Group Fairness by Probabilistic Modeling with Latent Fair Decisions

YooJung Choi, Meihua Dang, and Guy Van den Broeck
Computer Science Department
University of California, Los Angeles
{yjchoi,mhdang,guyvdb}@cs.ucla.edu

Abstract

Machine learning systems are increasingly being used to make impactful decisions such as loan applications and criminal justice risk assessments, and as such, ensuring fairness of these systems is critical. This is often challenging as the labels in the data are biased. This paper studies learning fair probability distributions from biased data by explicitly modeling a latent variable that represents a hidden, unbiased label. In particular, we aim to achieve demographic parity by enforcing certain independencies in the learned model. 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, an expressive and tractable probabilistic model, 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.

1 Introduction

As machine learning algorithms are being increasingly used in real-world decision making scenarios, there has been growing concern that these methods may produce decisions that discriminate against particular groups of people. The relevant applications include online advertising, hiring, loan approvals, and criminal risk assessment (Datta, Tschantz, and Datta 2015; Barocas and Selbst 2016; Chouldechova 2017; Berk et al. 2018). 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.

In this paper, we study the problem of enforcing demographic parity in probabilistic classifiers. In particular, we focus on the fact that class labels in the data are often biased, and then propose 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, we show 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.

Our 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. Secondly, we propose an algorithm to learn probabilistic circuits (PCs) (Vergari, Di Mauro, and Van den Broeck 2019), a type of tractable probabilistic models, 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 we have a latent variable. Finally, we evaluate our approach empirically on synthetic and real-world datasets, comparing against existing fair learning methods as well as a baseline we propose 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.

2 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, Hardt, and Narayanan (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, Price, and Srebro 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, Price, and Srebro 2016; Pleiss et al. 2017), and (3) in-processing which incorporates fairness constraints directly in learning/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, G'Sell, and Chouldechova 2020; Blum and Stangl 2020; Jiang and Nachum 2020). For instance, Blum and Stangl (2020) studies 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; whereas, 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. 2020). 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 Latent Fair Decisions

We use uppercase letters (e.g., X) for discrete random variables (RVs) and lowercase letters (x) for their assignments. Negation of a binary assignment x is denoted by $\bar{x}$. Sets of RVs are denoted by bold uppercase letters ($\mathbf{X}$), and their joint assignments by bold lowercase ($\mathbf{x}$). Let S denote a *sensitive attribute*, such as gender or race, and let $\mathbf{X}$ be the *non-sensitive attributes* or features. In this paper, we assume S is a binary variable 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 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.

3.1 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 recidivism 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 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

Figure 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 $\mathbf{X}$.

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. In other words, the probability of D being positive depends on both 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 latent fair decision 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 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 $s, \mathbf{x}, d$,

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

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 learn the relationship between D_f and other features, and may assume the latent decision variable to be 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 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 by Calders and Verwer (2010) which had a naive Bayes assumption among the non-sensitive features; i.e., variables in $\mathbf{X}$ are conditionally independent given the sensitive attribute S and D_f. 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.

The latent variable not only encodes the intuition that observed labels may be biased, but it also has advantages in achieving high likelihood with respect to data. Consider an alternative way to satisfy statistical parity: by directly enforcing independence between the observed decision variable D and sensitive attributes $\mathbf{S}$: see Figure 1b. We will show that, on the same data, our proposed model can always achieve marginal likelihood at least as high as the model without a latent decision variable. We can enforce the independence of D and $\mathbf{S}$ by setting the latent variable D_f to always be equal to D, which results in a marginal distribution over $S, \mathbf{X}, D$ with the same independencies as in Figure 1b:

$$\begin{aligned}&\Pr(s\mathbf{x}d) \\ &= \Pr(\mathbf{x} \mid s, D_f{=}1)\Pr(d \mid s, D_f{=}1)\Pr(s)\Pr(D_f{=}1) \\ &\quad + \Pr(\mathbf{x} \mid s, D_f{=}0)\Pr(d \mid s, D_f{=}0)\Pr(s)\Pr(D_f{=}0) \\ &= \Pr(\mathbf{x} \mid sd)\Pr(s)\Pr(d)\end{aligned}$$

Thus, any fair distribution without the latent decision can also be represented by our latent variable approach. In addition, our approach will achieve strictly better likelihood if the observed data does not satisfy demographic parity, because it can also model distributions where D and S are dependent.

