \mathbf{u}) > P(d \mid \boldsymbol{x}, \mathbf{u}, w_i)$ for $i = 1, \ldots, n-1$ implies that $P(d \mid \boldsymbol{x}, \mathbf{u}) < P(d \mid \boldsymbol{x}, \mathbf{u}, w_n)$. First, it implies that $P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u}, w_i) > P(\boldsymbol{x}, \mathbf{u}) \cdot P(d, \boldsymbol{x}, \mathbf{u}, w_i)$ for all $i \leq n-1$ which leads to:

$$\sum_{i=1}^{n-1} (P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u}, w_i)) > \sum_{i=1}^{n-1} (P(\boldsymbol{x}, \mathbf{u}) \cdot P(d, \boldsymbol{x}, \mathbf{u}, w_i)).$$

Subtracting both sides from $P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u})$ we get:

$$
\begin{aligned}
&P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u}) - \sum_{i=1}^{n-1} (P(\boldsymbol{x}, \mathbf{u}) \cdot P(d, \boldsymbol{x}, \mathbf{u}, w_i)) \\
&\qquad > P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u}) - \sum_{i=1}^{n-1} (P(d, \boldsymbol{x}, \mathbf{u}) \cdot P(\boldsymbol{x}, \mathbf{u}, w_i)) \\
\implies &\frac{P(d, \boldsymbol{x}, \mathbf{u}) - \sum_{i=1}^{n-1} P(d, \boldsymbol{x}, \mathbf{u}, w_i)}{P(\boldsymbol{x}, \mathbf{u}) - \sum_{i=1}^{n-1} P(\boldsymbol{x}, \mathbf{u}, w_i)} > \frac{P(d, \boldsymbol{x}, \mathbf{u})}{P(\boldsymbol{x}, \mathbf{u})} \\
\implies &P(d \mid \boldsymbol{x}, \mathbf{u}, w_n) > P(d \mid \boldsymbol{x}, \mathbf{u}).
\end{aligned}
$$

□

Lemma 2. *Let P be a distribution over $D \cup \mathbf{Z}$, $\boldsymbol{x}$ an assignment to $\mathbf{X} \subseteq \mathbf{Z}$, and $\mathbf{U} \subseteq \mathbf{Z} \setminus \mathbf{X}$. Then,*

$$\arg\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} P(d \mid \boldsymbol{x}, \mathbf{u}) = \arg\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} \frac{P(\boldsymbol{x}, \mathbf{u} \mid d)}{P(\boldsymbol{x}, \mathbf{u} \mid \overline{d})}.$$

Proof. Since $P(d \mid \boldsymbol{x}, \mathbf{u}) = \frac{P(d, \boldsymbol{x}, \mathbf{u})}{P(d, \boldsymbol{x}, \mathbf{u}) + P(\overline{d}, \boldsymbol{x}, \mathbf{u})} = \frac{1}{1 + P(\overline{d}, \boldsymbol{x}, \mathbf{u}) / P(d, \boldsymbol{x}, \mathbf{u})}$, we obtain that

$$\arg\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} P(d \mid \boldsymbol{x}, \mathbf{u}) = \arg\min_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} \frac{P(\overline{d}, \boldsymbol{x}, \mathbf{u})}{P(d, \boldsymbol{x}, \mathbf{u})} = \arg\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} \frac{P(d, \boldsymbol{x}, \mathbf{u})}{P(\overline{d}, \boldsymbol{x}, \mathbf{u})} = \arg\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} \frac{P(\boldsymbol{x}, \mathbf{u} \mid d)}{P(\boldsymbol{x}, \mathbf{u} \mid \overline{d})}.$$

□

Combining these observations, we see that the upper bound in Equation 4.3 can be computed easily if we can efficiently maximize and minimize quantities of the form $P(\boldsymbol{x}, \mathbf{u} \mid d) / P(\boldsymbol{x}, \mathbf{u} \mid \overline{d})$ over values of $\mathbf{U} = \mathbf{Z} \setminus \mathbf{Y}$ for some given assignment $\boldsymbol{x} \in \mathsf{val}(\mathbf{X})$. In fact, we derive an algorithm with worst-case quadratic time complexity (in the size of the circuit) for PCs that satisfy additional structural constraints.

In particular, Algorithm 3 maximizes the ratio between two PCs that are deterministic and compatible. A circuit is deterministic if the children of every sum node have disjoint supports (denoted by $\mathsf{supp}(n)$). In other words, for every complete assignment $\boldsymbol{z}$, at most one of the children

Algorithm 3 Best Ratio: $\mathrm{BR}(n, m)$

Input: deterministic and compatible PCs n and m over $\mathbf{Z}$; an assignment $\boldsymbol{x} \in \mathsf{val}(\mathbf{X})$ for $\mathbf{X} \subset \mathbf{Z}$
Output: $\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} n(\boldsymbol{x}, \mathbf{u}) / m(\boldsymbol{x}, \mathbf{u})$ where $\mathbf{U} = \mathbf{Z} \setminus \mathbf{X}$

1: **if** n, m are leaf nodes **then**
2: **if** $\mathsf{supp}(n) \cap \mathsf{supp}(m) \neq \emptyset$ and $n(\boldsymbol{x}) \neq 0, m(\boldsymbol{x}) \neq 0$ **then**
3: $\mathrm{BR}(n, m) \leftarrow 1$
4: **else**
5: $\mathrm{BR}(n, m) \leftarrow 0$
6: **else if** n, m are product nodes **then**
7: $\mathrm{BR}(n, m) \leftarrow \prod_{i=1}^{|\mathsf{ch}(n)|} \mathrm{BR}(n_i, m_i)$
8: **else** ▷ n, m are sum nodes
9: $\mathrm{BR}(n, m) \leftarrow \max_{n_i \in \mathsf{ch}(n), m_j \in \mathsf{ch}(m)} \frac{\theta_i}{\theta_j} \mathrm{BR}(n_i, m_j)$

nodes will have a non-zero output. In addition, two circuits are compatible if they are: (1) smooth and decomposable; and (2) any pair of product nodes, one from each circuit, that are defined over the same set of variables decompose the variables in the same way. We refer the readers to [Vergari et al., 2021] for a more detailed discussion of compatibility. We can then apply this algorithm to two compatible PCs, representing the conditional distributions $P(\mathbf{Z} \mid d)$ and $P(\mathbf{Z} \mid \overline{d})$. For instance, consider a PC over variables $D \cup \mathbf{Z}$ whose root is a decision node on D as in **??**: then the two subcircuits rooted at each child of the root node exactly corresponds to the conditional distributions given D, and they are deterministic and share a vtree (i.e. are compatible). Moreover, we can easily tweak the algorithm to minimize the ratio, by changing Line 9 to return the minimum over non-zero values of the recursive calls if they exist, or zero otherwise.

Proof of Correctness. We proceed via induction. For the leaves, as they are compatible, by definition their supports are either identical or completely disjoint. Thus, the maximum ratio is 1, 0, or undefined (we also propagate 0 in this case).

Next, consider two compatible product nodes. As they decompose the variables identically, we can order their children nodes such that $n(\boldsymbol{z}) = \prod_i n_i(\boldsymbol{z}_i)$ and $m(\boldsymbol{z}) = \prod_i m_i(\boldsymbol{z}_i)$, where n_i and m_i are over the same set of variables $\mathbf{Z}_i$. Let us write $\mathbf{U}_i = \mathbf{U} \cap \mathbf{Z}_i$ and $\mathbf{X}_i = \mathbf{X} \cap \mathbf{Z}_i$. Then, we

have:

$$\max_{\mathbf{u}\in\mathsf{val}(\mathbf{U})} \frac{n(\boldsymbol{x},\mathbf{u})}{m(\boldsymbol{x},\mathbf{u})} = \max_{\mathbf{u}\in\mathsf{val}(\mathbf{U})} \frac{\prod_i n_i(\boldsymbol{x}_i,\mathbf{u}_i)}{\prod_i m_i(\boldsymbol{x}_i,\mathbf{u}_i)} = \prod_i \max_{\mathbf{u}_i\in\mathsf{val}(\mathbf{U}_i)} \frac{n_i(\boldsymbol{x}_i,\mathbf{u}_i)}{m_i(\boldsymbol{x}_i,\mathbf{u}_i)},$$

leading to Line 7 in Algorithm 3.

Finally, consider two deterministic sum nodes. Then for any $\boldsymbol{z}$, at most one children each of n and m would evaluate non-zero values. That is, the sum nodes can effectively be treated as maximization nodes: e.g. $n(\boldsymbol{z}) = \sum_i n_i(\boldsymbol{z}) = \max_i n_i(\boldsymbol{z})$. Moreover, among all pairs of children n_i, m_j, the ratio $n_i(\boldsymbol{z})/m_j(\boldsymbol{z})$ for any fixed $\boldsymbol{z}$ would be non-zero for at most one pair (again, we treat the ratio that is undefined as 0). Therefore, we have:

$$\frac{n(\boldsymbol{z})}{m(\boldsymbol{z})} = \frac{\sum_i n_i(\boldsymbol{z})}{\sum_j m_j(\boldsymbol{z})} = \frac{\max_i n_i(\boldsymbol{z})}{\max_j m_j(\boldsymbol{z})} = \max_{i,j} \frac{n_i(\boldsymbol{z})}{m_j(\boldsymbol{z})}.$$

Thus, we can break down the maximization as the following, corresponding to Line 9:

$$\max_{\mathbf{u}\in\mathsf{val}(\mathbf{U})} \frac{n(\boldsymbol{x},\mathbf{u})}{m(\boldsymbol{x},\mathbf{u})} = \max_{\mathbf{u}\in\mathsf{val}(\mathbf{U})} \max_{i,j} \frac{n_i(\boldsymbol{x},\mathbf{u})}{m_j(\boldsymbol{x},\mathbf{u})} = \max_{i,j} \max_{\mathbf{u}\in\mathsf{val}(\mathbf{U})} \frac{n_i(\boldsymbol{x},\mathbf{u})}{m_j(\boldsymbol{x},\mathbf{u})}.$$

□

Furthermore, we can similarly search for top-k patterns ranked by their divergence scores. The upper bound on divergence score used in Section 4.2.2 in fact does not require the naive Bayes assumption, and only needs efficient maximization/minimization of the conditional probability of extensions. Thus, given the kinds of PC structure described above, we can also compute a non-trivial bound and extend the branch-and-bound search approach in a straightforward manner to mine divergence patterns in PCs as well.

Table 4.5: Dataset statistics (number of examples, number of sensitive features S, non-sensitive features N, and number of potential patterns) and speedup of top-k search v.s. naive enumeration, in terms of the fraction of search space explored.

						Disc.	Divergence		
Dataset	Size	S	N	# Pat.	k	$\delta=0.1$	$\delta=0.01$	$\delta=0.05$	$\delta=0.10$
COMPAS	48834	4	3	15K	1	2.73x	2.17x	1.40x	1.16x
					10	2.68x	1.85x	1.26x	1.10x
					100	2.52x	1.46x	1.13x	1.04x
Income	195665	2	6	11K	1	1.22x	1.50x	1.32x	1.13x
					10	1.20x	1.40x	1.26x	1.08x
					100	1.13x	1.31x	1.15x	1.02x
Adult	32561	4	9	11M	1	1.32x	24.20x	16.72x	10.88x
					10	1.31x	20.44x	14.75x	9.82x
					100	1.29x	16.10x	11.87x	8.40x

4.4.1 Empirical Evaluation

We again evaluate the discrimination pattern mining algorithm on the *COMPAS* and *Adult* datasets, as well as the *Income* [Ding et al., 2021] dataset used for predicting income levels. From each dataset, a probabilistic circuit is learned using the STRUDEL algorithm [Dang et al., 2022], which returns deterministic and structured decomposable PCs as required by our search algorithm.

We first evaluate the efficiency of our branch-and-bound search algorithm to find discrimination patterns. As our approach is the first non-trivial method for a general class of probabilistic circuits, we see whether it is more efficient than a naive solution that enumerates all possible patterns. We mine the top-k patterns for two ranking heuristics (discrimination and divergence score), three values of k (1, 10, 100), and three threshold values δ (0.01, 0.05, 0.1). Table 4.5 reports the speedup in terms of the proportion of the search space visited by our algorithm compared to the naive approach. Note that only the settings in which $\delta = 0.1$ for ranking by discrimination score are reported, because the results are identical for smaller values of δ. We observe that pruning is effective, resulting in consistent speedup, including some significant improvement in performance as high as 24x speedup in the case of mining top-k divergence patterns on the Adult dataset.

Moreover, note that our method must compute an upper bound at every search step, which has a worst-case quadratic time complexity. However, we see that pruning the search space still improves the overall run time of the algorithm, even with this extra computation. For example, our method explores a little less than half the search space for top-k discrimination patterns with $\delta = 0.1$ on the COMPAS dataset, and it takes about 60% of the time taken by naive enumeration; concretely, it takes 24.4s, 24.6s, and 25.3s for $k = 1, 10, 100$, respectively, while the naive approach takes 40.2s.

4.5 Discussion

This chapter introduced a novel definition of fair probability distribution in terms of discrimination patterns which considers exponentially many (partial) observations of features. We have also presented algorithms to search for discrimination patterns in naive Bayes networks and to learn a high-quality fair naive Bayes classifier from data. We empirically demonstrated the efficiency of our search algorithm and the ability to eliminate exponentially many discrimination patterns by iteratively removing a small fraction at a time.

This approach to fair distribution implies group fairness such as statistical parity. However, ensuring group fairness in general is always with respect to a distribution and is only valid under the assumption that this distribution is truthful. While our approach guarantees some level of group fairness of naive Bayes classifiers, this is only true if the naive Bayes assumption holds. That is, the group fairness guarantees do not extend to using the classifier on an arbitrary population.

There is always a tension between three criteria of a probabilistic model: its fidelity, fairness, and tractability. Our initial approach aims to strike a balance between them by giving up some likelihood to be tractable (naive Bayes assumption) and more fair. There are certainly other valid approaches: learning a more general graphical model to increase fairness and truthfulness, which would in general make it intractable, or making the model less fair in order to make it more truthful and tractable.

As a step towards achieving a better balance with more expressive models, Section 4.4 introduced

a search algorithm to mine discrimination pattern from probabilistic circuits. Then a promising future direction is to learn fair probabilistic circuits by eliminating discrimination patterns. Moreover, given the intractability of an exact search algorithm, I also hope to explore approximate methods.

CHAPTER 5

Robust Decision Making

In many applications, one can define a large set of features to support the classification task at hand. At test time, however, these become prohibitively expensive to evaluate. This chapter considers the problem of removing costly features from a probabilistic classifier—derived from a probabilistic circuit or a Bayesian network for example—while staying robust to these changes and maintaining its classification behavior. To this end, we propose a closeness metric between Bayesian classifiers, called the *expected classification agreement (ECA).* Next, we present the first algorithm to compute it exactly[1] and a trimming algorithm that maximizes the expected agreement subject to a budgetary constraint, utilizing new theoretical insights to perform branch-and-bound search in the space of feature sets, while computing bounds on the ECA. Experiments on benchmark datasets investigate both the runtime cost of trimming and its effect on the robustness and accuracy of the resulting classifier.

5.1 Introduction

Bayesian classification plays a prominent role throughout machine learning [Wu et al., 2008; Laidlaw et al., 1998; Metsis et al., 2006]. In this setting, one has a model that specifies a probability distribution $\Pr(.)$ over a set of variables, including class variable C and attributes or features $\mathbf{F} = \{F_1, \ldots, F_n\}$. Given a particular instance, described as an assignment to features $\mathbf{f} = \{f_1, \ldots, f_n\}$,

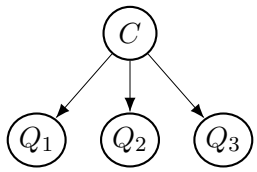
[1]The algorithm was initially developed for a closely related query called the expected same-decision probability on Bayesian networks, based on compiling the network into a tractable circuit representation for weighted model counting [Choi et al., 2017]. Here it is presented for ECA on probabilistic circuits, which can also be compiled from Bayesian networks.

this model is used to compute the posterior probability $\Pr(C|f_1, \ldots, f_n)$ which is then compared against a threshold T to classify the instance.

In practice, observing features often has a cost, and one typically needs to keep it within a given budget. For example, features in a medical diagnosis may be invasive, time-consuming, or expensive medical tests [Kononenko, 2001]. Similar issues arise in active sensing [Gao and Koller, 2011], adaptive testing [Millán and Pérez-De-La-Cruz, 2002; Munie and Shoham, 2008], and robotics [Kollar and Roy, 2008]. This problem has been studied from different angles, often under the umbrella of *feature selection*. For example, one may select features at learning time based on their relevance, redundancy, or classification accuracy [Kira and Rendell, 1992; Yu and Liu, 2004]. Alternatively, features may be selected at prediction time based on their expected misclassification cost or information gain [Bilgic and Getoor, 2011; Krause and Guestrin, 2009; Zhang and Ji, 2010]. Such probabilistic objectives are computed on the distribution of the Bayesian classifier.

In this chapter, I approach the problem from a different perspective, namely, classifier *trimming*. In addition to selecting features that fit the budget, trimming adjusts the threshold T to induce a new classifier. Moreover, instead of simply optimizing the predictive accuracy, we want the trimming to be robust—that is, to preserve the original classifier's *general behavior*. This is due to the following two reasons. First, Bayesian classifiers often incorporate significant expert knowledge in the form of priors, structural assumptions, and choice of distribution class [Lucas, 2001]. This is particularly true in medical applications where data is scarce [Bellazzi and Zupan, 2008]. Second, two classifiers with the same predictive quality can exhibit vastly different behavior and failure modes. For example, Zhao et al. [2017] describe two classifiers with a similar accuracy, but markedly different amounts of gender bias in their predictions. In either scenario, it is essential to retain the desired behavior of the original classifier during trimming.

Figure 5.1 depicts a classifier utilizing three features $\mathbf{Q} = \{Q_1, Q_2, Q_3\}$ with a threshold of $T = 0.07$. Consider two possible trimmings of this classifier: one obtained by removing Q_2 and adjusting the threshold to 0.10, the other with Q_1 removed and the threshold changed to 0.30. The trimmed classifiers are clearly less expensive than the original one, but how do we quantitatively



$\Pr(C=+)$
0.1

C	$\Pr(Q_1=+\|C)$
$+$	0.9
$-$	0.3

C	$\Pr(Q_2=+\|C)$
$+$	0.9
$-$	0.6

C	$\Pr(Q_3=+\|C)$
$+$	0.4
$-$	0.2

Figure 5.1: Naive Bayes classifier for a quiz scenario where answers on $\mathbf{Q} = \{Q_1, Q_2, Q_3\}$ (features) depend on knowledge C (class)

compare and choose between these trimmings? To answer this question, this chapter introduces the notion of *expected classification agreement (ECA)*. It is an expectation of the two classifiers agreeing on instances, measuring how much behavior from the original classifier is preserved.

Probabilistic graphical models, such as Bayesian networks, are often used to represent the Bayesian classifier's distribution. We propose an algorithm to find the best trimming of a Bayesian network classifier subject to a budgetary constraint. The algorithm selects features and chooses a new classification threshold in order to maximize the ECA. We also propose a specialized algorithm for the case of naive Bayes classifiers that exploits the naive Bayes independence assumptions for more efficient trimming. These novel trimming algorithms are based on the following progression of ideas. We introduce the first algorithm to compute the ECA between the classifier and its trimming, using tractable circuits. Next, we propose an upper bound on the ECA that can be computed more efficiently, enabled by our formulation that adjusts the threshold. Lastly, we use this upper bound to effectively trim classifiers with branch-and-bound search.

Finally, with evaluation on real-world data, we show that our approach finds robust trimmings and demonstrate the relationship between robustness and accuracy. We also illustrate the importance of optimizing the threshold for both classification similarity and efficiency of search. Moreover, we show that our trimming approach consistently returns a classifier that is significantly more similar to the original classifier than selecting features based on information gain.

5.2 Expected Classification Agreement

A binary Bayesian classifier is a tuple $\alpha = (C, \mathbf{F}, T)$, where C is a binary class variable, $\mathbf{F}$ are (possibly multi-valued) features, and T is a threshold. On a joint probability distribution $\Pr(.)$ over variables C and $\mathbf{F}$, the classification function is

$$C_T(\mathbf{f}) = \begin{cases} 1, & \text{if } \Pr(C = 1|\mathbf{f}) \geq T \\ 0, & \text{otherwise.} \end{cases}$$

For example, with a threshold of 0.5, an instance will be classified into the more probable class after observing its features. Next, we motivate and define our proposed closeness measure between classifiers, quantifying their expected agreement.

5.2.1 Example and Motivation

Consider the scenario shown in Figure 5.1, where an instructor uses a quiz to test students' knowledge. The quiz contains three independent questions: Q_1 is strongly indicative of being knowledgeable, Q_2 is an easy question, and Q_3 is a hard question (only 40% of the knowledgeable students answer it correctly). The subject of this quiz is quite difficult, and only 10% of the students are expected to master it, as reflected by the prior on class variable C. Hence, the instructor sets a lenient threshold of $T = 0.07$ to avoid failing students who may have grasped the subject. According to this classifier, a student will pass the quiz precisely when their answer matches one of the following three (out of eight) outcomes: $\{Q_1 = +, Q_2 = +, Q_3 = +\}$, $\{Q_1 = +, Q_2 = +, Q_3 = -\}$, and $\{Q_1 = +, Q_2 = -, Q_3 = +\}$. Moreover, the probability of seeing one of these outcomes is 32%: the fraction of students that are expected to pass the quiz. Suppose now that we drop questions Q_1 and Q_2, relying solely on question Q_3 to evaluate students (using the same threshold). Since $\Pr(C = +|Q_3)$ is always greater than $T = 0.07$, all students will pass the quiz, completely ignoring the test results. Alternatively, we can make more intuitive use of the test question and pass only the students who answered Q_3 correctly. This is equivalent to comparing $\Pr(C = +|Q_3)$ against a new

threshold of $T = 0.15$. Using this new threshold, we will now obtain the same student assessment on five test outcomes,[2] whose probabilities add up to 75%. This is the expected classification agreement (ECA). In particular, we say that the two classifiers $\alpha = (C, \{Q_1, Q_2, Q_3\}, 0.07)$ and $\beta = (C, \{Q_3\}, 0.15)$ have an ECA of 75%.

5.2.2 Formalization

We now formalize the notion of ECA and classifier trimming.

Definition 5.1. *Let $\alpha = (C, \mathbf{F}, T)$ be a Bayesian classifier using distribution $\Pr(.)$. The classifier $\beta = (C, \mathbf{F}', T')$ is a* trimming *of α if it uses the same class variable C and distribution $\Pr(.)$ as α, and a subset of its features (i.e., $\mathbf{F}' \subset \mathbf{F}$).*

Definition 5.2. *Let $\alpha = (C, \mathbf{F}, T)$ be a Bayesian classifier and let $\beta = (C, \mathbf{F}', T')$ be one of its trimmings. The* expected classification agreement (ECA) *between these classifiers is:*

$$\mathrm{ECA}(\alpha, \beta) = \sum_{\mathbf{f}} [C_T(\mathbf{f}) = C_{T'}(\mathbf{f}')] \cdot \Pr(\mathbf{f}).$$

Here, $\mathbf{f}'$ is the subset of instantiation $\mathbf{f}$ pertaining to variables in $\mathbf{F}'$, and $[.]$ is an indicator function (evaluates to 1 when its argument is true and to 0 otherwise).

Section 5.1 asks to compare trimmings of classifier α in Figure 5.1. The first trimming has $\mathrm{ECA}(\alpha, (C, \{Q_1, Q_3\}, 0.10)) = 91\%$ while the second has $\mathrm{ECA}(\alpha, (C, \{Q_2, Q_3\}, 0.30)) = 68\%$.

We are now ready to define the *classifier trimming problem* more formally. The input to this problem is a binary Bayesian classifier $\alpha = (C, \mathbf{F}, T)$, a positive cost for each feature in $\mathbf{F}$, and a budget B. The output is a subset of features $\mathbf{F}^\star \subseteq \mathbf{F}$ whose sum of costs is at most B and a threshold $T^\star$, leading to a trimmed classifier $\beta^\star = (C, \mathbf{F}^\star, T^\star)$ that maximizes the ECA with α:

[2] The two classifiers will disagree on $\{Q_1 = +, Q_2 = +, Q_3 = -\}$, $\{Q_1 = -, Q_2 = +, Q_3 = +\}$ and $\{Q_1 = -, Q_2 = -, Q_3 = +\}$.

Algorithm 4 ECA-TRIM$(\mathbf{I}, \mathbf{E}, b)$

Input: α : Bayesian classifier $(C, \mathbf{F}, T)$; B : budget
Data: $\mathbf{I} \leftarrow \emptyset$, $\mathbf{E} \leftarrow \emptyset$: set of included/excluded features; $b \leftarrow B$: remaining budget; $\mathbf{F}^\star, M^\star, T^\star$: optimal subset, MAA value, and threshold
Output: Optimal trimmed classifier $\beta^\star = (C, \mathbf{F}^\star, T^\star)$

if $b \geq 0$ **then**
  $(m, T_m) \leftarrow \mathrm{MAA}(\mathbf{I})$
  if $m > M^\star$ **then** $M^\star \leftarrow m$; $\mathbf{F}^\star \leftarrow \mathbf{I}$; $T^\star \leftarrow T_m$
if $\min_{F \in \mathbf{F} \setminus (\mathbf{I} \cup \mathbf{E})} \mathrm{cost}(F) \leq b$ **then**
  $m \leftarrow \mathrm{UB}(\mathbf{F} \setminus \mathbf{E})$
  if $m \leq M^\star$ **then return**
  $F \leftarrow$ a feature from $\mathbf{F} \setminus (\mathbf{I} \cup \mathbf{E})$
  ECA-TRIM$\big(\mathbf{I} \cup \{F\}, \mathbf{E}, b - \mathrm{cost}(F)\big)$
  ECA-TRIM$\big(\mathbf{I}, \mathbf{E} \cup \{F\}, b\big)$

$\beta^\star = \arg\max_\beta \mathrm{ECA}(\alpha, \beta)$. I.e., we wish to find a solution to the following optimization problem:

$$\mathrm{ECA}^\star = \max_{\mathbf{F}' \subseteq \mathbf{F}} \max_{T'} \mathrm{ECA}(\alpha, (C, \mathbf{F}', T')) \quad \text{s.t.} \sum_{F' \in \mathbf{F}'} \mathrm{cost}(F') \leq B$$

This problem can alternatively be described as feature selection using the following criterion.

Definition 5.3. *Let $\alpha = (C, \mathbf{F}, T)$ be a Bayesian classifier. The* maximum achievable agreement (MAA) *for feature subset $\mathbf{F}' \subseteq \mathbf{F}$ is defined as:*

$$\mathrm{MAA}_\alpha(\mathbf{F}') = \max_{T'} \mathrm{ECA}(\alpha, (C, \mathbf{F}', T')).$$

The $\mathrm{MAA}_\alpha(\mathbf{F}')$ corresponds to the maximum ECA that is achievable by a trimmed classifier with features $\mathbf{F}'$. Hence, the classifier trimming problem reduces to searching for the subset of features that fits within the budget and maximizes the MAA. We will drop the subscript α when clear from context.

5.3 Searching for an Optimal Trimming

This section describes our approach to search for an optimal trimming of Bayesian classifiers, or equivalently, selecting a feature subset with optimal MAA.[3] Our approach is based on a branch-and-bound search algorithm similar to Narendra and Fukunaga [1977] and Kolesar [1967]. As shown in Algorithm 4, we run a depth-first search through a binary tree where each node is branched into two nodes: one that includes and one that excludes a feature. Each node then represents the set of features that are included by the path from the root to that node. The algorithm computes the MAA at each node if the represented feature subset fits within the budget, keeping track of the best subset and its MAA at each point in search, as in Lines 1–3. In particular, this means that we compute the MAA even if the subset does not exhaust the budget, because MAA does not necessarily increase as the subset size grows.

The essence of the algorithm is pruning subtrees without affecting the optimality of the solution. Suppose given any node, we know the largest value of MAA that its descendants can achieve (UB). Then we can safely prune the subtree rooted at that node if the bound does not exceed the current best score (Line 6). Formally, let $\mathbf{E}$ be the set of features that were excluded by the path to a certain node. Each descendant node will then represent a subset of $\mathbf{F} \setminus \mathbf{E}$. Hence, an upper bound on MAA for all subsets of $\mathbf{F} \setminus \mathbf{E}$ would allow pruning of intermediate nodes in the search tree.

5.3.1 Maximum Potential Agreement

We now introduce an upper bound for the MAA and show how it can be used in the search for an optimal trimming.

Definition 5.4. *Consider a Bayesian classifier* $\alpha = (C, \mathbf{F}, T)$ *. Let* $\mathbf{F}' \subseteq \mathbf{F}$ *be a subset of its*

[3]Code available at `https://github.com/UCLA-StarAI/TrimBN`.

features, and let $\mathbf{R} = \mathbf{F} \setminus \mathbf{F}'$. *The* maximum potential agreement *(MPA) is*

$$\mathrm{MPA}_\alpha(\mathbf{F}') = \sum_{\mathbf{f}'} \max_c \left\{ \sum_{\mathbf{r}} [C_T(\mathbf{f}'\mathbf{r}) = c] \cdot \Pr(\mathbf{f}'\mathbf{r}) \right\}.$$

Intuitively, the MPA is the expected agreement between a Bayesian classifier α and a hypothetical classifier γ that classifies an instance $\mathbf{f}'$ into the class that is more likely after observing the remaining features in $\mathbf{R}$. Note that such classifier γ is not a Bayesian classifier as it does not test the posterior $\Pr(c|\mathbf{f}')$ against a threshold. However, the MPA is still a useful computational tool due to its relationship to the MAA.

Proposition 5.1. *The MPA is an upper bound on the MAA:* $\mathrm{MAA}_\alpha(\mathbf{F}') \leq \mathrm{MPA}_\alpha(\mathbf{F}')$.

Proof. Let $\mathbf{R} = \mathbf{F} \setminus \mathbf{F}'$. The proposition follows from the definitions of MAA and MPA as follows:

$$\mathrm{MAA}_\alpha(\mathbf{F}') = \max_{T'} \sum_{\mathbf{f}'\mathbf{r}} [C_T(\mathbf{f}'\mathbf{r}) = C_{T'}(\mathbf{f}')] \cdot \Pr(\mathbf{f}'\mathbf{r}) \leq \sum_{\mathbf{f}'} \max_{T'} \sum_{\mathbf{r}} [C_T(\mathbf{f}'\mathbf{r}) = C_{T'}(\mathbf{f}')] \cdot \Pr(\mathbf{f}'\mathbf{r})$$

$$= \sum_{\mathbf{f}'} \max_c \left\{ \sum_{\mathbf{r}} [C_T(\mathbf{f}'\mathbf{r}) = c] \cdot \Pr(\mathbf{f}'\mathbf{r}) \right\} = \mathrm{MPA}_\alpha(\mathbf{F}'). \tag{5.1}$$

Equation 5.1 comes from the fact that, for a fixed instance $\mathbf{f}'$, choosing a threshold T' is equivalent to choosing to classify that instance positively or negatively. □

In addition, the MPA is monotonically increasing, a property that we utilize later in the proposed algorithms.

Proposition 5.2. *For any* $\mathbf{F}_1 \subseteq \mathbf{F}_2$, $\mathrm{MPA}_\alpha(\mathbf{F}_1) \leq \mathrm{MPA}_\alpha(\mathbf{F}_2)$.

Proof. Let $\mathbf{R}_1 = \mathbf{F} \setminus \mathbf{F}_1$ and $\mathbf{R}_2 = \mathbf{F} \setminus \mathbf{F}_2$. Say $\mathbf{G} = \mathbf{F}_2 \setminus \mathbf{F}_1 = \mathbf{R}_1 \setminus \mathbf{R}_2$. Then,

$$\mathrm{MPA}(\mathbf{F}_1) = \sum_{\mathbf{f}_1} \max_c \left\{ \sum_{\mathbf{g}\mathbf{r}_2} [C_T(\mathbf{f}_1\mathbf{g}\mathbf{r}_2) = c] \cdot \Pr(\mathbf{f}_1\mathbf{g}\mathbf{r}_2) \right\} \tag{5.2}$$

$$\leq \sum_{\mathbf{f}_1\mathbf{g}} \max_c \left\{ \sum_{\mathbf{r}_2} [C_T(\mathbf{f}_1\mathbf{g}\mathbf{r}_2) = c] \cdot \Pr(\mathbf{f}_1\mathbf{g}\mathbf{r}_2) \right\} = \mathrm{MPA}(\mathbf{F}_2). \tag{5.3}$$

Equation 5.2 is from the assumption that $\mathbf{R}_1 = \mathbf{G} \cup \mathbf{R}_2$, and Equation 5.3 follows from the fact that sum of maxima is an upper bound on the maximum of sums. □

These two propositions together imply that the MPA of $\mathbf{F}'$ also upper-bounds the MAA of all subsets of $\mathbf{F}'$.

Corollary 5.2.1. *For any* $\mathbf{F}_1 \subseteq \mathbf{F}_2$, $\mathrm{MAA}_\alpha(\mathbf{F}_1) \leq \mathrm{MPA}_\alpha(\mathbf{F}_2)$.

Therefore, we can use the MPA as an upper bound on the MAA of a node's descendants in the branch-and-bound search algorithm for optimal trimming.

Lastly, we provide an observation that leads to a computational gain, especially in the case of naive Bayes models.

Proposition 5.3. *If* $\mathbf{F}'$ *and* $\mathbf{F} \setminus \mathbf{F}'$ *are independent given* C*, then* $\mathrm{MAA}_\alpha(\mathbf{F}') = \mathrm{MPA}_\alpha(\mathbf{F}')$.

Proof. See Section A.4. □

The above property is useful because it is generally easier to compute $\mathrm{MPA}(\mathbf{F}')$ than $\mathrm{MAA}(\mathbf{F}')$, as we can maximize each instantiation $\mathbf{f}'$ separately. Moreover, in naive Bayes models, the quantity MAA that we wish to optimize is now monotonic. Thus, we need to compute this quantity only for those subsets that exhaust the budget, instead of every subset that fits within budget.

5.4 Probabilistic Reasoning Algorithms

This section describes the algorithms to compute the expected classification agreement (ECA), maximum achievable agreement (MAA), and maximum potential agreement (MPA) on probabilistic circuits. Recall from Section 2.2 that Bayesian networks can be compiled into tractable circuit

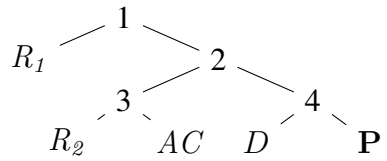


Figure 5.2: Constrained vtree where $\mathbf{F} = \{R_1, R_2, AC\}$ and $\mathbf{F}' = \{R_1\}$. The $\mathbf{F}'$-constr. node is 2 and $\mathbf{F}$-constr. node is 4.

representations, and thus we can use the algorithms presented in this section to run inference on Bayesian networks as well.[4]

5.4.1 Computing the ECA using Constrained Circuits

Our approach to compute the ECA using probabilistic circuits is shown in Algorithm 5. It requires a special type of PSDDs that are normalized for constrained vtrees [Oztok et al., 2016].

Definition 5.5. *A vtree node v is $\mathbf{X}$-constrained, denoted $v_{\mathbf{X}}$, iff it appears on the right-most path of the vtree and $\mathbf{X}$ is exactly the set of variables outside v. A vtree is $\mathbf{X}$-constrained iff it has an $\mathbf{X}$-constrained node, and a PSDD is $\mathbf{X}$-constrained iff it is normalized for an $\mathbf{X}$-constrained vtree.*

The ECA between a classifier and its trimming—using the variables $\mathbf{F}$ and $\mathbf{F}'$, respectively—can be computed efficiently for a PSDD that is $\mathbf{F}$-constrained as well as $\mathbf{F}'$-constrained. Figure 5.2 depicts an example of such vtree. Note that the $\mathbf{F}'$-constrained node $v_{\mathbf{F}'}$ is always an ancestor of the $\mathbf{F}$-constrained node $v_{\mathbf{F}}$.

To understand the algorithm, first consider the following alternate expression for the ECA between $\alpha = (C, \mathbf{F}, T)$ and $\beta = (C, \mathbf{F}', T')$.

$$\mathrm{ECA}(\alpha, \beta) = \sum_{\mathbf{f}' : C_{T'}(\mathbf{f}')=1} \Pr(C_T(\mathbf{f}) = 1 | \mathbf{f}') \cdot \Pr(\mathbf{f}') + \sum_{\mathbf{f}' : C_{T'}(\mathbf{f}')=0} \Pr(C_T(\mathbf{f}) = 0 | \mathbf{f}') \cdot \Pr(\mathbf{f}'). \quad (5.4)$$

[4]Even though compiling the circuit is computationally heavy in general, computing the ECA and hence the MPA is efficient once we have successfully compiled the circuit. Moreover, we can sometimes efficiently compile certain networks (e.g. high treewidth) in which traditional inference techniques become infeasible [Choi et al., 2013].

Algorithm 5 $\mathrm{ECA}(\alpha, \beta)$

Input: Classifiers $\alpha = (C, \mathbf{F}, T)$ and $\beta = (C, \mathbf{F}', T')$; an $\mathbf{F}$- and $\mathbf{F}'$-constrained PC $\mathcal{C}$
Output: $\mathrm{ECA}(\alpha, \beta)$

1: $\mathsf{N} \leftarrow$ FEEDFORWARDORDER($\mathcal{C}$)
2: **for each** $n \in \mathsf{N}$ **do**
3: **if** n is a leaf node **then**
4: **if** $n = [C = 0]$ **then** $r_1(n) \leftarrow 0$ **else** $r_1(n) \leftarrow 1$
5: **else if** n is a product node **then**
6: $r_i(n) \leftarrow \prod_{c \in \mathsf{ch}(n)} r_i(c)$ for $i = 1, 2, 3$
7: **else** ▷ a sum node
8: $r_i(n) \leftarrow \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} r_i(c)$ for $i = 1, 2, 3$
9: **if** n is $\mathbf{F}$-constrained **then**
10: **if** $r_1(n) \geq T$ **then** $r_2(n) \leftarrow 1$ **else** $r_2(n) \leftarrow 0$
11: **else if** n is $\mathbf{F}'$-constrained **then**
12: **if** $r_1(n) \geq T'$ **then** $r_3(n) \leftarrow r_2(n)$ **else** $r_3(n) \leftarrow 1 - r_2(n)$
13: **return** $r_3(\text{root})$

Here, $\Pr(C_T(\mathbf{f}) = 1|\mathbf{f}')$ is the expected probability that observing all features in $\mathbf{F}$ will lead to a positive classification, given that $\mathbf{f}'$ is already fixed. In other words,

$$\Pr(C_T(\mathbf{f}) = 1|\mathbf{f}') = \sum_{\mathbf{f}:\mathbf{f}' \subset \mathbf{f}} [C_T(\mathbf{f}) = 1] \Pr(\mathbf{f}|\mathbf{f}').$$

Algorithm 5 computes the ECA in a single feedforward pass through the circuit, caching three values at each node. First, note that $r_1(.)$ simply corresponds to evaluating the PC with $C = 1$ as evidence: that is, $r_1(n) = \Pr_n(C = 1) = \Pr(C = 1|\gamma_n)$. Recall that γ_n denotes the context of node n (Definition 3.2). At each $\mathbf{F}$-constrained node, its context corresponds to some assignment $\mathbf{f}$; then the condition in Line 10 corresponds to checking whether $\Pr(C = 1|\mathbf{f}) \geq T$, or equivalently $C_T(\mathbf{f}) = 1$. Therefore, $r_2(n)$ contains the expected probability of positive classification given by the distribution at node n: $\Pr_n(C_T(\mathbf{f}) = 1)$. Similarly, Line 12 checks $C_{T'}(\mathbf{f}')$ for some $\mathbf{f}'$ and propagates the expected probability that observing the remaining features will lead to the same

classification. In other words, we have:

$$r_3(n) = \sum_{\mathbf{f}'} \Pr_n(C_T(\mathbf{f}) = C_{T'}(\mathbf{f}'))$$

At the root node, this exactly corresponds to the ECA between α and β.