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.

4 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. In the following, we give some background on *probabilistic circuits (PCs)* and show how they satisfy each of the above criteria. Then we will describe our proposed algorithm to learn fair probabilistic circuits from data.

4.1 Probabilistic Circuits

Representation *Probabilistic circuits (PCs)* (Vergari et al. 2020) refer to a family of tractable probabilistic models

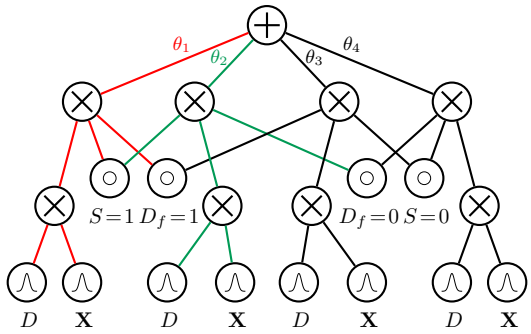

Figure 2: A probabilistic circuit over variables $S, \mathbf{X}, D, D_f$

including arithmetic circuits (Darwiche 2002, 2003), sum-product networks (Poon and Domingos 2011), and cutset networks (Rahman, Kothalkar, and Gogate 2014). A probabilistic circuit $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ over RVs $\mathbf{X}$ is characterized by its structure $\mathcal{G}$ and parameters $\boldsymbol{\theta}$. 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 $[X = 1]$. Parameters $\boldsymbol{\theta}$ are each associated with an input edge to a sum node. Note that a subcircuit rooted at an inner node of a PC is itself a valid PC. Figure 2 depicts an example probabilistic circuit.[1]

Let $\mathsf{ch}(n)$ be the set of children nodes of an inner node n. Then a probabilistic circuit $\mathcal{C}$ over RVs $\mathbf{X}$ defines a joint distribution $\Pr_{\mathcal{C}}(\mathbf{X})$ in a recursive way as follows:

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

Intuitively, a product node n defines a factorized distribution, and a sum node n defines a mixture model parameterized by weights $\{\theta_{n,c}\}_{c \in \mathsf{ch}(n)}$. $\Pr_n$ is also called the *output* of n.

Properties and inference A strength of probabilistic circuits is that (1) they are expressive, achieving high likelihoods on density estimation tasks (Rahman and Gogate 2016; Liang, Bekker, and Van den Broeck 2017; Peharz et al. 2020), and (2) they support tractable probabilistic inference, enabled by certain structural properties. In particular, PCs support efficient marginal inference if they are smooth and decomposable. A circuit is said to be *smooth* if for every sum node all of its children depend on the same set of variables; it is *decomposable* if for every product node its children depend on disjoint sets of variables (Darwiche and Marquis 2002). Given a smooth and decomposable probabilistic circuit, computing the marginal probability for any partial evidence is reduced to simply evaluating the circuit bottom-up. This also implies tractable computation of conditional probabilities, which are ratios of marginals. Thus, we can make predictions in time linear in the size of the circuit.

Another useful structural property is *determinism*; a circuit 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. We also exploit this property for learning PCs with latent variables, which we will later describe in detail.

Encoding independence assumptions Next, we demonstrate how we encode the independence assumptions of a fair distribution as in Figure 1a in a probabilistic circuit. Recall the example PC in Figure 2: regardless of parameterization, this circuit structure always encodes a distribution where D is independent of $\mathbf{X}$ given S and D_f. To prove this, we first observe that the four product nodes in the second layer each correspond to four possible assignments to S and D_f. For instance, the left-most product node returns a non-zero output only if the input sets both $S = 1$ and $D_f = 1$. 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. With no restriction on these parameters, the circuit structure 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:

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

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. We first study parameter learning in the next section, then structure learning in Section 4.3.