5.4.2 Computing the MPA

We now describe how we compute the MPA at each search step. First, the MPA can be expressed as follows:

$$\mathrm{MPA}(\mathbf{F}') = \sum_{\mathbf{f}'} \max\Big(\Pr(C_T(\mathbf{f}) = 1|\mathbf{f}'), 1 - \Pr(C_T(\mathbf{f}) = 1|\mathbf{f}')\Big) \cdot \Pr(\mathbf{f}'). \tag{5.5}$$

Note that $\Pr(C_T(\mathbf{f}) = 1|\mathbf{f}')$ is the expected probability of positive class given $\mathbf{f}'$, which was a central component to computing the ECA. Moreover, the marginal probabilities of $\mathbf{f}'$ are also computed as part of the feedforward evaluation for the ECA. Thus, we can exploit this connection to compute the MPA: it is as straightforward as replacing Line 12 in Algorithm 5 to propagate the maximum between $r_2(n)$ and $1 - r_2(n)$.

With the ability to compute the MPA, we can now search for optimal trimmings of naive Bayes classifiers. The condition in Proposition 5.3 holds for all feature subsets of a naive Bayes model, and thus the MAA of a subset is always equal to its MPA. An optimal trimming is then found as shown in Algorithm 4 where both the upper bound and value of MAA are computed using the MPA algorithm described before.

5.4.3 Computing the MAA

Searching for an optimal trimming of arbitrary Bayesian network classifiers requires the computation of MAA, which involves tuning the trimmed classifier's threshold to maximize the ECA. First, we utilize the observation that a change in threshold affects the value of ECA only if the class

Algorithm 6 COMPUTE-MAA

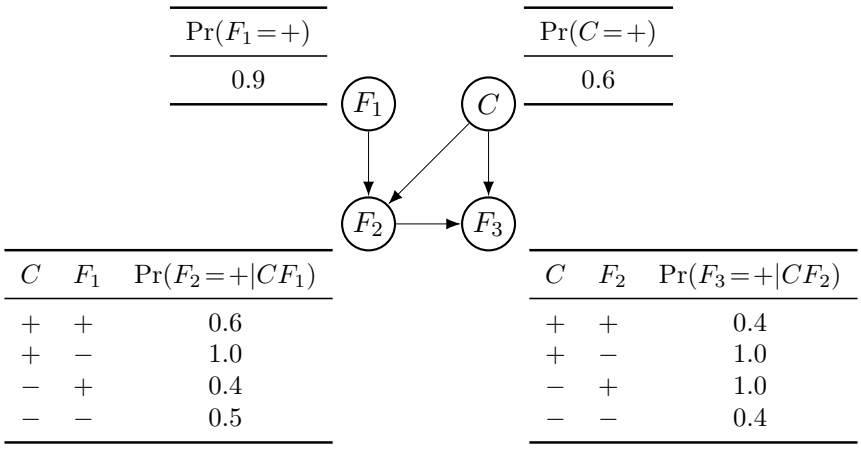
Input: α : Bayesian network classifier $(C, \mathbf{F}, T)$; $\mathbf{F}' \subset \mathbf{F}$
Data: CPR$(i) \leftarrow \Pr(C = 1|\mathbf{f}'_i)$; MAR$(i) \leftarrow \Pr(\mathbf{f}'_i)$; POS$(i) \leftarrow \Pr(C_T(\mathbf{f}) = 1|\mathbf{f}'_i)$ for all i
Output: The score $\mathrm{MAA}(\mathbf{F}')$ and the optimal threshold T'

1: Sort instances $\mathbf{f}'_i$ in nondecreasing order of CPR(i)
2: $m \leftarrow \sum_i$ POS$(i) \cdot$ MAR(i); $m^\star \leftarrow m$
3: $t^\star \leftarrow [0,$ CPR$(1)]$
4: **for** i in $1, 2, \ldots$ **do**
5: $\quad m \leftarrow m -$ MAR$(i) \cdot (2$POS(i) $- 1)$
6: $\quad$ **if** $m > m^\star$ **then**
7: $\quad\quad m^\star \leftarrow m$; $t^\star \leftarrow ($CPR$(i),$ CPR$(i+1)]$
8: **return** $\mathrm{MAA}(\mathbf{F}') = m^\star$ and any $T' \in t^\star$

probability given some instance lies on a different side of the threshold after the change. For example, recall the trimmed classifier using only Q_3 from the quiz example in Section 5.2. We showed that a threshold of 0.15 will result in passing only the students who answered Q_3 correctly. In fact, any threshold between $\Pr(C{=}+|Q_3{=}-){=}0.08$ and $\Pr(C{=}+|Q_3{=}+){=}0.18$ results in the same behavior and hence the same ECA. Therefore, the number of threshold values we need to consider is finite and in fact linear in the number of possible instances $\mathbf{f}'$.

Algorithm 6 shows the pseudocode to compute the MAA and the optimal threshold given the cached values from the ECA algorithm as inputs. Starting from $T' = 0$, the threshold is repeatedly incremented to just above the next lowest class probability given some feature instance $\mathbf{f}'$. With each threshold change, the ECA value is also updated by subtracting the expected probability of positive classification given that instance and adding the complement of it, weighted by the marginal probability of that instance, as in Line 5 of Algorithm 6. In other words, that instance $\mathbf{f}'$ is moved from the first sum to the second in Equation 5.4. At the end, the highest value of ECA and its corresponding threshold is reported.

Table 5.1 offers a visualization of the algorithm. Here, we wish to compute $\mathrm{MAA}(\{F_1, F_2\})$ with respect to the Bayesian network classifier $\alpha = (C, \{F_1, F_2, F_3\}, 0.55)$ in Figure 5.3. Each table row corresponds to a feature instance, sorted by the class probability. We consider five different cutoff points, and the ECA value at each cutoff point is the sum of expected probability of positive



C	F_1	$\Pr(F_2=+\|CF_1)$
$+$	$+$	0.6
$+$	$-$	1.0
$-$	$+$	0.4
$-$	$-$	0.5

C	F_2	$\Pr(F_3=+\|CF_2)$
$+$	$+$	0.4
$+$	$-$	1.0
$-$	$+$	1.0
$-$	$-$	0.4

Figure 5.3: A Bayesian network over features $\{F_1, F_2, F_3\}$ and class C.

$\{F_1, F_2\}$	$\Pr(c\|F_1F_2)$	$+$ class pr.	$-$ class pr.
$\{-,+\}$	0.75	0.04	0.04
$\{+,+\}$	0.69	0.20	0.27
$\{+,-\}$	0.50	0.30	0.13
$\{-,-\}$	0.00	0.00	0.02

Table 5.1: Table to calculate the $\mathrm{MAA}(\{F_1, F_2\})$

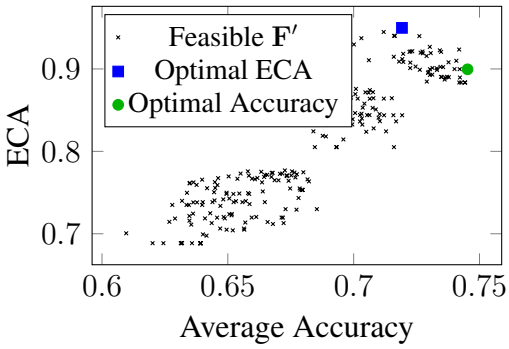
class for instances above the line and the expected probability of negative class below the line. In this case, $\mathrm{MAA}(\{F_1, F_2\}) = 0.56$ with the optimal threshold $T^\star \in (0, 0.50]$, indicated by a dotted line in the table.

5.5 Empirical Evaluation

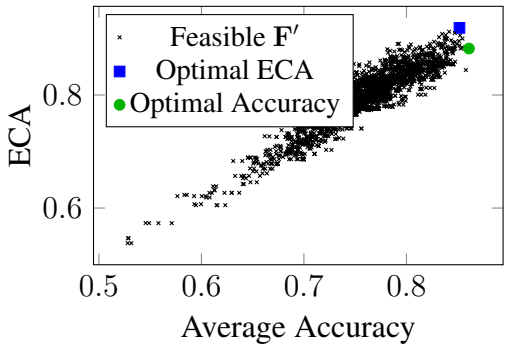
We now empirically evaluate our proposed algorithms on several naive Bayes and general Bayesian network benchmarks.

5.5.1 Accuracy vs. Agreement

We evaluate our method on real-world datasets from the UCI repository [Bache and Lichman, 2013]. We randomly split each dataset into 80/20 train and test sets and learn a naive Bayes classifier using



(a) ECA and average accuracy for `pima`



(b) ECA and average accuracy for `heart`

		Agreement	Accuracy
`pima`	Opt. ECA	0.9863	0.7123
	Opt. Acc.	0.9452	0.7260
`heart`	Opt. ECA	0.9245	0.8491
	Opt. Acc.	0.9057	0.7925

(c) Test agreement and accuracy

Figure 5.4: (a),(b) ECA and average accuracy achieved by feasible feature subsets. (c) evaluation of subsets with highest ECA and accuracy.

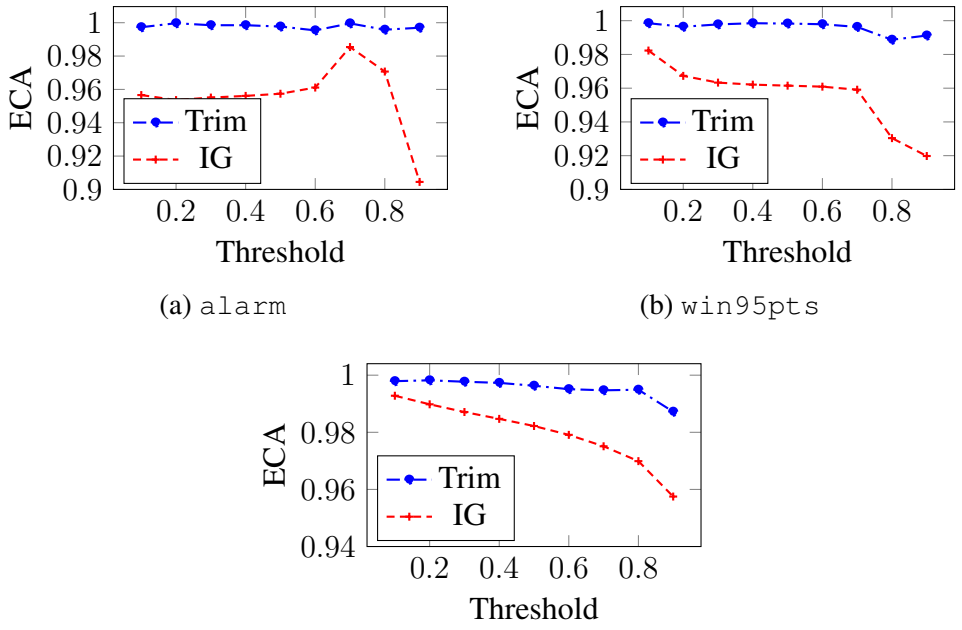
the training set. With the budget set as half the number of features and threshold as 0.5, we compute the ECA of each feasible feature subset. In addition, we compute the average classification accuracy of each feature subset using 10-fold cross validation on the training set.

Figure 5.4 shows the ECA and average accuracy achieved by each feasible subset, and the subsets with highest ECA and highest accuracy are highlighted. We can see that optimizing the ECA tends towards higher accuracy. More interestingly, we can observe a Pareto frontier where one cannot increase the ECA without sacrificing average accuracy, and vice versa. This suggests that one may need to make a tradeoff between classifier agreement (i.e., robustness) and accuracy when selecting features. Moreover, we evaluate the subsets with highest ECA and accuracy on the test set and report their empirical classification agreement and accuracy in Figure 5.4c. The subset chosen for optimal ECA on the training set also achieves high classification agreement on the test set. Surprisingly, on network `heart`, it also achieves higher test accuracy than the subset

with the highest average cross-validation accuracy on the training set. A possible explanation is that choosing subsets based on their cross-validation classification accuracy does not generalize well to the test set. It may introduce additional overfitting that ECA does not suffer from: if the original model generalizes well, we also expect our trimmed classifier to generalize well. We also evaluated accuracy and agreement of feature selection by information gain, but it neither outperformed optimizing the ECA nor the average cross-validation accuracy. In addition, our method achieves higher accuracy on most splits of the `heart` data, which suggests that this may be a property of the dataset. In particular, the average cross-validation accuracy of the original full classifier for `pima` was approximately 0.720, which was lower than the average accuracy of about 21% of the candidate subsets. As our method optimizes for agreement with this original classifier, which has relatively low accuracy, the resulting trimming may have lower accuracy than if we were to actively optimize for average accuracy. On the other hand, the original classifier for `heart` had average accuracy 0.866, which was lower than only 2% of the candidate subsets. Hence, in this case, optimizing the ECA to closely mimic the original classifier's behavior also results in relatively high classification accuracy.

5.5.2 Trimming General Networks

Next, we evaluate the quality of trimmed classifiers on general Bayesian networks from the UAI 2008 evaluation and a tree-augmented naive Bayes model for mammography reports [Gimenez et al., 2014]. We run our method with T in $\{0.1, 0.2, \ldots, 0.9\}$ and the budget set to ⅓ the number of features. For the UAI networks, we randomly chose a root node to be the class variable and used the set of all leaf nodes as the feature set $\mathbf{F}$. Each setting was repeated for three randomly selected class variables. For the mammography network, we used the (root) decision node as the class variable and chose 17 out of the 20 variables to be the feature set. Training data was not available for these networks, so we compare against feature selection by information gain instead of classification accuracy. Figure 5.5 highlights the results. The trimmed classifier by our algorithm consistently achieves higher expected classification agreement, demonstrating that robustness is



(a) `alarm` (b) `win95pts`

(c) `mammography`

Figure 5.5: Comparing ECA of features selected by classifier trimming and information gain

not easily achieved by other feature selection methods. We also want to stress that the features selected using information gain differ for different class variables, but stay the same across different initial threshold values. On the other hand, our algorithm is sensitive to the original threshold, and thus results in trimmed classifiers with similar behavior as the original classifier with a particular threshold.

5.6 Conclusion

This chapter developed a novel operator on Bayesian classifiers: to trim the set of features to fit within a budget, while simultaneously adjusting the classification threshold. Our objective was to optimize the expected classification agreement between the original classifier and its trimmed counterpart. By analyzing the properties of classifier agreement and its maximum potential agreement, we developed a branch-and-bound search algorithm to find optimal trimmings. Experiments on naive and general Bayesian networks demonstrated the effectiveness of our approach in finding

robust trimmings of classifiers, especially compared to optimizing more traditional objectives such as expected SDP and information gain.

CHAPTER 6

Probabilistic Inference by Circuit Transformations

A standard technique for exact inference on probabilistic circuits is to enforce certain structural constraints such that the query of interest can be computed efficiently, often simply through a feedforward pass through the circuit. The inference algorithms discussed in the previous chapters also followed such technique. For example, marginal probabilities can be computed in a feedforward evaluation of smooth and decomposable PCs; and the expected classification accuracy can be computed in linear time for deterministic and constrained PCs.

Harder queries tend to require more restrictive structural properties, and enforcing them on a given PC can be highly expensive. Alternatively, learning probabilistic circuit structures with more constraints might degrade the likelihood of the learned models, compared to learning PCs with less restrictive conditions and thus more freedom and flexibility in exploring the candidate structures.

Instead, this chapter proposes a new approach to probabilistic circuit inference: *through a series of circuit transformations*. A circuit transformation, or operation, manipulates the structure of a given PC, which may or may not change the distribution represented by the PC. Here, we show how the transformations *prune* and *split* can be performed iteratively to solve marginal MAP queries. We also briefly discuss decomposing many queries into pipelines of atomic circuit operations, which allows us to easily derive the tractability conditions and inference algorithms for new queries. Through this new approach, I aim to push the boundaries of our probabilistic reasoning framework and hope to more easily address various problems in the field of trustworthy AI.

6.1 Marginal MAP

Perhaps the most widely supported queries for tractable inference by different kinds of PCs are: marginal inference, which computes the probability of a partial assignment; and the most probable explanations (MPE),[1] which computes for a given partial assignment (or evidence) the most likely state of all the remaining variables. However, many related inference tasks remain hard even on those PCs tractable for marginals and MPE [Rahman et al., 2021; Rouhani et al., 2018]. In particular, marginal MAP (maximum a posteriori hypothesis) is a closely related problem that still appears to be hard for most probabilistic circuits, despite being used in many applications including image segmentation, planning, and diagnosis, among others [Lee et al., 2014; Kiselev and Poupart, 2014; Bioucas-Dias and Figueiredo, 2016]. A marginal MAP (MMAP) problem, unlike MPE, computes the most likely state of a subset of variables, while marginalizing out the others.

Definition 6.1. *Suppose $p(\mathbf{X})$ is a probability distribution over a set of variables $\mathbf{X}$ which is partitioned into three subsets $\mathbf{Q}$, $\mathbf{E}$, and $\mathbf{H}$, referred to as the query, evidence, and hidden variables, respectively. Given some evidence $\boldsymbol{e} \in \mathsf{val}(\mathbf{E})$, the marginal MAP problem $\mathsf{MMAP}(\mathbf{Q}, \boldsymbol{e})$ is defined as follows:*

$$\arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} p(\boldsymbol{q}, \boldsymbol{e}) \;=\; \arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} \sum_{\boldsymbol{h} \in \mathsf{val}(\mathbf{H})} p(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{e}).$$

Note that if $\mathbf{H}$ is empty, this corresponds to an MPE (most probable explanations) problem.

Although these queries appear closely related, a PC that can tractably solve both marginals and MPE queries does not necessarily solve the marginal MAP tractably. In fact, exactly solving marginal MAP is known to be NP-hard, even for tractable PCs [de Campos, 2011]. This remains to be the case when solving it approximately [Conaty et al., 2017; Mei et al., 2018].

[1] MPE is sometimes referred to as MAP (maximum a posteriori hypothesis). To avoid confusion, this chapter will use the terms MPE and marginal MAP.

6.1.1 Exact Solvers

Most existing marginal MAP solvers on PCs are based on variations of branch-and-bound search [Mei et al., 2018; Huang et al., 2006], as has been the case for exact marginal MAP solvers for probabilistic graphical models [Park and Darwiche, 2002; Marinescu and Dechter, 2009; Marinescu et al., 2018]. Alternatively, probabilistic circuits satisfying more restrictive structural constraints can support efficient inference of marginal MAP and related queries [Oztok et al., 2016; Choi et al., 2017]. These structural constraints can be generalized into the notion of $\mathbf{Q}$-*determinism* [Choi et al., 2020b].

Definition 6.2. *Suppose $\mathcal{C}$ is a PC over variables $\mathbf{X}$ and let $\mathbf{Q} \subseteq \mathbf{X}$ be a subset. A sum node in $\mathcal{C}$ is $\mathbf{Q}$-deterministic if computing the marginal probability for any partial assignment $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ makes at most one of its children evaluate to a nonzero output. A PC $\mathcal{C}$ is $\mathbf{Q}$-deterministic if all sum nodes containing variables in $\mathbf{Q}$ are $\mathbf{Q}$-deterministic.*

Then, solving a marginal MAP problem $\mathsf{MMAP}(\mathbf{Q}, \boldsymbol{e})$ of a $\mathbf{Q}$-deterministic PC simply amounts to evaluating the circuit bottom-up similar to computing a marginal, except that every sum node that contains a variable in $\mathbf{Q}$ takes the weighted maximum of its inputs, instead of the weighted sum.

As one may intuit from the complexity of marginal MAP, enforcing this structural constraint on an arbitrary PC is an intractable task in general, as we also later demonstrate empirically. Furthermore, even if one somehow learns or constructs a PC that satisfies $\mathbf{Q}$-determinism, this would support tractable marginal MAP only for this specific $\mathbf{Q}$. This is clearly infeasible in applications where one wishes to answer different marginal MAP queries using the probabilistic model.

This chapter proposes a novel approach to marginal MAP inference: probabilistic circuit transformations. In particular, I will show that large parts of the circuit may be irrelevant to the marginal MAP problem at hand, and thus can be pruned away without affecting the solution. This in a sense "specializes" the PC to a particular MMAP instance and makes it more amenable to solving. I then develop an efficient algorithm to determine which parts of the circuit can be safely pruned,

using a novel edge bound. Lastly, I propose an exact MMAP solver that leverages this pruning algorithm and iteratively transforms the PC structure until the MMAP solution can be easily read from it. Empirical evaluation on real-world benchmark datasets shows that this approach can solve more marginal MAP instances with faster run time than existing solvers.

6.2 Circuit Pruning For Marginal MAP

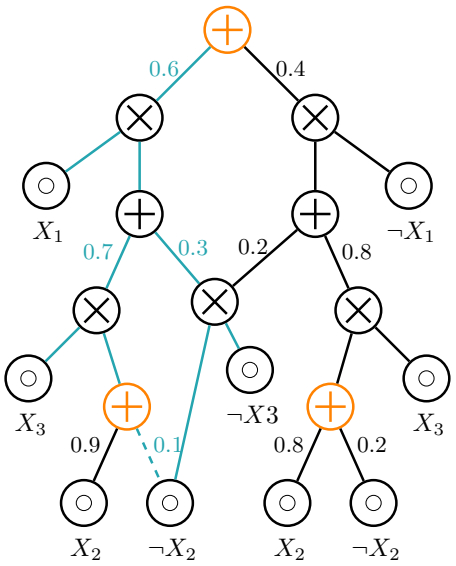
In the following sections, we assume a PC that satisfies smoothness and decomposability. Moreover, for simplicity of exposition, we consider only the marginal MAP problems without any evidence. This is because a given evidence can be incorporated into the PC by setting the leaf nodes (just like for computing marginals), and then we can equivalently solve the marginal MAP problem with no evidence on the resulting PC.

We now describe the main contribution behind our proposed marginal MAP solver: pruning parts of a probabilistic circuit without affecting its MMAP solution. This is motivated by two key observations.

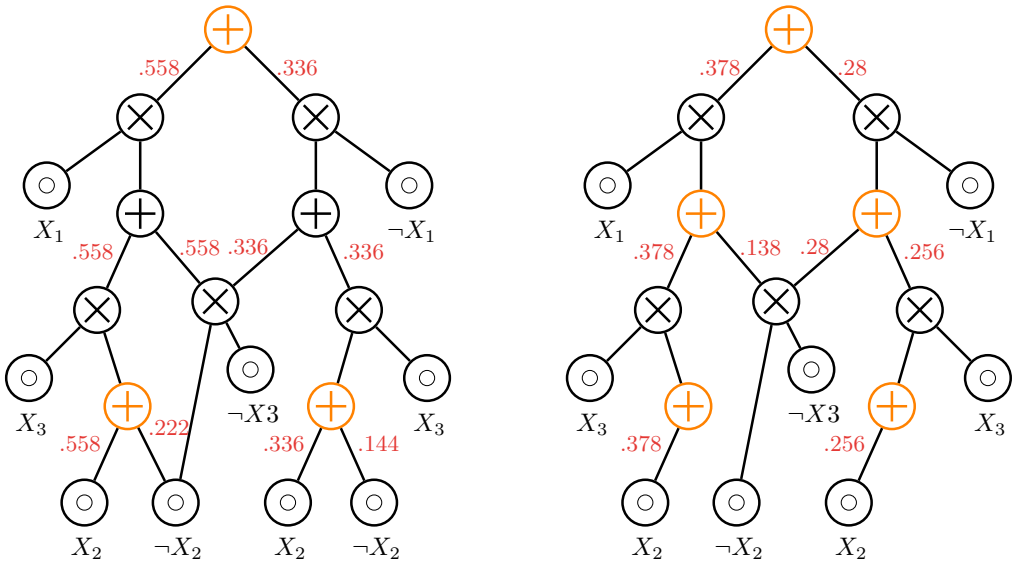
6.2.1 Motivation

Consider the following two observations.

(i): Computing the marginal probability of any partial assignment $\boldsymbol{q}$ is equivalent to evaluating a sub-circuit in which every $\mathbf{Q}$-deterministic sum node has one input. In other words, the sub-circuit for $\boldsymbol{q}$ includes the parts of the PC that are used or "activated" when computing the marginal of $\boldsymbol{q}$. Let us call this the $\boldsymbol{q}$-subcircuit and denote it by $\mathcal{C}'_{\boldsymbol{q}}$. Formally, an edge (n, c) is said to be in the $\boldsymbol{q}$-subcircuit if $\boldsymbol{q} \in \gamma_{(n,c)}\big|_{\mathbf{Q}}$; i.e., the context of (n, c) (Definition 3.2) reduced to variables in Q contains the assignment $\boldsymbol{q}$. We illustrate this with the example PC in Figure 6.1a. Suppose $\mathbf{Q} = \{X_1, X_2\}$ and we wish to compute the marginal probability of $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$. Recall from Chapter 2 that this corresponds to setting the input units for $\neg X_1$ and X_2 to 0 and



(a) Example PC



(b) PC before and after pruning. Sum edges are labeled with edge bounds.

Figure 6.1: A smooth and decomposable PC over variables $\{X_1, X_2, X_3\}$. Orange sum nodes are $\mathbf{Q}$-deterministic for $\mathbf{Q} = \{X_1, X_2\}$; blue edges form the sub-circuit for joint assignment $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$.

all others to 1, then evaluating the circuit in a bottom-up fashion. We can quickly check that the output is $0.6 \cdot (0.7 \cdot 0.1 + 0.3) = 0.222$, which is equivalent to simply evaluating the sub-circuit highlighted in blue with its input units set to 1. Moreover, observe that every $\mathbf{Q}$-deterministic sum

node (highlighted in orange) that is included in this sub-circuit has exactly one input.

(ii): If we remove an edge that does not appear in the sub-circuit for any assignment $\boldsymbol{q}$, then the (unnormalized) probability of $\boldsymbol{q}$ is unchanged in the resulting PC. This directly follows from observation (i). For example, removing any non-colored edge from the PC in Figure 6.1a does not affect the marginal for $\boldsymbol{q}$, as defined previously, in the resulting circuit. Moreover, if an edge in the sub-circuit for $\boldsymbol{q}$ is removed, then the probability of $\boldsymbol{q}$ decreases in the resulting PC. Again visiting Figure 6.1a, removing the edge represented by the dashed line will drop the probability of $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$ from 0.222 to $0.6 \cdot 0.3 = 0.18$.

We can apply observations (i) and (ii) to the marginal MAP state, denoted by $\boldsymbol{q}^\star$, to conclude that any edge that does not appear in the $\boldsymbol{q}^\star$-subcircuit (namely the "solution sub-circuit") can be pruned away while keeping the MMAP problem equivalent. That is, removing an edge that is not in the solution sub-circuit will not affect the probability of $\boldsymbol{q}^\star$ but may decrease the probabilities of other assignments to $\mathbf{Q}$; hence, $\boldsymbol{q}^\star$ remains as the solution for marginal MAP problem in the pruned circuit. Solving a MMAP instance by solving the equivalent problem on a pruned circuit can have the following important benefits. First, the complexity of inference algorithms on PCs generally depends on the size of the circuit, and thus reducing the size by pruning edges is desirable. In addition, because pruning as described above keeps the marginal MAP probability while potentially decreasing other marginal probabilities, it effectively increases the gap between the solution and other states. This can not only lead to more iterations of pruning, further specializing the circuit to the MMAP problem, but also arguably make the problem easier to solve. For example, in the extreme case that all edges other than the solution sub-circuit are pruned, the resulting MMAP problem becomes trivial to solve.

Given these benefits, we naturally raise the following question: *can we efficiently determine which edges do not appear in the solution sub-circuit (i.e. $\boldsymbol{q}^\star$-subcircuit)?* The challenge is to do this without knowing a priori the marginal MAP state $\boldsymbol{q}^\star$. In the following section, we propose an algorithm that efficiently computes, for every edge, an upper bound on the output of any sub-circuit that includes the edge, which gives a positive answer to the previous question.

6.2.2 Edge Bounds

We will now define more formally our edge bounds and the algorithm to efficiently compute them.

Definition Abusing notation, let us denote by $\mathsf{MMAP}(\mathbf{Q}|_{(n,c)})$ the largest marginal probability obtainable by an assignment $\boldsymbol{q}$ whose $\boldsymbol{q}$-subcircuit includes the edge (n, c). Formally,

$$\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) := \max_{\boldsymbol{q}:(n,c)\in\mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}). \tag{6.1}$$

Intuitively, this corresponds to a marginal MAP problem where the possible states have been reduced from $\mathsf{val}(\mathbf{Q})$ to those that "activate" the edge (n, c) when computing their marginal probability. Moreover, suppose we define a hypothetical edge from the root to output, denoted $(\cdot, \text{root})$. Then by definition, the MMAP reduced to this edge, i.e. $\mathsf{MMAP}(\mathbf{Q}|_{(\cdot,\text{root})})$ is simply the MMAP problem $\mathsf{MMAP}(\mathbf{Q})$.

For each edge (n, c), we wish to obtain an edge bound $\mathsf{EB}(n, c)$ that satisfies the following:

$$\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) \leq \mathsf{EB}(n, c).$$

Let us also introduce EB for each node n, which may be useful as intermediate quantities as will be apparent later.

$$\mathsf{MMAP}(\mathbf{Q}|_{n}) \leq \mathsf{EB}(n).$$

It is important to note that the edge bound $\mathsf{EB}(n, c)$ is not a bound on some output from the edge or either of the nodes connected by it. Rather, it bounds from above the output of the PC at the root, using the edge to limit the state space. Suppose we are given such edge bound; clearly, if we also have a lower bound on the marginal MAP probability, we can safely prune any edge whose EB is smaller than the given lower bound.

Computing the Edge Bound To develop an edge bound with the properties described above, we first observe that every $\boldsymbol{q}$-subcircuit that includes an edge (n, c) must also include the node n. Then, we can conclude that $\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) \leq \mathsf{MMAP}(\mathbf{Q}|_n)$. Suppose we have an upper bound on the MMAP reduced to node n. Such bound will also be at least as large as the MMAP reduced to edge (n, c), and can be used as edge bound $\mathsf{EB}(n, c)$. However, $\mathsf{EB}(n, c)$ need not be as large as $\mathsf{MMAP}(\mathbf{Q}|_n)$, so there may be some opportunity to tighten the bound going from n to (n, c).

As a base case of the top-down recursion, we need an upper-bound of MMAP at the root. For this, we use the algorithm by Huang et al. [2006], shown in Algorithm 7, which not only computes the upper-bound on marginal MAP at the root node but also bounds the output of every node, via a single feedforward pass on the PC. Formally, for every node $n \in \mathcal{C}$ it computes an upper bound on:

$$\max_{\boldsymbol{q}: n \in \mathcal{C}'_{\boldsymbol{q}}} n(\boldsymbol{q}) = \max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} n(\boldsymbol{q}), \tag{6.2}$$

and stores it in m_n. Let us denote the upper-bound at the root by $\mathsf{m}_{\mathcal{C}}$.

Intuitively, our proposed edge bound $\mathsf{EB}(n, c)$ aims to upper-bound the largest value returned by Algorithm 7 on a $\boldsymbol{q}$-subcircuit that includes the edge (n, c). In other words, for each edge (n, c), we bound from above the following:

$$\max_{\boldsymbol{q}: (n,c) \in \mathcal{C}'_{\boldsymbol{q}}} \mathsf{m}_{\mathcal{C}'_{\boldsymbol{q}}} \geq \mathsf{MMAP}(\mathbf{Q}|_{(n,c)}).$$

$\mathsf{EB}(n)$ then similarly upper-bounds $\mathsf{MMAP}(\mathbf{Q}|_n)$. It is worth pointing out that this bounds the output *at the root* for states $\boldsymbol{q}$ that includes n in their sub-circuits, whereas m_n by Algorithm 7 upper-bounds the output *at each node*.

Let us now describe the recursive steps. First, suppose we want to compute $\mathsf{EB}(n)$ where $\mathsf{EB}(p, n)$ for every parent p of n (i.e. $n \in \mathsf{ch}(p)$) has been computed by the recursion. In order to make sure that $\mathsf{EB}(n)$ upper-bounds the marginal MAP reduced to n, we observe that if $\boldsymbol{q}$ is the solution to $\mathsf{MMAP}(\mathbf{Q}|_n)$ then the $\boldsymbol{q}$-subcircuit must also include one of the edges (p, n). Thus,

Algorithm 7 OUTPUT-BOUNDS$(\mathcal{C}, \mathbf{Q})$

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: m_n storing output bounds for each node n

```
N ← FEEDFORWARDORDER(C)
for each n ∈ N do
    if n is an input unit then
        m_n ← C_n^max(x_φ(n))
    else if n is a product unit then
        m_n ← ∏_{c∈ch(n)} m_c
    else if n is Q-deterministic then
        m_n ← max_{c∈ch(n)} θ_{n,c} m_c
    else
        m_n ← Σ_{c∈ch(n)} θ_{n,c} m_c
```

$\mathsf{EB}(n) = \max_p(\mathsf{EB}(p, n))$ is a valid edge bound:

$$\max_{p:n\in\mathsf{ch}(p)} \mathsf{EB}(p, n) \geq \max_{p:n\in\mathsf{ch}(p)} \mathsf{MMAP}(\mathbf{Q}|_{(p,n)}) = \mathsf{MMAP}(\mathbf{Q}|_n)$$

Next, suppose we wish to compute $\mathsf{EB}(n, c)$ from a given $\mathsf{EB}(n)$. We consider the three possible cases of n being a $\mathbf{Q}$-deterministic sum node, a non $\mathbf{Q}$-deterministic sum node, and a product node. For the latter two cases, the edge bounds are simply propagated from the node. This is because any sub-circuit that includes such node will also include both of its input edges, and thus their bounds will be the same.

Finally, we consider the edge bound $\mathsf{EB}(n, c)$ for an input edge to a $\mathbf{Q}$-deterministic sum node. To illustrate the intuition, we use the example PC in Figure 6.1a. Suppose we want the edge bound between the root and its right input, denoted $\mathsf{EB}(\text{root}, r)$. Running Algorithm 7, we get the upper bound $\mathsf{m}_{\text{root}} = 0.558$ at the root and $\mathsf{m}_l = 0.93$ and $\mathsf{m}_r = 0.84$ for its left and right input, respectively. Note that for every $\boldsymbol{q}$ that includes this edge in its sub-circuit,[2] the marginal $\mathcal{C}(\boldsymbol{q})$ must

[2]This corresponds to $\{X_1 = 0, X_2 = 0\}$ and $\{X_1 = 0, X_2 = 1\}$.

be $0.4 \cdot r(\boldsymbol{q})$, leading to:

$$\max_{\boldsymbol{q}:(\text{root},r)\in\mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}) = 0.4 \cdot \max_{\boldsymbol{q}:(\text{root},r)\in\mathcal{C}'_{\boldsymbol{q}}} r(\boldsymbol{q}) \leq 0.4 \cdot \mathsf{m}_r.$$

Thus, we can use $0.4 \cdot \mathsf{m}_r = 0.336$ as the edge bound for (root, r). Similarly, we can derive the edge bound for (root, l) as $0.6 \cdot \mathsf{m}_l = 0.558$. This can be expressed as:

$$\mathsf{EB}(\text{root}, c) = \mathsf{EB}(\text{root}) - \mathsf{m}_{\text{root}} + \theta_{\text{root},c}\mathsf{m}_c \tag{6.3}$$

for any $c \in \mathsf{ch}(\text{root})$. Note that this holds trivially because $\mathsf{EB}(\text{root}) = \mathsf{m}_{\text{root}}$ as the base case. However, we can generalize this to derive the expression for edge bound from an inner $\mathbf{Q}$-deterministic node.

Let us again use Figure 6.1a as an example; this time we consider the blue dashed edge, denoting it (n, c). Recall that $\mathsf{EB}(n, c)$ aims to upper-bound what Algorithm 7 would return at the root of a $\boldsymbol{q}$-subciruit that includes edge (n, c). In such sub-circuit, (n, c) would be the only input edge to node n, and thus the algorithm would propagate up $\theta_{n,c}\mathsf{m}_c = 0.1$ instead of m_n. This hints at a similar expression as Equation 6.3 where we subtract the contribution of m_n and add $\theta_{n,c}\mathsf{m}_c$. However, a key observation is that m bounds from Algorithm 7 concern the output of each node, whereas the edge bounds concern the output of the root node. Thus, we need to consider how the contribution of m_n gets scaled when it is propagated up to the root node. In this instance, it would be multiplied by $0.7 \cdot 0.6$, which is the product of edge parameters that lie in the path from n to the root. In other words, we get the following expression:

$$\mathsf{EB}(n, c) = \mathsf{EB}(n) + 0.7 \cdot 0.6(-\mathsf{m}_n + \theta_{n,c}\mathsf{m}_c)$$

The pseudocode for this recursive algorithm is described in Algorithm 8.

Proposition 6.1. *Given a smooth and decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a subset $\mathbf{Q} \subset \mathbf{X}$, Algorithm 8 computes an upper bound on Equation* 6.1 *for every edge in $\mathcal{C}$.*

Algorithm 8 EDGE-BOUNDS($\mathcal{C}, \mathbf{Q}$)

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: $\mathsf{r}_{n,c}$ storing edge bounds for each edge (n, c)

$\mathsf{m} \leftarrow$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$)
$\mathsf{t}_{\text{root}} \leftarrow 1$
$\mathsf{r}_{\text{root}} \leftarrow \mathsf{m}_{\text{root}}$
$\mathsf{N} \leftarrow$ BACKWARDORDER($\mathcal{C}$)
for each $n \in \mathsf{N}$ s.t. $\mathsf{t}_n > 0$, $c \in \mathsf{ch}(n)$ **do**
    if n is a product unit **then**
        $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n$
        $\mathsf{r}_c \leftarrow \max(\mathsf{r}_c, \mathsf{r}_{n,c})$
        $\mathsf{t}_c \leftarrow \min(\mathsf{t}_c, \mathsf{t}_n)$
    else if n is a sum unit **then**
        if n is $\mathbf{Q}$-deterministic **then**
            $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n + \mathsf{t}_n(\theta_{n,c}\mathsf{m}_c - \mathsf{m}_n)$
        else
            $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n$
        $\mathsf{r}_c \leftarrow \max(\mathsf{r}_c, \mathsf{r}_{n,c})$
        $\mathsf{t}_c \leftarrow \min(\mathsf{t}_c, \theta_{n,c}\mathsf{t}_n)$

Pruning example We refer to the Appendix for a formal proof of the above proposition, and instead conclude this section with an example round of pruning. Suppose we wish to prune edges from the PC in Figure 6.1a, for an MMAP problem with $\mathbf{Q} = \{X_1, X_2\}$. First, we compute the edge bounds as shown in the left circuit in Figure 6.1b. To perform pruning, we need a lower bound on the marginal MAP probability to compare against. The probability of any $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ state suffices; suppose we use $\boldsymbol{q} = \{X_1 = 0, X_2 = 1\}$ with $p(\boldsymbol{q}) = 0.256$. Then we can prune two edges, resulting in the circuit on the right in Figure 6.1b. More notably, all sum nodes in the resulting circuit become $\mathbf{Q}$-deterministic (highlighted in orange). In particular, as we will discuss more in the next section, this allows us to answer the marginal MAP query via a single feedforward pass. Running Algorithm 7 on this PC, the output at the root is 0.378 which exactly corresponds to the marginal MAP solution $p(X_1 = 1, X_2 = 1) = 0.378$.