4.2 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. 2014). 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:

$$\theta_{n,c} = F_{\mathcal{D}}(n, c) / \sum_{c \in \mathsf{ch}(n)} F_{\mathcal{D}}(n, c) \tag{1}$$

Here, $F_{\mathcal{D}}(n, c)$ is called the *circuit flow* of edge (n, c) given $\mathcal{D}$, and it counts the number of data samples in $\mathcal{D}$ that "activate" this edge. For example, in Figure 2, the edges activated

[1] The features $\mathbf{X}$ and D are shown as leaf nodes for graphical conciseness, but refer to sub-circuits over the respective variables.

by sample $\{D_f = 1, S = 1, d, \mathbf{x}\}$, for any assignments $d, \mathbf{x}$, are colored red.[2]

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 2009). 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 now propose an EM parameter learning algorithm for PCs that does not explicitly complete the data, but rather utilizes circuit flows. In particular, we introduce the notion of *expected flows*, which is defined as the following for a given circuit $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ over RVs $\mathbf{Z}$ and an incomplete dataset $\mathcal{D}$:

$$\begin{aligned}\mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c) :=& \mathbb{E}_{\Pr_{\mathcal{C}}}[F_{\mathcal{D}_i}(n, c)] \\ =& \sum_{\mathcal{D}_i \in \mathcal{D}} \sum_{\mathbf{z} \models \mathcal{D}_i} \Pr_{\mathcal{C}}(\mathbf{z} | \mathcal{D}_i) \cdot F_{\mathbf{z}}(n, c).\end{aligned}$$

Here, $\mathcal{D}_i$ denotes the i-th sample in the dataset, and $\mathbf{z} \models \mathcal{D}_i$ are the possible completions of sample $\mathcal{D}_i$. For example, in Figure 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. Then, using expected flows, we can perform both the E- and M-step by the following closed-form solution.

Proposition 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:

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

Note that this is very similar to the ML estimate from complete data in Equation 1, except that expected flows are used instead of circuit flows. Furthermore, 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. We refer to Appendix A for details on computing the expected flows and proof for above proposition.

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 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 $\Pr(d|s, d_f) = [d = d_f]$.

4.3 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, Vergari, and Van den Broeck 2020).[3] 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.

5 Experiments

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) by Calders and Verwer (2010) and PCs without latent variable (NLATPC) as described in Sec 3. We also compare against existing 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 subject to fairness constraints; 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 using modified objective functions.

[2] See Appendix A for a formal definition and proof of Equation 1.

[3] PCs learned this way also satisfy properties such as structured decomposability that are not necessary for our use case.

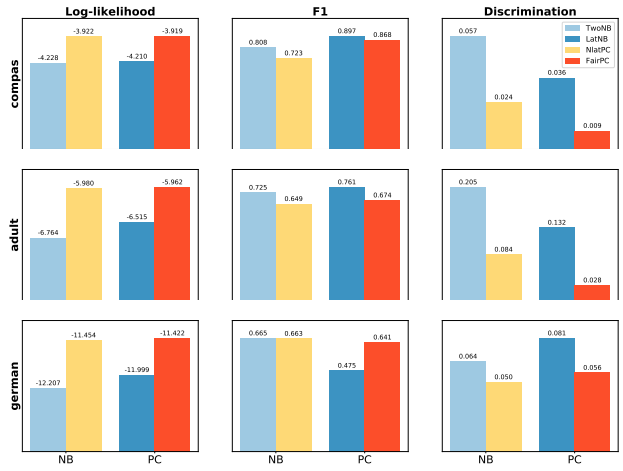

Figure 3: Comparison of fair probability distributions. **Columns:** log-likelihood, F1-score, discrimination score (higher is better for the first two; lower is better for last). **Rows:** COMPAS, Adult, German datasets. The four bars in each graph from left to right are: 1) 2NB, 2) LATNB, 3) NLATPC,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

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.

5.1 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 without discrimination with respect to a protected attribute: "sex" for Adult and German, and "ethnicity" for COMPAS. As pre-processing, we discretize numerical features (e.g. age), 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 summarizes the result. The bars, from left to right, correspond to 2NB, LATNB, NLATPC, and FAIRPC. First and last two bars in each graph correspond to NB and PC models, respectively. Blue bars denote non-latent model, and yellow/orange denote latent-variable approach.

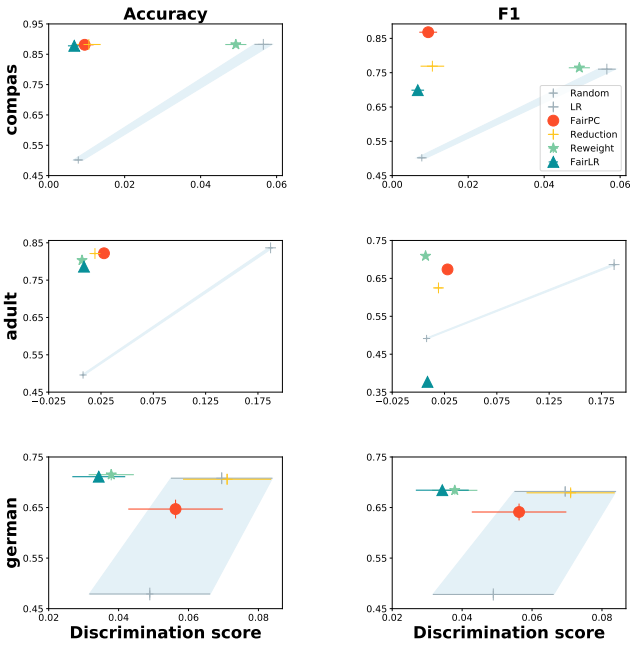

Figure 4: Predictive performance (y-axis) vs. discrimination score (x-axis) for FAIRPC and fair classification methods (FAIRLR, REDUCTION, REWEIGHT), in addition with two trivial baselines (RAND and LR). **Columns:** accuracy, F1-score. **Rows:** COMPAS, Adult, German datasets.

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 argument made in Section 3 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 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. Moreover, as expected, PCs 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 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 logis-

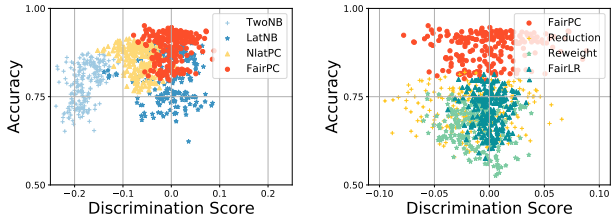

Figure 5: Accuracy (y-axis) vs. discrimination score (x-axis) on synthetic datasets. We compare FAIRPC with 2NB, LATNB, NLATPC (left) and with REDUCTION, REWEIGHT, FAIRLR (right). Each dot is a single run on a generated dataset using the method indicated by its color.

tic 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 highly noisy to draw a statistically significant conclusion. This may be explained by the fact that the dataset is relatively small with 1000 samples.

5.2 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. The sub-circuits of $\mathcal{C}_{true}$ over features $\mathbf{X}$ are randomly generated tree distributions, and their parameters are randomly initialized with 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 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. Moreover FAIRPC also outperforms the other probabilistic methods in terms of likelihood; see Appendix B.

We also compare FAIRPC to FAIRLR, REDUCTION, and REWEIGHT, the results visualized in Figure 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.

5.3 Additional experiments

Appendix B includes learning curves, statistical tests, and detailed performance of our real-world data experiments, as well as the following additional experiments. We empirically validated that initializing parameters using prior knowledge as described in Section 4.2 indeed converges closer to the true distribution of $\Pr(D|S, D_f)$ than randomly initializing parameters. In addition, as mentioned in Section 4.2, our method can be applied even on datasets with missing values, with no change to the algorithm. We demonstrate this empirically and show that our approach still gets comparably good performance for density estimation.

6 Conclusion

In this paper, we 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 showed that our method consistently achieves the highest log-likelihoods and a low discrimination score. 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.

Acknowledgments This work is partially supported by NSF grants #IIS-1943641, #IIS-1633857, #CCF-1837129, DARPA grant #N66001-17-2-4032, a Sloan Fellowship, Intel, and Facebook.

References

Agarwal, A.; Beygelzimer, A.; Dudik, M.; Langford, J.; and Wallach, H. 2018. A Reductions Approach to Fair Classifi-

cation. In *International Conference on Machine Learning*, 60–69.

Barocas, S.; Hardt, M.; and Narayanan, A. 2019. *Fairness and Machine Learning*. fairmlbook.org. http://www.fairmlbook.org.

Barocas, S.; and Selbst, A. D. 2016. Big data's disparate impact. *Calif. L. Rev.* 104: 671.

Berk, R.; Heidari, H.; Jabbari, S.; Kearns, M.; and Roth, A. 2018. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research* 0049124118782533.

Blum, A.; and Stangl, K. 2020. Recovering from Biased Data: Can Fairness Constraints Improve Accuracy? In *1st Symposium on Foundations of Responsible Computing*.