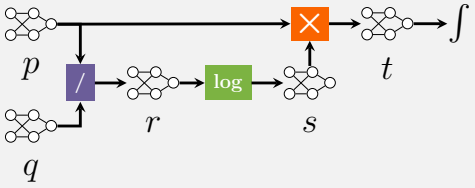
Thus, pruning not only has the immediate effect of decreasing the circuit size, but also changes the PC and its distribution in such a way that can make it easier to solve the marginal MAP problem.

Intermezzo 3: Inference by Composition of Circuit Transformations

Characterizing the tractability of various circuit transformations can also allow us to systematically derive probabilistic circuit inference algorithms, by decomposing queries into modular operations. Specifically, Vergari, Choi, Liu, Teso, and Van den Broeck [2021] proposed to represent queries as *circuit pipelines*: computational graphs whose intermediate operations transform and combine the input circuits into other circuits.

First, we can build a set of simple circuit transformations—sums, products, powers, logarithms, and exponentials—and characterize their tractability in terms of structural constraints on the input circuits and the guaranteed properties of the output circuits. For example, we can take two compatible probabilistic circuits p and q and, in quadratic time, construct a decomposable circuit that represents their product $p \cdot q$ [Shen et al., 2016].

Then given a query that is composed of these simple operations, we can analyze its tractability by propagating the sufficient conditions for tractability of the intermediate operations in the pipeline.




```
function kld(p, q)
    r = quotient(p, q)
    s = log(r)
    t = product(p, s)
    return integrate(t)
end
```

For instance, the figure above on the left shows the pipeline for computing the Kullback-Leibler Divergence between two distributions p and q encoded by circuits, defined as:

$$\mathrm{KL}\left(p \parallel q\right) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\mathbf{X}.$$

We can identify a general class of models that supports its tractable computation: by tracing the conditions for tractable quotient, logarithm, and product over circuits such that the output circuit (i.e., t) admits tractable integration, we can derive a set of sufficient conditions for the input circuits. Moreover, the inference algorithm for KLD can quickly be implemented in a few lines of code as shown on the right. The tractability conditions and inference algorithms can be derived similarly for various information-theoretic quantities, including but not limited to the cross entropy, Shannon entropy, Rényi entropy and divergence, and the squared loss.

In addition, recall the task of computing expected predictions discussed in Intermezzo 1. We can alternatively derive the tractability conditions for it by observing that an expected prediction $\mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})]$ can be expressed as $\int f(\boldsymbol{x}) \cdot p(\boldsymbol{x})$. Therefore, if we can efficiently take the product $f \cdot p$ and output it as a smooth and decomposable circuit (which is the case if f and p are compatible circuits), then we can compute the expected prediction by composing the product operation and marginalization.

Algorithm 9 ITER-SOLVE($\mathcal{C}, \mathbf{Q}$)

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: MMAP($\mathbf{Q}$)

$u \leftarrow$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$)
$l \leftarrow$ LOWER-BOUND($\mathcal{C}, \mathbf{Q}$)
$\mathbf{V} \leftarrow \mathbf{Q}$
while $u > l$ **do**
    $\mathsf{r} \leftarrow$ EDGE-BOUNDS($\mathcal{C}, \mathbf{Q}$)
    for all $(n, c) \in \mathcal{C}$ s.t. $\mathsf{r}_{n,c} \leq l$ **do**
        $\mathcal{C} \leftarrow$ PRUNE-EDGE($\mathcal{C}, (n, c)$)
    $X \leftarrow$ PICK-VAR($\mathbf{V}$); $\mathbf{V} \leftarrow \mathbf{V} \setminus \{X\}$
    $\mathcal{C} \leftarrow$ SPLIT($\mathcal{C}, X$)
    $u \leftarrow \min(u,$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$))
    $l \leftarrow \max(l,$ LOWER-BOUND($\mathcal{C}, \mathbf{Q}$))
return u

6.3 Iterative Marginal MAP Solver

We are now ready to show how the pruning algorithm from the previous section can be leveraged to solve marginal MAP exactly.

As discussed briefly in Section 6.1.1, we can tractably answer a marginal MAP query for a $\mathbf{Q}$-deterministic PC. Thus, a naive solver may try to transform the input PC into a $\mathbf{Q}$-deterministic one to solve a marginal MAP instance. For example, one could apply the *split* operation [Liang et al., 2017; Dang et al., 2020] on the root for each variable in $\mathbf{Q}$. Splitting on a variable $Q \in \mathbf{Q}$ effectively turns the root of the PC into a Q-deterministic sum node while maintaining the distribution represented by it; thus, splitting on every variable in $\mathbf{Q}$ would result in a $\mathbf{Q}$-deterministic circuit. However, this would be highly intractable as each split operation could at most double the size of the PC.

Instead, we propose to prune the circuit as well as split on a query variable in each iteration. While the circuit could grow exponentially in the worst case, we show empirically in the next section that pruning plays a crucial role in indeed keeping the circuit size from growing too much. In fact, in many instances, it decreases the circuit size over the iterations.

A pseudocode of our approach is shown in Algorithm 9.[3] The solver maintains an upper and lower bound on marginal MAP and updates it after every prune and split. The upper bound is computed using Algorithm 7 as discussed in Section 6.2.2. The marginal probability of any instantiation of $\mathbf{Q}$ can be used as a lower bound on the MMAP probability. In particular, we use the solution to a different MMAP instance whose query variables include $\mathbf{Q}$ and can be solved efficiently; more details can be found in the following section. In each iteration, we first prune all edges whose edge bound, computed by Algorithm 8, does not exceed the current lower bound. Then we split on a variable chosen according to some heuristic (discussed further in the next section). The solver is guaranteed to converge after at most $|\mathbf{Q}|$ iterations, at which point the PC must be $\mathbf{Q}$-deterministic, allowing exact computation of MMAP. Furthermore, each prune and split improves the bounds, and thus the solver may also terminate before splitting on all query variables. That is, pruning can decrease the upper bound as we saw in Figure 6.1b, and a split operation also improves the bounds by adding a new $\mathbf{Q}$-deterministic node at the root. Lastly, we again emphasize that our marginal MAP solver only assumes smoothness and decomposability; determinism is not required. For example, this implies that we can also exactly solve MPE for non-deterministic PCs.

6.3.1 Lower Bound

We now describe the algorithm to compute the lower bound used in our iterative solver. First, note that the probability of any assignment to query variables can be used as a lower bound for marginal MAP by definition. A simple and common approach to approximate the marginal MAP state is to solve MPE instead and reduce the MPE state to the query variables. We use a similar approach but with a key additional guarantee: after splitting on all query variables, it exactly solves the marginal MAP problem. A pseudocode of our method is shown in Algorithm 10. Note the similarity of its feedforward pass to Algorithm 7: they both evaluate the circuit while replacing some sum nodes to take the weighted maximum. However, our algorithm not only replaces the $\mathbf{Q}$-deterministic sum

[3] Our marginal MAP solver is implemented in https://github.com/Juice-jl/ProbabilisticCircuits.jl.

Algorithm 10 LOWER-BOUND$(\mathcal{C}, \mathbf{Q})$

Input: a PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: an assignment $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$

```
N ← FEEDFORWARDORDER(C)
for each n ∈ N do
    if n is an input unit then m_n ← 1.0
    else if n is a product unit then m_n ← ∏_{c∈ch(n)} m_c
    else if n or its descendant is Q-deterministic then m_n ← max_{c∈ch(n)} θ_{n,c} m_c
    else m_n ← Σ_{c∈ch(n)} θ_{n,c} m_c
return EXTRACT-STATE(C, Q, m)

procedure EXTRACT-STATE(n,Q,m)
    if n is an input unit then
        if Variable(n) ∈ Q then return {Literal(n)} else return {}
    else if n is a product unit then
        return ⋃_{c∈ch(n)} EXTRACT-STATE(c, Q, m)
    else
        return EXTRACT-STATE(arg max_{c∈ch(n)} θ_{n,c} m_c, Q, m)
```

nodes but all of their ancestors as well. This is so that we can extract a state $\boldsymbol{q}$ by a backward pass, following the edges that were selected by the weighted maximum. Moreover, if the input PC $\mathcal{C}$ is $\mathbf{Q}$-deterministic, this algorithm behaves the same as Algorithm 7 and exactly solves the MMAP problem.

6.3.2 Split Heuristics

This section describes the two variable split heuristics that are evaluated in Section 6.4. (Pruned) selects variables based on the number of pruned edges associated with the variable; (UB) selects variables by the expected change in upper bound after splitting on a variable, which can be computed efficiently via a single pass on the circuit.

Using the (Pruned) heuristic, at every iteration we split on the query variable that had the most number of associated edges pruned. In other words, for each query variable $Q \in \mathbf{Q}$ that is yet to be split on, we count how many edges of a Q-deterministic sum node have been pruned (this value

can be cached to minimize redundant calculations) and choose the variable with the highest count. Intuitively, using this heuristic would tend to minimize a size blow-up by each split.

On the other hand, (UB) aims to maximize opportunities for pruning in the iteration following each split. To compute the heuristic, we first compute for each query variable $Q \in \mathbf{Q}$ the MMAP upper-bounds as described in Algorithm 7, one setting $Q = 0$ as evidence and the other $Q = 1$. Because splitting the root on Q would introduce a deterministic sum node whose children set Q to 0 and 1, these bounds equal the edge bounds on the two input edges to the root after splitting. Let us denote these bounds $B_{Q=0}$ and $B_{Q=1}$ respectively, the lower bound in the current iteration as lb, and the candidate query variables by $\mathbf{Q}' \subseteq \mathbf{Q}$ (i.e. query variables that have not been split on in the previous iterations). Then the (UB) heuristic selects a variable as follows:

$$\begin{cases} \arg\min_{Q:\min(B_{Q=0},B_{Q=1})<lb} \max(B_{Q=0}, B_{Q=1}) & \text{if } \exists\, Q \in \mathbf{Q}' \text{ s.t. } \min(B_{Q=0}, B_{Q=1}) < lb, \\ \arg\min_{Q \in \mathbf{Q}'} B_{Q=0} + B_{Q=1} & \text{otherwise.} \end{cases}$$

In other words, if any variable would have a corresponding edge bound drop below the lower bound, we prioritize selecting from those variables as this guarantees a large part of the circuit is pruned in the next iteration. Then we choose the variable that would decrease the upper bound the most, which would, intuitively, result in more edges being pruned in the next iteration. Note that computing this heuristic requires additional passes through the PC, but as shown in the next section, it makes pruning much more effective and the resulting solver more efficient, despite the added time to compute the heuristic.

6.4 Empirical Evaluation

We evaluated the iterative solver on probabilistic circuits learned from twenty widely-used benchmark datasets. The number of variables ranges from 16 to 1,556, and the size of PCs, learned using Strudel [Dang et al., 2020], ranges from 3,177 to 745,815. We generated marginal MAP instances with two different proportions of query, evidence, and hidden variables—30%, 30%, 40% and 50%,

Table 6.1: Average run time in seconds (with 1-hour time limit for each instance) and the number of instances solved for different proportions of (query, evidence, hidden) variables.

	(30%, 30%, 40%)			(50%, 20%, 30%)		
Dataset	**MaxSPN**	**(Pruned)**	**(UB)**	**MaxSPN**	**(Pruned)**	**(UB)**
NLTCS	**0.004** (10)	0.35 (10)	0.54 (10)	**0.01** (10)	0.39 (10)	0.63 (10)
MSNBC	**0.01** (10)	0.29 (10)	0.50 (10)	**0.03** (10)	0.43 (10)	0.73 (10)
KDD	**0.02** (10)	0.42 (10)	0.64 (10)	**0.04** (10)	0.49 (10)	0.68 (10)
Plants	**0.27** (10)	0.99 (10)	1.36 (10)	2.95 (10)	**2.61** (10)	2.72 (10)
Audio	188.59 (10)	16.57 (10)	**2.87** (10)	2041.33 (6)	15.61 (10)	**13.70 (10)**
Jester	265.50 (10)	16.16 (10)	**6.17** (10)	2913.04 (2)	44.16 (10)	**14.74 (10)**
Netflix	344.71 (10)	22.23 (10)	**5.61** (10)	– (0)	936.83 (10)	**47.18 (10)**
Accidents	**0.54** (10)	2.00 (10)	2.00 (10)	109.56 (10)	19.81 (10)	**15.86** (10)
Retail	**0.03** (10)	0.47 (10)	0.61 (10)	**0.06** (10)	0.67 (10)	0.81 (10)
Pumsb-star	273.70 (10)	106.04 (10)	**6.04** (10)	2208.27 (7)	54.32 (10)	**20.88 (10)**
DNA	2809.44 (4)	65.27 (10)	**9.16 (10)**	– (0)	2634.41 (3)	**505.75 (9)**
Kosarek	1.60 (10)	**0.81** (10)	0.98 (10)	48.74 (10)	**2.65** (10)	3.41 (10)
MSWeb	25.70 (10)	3.63 (10)	**0.96** (10)	1543.49 (10)	48.89 (10)	**1.28** (10)
Book	– (0)	56.47 (10)	**7.25 (10)**	– (0)	907.51 (9)	**46.50 (10)**
EachMovie	– (0)	2563.02 (3)	**93.66 (10)**	– (0)	3293.78 (1)	**1216.89 (8)**
WebKB	– (0)	3378.03 (2)	**102.37 (10)**	– (0)	– (0)	**575.68 (10)**
Reuters-52	– (0)	1238.10 (7)	**22.91 (10)**	– (0)	3107.57 (3)	**120.58 (10)**
20 NewsGrp.	– (0)	2882.95 (3)	**88.13 (10)**	– (0)	– (0)	**504.52 (9)**
BBC	– (0)	– (0)	**766.93 (9)**	– (0)	– (0)	**2757.18 (3)**
Ad	– (0)	– (0)	**344.81 (10)**	– (0)	– (0)	**1254.37 (8)**
Total Solved	124	155	**199**	105	146	**187**

20%, 30%, respectively—randomly dividing the variables and generating evidence while ensuring its probability is nonzero. We generated 10 instances for each dataset and each proportion.

For comparison, we also solved the marginal MAP problems using MaxSPN[4] which is a search-based exact solver for (marginal) MAP on sum-product networks [Mei et al., 2018]. All experiments were ran on a Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz.

Table 6.1 summarizes the results. First, we compare the two heuristics. (Pruned) is comparable or faster than (UB) on relatively easy datasets, but is significantly slower on most of the datasets. Moreover, (Pruned) failed to solve any instance on BBC and Ad datasets, whereas (UB) was able

[4]Specifically, we use the forward checking technique with ordering and stage, which was shown to be the best performing among the exact solvers by Mei et al. [2018].

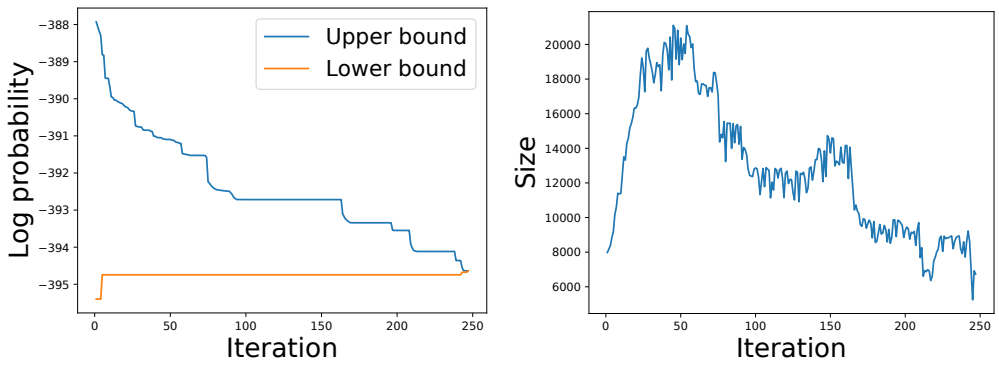


Figure 6.2: Upper and lower bounds (top) and circuit size (bottom) in each iteration of the solver on an example instance on EachMovie dataset.

to solve at least one instance in all datasets. In fact, it was able to solve all 20 instances (10 for each proportion) on 15 out of the 20 datasets. This clearly demonstrates the importance of variable split heuristics and the benefit of explicitly choosing splits that lead to better bounds.

Next, we compare our iterative solver to the search-based approach of MaxSPN. We observe that MaxSPN is faster than our algorithm on easy instances (sub-1 second average run time). This is likely because there is a minimum overhead of performing circuit transformations. On the other hand, our iterative approach clearly outperforms MaxSPN on all other datasets, both in terms of average run time and the number of instances solved.

Lastly, we examine more closely an example run of our solver to empirically demonstrate the benefits of pruning a PC for a specific marginal MAP problem; see Figure 6.2. As we expected, iterative prune and split improve the upper and lower bounds until they converge. The next plot on circuit size clearly illustrates the importance of pruning the circuit. Even though split operations can increase the circuit size, we are very effective at pruning away irrelevant parts of the circuit for MMAP that the circuit size actually decreases over time. Indeed, the size at the point of convergence is smaller than the initial size. Judging by the rate of increase in the early iterations, it is not hard to imagine that without pruning, the circuit would quickly grow too large to run any inference.

6.5 Conclusion

We have introduced a novel approach to marginal MAP inference on probabilistic circuits. It is fundamentally distinct from existing solvers, which are based on a branch and bound search [Mauá et al., 2020; Mei et al., 2018; Huang et al., 2006] using the tractable circuit to prune the search. Instead, we showed that the circuit can be pruned by keeping edges that are relevant to the marginal MAP state. Furthermore, our edge bounds algorithm can effectively find such edges to prune. What remains to solve marginal MAP is to perform simple splits on the circuit, tightening the bounds, and providing more opportunity to prune edges, until a marginal MAP solution is found. Our experiments empirically show that this novel approach to marginal MAP outperforms the search-based approach on a large number of real-world learned probabilistic circuits.

CHAPTER 7

Conclusion

This dissertation proposed a framework of probabilistic modeling and reasoning to address trustworthy AI issues, in particular focusing on the use of probabilistic circuits to measure and enforce different notions of fairness and decision robustness.

The fair and robust machine learning methods as currently described in this thesis are limited in the sense that they assume a generative model will be used as both the prediction/decision-making system as well as the probabilistic model of the underlying distribution. However, it is well known for many real-world applications that specialized discriminative models tend to perform better on making classifications and predictions, compared to generative models whose strengths are more general and flexible reasoning. Therefore, a more broadly applicable and impactful framework would be one that can reason about different prediction models with respect to a distribution represented by a separate probabilistic model.

The initial foundations to develop such reasoning framework have been laid by a tractable inference scheme called *expected predictions* [Khosravi et al., 2019b,a, 2020] (Intermezzo 1). The task is to compute the expected output of a function (in addition to related queries such as its moments), defined by some discriminative model, with respect to a distribution represented by a probabilistic circuit. For instance, we can compute the expected (average) decisions for different demographic groups to audit a model for fairness. In particular, Chapter 3 proposed a fair probabilistic circuit learning algorithm to model the hidden fair labels from data with biased labels. While the PC was also used subsequently to make classifications, we can instead exploit expected predictions to reason about the fairness properties of a given prediction model with respect

to the learned PC. This would allow us to properly handle the uncertainty in the true labels even for classifiers that are not by nature probabilistic.

Furthermore, the current results require strong constraints on probabilistic circuits to solve hard queries such as discrimination pattern mining (Chapter 4) and the expected classification agreement (Chapter 5), which can again limit the scalability and practicality of the proposed approach. Specifically, requiring a constrained vtree may result in a size blow-up of the circuit, nullifying the benefits of linear-time exact inference. As a first step to address this, Chapter 6 introduced a new inference technique that requires less restrictive properties on the PCs and instead answers queries through a series of circuit transformations. A natural future direction to explore would be to employ similar techniques to solve other queries, in particular those relevant to trustworthy AI.