Calders, T.; and Verwer, S. 2010. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* 21(2): 277–292.

Calmon, F.; Wei, D.; Vinzamuri, B.; Ramamurthy, K. N.; and Varshney, K. R. 2017. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, 3992–4001.

Choi, A.; and Darwiche, A. 2017. On Relaxing Determinism in Arithmetic Circuits. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)*.

Choi, Y.; Farnadi, G.; Babaki, B.; and Van den Broeck, G. 2020. Learning Fair Naive Bayes Classifiers by Discovering and Eliminating Discrimination Patterns. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

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

Chow, C. K.; and Liu, C. N. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* .

Corbett-Davies, S.; Pierson, E.; Feller, A.; Goel, S.; and Huq, A. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 797–806. ACM.

Dang, M.; Vergari, A.; and Van den Broeck, G. 2020. Strudel: Learning Structured-Decomposable Probabilistic Circuits. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*.

Darwiche, A. 2002. A Logical Approach to Factoring Belief Networks. In *Proceedings of KR*, 409–420.

Darwiche, A. 2003. A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM* 50(3): 280–305.

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

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

Datta, A.; Tschantz, M. C.; and Datta, A. 2015. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. *Proceedings on privacy enhancing technologies* 2015(1): 92–112.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, 214–226. ACM.

Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 259–268. ACM.

Fogliato, R.; G'Sell, M.; and Chouldechova, A. 2020. Fairness Evaluation in Presence of Biased Noisy Labels. In *AISTATS*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, 3315–3323.

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

Jiang, R.; Pacchiano, A.; Stepleton, T.; Jiang, H.; and Chiappa, S. 2019. Wasserstein fair classification. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Kamiran, F.; and Calders, T. 2009. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, 1–6. IEEE.

Kamishima, T.; Akaho, S.; Asoh, H.; and Sakuma, J. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 35–50. Springer.

Kearns, M.; Neel, S.; Roth, A.; and Wu, Z. S. 2018. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *International Conference on Machine Learning*, 2564–2572.

Kisa, D.; Van den Broeck, G.; Choi, A.; and Darwiche, A. 2014. Probabilistic Sentential Decision Diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

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

Kusner, M. J.; Loftus, J.; Russell, C.; and Silva, R. 2017. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, 4066–4076.

Liang, Y.; Bekker, J.; and Van den Broeck, G. 2017. Learning the Structure of Probabilistic Sentential Decision Diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. URL http://starai.cs.ucla.edu/papers/LiangUAI17.pdf.

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

Peharz, R.; Vergari, A.; Stelzner, K.; Molina, A.; Shao, X.; Trapp, M.; Kersting, K.; and Ghahramani, Z. 2020. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, 334–344. PMLR.

Pleiss, G.; Raghavan, M.; Wu, F.; Kleinberg, J.; and Weinberger, K. Q. 2017. On fairness and calibration. In *Advances in Neural Information Processing Systems*, 5680–5689.

Poon, H.; and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 689–690. IEEE.

Propublica. 2016. COMPAS Analysis. https://github.com/propublica/compas-analysis.

Rahman, T.; and Gogate, V. 2016. Merging Strategies for Sum-Product Networks: From Trees to Graphs. In *UAI*.

Rahman, T.; Kothalkar, P.; and Gogate, V. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, 630–645. Springer.

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

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

Vergari, A.; Choi, Y.; Peharz, R.; and Van den Broeck, G. 2020. Probabilistic circuits: Representations, inference, learning and applications. *AAAI Tutorial* .

Vergari, A.; Di Mauro, N.; and Van den Broeck, G. 2019. Tractable probabilistic models: Representations, algorithms, learning, and applications. http://web.cs.ucla.edu/ guyvdb/slides/TPMTutorialUAI19.pdf.

Woodworth, B.; Gunasekar, S.; Ohannessian, M. I.; and Srebro, N. 2017. Learning Non-Discriminatory Predictors. In *Conference on Learning Theory*, 1920–1953.

Zafar, M. B.; Valera, I.; Gomez Rodriguez, M.; and Gummadi, K. P. 2017. Fairness Constraints: Mechanisms for Fair Classification. In *20th International Conference on Artificial Intelligence and Statistics*, 962–970.