Lastly, even though this thesis demonstrated that several fairness and robustness questions can be answered through probabilistic inference, the reasoning and learning methods were still contained within their respective area. However, I believe there is an unrealized potential to seamlessly bring together various subfields of trustworthy AI within a single framework. By advancing efficient and flexible reasoning in terms of both the inference tasks and the types of models we can handle, I hope we can build a unified framework that would allow us to easily audit various properties—such as fairness, explainability, privacy, robustness, and more—and develop more trustworthy automated decision-making systems.

Appendix A

Proofs

A.1 Degree of Discrimination Bound

A.1.1 Proof of Proposition 4.1

We first derive how $\widetilde{\Delta}$ represents the degree of discrimination Δ for some pattern $\mathbf{xy}$.

$$\Delta_{P,d}(\mathbf{x},\mathbf{y}) = P(d|\mathbf{xy}) - P(d|\mathbf{y}) = \frac{P(\mathbf{x}|d)P(d\mathbf{y})}{P(\mathbf{x}|d)P(d\mathbf{y}) + P(\mathbf{x}|\overline{d})P(\overline{d}\mathbf{y})} - P(d|\mathbf{y})$$
$$= \frac{P(\mathbf{x}|d)P(d|\mathbf{y})}{P(\mathbf{x}|d)P(d|\mathbf{y}) + P(\mathbf{x}|\overline{d})P(\overline{d}|\mathbf{y})} - P(d|\mathbf{y}) = \widetilde{\Delta}(P(\mathbf{x}|d), P(\mathbf{x}|\overline{d}), P(d|\mathbf{y}))$$

Clearly, if $l \leq \gamma \leq u$ then $\min_{l\leq\gamma\leq u} \widetilde{\Delta}(\alpha,\beta,\gamma) \leq \widetilde{\Delta}(\alpha,\beta,\gamma) \leq \max_{l\leq\gamma\leq u} \widetilde{\Delta}(\alpha,\beta,\gamma)$. Therefore, if $l \leq P(d|\mathbf{yy}') \leq u$, then the following holds for any $\mathbf{x}$:

$$\min_{l\leq\gamma\leq u} \widetilde{\Delta}(P(\mathbf{x}|d), P(\mathbf{x}|\overline{d}), \gamma) \leq \widetilde{\Delta}(P(\mathbf{x}|d), P(\mathbf{x}|\overline{d}), P(d|\mathbf{yy}'))$$
$$= \Delta_{P,d}(\mathbf{x},\mathbf{yy}') \leq \max_{l\leq\gamma\leq u} \widetilde{\Delta}(P(\mathbf{x}|d), P(\mathbf{x}|\overline{d}), \gamma).$$

Next, suppose $\mathbf{x}'_u = \arg\max_{\mathbf{x}'} P(d|\mathbf{xx}')$ and $\mathbf{x}'_l = \arg\min_{\mathbf{x}'} P(d|\mathbf{xx}')$. Then from Lemma 1, we also have that $\mathbf{x}'_u = \arg\max_{\mathbf{x}'} P(d|\mathbf{xx}'\mathbf{yy}')$ and $\mathbf{x}'_l = \arg\min_{\mathbf{x}'} P(d|\mathbf{xx}'\mathbf{yy}')$ for any $\mathbf{yy}'$. Therefore,

$$\min_{l\leq\gamma\leq u} \widetilde{\Delta}\left(P(\mathbf{xx}'_l|d), P(\mathbf{xx}'_l|\overline{d}), \gamma\right) \leq \widetilde{\Delta}\left(P(\mathbf{xx}'_l|d), P(\mathbf{xx}'_l|\overline{d}), P(d|\mathbf{yy}')\right) = \Delta_{P,d}(\mathbf{xx}'_l, \mathbf{yy}')$$

$$= P(d|\mathbf{x}\mathbf{x}'_l\mathbf{y}\mathbf{y}') - P(d|\mathbf{y}\mathbf{y}') \leq P(d|\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - P(d|\mathbf{y}\mathbf{y}') = \Delta_{P,d}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}')$$
$$\leq \Delta_{P,d}(\mathbf{x}\mathbf{x}'_u, \mathbf{y}\mathbf{y}') = \widetilde{\Delta}\left(P(\mathbf{x}\mathbf{x}'_u|d), P(\mathbf{x}\mathbf{x}'_u|\overline{d}), P(d|\mathbf{y}\mathbf{y}')\right) \leq \max_{l\leq\gamma\leq u} \widetilde{\Delta}\left(P(\mathbf{x}\mathbf{x}'_u|d), P(\mathbf{x}\mathbf{x}'_u|\overline{d}), \gamma\right).$$

A.1.2 Computing the Discrimination Bound

If $\alpha = P(\mathbf{x}|d) = 0$ and $\beta = P(\mathbf{x}|\overline{d}) = 0$, then the probability of $\mathbf{x}$ is zero and thus $P(d|\mathbf{x}\mathbf{y})$ is ill-defined. Therefore, we will assume that either α or β is nonzero.

Let us write $\widetilde{\Delta}_{\alpha,\beta}(\gamma) = \widetilde{\Delta}(\alpha, \beta, \gamma)$ to denote the function restricted to fixed α and β. If $\alpha = \beta$, then $\widetilde{\Delta}_{\alpha,\beta} = 0$. Also, $\widetilde{\Delta}_{0,\beta}(\gamma) = -\gamma$ and $\widetilde{\Delta}_{\alpha,0}(\gamma) = 1 - \gamma$. Thus, in the following analysis we assume α and β are non-zero and distinct.

If $0 < \alpha \leq \beta \leq 1$, $\widetilde{\Delta}_{\alpha,\beta}$ is negative and convex in γ within $0 \leq \gamma \leq 1$. On the other hand, if $0 < \beta \leq \alpha \leq 1$, then $\widetilde{\Delta}_{\alpha,\beta,\gamma}$ is positive and concave. This can quickly be checked using the following derivatives.

$$\frac{d}{d\gamma}\widetilde{\Delta}_{\alpha,\beta}(\gamma) = \frac{\alpha\beta}{\left(\alpha\gamma + \beta(1-\gamma)\right)^2} - 1, \qquad \frac{d^2}{d\gamma^2}\widetilde{\Delta}_{\alpha,\beta}(\gamma) = \frac{-2\alpha\beta(\alpha-\beta)}{\left(\alpha\gamma + \beta(1-\gamma)\right)^3}$$

Furthermore, the sign of the derivative at $\gamma = 0$ is different from that at $\gamma = 1$, and thus there must exist a unique optimum in $0 \leq \gamma \leq 1$.

Solving for $\frac{d}{d\gamma}\widetilde{\Delta}_{\alpha,\beta}(\gamma) = 0$, we get $\gamma = \frac{\beta \pm \sqrt{\alpha\beta}}{\beta-\alpha}$. The solution corresponding to the feasible space $0 \leq \gamma \leq 1$ is: $\gamma_{\text{opt}} = \frac{\beta-\sqrt{\alpha\beta}}{\beta-\alpha}$. The optimal value is derived as the following.

$$\widetilde{\Delta}_{\alpha,\beta}(\gamma_{\text{opt}}) = \frac{\alpha\left(\frac{\beta-\sqrt{\alpha\beta}}{\beta-\alpha}\right)}{(\alpha-\beta)\left(\frac{\beta-\sqrt{\alpha\beta}}{\beta-\alpha}\right) + \beta} - \frac{\beta-\sqrt{\alpha\beta}}{\beta-\alpha} = \frac{\alpha(\beta-\sqrt{\alpha\beta})}{\sqrt{\alpha\beta}(\beta-\alpha)} - \frac{\beta-\sqrt{\alpha\beta}}{\beta-\alpha} = \frac{2\sqrt{\alpha\beta}-\alpha-\beta}{\beta-\alpha}.$$

Next, suppose that the feasible space is restricted to $l \leq \gamma \leq u$. Then the optimal solution is: γ_{opt} if $l \leq \gamma_{\text{opt}} \leq u$; l if $\gamma_{\text{opt}} < l$; and u if $\gamma_{\text{opt}} > u$.

A.1.3 Proof of Lemma 1

Now we prove that we can maximize the posterior decision probability by maximizing each variable independently. It suffices to prove that for a single variable V and all evidence $\mathbf{w}$, $\arg\max_v P(d|v\mathbf{w}) = \arg\max_v \frac{P(v|d)}{P(v|\overline{d})}$. We first express $P(d|v\mathbf{w})$ as the following:

$$P(d|v\mathbf{w}) = \frac{P(v|d)P(d|\mathbf{w})}{P(v|d)P(d|\mathbf{w}) + P(v|\overline{d})P(\overline{d}|\mathbf{w})} = \frac{1}{1 + \frac{P(v|\overline{d})P(\overline{d}|\mathbf{w})}{P(v|d)P(d|\mathbf{w})}}$$

Then clearly,

$$\arg\max_v P(d|v\mathbf{w}) = \arg\min_v \frac{P(v|\overline{d})P(\overline{d}|\mathbf{w})}{P(v|d)P(d|\mathbf{w})} = \arg\max_v \frac{P(v|d)}{P(v|\overline{d})}.$$

A.2 Divergence Score

A.2.1 Derivation of Equation 4.2

We want to find the closed form solution of the optimization problem in Equation 4.1. Because P and Q differs only in two assignments, we can write the KL divergence as follows:

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{d\mathbf{z}} P(d\mathbf{z}) \log\left(\frac{P(d\mathbf{z})}{Q(d\mathbf{z})}\right) = P(d\mathbf{xy}) \log\left(\frac{P(d\mathbf{xy})}{Q(d\mathbf{xy})}\right) + P(\overline{d}\mathbf{xy}) \log\left(\frac{P(\overline{d}\mathbf{xy})}{Q(\overline{d}\mathbf{xy})}\right)$$

Let r be the change in probability of $d\mathbf{xy}$. That is, $r = Q(d\mathbf{xy}) - P(d\mathbf{xy})$. For Q to be a valid probability distribution, we must have $Q(d\mathbf{xy}) + Q(\overline{d}\mathbf{xy}) = P(\mathbf{xy})$. Then we have $Q(d\mathbf{xy}) = P(d\mathbf{xy}) + r$, and $Q(\overline{d}\mathbf{xy}) = P(\mathbf{xy}) - Q(d\mathbf{xy}) = P(\overline{d}\mathbf{xy}) - r$. We can then express the KL divergence between P and Q as a function of P and r:

$$g_{P,d,\mathbf{x},\mathbf{y}}(r) \triangleq P(d\mathbf{xy}) \log\left(\frac{P(d\mathbf{xy})}{P(d\mathbf{xy}) + r}\right) + P(\overline{d}\mathbf{xy}) \log\left(\frac{P(\overline{d}\mathbf{xy})}{P(\overline{d}\mathbf{xy}) - r}\right)$$

Moreover, the discrimination score of pattern $\mathbf{xy}$ w.r.t Q can be expressed using P and r as the

following:

$$Q(d|\mathbf{xy}) - Q(d|\mathbf{y}) = \frac{P(d\mathbf{xy}) + r}{P(\mathbf{xy})} - \frac{P(d\mathbf{y}) + r}{P(\mathbf{y})} = P(d|\mathbf{xy}) - P(d|\mathbf{y}) + r\left(\frac{1}{P(\mathbf{xy})} - \frac{1}{P(\mathbf{y})}\right)$$
$$= \Delta_{P,d}(\mathbf{x},\mathbf{y}) + r\left(\frac{1}{P(\mathbf{xy})} - \frac{1}{P(\mathbf{y})}\right).$$

The heuristic $\mathrm{Div}_{P,d,\delta}(\mathbf{x},\mathbf{y})$ is then written using r as follows:

$$\min_r g_{P,d,\mathbf{x},\mathbf{y}}(r) \tag{A.1}$$
$$\text{s.t.} \quad \left|\Delta_{P,d}(\mathbf{x},\mathbf{y}) + r\left(\frac{1}{P(\mathbf{xy})} - \frac{1}{P(\mathbf{y})}\right)\right| \leq \delta, \quad -P(d\mathbf{xy}) \leq r \leq P(\bar{d}\mathbf{xy})$$

The objective function $g_{P,d,\mathbf{x},\mathbf{y}}$ is convex in r with its unconstrained global minimum at $r = 0$. Note that this is a feasible point if and only if $|\Delta_{P,d}(\mathbf{x},\mathbf{y})| \leq \delta$; in other words, when the pattern $\mathbf{xy}$ is already fair. Otherwise, the optimum must be either of the extreme points of the feasible space, whichever is closer to 0. The extreme points for the first set of inequalities are:

$$r_1 = \frac{\delta - P(d|\mathbf{xy}) + P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}, \qquad r_2 = \frac{-\delta - P(d|\mathbf{xy}) + P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}.$$

If $\Delta_{P,d}(\mathbf{x},\mathbf{y}) > \delta$, then $r_2 \leq r_1 < 0$. In such case, $g(r_2) \geq g(r_1)$ and $-P(d\mathbf{xy}) \leq r_1 \leq P(\bar{d}\mathbf{xy})$ as shown below:

$$r_1 < 0 \leq P(\bar{d}\mathbf{xy}),$$
$$-r_1 = \frac{-\delta + P(d|\mathbf{xy}) - P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} \leq \frac{P(d|\mathbf{xy}) - P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} \leq \frac{P(d|\mathbf{xy}) - P(d\mathbf{x}|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} = P(d\mathbf{xy}).$$

Similarly, if $\Delta_{P,d}(\mathbf{x},\mathbf{y}) < -\delta$, then $r_1 \geq r_2 > 0$. Also, $g(r_1) \geq g(r_2)$ and $-P(d\mathbf{xy}) \leq r_2 \leq P(\bar{d}\mathbf{xy})$ as shown below:

$$r_2 > 0 \geq -P(d\mathbf{xy}),$$

$$r_2 \leq \frac{-P(d|\mathbf{xy}) + P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} \leq \frac{P(\bar{d}|\mathbf{xy}) - P(\bar{d}|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} = P(\bar{d}\mathbf{xy}).$$

Hence, the optimal solution $r^\star$ is

$$r^\star = \begin{cases} 0, & \text{if } |\Delta_{P,d}(\mathbf{x},\mathbf{y})| \leq \delta, \\ \frac{\delta - \Delta_{P,d}(\mathbf{x},\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}, & \text{if } \Delta_{P,d}(\mathbf{x},\mathbf{y}) > \delta, \\ \frac{-\delta - \Delta_{P,d}(\mathbf{x},\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}, & \text{if } \Delta_{P,d}(\mathbf{x},\mathbf{y}) < -\delta, \end{cases}$$

and the divergence score is $\mathrm{Div}_{P,d,\delta}(\mathbf{x},\mathbf{y}) = g_{P,d,\mathbf{x},\mathbf{y}}(r^\star)$.

A.2.2 Upper Bounds on Divergence Score

Here we present two upper bounds on the divergence score for pruning the search tree. The first bound uses the observation that the hypothetical distribution Q with $\Delta_{Q,d}(\mathbf{x},\mathbf{y}) = 0$ is always a feasible hypothetical fair distribution.

Proposition A.1. *Let P be a Naive Bayes distribution over $D \cup \mathbf{Z}$, and let $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$. For all possible valid extensions $\mathbf{x}'$ and $\mathbf{y}'$, the following holds:*

$$\mathrm{Div}_{P,d,\delta}(\mathbf{xx}', \mathbf{yy}') \leq \; P(d\mathbf{xy}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(d|\mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(d|\mathbf{z})} + P(\bar{d}\mathbf{xy}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(\bar{d}|\mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(\bar{d}|\mathbf{z})}$$

Proof. Consider the following point:

$$r_0 = \frac{-P(d|\mathbf{xy}) + P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})}.$$

First, we show that above r_0 is always a feasible point in Problem A.1:

$$\left| \Delta_{P,d}(\mathbf{x},\mathbf{y}) + r_0 \left(\frac{1}{P(\mathbf{xy})} - \frac{1}{P(\mathbf{y})} \right) \right| = |\Delta_{P,d}(\mathbf{x},\mathbf{y}) - \Delta_{P,d}(\mathbf{x},\mathbf{y})| = 0 \leq \delta,$$

$$r_0 = \frac{P(\overline{d}|\mathbf{xy}) - P(\overline{d}|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} \leq \frac{P(\overline{d}|\mathbf{xy}) - P(\overline{d}\mathbf{x}|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} = P(\overline{d}\mathbf{xy}),$$
$$-r_0 = \frac{P(d|\mathbf{xy}) - P(d|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} \leq \frac{P(d|\mathbf{xy}) - P(d\mathbf{x}|\mathbf{y})}{1/P(\mathbf{xy}) - 1/P(\mathbf{y})} = P(d\mathbf{xy}).$$

Then the divergence score for any pattern must be smaller than $g_{P,d,\mathbf{x},\mathbf{y}}(r_0)$:

$$\begin{aligned}\mathrm{Div}_{P,d,\delta}(\mathbf{x},\mathbf{y}) &\leq g_{P,d,\mathbf{x},\mathbf{y}}(r_0) = P(d\mathbf{xy}) \log \frac{P(d|\mathbf{xy})}{P(d|\overline{\mathbf{x}}\mathbf{y})} + P(\overline{d}\mathbf{xy}) \log \frac{P(\overline{d}|\mathbf{xy})}{P(\overline{d}|\overline{\mathbf{x}}\mathbf{y})} \\ &\leq P(d\mathbf{xy}) \log \frac{P(d|\mathbf{xy})}{\min_{\mathbf{x}} P(d|\mathbf{xy})} + P(\overline{d}\mathbf{xy}) \log \frac{P(\overline{d}|\mathbf{xy})}{\min_{\mathbf{x}} P(\overline{d}|\mathbf{xy})}.\end{aligned}$$

Here, we use $\overline{\mathbf{x}}$ to mean that $\mathbf{x}$ does not hold. In other words,

$$P(d|\overline{\mathbf{x}}\mathbf{y}) = \frac{P(d\mathbf{y}) - P(d\mathbf{xy})}{P(\mathbf{y}) - P(\mathbf{xy})} = \sum_{\mathbf{x}} P(d|\mathbf{xy}) P(\mathbf{x}|\overline{\mathbf{x}}\mathbf{y}).$$

We can then use this to bound the divergence score any pattern extended from $\mathbf{xy}$:

$$\begin{aligned}&\mathrm{Div}_{P,d,\delta}(\mathbf{xx}', \mathbf{yy}') \\ &\leq P(d\mathbf{xx}'\mathbf{yy}') \log \frac{P(d|\mathbf{xx}'\mathbf{yy}')}{\min_{\mathbf{xx}'} P(d|\mathbf{xx}'\mathbf{yy}')} + P(\overline{d}\mathbf{xx}'\mathbf{yy}') \log \frac{P(\overline{d}|\mathbf{xx}'\mathbf{yy}')}{\min_{\mathbf{xx}'} P(\overline{d}|\mathbf{xx}'\mathbf{yy}')} \\ &\leq P(d\mathbf{xy}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(d|\mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(d|\mathbf{z})} + P(\overline{d}\mathbf{xy}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(\overline{d}|\mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(\overline{d}|\mathbf{z})}.\end{aligned}$$

□

We can also bound the divergence score using the maximum and minimum possible discrimination scores shown in Proposition 4.1, in place of the current pattern's discrimination. Let us denote the bounds for discrimination score as follows:

$$\overline{\Delta}(\mathbf{x},\mathbf{y}) = \max_{l \leq \gamma \leq u} \widetilde{\Delta}\left(P(\mathbf{xx}'_u|d), P(\mathbf{xx}'_u|\overline{d}), \gamma\right), \quad \underline{\Delta}(\mathbf{x},\mathbf{y}) = \min_{l \leq \gamma \leq u} \widetilde{\Delta}\left(P(\mathbf{xx}'_l|d), P(\mathbf{xx}'_l|\overline{d}), \gamma\right).$$

Proposition A.2. *Let P be a Naive Bayes distribution over $D \cup \mathbf{Z}$, and let $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$. For all possible valid extensions $\mathbf{x}'$ and $\mathbf{y}'$,* $\mathrm{Div}_{P,d,\delta}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') \leq \max\left(g_{P,d,\mathbf{x}\mathbf{x}',\mathbf{y}\mathbf{y}'}(r_u), g_{P,d,\mathbf{x}\mathbf{x}',\mathbf{y}\mathbf{y}'}(r_l)\right)$ *where*

$$r_u = \frac{\delta - \overline{\Delta}(\mathbf{x}, \mathbf{y})}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')}, \qquad r_l = \frac{-\delta - \underline{\Delta}(\mathbf{x}, \mathbf{y})}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')}.$$

Proof. The proof proceeds by case analysis on the discrimination score of extended patterns $\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}'$.