Zemel, R.; Wu, Y.; Swersky, K.; Pitassi, T.; and Dwork, C. 2013. Learning fair representations. In *International Conference on Machine Learning*, 325–333.

A Parameter Learning using Expected Flows

Here we formally define the circuit flow and expected flows, and provide details on EM parameter learning using expected flows as well as proof of correctness. We assume smooth, decomposable, and deterministic PCs in the following sections.

A.1 Definitions

Definition 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 $\mathsf{supp}(n) := \{\mathbf{z} : \mathcal{C}_n(\mathbf{z}) > 0\}$ is 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.

We can now express circuit flows and expected flows in terms of contexts. 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.

Definition 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

$$F_{\mathbf{z}}(n, c) = [\mathbf{z} \in \gamma_n \cap \gamma_c]. \tag{2}$$

Definition 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

$$\mathrm{EF}_{\mathbf{e},\theta}(n, c) := \mathbb{E}_{\mathbf{z} \sim \Pr_{\mathcal{C}}(\cdot|\mathbf{e})}[F_{\mathbf{z}}(n, c)]. \tag{3}$$

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 $\mathbf{z}$. Similarly, 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 $\mathbf{E} \subseteq \mathbf{Z}$.

A.2 Computing the Expected Flow

Next we describe how to compute the expected flows. First, focusing on the expected flow given a single partial assignment $\mathbf{e}$, we can express the expected flow as the following using Equations 2 and 3.

$$\mathrm{EF}_{\mathbf{e},\theta}(n, c) = \mathbb{E}_{\mathbf{z} \sim \Pr_{\mathcal{C}}(\cdot|\mathbf{e})}[\mathbf{z} \in \gamma_n \cap \gamma_c] = \Pr_{\mathcal{C}}(\gamma_n \cap \gamma_c|\mathbf{e}) \tag{4}$$

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,

$$\sum_{p \in \mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p, n) = \sum_{p \in \mathsf{pa}(n)} \Pr_{\mathcal{C}}(\gamma_p \cap \gamma_n | \mathbf{e}) = \Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}). \quad (5)$$

We can observe from Equations 4 and 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,

$$\begin{aligned}\mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(n, c) &= \Pr_{\mathcal{C}}(\gamma_n \cap \gamma_c | \mathbf{e}) = \frac{\Pr_{\mathcal{C}}(\mathbf{e}, \gamma_c | \gamma_n) \Pr_{\mathcal{C}}(\gamma_n)}{\Pr_{\mathcal{C}}(\mathbf{e})} \\ &= \frac{\Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}) \Pr_{\mathcal{C}}(\mathbf{e}, \gamma_c | \gamma_n) \Pr_{\mathcal{C}}(\gamma_n)}{\Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}) \Pr_{\mathcal{C}}(\mathbf{e})} \\ &= \frac{\Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}) \Pr_{\mathcal{C}}(\mathbf{e}, \gamma_c | \gamma_n)}{\Pr_{\mathcal{C}}(\mathbf{e} | \gamma_n)} \\ &= \left(\sum_{p \in \mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p, n) \right) \frac{\theta_{n,c} \Pr_c(\mathbf{e})}{\Pr_n(\mathbf{e})}. \quad (6)\end{aligned}$$

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:

$$\begin{aligned}\gamma_n &= \gamma_n \cap \mathsf{supp}(n) \subseteq \gamma_n \cap \mathsf{supp}(c) \quad (7) \\ &\subseteq \bigcup_{p \in \mathsf{pa}(c)} \gamma_p \cap \mathsf{supp}(c) = \gamma_c\end{aligned}$$

where Equation 7 follows from Definition 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:

$$\begin{aligned}\mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(n, c) &= \Pr_{\mathcal{C}}(\gamma_n \cap \gamma_c | \mathbf{e}) = \Pr_{\mathcal{C}}(\gamma_n | \mathbf{e}) \\ &= \sum_{p \in \mathsf{pa}(n)} \mathrm{EF}_{\mathbf{e},\boldsymbol{\theta}}(p, n). \quad (8)\end{aligned}$$

Therefore, Equations 6 and 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.