First, if $|\Delta(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}')| \leq \delta$, $\mathrm{Div}_{P,d,\delta}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') = 0$ which is the global minimum, and thus is smaller than both $g(r_u)$ and $g(r_l)$.

Next, suppose $\Delta(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') > \delta$. Then from Proposition 4.1,

$$r_u = \frac{\delta - \overline{\Delta}(\mathbf{x}, \mathbf{y})}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')} \leq r^\star = \frac{\delta - \Delta_{P,d}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}')}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')} < 0.$$

As g is convex with its minimum at 0, we can conclude $\mathrm{Div}_{P,d,\delta}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') = g(r^\star) \leq g(r_u)$.

Finally, if $\Delta(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') < -\delta$, we have

$$r_l = \frac{-\delta - \underline{\Delta}(\mathbf{x}, \mathbf{y})}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')} \geq r^\star = \frac{-\delta - \Delta_{P,d}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}')}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')} > 0.$$

Similarly, this implies $\mathrm{Div}_{P,d,\delta}(\mathbf{x}\mathbf{x}', \mathbf{y}\mathbf{y}') = g(r^\star) \leq g(r_l)$. Because the divergence score is always smaller than either $g(r_u)$ or $g(r_l)$, it must be smaller than $\max(g(r_u), g(r_l))$. □

Lastly, we show how to efficiently compute an upper bound on $g_{P,d,\mathbf{x}\mathbf{x}',\mathbf{y}\mathbf{y}'}(r_u)$ $g_{P,d,\mathbf{x}\mathbf{x}',\mathbf{y}\mathbf{y}'}(r_l)$ from Proposition A.2 for all patterns extended from $\mathbf{x}\mathbf{y}$. This is necessary for pruning during the search for discrimination patterns with high divergence scores. First, note that r_u and r_l can be expressed as

$$\frac{c}{1/P(\mathbf{x}\mathbf{x}'\mathbf{y}\mathbf{y}') - 1/P(\mathbf{y}\mathbf{y}')}, \tag{A.2}$$

where $c = \delta - \overline{\Delta}(\mathbf{x}, \mathbf{y})$ for r_u and $c = -\delta - \underline{\Delta}(\mathbf{x}, \mathbf{y})$ for r_l. Hence, it suffices to derive the following

bound.

$$
\begin{aligned}
&g_{P,d,\mathbf{xx}',\mathbf{yy}'}\left(\frac{c}{1/P(\mathbf{xx}'\mathbf{yy}')-1/P(\mathbf{yy}')}\right)\\
&= P(d\mathbf{xx}'\mathbf{yy}')\log\left(\frac{P(d\mathbf{xx}'\mathbf{yy}')}{P(d\mathbf{xx}'\mathbf{yy}')+\frac{c}{1/P(\mathbf{xx}'\mathbf{yy}')-1/P(\mathbf{yy}')}}\right)\\
&\quad + P(\bar{d}\mathbf{xx}'\mathbf{yy}')\log\left(\frac{P(\bar{d}\mathbf{xx}'\mathbf{yy}')}{P(\bar{d}\mathbf{xx}'\mathbf{yy}')-\frac{c}{1/P(\mathbf{xx}'\mathbf{yy}')-1/P(\mathbf{yy}')}}\right)\\
&= P(d\mathbf{xx}'\mathbf{yy}')\log\left(\frac{P(d|\mathbf{xx}'\mathbf{yy}')(1-P(\mathbf{xx}'|\mathbf{yy}'))}{P(d|\mathbf{xx}'\mathbf{yy}')(1-P(\mathbf{xx}'|\mathbf{yy}'))+c}\right)\\
&\quad + P(\bar{d}\mathbf{xx}'\mathbf{yy}')\log\left(\frac{P(\bar{d}|\mathbf{xx}'\mathbf{yy}')(1-P(\mathbf{xx}'|\mathbf{yy}'))}{P(\bar{d}|\mathbf{xx}'\mathbf{yy}')(1-P(\mathbf{xx}'|\mathbf{yy}'))-c}\right)\\
&\leq \begin{cases} 0 & \text{if } c=0\\ P(d\mathbf{xy})\log\frac{(\max_{\mathbf{z}\models\mathbf{xy}}P(d|\mathbf{z}))(1-\min_{\mathbf{x}'\mathbf{y}'}P(\mathbf{xx}'|\mathbf{yy}'))}{(\min_{\mathbf{z}\models\mathbf{xy}}P(d|\mathbf{z}))(1-\max_{\mathbf{x}'\mathbf{y}'}P(\mathbf{xx}'|\mathbf{yy}'))+c} & \text{if } c<0\\ P(\bar{d}\mathbf{xy})\log\frac{(\max_{\mathbf{z}\models\mathbf{xy}}P(\bar{d}|\mathbf{z}))(1-\min_{\mathbf{x}'\mathbf{y}'}P(\mathbf{xx}'|\mathbf{yy}'))}{(\min_{\mathbf{z}\models\mathbf{xy}}P(\bar{d}|\mathbf{z}))(1-\max_{\mathbf{x}'\mathbf{y}'}P(\mathbf{xx}'|\mathbf{yy}'))-c} & \text{if } c>0\end{cases}
\end{aligned}
$$

A.3 Proof of Proposition 4.2

The probability values of positive decision in terms of naive Bayes parameters θ are as follows:

$$
\begin{aligned}
P_\theta(d|\mathbf{xy}) &= \frac{P_\theta(d\mathbf{xy})}{P_\theta(\mathbf{xy})} = \frac{\theta_d\prod_x\theta_{x|d}\prod_y\theta_{y|d}}{\theta_d\prod_x\theta_{x|d}\prod_y\theta_{y|d}+\theta_{\bar{d}}\prod_x\theta_{x|\bar{d}}\prod_y\theta_{y|\bar{d}}} = \frac{1}{1+\frac{\theta_{\bar{d}}\prod_x\theta_{x|\bar{d}}\prod_y\theta_{y|\bar{d}}}{\theta_d\prod_x\theta_{x|d}\prod_y\theta_{y|d}}},\\
P_\theta(\bar{d}|\mathbf{y}) &= \frac{P_\theta(d\mathbf{y})}{P_\theta(\mathbf{y})} = \frac{1}{1+\frac{\theta_{\bar{d}}\prod_y\theta_{y|\bar{d}}}{\theta_d\prod_y\theta_{y|d}}}.
\end{aligned}
$$

For simplicity of notation, let us write:

$$
r_{\mathbf{x}} = \frac{\prod_x\theta_{x|\bar{d}}}{\prod_x\theta_{x|d}}, \quad r_{\mathbf{y}} = \frac{\theta_{\bar{d}}\prod_y\theta_{y|\bar{d}}}{\theta_d\prod_y\theta_{y|d}}. \tag{A.3}
$$

Then the degree of discrimination is $\Delta_{P_\theta,d}(\mathbf{x},\mathbf{y}) = P_\theta(d|\mathbf{xy}) - P_\theta(d|\mathbf{y}) = \frac{1}{1+r_\mathbf{x}r_\mathbf{y}} - \frac{1}{1+r_\mathbf{y}}$. Now we express the fairness constraint $|\Delta_{P_\theta,d}(\mathbf{x},\mathbf{y})| \leq \delta$ as the following two inequalities:

$$-\delta \leq \frac{(1+r_\mathbf{y}) - (1+r_\mathbf{x}r_\mathbf{y})}{(1+r_\mathbf{x}r_\mathbf{y})\cdot(1+r_\mathbf{y})} \leq \delta.$$

After simplifying,

$$r_\mathbf{y} - r_\mathbf{x}r_\mathbf{y} \geq -\delta(1 + r_\mathbf{x}r_\mathbf{y} + r_\mathbf{y} + r_\mathbf{x}r_\mathbf{y}^2), \qquad r_\mathbf{y} - r_\mathbf{x}r_\mathbf{y} \leq \delta(1 + r_\mathbf{x}r_\mathbf{y} + r_\mathbf{y} + r_\mathbf{x}r_\mathbf{y}^2).$$

We further express this as the following two signomial inequality constraints:

$$\left(\frac{1-\delta}{\delta}\right) r_\mathbf{x}r_\mathbf{y} - \left(\frac{1+\delta}{\delta}\right) r_\mathbf{y} - r_\mathbf{x}r_\mathbf{y}^2 \leq 1, \qquad -\left(\frac{1+\delta}{\delta}\right) r_\mathbf{x}r_\mathbf{y} + \left(\frac{1-\delta}{\delta}\right) r_\mathbf{y} - r_\mathbf{x}r_\mathbf{y}^2 \leq 1$$

Note that $r_\mathbf{x}$ and $r_\mathbf{y}$ according to Equation A.3 are monomials of θ, and thus above constraints are also signomial with respect to the optimization variables θ.

A.4 Proof of Proposition 5.3

Proof. We have $\mathrm{MAA}(\mathbf{F}') \leq \mathrm{MPA}(\mathbf{F}')$ from Proposition 5.1. To show $\mathrm{MAA}(\mathbf{F}') \geq \mathrm{MPA}(\mathbf{F}')$ under the independence assumption, we will use the following claim.

Claim 1. *Suppose features* $\mathbf{F}'$ *and* $\mathbf{R}$ *are independent given class* C*. For any instances* $\mathbf{f}'_1$ *and* $\mathbf{f}'_2$*, if* $\Pr(c|\mathbf{f}'_1) \geq \Pr(c|\mathbf{f}'_2)$*, then at least one of the following must hold:*

1. $\sum_\mathbf{r} [C_T(\mathbf{f}'_1\mathbf{r}) = c] \cdot \Pr(\mathbf{r}|\mathbf{f}'_1) \geq 0.5$*, or*
2. $\sum_\mathbf{r} [C_T(\mathbf{f}'_2\mathbf{r}) = \bar{c}] \cdot \Pr(\mathbf{r}|\mathbf{f}'_2) \geq 0.5$.

If inequality 1 is true, we say $\mathbf{f}'_1$ *"favors positive classification", and if inequality 2 is true, we say* $\mathbf{f}'_2$ *"favors negative classification".*

Above claim states that, if $\mathbf{F}'$ and $\mathbf{R}$ are independent given C and if positive class c is more

likely given $\mathbf{f}'_1$ than given $\mathbf{f}'_2$, then we cannot have $\mathbf{f}'_1$ favor negative classification while $\mathbf{f}'_2$ favor positive classification.

Assuming the claim is true, we can choose a T' such that $Q_1 < T' \leq Q_2$, where $Q_1 = \max_{\mathbf{f}'} \Pr(c|\mathbf{f}')$ such that $\mathbf{f}'$ favors negative class and $Q_2 = \min_{\mathbf{f}'} \Pr(c|\mathbf{f}')$ such that $\mathbf{f}'$ favors positive class. Then, an instance $\mathbf{f}'$ favors positive class if and only if $\Pr(c|\mathbf{f}') \geq T'$. Thus, we obtain the following inequality:

$$\begin{aligned}
\mathrm{MAA}(\mathbf{F}') &\geq \sum_{\mathbf{f}} [C_T(\mathbf{f}) = C_{T'}(\mathbf{f}')] \cdot \Pr(\mathbf{f}) && \text{(A.4)} \\
&= \sum_{\mathbf{f}'} \max_c \left\{ \sum_{\mathbf{r}} [C_T(\mathbf{f}'\mathbf{r}) = c] \cdot \Pr(\mathbf{r}|\mathbf{f}') \right\} \cdot \Pr(\mathbf{f}') && \text{(A.5)} \\
&= \mathrm{MPA}(\mathbf{F}')
\end{aligned}$$

Equation A.4 follows from the definition of MAA as the maximum of ECA across different T'. Equation A.5 holds because T' was explicitly constructed such that the trimmed classifier classifies each instance $\mathbf{f}'$ into the class that it favors in the original classifier.

Now we prove the claim by considering two possible cases. Suppose there exists $\mathbf{f}'_1$ and $\mathbf{f}'_2$ such that $\Pr(c|\mathbf{f}'_1) \geq \Pr(c|\mathbf{f}'_2)$ but $\mathbf{f}'_1$ favors negative class while $\mathbf{f}'_2$ favors positive class. Since $\mathbf{F}'$ and $\mathbf{R}$ are independent given C, we can write the following in log-odds domain: $\log O(c|\mathbf{f}'\mathbf{r}) = \log O(c) + w_{\mathbf{f}'} + w_{\mathbf{r}}$, where $w_{\mathbf{x}} = \log \frac{\Pr(\mathbf{x}|c)}{\Pr(\mathbf{x}|\overline{c})}$. Then $\Pr(c|\mathbf{f}'\mathbf{r}) \geq T$ if and only if $\log O(c|\mathbf{f}'\mathbf{r}) \geq \lambda = \log \frac{T}{1-T}$.

Case 1: $\Pr(c|\mathbf{f}'_2) < T$. Equivalently, $\log O(c|\mathbf{f}'_2) < \lambda$. Also, $\log O(c|\mathbf{f}'_2) \leq \log O(c|\mathbf{f}'_1)$ by assumption. Then $\forall\, \mathbf{r}\ \log O(c|\mathbf{f}'_2\mathbf{r}) \geq \lambda \implies \log O(c|\mathbf{f}'_1\mathbf{r}) \geq \lambda$. For such $\mathbf{r}$, $w_{\mathbf{r}} > 0$ and thus $\Pr(\mathbf{r}|c) > \Pr(\mathbf{r}|\overline{c})$, which implies:

$$\Pr(\mathbf{r}|\mathbf{f}'_1) = \Pr(\mathbf{r}|c)\Pr(c|\mathbf{f}'_1) + \Pr(\mathbf{r}|\overline{c})\Pr(\overline{c}|\mathbf{f}'_1) = \Pr(\mathbf{r}|\mathbf{f}'_2) + \alpha(\Pr(\mathbf{r}|c) - \Pr(\mathbf{r}|\overline{c})) \geq \Pr(\mathbf{r}|\mathbf{f}'_2),$$

where $\alpha = \Pr(c|\mathbf{f}'_1) - \Pr(c|\mathbf{f}'_2) \geq 0$.

Combining these, we have

$$\sum_{\mathbf{r}} [C_T(\mathbf{f}'_1\mathbf{r}) = c] \Pr(\mathbf{r}|\mathbf{f}'_1) \geq \sum_{\mathbf{r}} [C_T(\mathbf{f}'_2\mathbf{r}) = c] \Pr(\mathbf{r}|\mathbf{f}'_2) > 0.5,$$

which is a contradiction of our assumption that $\mathbf{f}'_1$ favors negative class (i.e. $\sum_{\mathbf{r}} [C_T(\mathbf{f}'_1\mathbf{r}) = c] \Pr(\mathbf{r}|\mathbf{f}'_1) < 0.5$).

Case 2: $\Pr(c|\mathbf{f}'_2) \geq T$. Similarly, $\forall\, \mathbf{r}\, \Pr(c|\mathbf{f}'_1\mathbf{r}) < T$ implies $\Pr(c|\mathbf{f}'_2\mathbf{r}) < T$ and $\Pr(\mathbf{r}|c) < \Pr(\mathbf{r}|\overline{c})$. Thus,

$$\Pr(\mathbf{r}|\mathbf{f}'_2) = \Pr(\mathbf{r}|c) \Pr(c|\mathbf{f}'_2) + \Pr(\mathbf{r}|\overline{c}) \Pr(\overline{c}|\mathbf{f}'_2) = \Pr(\mathbf{r}|\mathbf{f}'_1) + \alpha(\Pr(\mathbf{r}|\overline{c}) - \Pr(\mathbf{r}|c)) \geq \Pr(\mathbf{r}|\mathbf{f}'_1).$$

This leads to the following:

$$\sum_{\mathbf{r}} [C_T(\mathbf{f}'_2\mathbf{r}) = \overline{c}] \Pr(\mathbf{r}|\mathbf{f}'_2) \geq \sum_{\mathbf{r}} [C_T(\mathbf{f}'_1\mathbf{r}) = \overline{c}] \Pr(\mathbf{r}|\mathbf{f}'_1) > 0.5,$$

which is a contradiction of our assumption that $\mathbf{f}'_2$ favors positive class and thus $\sum_{\mathbf{r}} [C_T(\mathbf{f}'_2\mathbf{r}) = \overline{c}] \Pr(\mathbf{r}|\mathbf{f}'_2) < 0.5$. This concludes the proof of the claim and the proposition. □

A.5 Proof of Proposition 6.1

Proposition 6.1. *Given a smooth and decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a subset $\mathbf{Q} \subset \mathbf{X}$, Algorithm 8 computes an upper bound on Equation* 6.1 *for every edge in $\mathcal{C}$.*

To prove above proposition, let us define some auxiliary circuit structures. First, running Algorithm 7 to compute m can be interpreted as a feedforward evaluation on a circuit obtained from $\mathcal{C}$ by replacing every $\mathbf{Q}$-deterministic sum node n with a node that simply returns the output of child node $c = \arg\max_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$ (i.e. they are "fixed" to select the same branch as Line 9 in

Algorithm 7). Suppose we unroll such circuit into a tree structure: i.e. create copies of any node with multiple parents and recurse down. We denote this circuit by $\mathcal{M}$. Then we have $\mathsf{m}_{\text{root}} = \mathcal{M}(\emptyset)$, where $\mathcal{M}(\emptyset)$ represents the circuit evaluation for marginal with no evidence. Moreover, for any node n' in $\mathcal{M}$ that corresponds to node n in $\mathcal{C}$, written as $n' \in \text{copy}(n)$, we have $\mathsf{m}_n = \mathcal{M}_{n'}(\emptyset)$.

In addition, for every node n' in $\mathcal{M}$, we define a circuit denoted $\mathcal{M}^{(n')}$ obtained from $\mathcal{M}$ by "fixing" the $\mathbf{Q}$-deterministic nodes that appear in the path from root to n' such that they select the branch that reaches n'. In other words, let $\mathbf{Q}'$ be $\mathbf{Q} \setminus \phi(n')$ and $\boldsymbol{q}' = \gamma_{n'}|_{\mathbf{Q}'}$. Note that because $\mathcal{M}$ is a tree structure, every assignment in the context of n' has the same value for variables in $\mathbf{Q}'$; this is given by the $\mathbf{Q}$-deterministic sum nodes in the path from n' to the root which is unique. Then $\mathcal{M}^{(n')}$ is identical to $\mathcal{M}$, except for the $\mathbf{Q}$-deterministic nodes that are ancestors of n', which output the child node whose context agrees with $\boldsymbol{q}'$.

Lemma 3. *Let $\mathcal{C}$ be a PC over variables $\mathbf{X}$ and $\mathcal{M}$ be its tree-unrolled max-sum circuit (as described above) for a set of query variables $\mathbf{Q}$. For any $\mathcal{M}^{(n')}$ constructed from $\mathcal{M}$ as above, the following statements hold:*

1. $\mathcal{M}^{(n')}(\emptyset) = \sum_{\boldsymbol{y} \in \mathsf{val}(\mathbf{X} \setminus \mathbf{Q})} \mathcal{M}^{(n')}(\boldsymbol{y})$.
2. *For any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$, $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}(\boldsymbol{q})$.*

Note that above statements also apply to $\mathcal{M} = \mathcal{M}^{(\text{root})}$. We now provide a proof of Proposition 6.1 using above lemma, which we will prove at the end of this section.

Proof. We will show that for every node n in $\mathcal{C}$, Algorithm 8 returns

$$\mathsf{r}_n \geq \max_{n' \in \text{copy}(n)} \mathcal{M}^{(n')}(\emptyset), \tag{A.6}$$

and for every edge (n, c) it returns

$$\mathsf{r}_{n,c} \geq \max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset). \tag{A.7}$$

Note that Equation A.7 implies that $\mathsf{r}_{n,c}$ upper-bounds the quantity $\mathsf{MMAP}(\,\mathbf{Q}|_{(n,c)})$ given by Equation 6.1:

$$\begin{aligned}\max_{(n',c')\in\mathsf{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) &\geq \max_{(n',c')\in\mathsf{copy}((n,c))} \max_{\boldsymbol{q}\in\gamma_{c'}|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{(n',c')\in\mathsf{copy}((n,c))} \max_{\boldsymbol{q}\in(\gamma_{n'}\cap\gamma_{c'})|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) \\ &= \max_{\boldsymbol{q}\in\bigcup_{(n',c')\in\mathsf{copy}((n,c))}(\gamma_{n'}\cap\gamma_{c'})|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{\boldsymbol{q}\in(\gamma_n\cap\gamma_c)|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{\boldsymbol{q}\in\gamma_{(n,c)}|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) \\ &= \max_{\boldsymbol{q}:(n,c)\in\mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}) = \mathsf{MMAP}(\,\mathbf{Q}|_{(n,c)})\end{aligned}$$

We will now prove that Equations A.6 and A.7 hold by induction. For the base case, r_{root} is set as m_{root}, which is exactly $\mathcal{M}(\emptyset) = \mathcal{M}^{(\text{root})}(\emptyset)$.

Next, assume Equation A.6 holds for a node n in $\mathcal{C}$, and we want to show that Equation A.7 holds for any of its input edges (n, c). If n is a product unit or a sum unit that is not $\mathbf{Q}$-deterministic, for any edge (n, c) and its copy (n', c') the circuits $\mathcal{M}^{(n')}$ and $\mathcal{M}^{(c')}$ are identical by definition. Then Equation A.7 holds as follows:

$$\max_{(n',c')\in\mathsf{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) = \max_{(n',c')\in\mathsf{copy}((n,c))} \mathcal{M}^{(n')}(\emptyset) = \max_{n'\in\mathsf{copy}(n)} \mathcal{M}^{(n')}(\emptyset) \leq \mathsf{r}_n = \mathsf{r}_{n,c}.$$

If n is a $\mathbf{Q}$-deterministic sum node, the circuits $\mathcal{M}^{(n')}$ and $\mathcal{M}^{(c')}$ can differ only by whether node n' is fixed to take c'. Thus, for any $\boldsymbol{y} \notin \gamma_{n'}|_{\mathbf{Y}}$ where $\mathbf{Y} = \mathbf{X} \setminus \mathbf{Q}$, $\mathcal{M}^{(n')}(\boldsymbol{y}) = \mathcal{M}^{(c')}(\boldsymbol{y})$. For $\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}$, we have

$$\mathcal{M}^{(n')}(\boldsymbol{y}) - \mathcal{M}^{(c')}(\boldsymbol{y}) = \Big(\prod_{\theta\in\mathsf{path}(n')} \theta\Big) \cdot \mathcal{M}^{(n')}_{n'}(\boldsymbol{y}) - \Big(\prod_{\theta\in\mathsf{path}(n')} \theta\Big) \cdot \theta_{n',c'} \cdot \mathcal{M}^{(c')}_{c'}(\boldsymbol{y})$$

where $\mathsf{path}(n')$ denotes the set of all edge parameters that appear in the path from root to node n'. Note that $\mathcal{M}^{(n')}_{n'}$, i.e. the subcircuit of $\mathcal{M}^{(n')}$ rooted at n', is identical to $\mathcal{M}_{n'}$ as the two max-sum circuits differ only in the ancestors of n'. Similarly, $\mathcal{M}^{(c')}_{c'}$ is equal to $\mathcal{M}_{c'}$. Then we can express the

circuit evaluation of $\mathcal{M}^{(c')}$ as

$$\begin{aligned}
\mathcal{M}^{(c')}(\emptyset) &= \sum_{\boldsymbol{y}\in\mathsf{val}(\mathbf{Y})} \mathcal{M}^{(c')}(\boldsymbol{y}) = \sum_{\boldsymbol{y}\notin\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) + \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) \\
&= \sum_{\boldsymbol{y}\notin\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(n')}(\boldsymbol{y}) + \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) = \mathcal{M}^{(n')}(\emptyset) - \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(n')}(\boldsymbol{y}) + \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) \\
&= \mathcal{M}^{(n')}(\emptyset) + \left(\prod_{\theta\in\mathsf{path}(n')} \theta\right)\left(\theta_{n',c'} \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}_{c'}(\boldsymbol{y}) - \sum_{\boldsymbol{y}\in\gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}_{n'}(\boldsymbol{y})\right) \\
&= \mathcal{M}^{(n')}(\emptyset) + \Big(\prod_{\theta\in\mathsf{path}(n')} \theta\Big)\left(\theta_{n',c'}\mathcal{M}_{c'}(\emptyset) - \mathcal{M}_{n'}(\emptyset)\right) = \mathcal{M}^{(n')}(\emptyset) + \Big(\prod_{\theta\in\mathsf{path}(n')} \theta\Big)\left(\theta_{n,c}\mathsf{m}_c - \mathsf{m}_n\right)
\end{aligned}$$

Because $\mathsf{m}_n \geq \theta_{n,c}\mathsf{m}_c$, above equation among copies of (n, c) can be bounded from above by:

$$\max_{(n',c')\in\mathsf{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) \leq \max_{n'\in\mathsf{copy}(n)} \mathcal{M}^{(n')}(\emptyset) + \Big(\min_{n'\in\mathsf{copy}(n)} \prod_{\theta\in\mathsf{path}(n')} \theta\Big)\left(\theta_{n,c}\mathsf{m}_c - \mathsf{m}_n\right).$$

We will show that $\mathsf{r}_{n,c} = \mathsf{r}_n + \mathsf{t}_n\left(\theta_{n,c}\mathsf{m}_c - \mathsf{m}_n\right)$ (Line 12 in Algorithm 8) is at most the right-hand side quantity of above inequality, thereby satisfying Equation A.7. First, we have $\mathsf{r}_n \geq \max_{(n')\in\mathsf{copy}(n)} \mathcal{M}^{(n')}(\emptyset)$ by the inductive hypothesis. Next, we want to show that $\mathsf{t}_n \leq \min_{n'\in\mathsf{copy}(n)} \prod_{\theta\in\mathsf{path}(n')} \theta$. For a given node c, suppose this holds for t_n of every parent node $n \in \mathsf{pa}(c)$. Then we have

$$\mathsf{t}_c = \min_{n\in\mathsf{pa}(c)} \theta_{n,c}\mathsf{t}_n \leq \min_{n\in\mathsf{pa}(c)} \theta_{n,c}\Big(\min_{n'\in\mathsf{copy}(n)} \prod_{\theta\in\mathsf{path}(n')} \theta\Big) = \min_{c'\in\mathsf{copy}(c)} \prod_{\theta\in\mathsf{path}(c')} \theta$$

For simplicity, we say $\theta_{n,c} = 1$ for a product node n.

Finally, assume that Equation A.7 holds for edges (p, n) where $p \in \mathsf{pa}(n)$, and we will show that Equation A.6 must hold then for node n. r_n, which is set to $\max_{p\in\mathsf{pa}(n)} \mathsf{r}_{p,n}$ in Algorithm 8, satisfies Equation A.6 as follows:

$$\max_{p\in\mathsf{pa}(n)} \mathsf{r}_{p,n} \geq \max_{p\in\mathsf{pa}(n)} \max_{(p',n')\in\mathsf{copy}((p,n))} \mathcal{M}^{(n')}(\emptyset) = \max_{n'\in\mathsf{copy}(n)} \mathcal{M}^{(n')}(\emptyset).$$

This concludes the proof of Proposition 6.1. □

Proof of Lemma 3. To show property (1) $\mathcal{M}^{(n')}(\emptyset) = \sum_{\boldsymbol{y} \in \mathsf{val}(\mathbf{X} \setminus \mathbf{Q})} \mathcal{M}^{(n')}(\boldsymbol{y})$, first observe that $\mathcal{M}^{(n')}$ fixes every $\mathbf{Q}$-deterministic node to always return the value of one of its children and thus can be simplified by removing those nodes and directly connecting its parent to the appropriate child node. This results in a smooth and decomposable PC with the normal types of sum and product nodes. Then (1) simply holds by the fact that smooth and decomposable PCs allow marginal inference by feedforward evaluation.

Property (2) $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}(\boldsymbol{q})$ holds for any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$ if and only if $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}'_{\boldsymbol{q}}(\emptyset)$, as computing the marginal probability of $\boldsymbol{q}$ is equivalent to evaluating the $\boldsymbol{q}$-subcircuit. Note that for any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$, the ancestor nodes of n' in $\mathcal{M}^{(n')}$ are equivalent to those in the $\boldsymbol{q}$-subcircuit. On the other hand, $\mathcal{M}^{(n')}_{n'}(\emptyset) = \mathcal{M}_{n'}(\emptyset)$ upper bounds $\max_{\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}} n(\boldsymbol{q})$ (recall Equation 6.2), hence must be at least $n(\boldsymbol{q})$. Therefore, at the root nodes, $\mathcal{M}^{(n')}$ must evaluate to at least $\mathcal{C}'_{\boldsymbol{q}}$. □

Bibliography

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69, 2018.

Francis Bach and Michael Jordan. Thin junction trees. *Advances in neural information processing systems*, 14, 2001.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. `http://www.fairmlbook.org`.

Riccardo Bellazzi and Blaz Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International journal of medical informatics*, 77(2):81–97, 2008.

Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, page 0049124118782533, 2018.

M. Bilgic and L. Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *JAIR*, 2011.

José M. Bioucas-Dias and Mário A. T. Figueiredo. Bayesian image segmentation using hidden fields: Supervised, unsupervised, and semi-supervised formulations. In *24th European Signal Processing Conference (EUSIPCO)*, pages 523–527, 2016.

Avrim Blum and Kevin Stangl. Recovering from biased data: Can fairness constraints improve accuracy? In *1st Symposium on Foundations of Responsible Computing*, 2020.

T. Calders and S. Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 2010.

Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, pages 3992–4001, 2017.

A. Choi, Y. Xue, and A. Darwiche. Same-Decision Probability: A confidence measure for threshold-based decisions. *IJAR*, 2012.

A. Choi, D. Kisa, and A. Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *ECSQARU*, pages 121–132, 2013.

Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)*, 2017.

YooJung Choi and Guy Van den Broeck. On robust trimming of bayesian network classifiers. In *IJCAI*, 2018.

YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal feature selection for decision robustness in bayesian networks. In *IJCAI*, 2017.

YooJung Choi, Golnoosh Farnadi, Behrouz Babaki, and Guy Van den Broeck. Learning fair naive bayes classifiers by discovering and eliminating discrimination patterns. In *AAAI*, 2020a.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. Oct 2020b.

YooJung Choi, Meihua Dang, and Guy Van den Broeck. Group fairness by probabilistic modeling with latent fair decisions. In *AAAI*, Feb 2021.

YooJung Choi, Tal Friedman, and Guy Van den Broeck. Solving marginal map exactly by probabilistic circuit transformations. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022.

Alexandra Chouldechova. Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

Diarmaid Conaty, Denis D Maua, and Casio P de Campos. Approximation complexity of maximum a posteriori inference in sum-product networks. In *The 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI, 2017.

Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*, 2018.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806. ACM, 2017.

Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*, sep 2020.

Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: A fast and accurate learner of structured-decomposable probabilistic circuits. *International Journal of Approximate Reasoning*, 140:92–115, jan 2022. ISSN 0888-613X.

Adnan Darwiche. A logical approach to factoring belief networks. In *Proceedings of KR*, pages 409–420, 2002.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009a. doi: 10.1017/CBO9780511811357.

Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009b.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

Sanjoy Dasgupta. Learning polytrees. *arXiv preprint arXiv:1301.6688*, 2013.

Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings. *Proceedings on Privacy Enhancing Technologies*, 2015(1):92–112, 2015.

Cassio P de Campos. New complexity results for map in bayesian networks. In *IJCAI*, volume 11, pages 2100–2106, 2011.

Rina Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.

Rina Dechter and Robert Mateescu. AND/OR search spaces for graphical models. *Artif. Intell.*, 171 (2-3):73–106, 2007. doi: 10.1016/j.artint.2006.11.003. URL `https://doi.org/10.1016/j.artint.2006.11.003`.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

Joseph G Ecker. Geometric programming: methods, computations and applications. *SIAM review*, 22(3):338–362, 1980.

M. Feldman, S.A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, 2015.

Riccardo Fogliato, Max G'Sell, and A. Chouldechova. Fairness evaluation in presence of biased noisy labels. In *AISTATS*, 2020.

T. Gao and D. Koller. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

Francisco J Gimenez, Yirong Wu, Elizabeth S Burnside, and Daniel L Rubin. A novel method to assess incompleteness of mammography reports. In *AMIA Annual Symposium Proceedings*, volume 2014, page 1758. American Medical Informatics Association, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2(2), 2017.

M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In *NeurIPS*, 2016.

Loren Henderson, Cedric Herring, Hayward Derrick Horton, and Melvin Thomas. Credit where credit is due?: Race, gender, and discrimination in the credit scores of business startups. *The Review of Black Political Economy*, 42(4):459–479, 2015.

Jinbo Huang, Mark Chavira, and Adnan Darwiche. Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 143–148, 2006.

Heinrich Jiang and Ofir Nachum. Identifying and correcting label bias in machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 702–712, 2020.

Ray Jiang, Aldo Pacchiano, Tom Stepleton, Heinrich Jiang, and Silvia Chiappa. Wasserstein fair classification. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*, jul 2019.

Faisal Kamiran and Toon Calders. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–6. IEEE, 2009.

Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pages 2564–2572, 2018.

Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. In *NeurIPS*, 2019a.

Pasha Khosravi, Yitao Liang, YooJung. Choi, and Guy Van den Broeck. What to expect of classifiers? reasoning about logistic regression with missing features. In *IJCAI*, 2019b.

Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. Handling missing data in decision trees: A probabilistic approach. In *The Art of Learning with Missing Values Workshop at ICML (Artemiss)*, jul 2020.

Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014a.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *KR*, 2014b.

Igor Kiselev and Pascal Poupart. Policy optimization by marginal-map probabilistic inference in generative models. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1611–1612, 2014.

Peter J Kolesar. A branch and bound algorithm for the knapsack problem. *Management science*, 13 (9):723–735, 1967.

Thomas Kollar and Nicholas Roy. Efficient optimization of information-theoretic exploration in slam. In *AAAI*, volume 8, pages 1369–1375, 2008.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193.

Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

A. Krause and C. Guestrin. Optimal value of information in graphical models. *JAIR*, 2009.

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pages 4066–4076, 2017.

David H Laidlaw, Kurt W Fleischer, and Alan H Barr. Partial-volume bayesian classification of material mixtures in mr volume data using voxel histograms. *IEEE transactions on medical imaging*, 17(1):74–86, 1998.

Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9, 2016.

Junkyu Lee, Radu Marinescu, and Rina Dechter. Applying marginal map search to probabilistic conformant planning: Initial results. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, August 2017. URL http://starai.cs.ucla.edu/papers/LiangUAI17.pdf.

Anji Liu, Stephan Mandt, and Guy Van den Broeck. Lossless compression with probabilistic circuits. In *International Conference on Learning Representations (ICLR)*, apr 2022.

Peter Lucas. Expert knowledge and its role in learning bayesian networks in medicine: an appraisal. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 156–166. Springer, 2001.

Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–510. ACM, 2011.

Koray Mancuhan and Chris Clifton. Combating discrimination using bayesian networks. *Artificial intelligence and law*, 22(2):211–238, 2014.

Radu Marinescu and Rina Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artif. Intell.*, 173(16-17):1457–1491, 2009. doi: 10.1016/j.artint.2009.07.003. URL https://doi.org/10.1016/j.artint.2009.07.003.

Radu Marinescu, Junkyu Lee, Rina Dechter, and Alexander T. Ihler. AND/OR search for marginal MAP. *J. Artif. Intell. Res.*, 63:875–921, 2018. doi: 10.1613/jair.1.11265. URL https://doi.org/10.1613/jair.1.11265.

Robert Mateescu, Rina Dechter, and Radu Marinescu. AND/OR multi-valued decision diagrams (aomdds) for graphical models. *J. Artif. Intell. Res.*, 33:465–519, 2008. doi: 10.1613/jair.2605. URL https://doi.org/10.1613/jair.2605.

Denis Deratani Mauá, Heitor Ribeiro Reis, Gustavo Perez Katague, and Alessandro Antonucci.

Two reformulation approaches to maximum-a-posteriori inference in sum-product networks. In *International Conference on Probabilistic Graphical Models*, pages 293–304. PMLR, 2020.

Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6:355–378, 2019.

Jun Mei, Yong Jiang, and Kewei Tu. Maximum a posteriori inference in sum-product networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69, 2006.

Eva Millán and José Luis Pérez-De-La-Cruz. A Bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction*, 12(2-3):281–330, 2002.

Michael Munie and Yoav Shoham. Optimal testing of structured knowledge. In *Proceedings of the 23rd National Conference on Artificial intelligence*, pages 1069–1074, 2008.

Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26(9):917–922, 1977.

U. Oztok, A. Choi, and A. Darwiche. Solving PP^{PP}-complete problems using knowledge compilation. In *KR*, 2016.

James D Park and Adnan Darwiche. Solving map exactly using systematic search. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 459–468, 2002.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Knot Pipatsrisawat and Adnan Darwiche. New compilation languages based on structured decomposability. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 517–522, 2008.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pages 5680–5689, 2017.

Hoifung Poon and Pedro Domingos. Sum-product networks: a new deep architecture. In *UAI*, pages 337–346, 2011.

Propublica. Compas analysis. `https://github.com/propublica/compas-analysis`, 2016.

Tahrima Rahman and Vibhav Gogate. Learning ensembles of cutset networks. In *AAAI*, pages 3301–3307. AAAI Press, 2016.

Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECML/PKDD*, 2014.

Tahrima Rahman, Sara Rouhani, and Vibhav Gogate. Novel upper bounds for the constrained most probable explanation task. *Advances in Neural Information Processing Systems*, 34, 2021.

Andrea Romei and Salvatore Ruggieri. A multidisciplinary survey on discrimination analysis. *The Knowledge Engineering Review*, 29(5):582–638, 2014.

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1–2):273–302, 1996.

Sara Rouhani, Tahrima Rahman, and Vibhav Gogate. Algorithms for the nearest assignment problem. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5096–5102, 2018.

Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*, pages 793–810. ACM, 2019.

Nikil Roashan Selvam, Guy Van den Broeck, and YooJung Choi. Certifying fairness of probabilistic circuits. *(submitted)*, May 2022.

Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable operations for arithmetic circuits of probabilistic models. In *Advances in Neural Information Processing Systems*, pages 3936–3944, 2016.

Alicia Solow-Niederman, YooJung Choi, and Guy Van den Broeck. The institutional life of algorithmic risk assessment. *BTLJ*, 2019.

Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. Fairtest: Discovering unwarranted associations in data-driven applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 401–416. IEEE, 2017.

Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. *AAAI Tutorial*, 2020.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2021.

Blake Woodworth, Suriya Gunasekar, Mesrob I Ohannessian, and Nathan Srebro. Learning non-discriminatory predictors. In *Conference on Learning Theory*, pages 1920–1953, 2017.

Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

Alice Xiang. Reconciling legal and technical approaches to algorithmic bias. *Tennessee Law Review*, 88:649, 2020.

Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *20th International Conference on Artificial Intelligence and Statistics*, pages 962–970, 2017.

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *ICML*, pages 325–333, 2013.

Y. Zhang and Q. Ji. Efficient sensor selection for active information fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 2010.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *EMNLP*, 2017.