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

// traverse PSDD nodes by visiting parents before children
for *n in PSDD* $\mathcal{C}$ **do**
  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**
      $\mathrm{EF}(n, c) \leftarrow \mathrm{EF}(n) \cdot \frac{\theta_{n,c} \cdot \Pr_{\mathcal{C}}(c)}{\Pr_{\mathcal{C}}(n)}$
  else if *n is a product node* **then**
    for c *in* $\mathsf{ch}(n)$ **do**
      $\mathrm{EF}(n, c) \leftarrow \mathrm{EF}(n)$

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.

A.3 Proof of Proposition 1

We now prove Proposition 1 which states that the following parameter update rule using expected flows is equivalent to an iteration of EM parameter learning for smooth, decomposable, and deterministic probabilistic circuits; i.e., equivalent to completing the dataset with weights then computing the maximum-likelihood parameters.

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

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,

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

B Additional Experiments

B.1 Real-world Data

Detailed results Table 1 reports the detailed results of experiments in Section 5.1. It compares 7 methods in terms of (1) log-likelihood, (2) accuracy, (3) F1-score, and (4) discrimination score on real world datasets. Bold number indicates the best result among fair probability distributions: FAIRPC, LATNB, NLATPC and 2NB. ↑ (resp. ↓) indicates that FAIRPC achieves better (resp. worse) result than the corresponding fair classification method: FAIRLR, REDUCTION or REWEIGHT.

Statistical tests Table 2 reports the pairwise Wilcoxon signed-rank test p-values for the comparisons of test log-likelihoods for each pair of probabilistic methods on real world datasets. Bold values indicate that two methods are statistically equivalent with confidence 99%.

Table 3 reports the pairwise McNemar's test p-values for the comparisons of test set prediction results for each pair of algorithms (columns) on all real world datasets (rows). Bold values indicate two methods make statistically equivalent predictions in terms of accuracy (similarity with the observed labels) with confidence 99%.

Learning curves Figure 6 shows the 10-fold CV training curves (test log-likelihoods and probability table w.r.t number of iterations) and ROC curves of FAIRPC, each line in the plot corresponding to one fold. The test set log-likelihoods are reported after the structure is learned, and thus only describe the EM parameter learning iterations. Among the three datasets, German has the highest variance perhaps from having fewer number of examples. On the other two datasets, we can observe that the learned parameters for the bias mechanism (i.e. for D, D_f, and S) are fairly consistent across different CV folds. For instance, the model learns that some negative labels ($D_f = 0$) for the majority group ($S = 0$) are flipped to positive labels in the observed data in COMPAS dataset, e.g., $P(D = 1|D_f = 0, S = 0) \approx 0.4$; whereas in the case of Adult dataset, positive labels ($D_f = 1$) of the minority group ($S = 1$) are observed as negative labels with some probability, e.g., $P(D = 0|D_f = 1, S = 1) \approx 0.6$. Moreover, the ROC curves show the predictive performance on these datasets.

B.2 Synthetic Data

Likelihood comparison Table 4 reports the average test log-likelihoods w.r.t. different numbers of non-sensitive variables, comparing the true distribution (True), LATNB, NLATPC, and FAIRPC initialized from random start (FAIRPC-rand) or prior knowledge (FAIRPC-prior). This shows that FAIRPC consistently achieves the best log-likelihoods, and LATNB performs the worst.

Initial parameters Table 4 also shows the test log-likelihoods of FAIRPC with random initialization of parameters or with prior knowledge (i.e. $D = D_f$). We can see that FAIRPC-prior achieves equal or slightly better results than FAIRPC-rand, but the difference is not significant. Figure 7 compares the initialization methods of FAIRPCbased on the conditional probability tables among D, D_f, and S w.r.t. the number of iterations. Each line in the plot is a single run with a certain random seed and number of non-sensitive features. From the plot, it is clear that prior knowledge outperforms random initialization in terms of convergence rate as well as the values they converge to. The probability tables of experiments initialized from prior knowledge converge very close to the true distribution in Section 5. However, random initialization has more variance, and some results are far from the true distribution, For example, the $P(D = 1|D_f = 0, S = 0)$ of several runs (light blue lines) are close to 1.0, far from 0.4; the $P(D = 1|D_f = 1, S = 0)$ of several runs (dark blue lines) are close to 0.4, far from 0.9. In these runs, the values of D_f are switched when $S = 0$.

B.3 Learning With Missing Values

As described in Section 4.2, 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 8 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 in Section 5. 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.

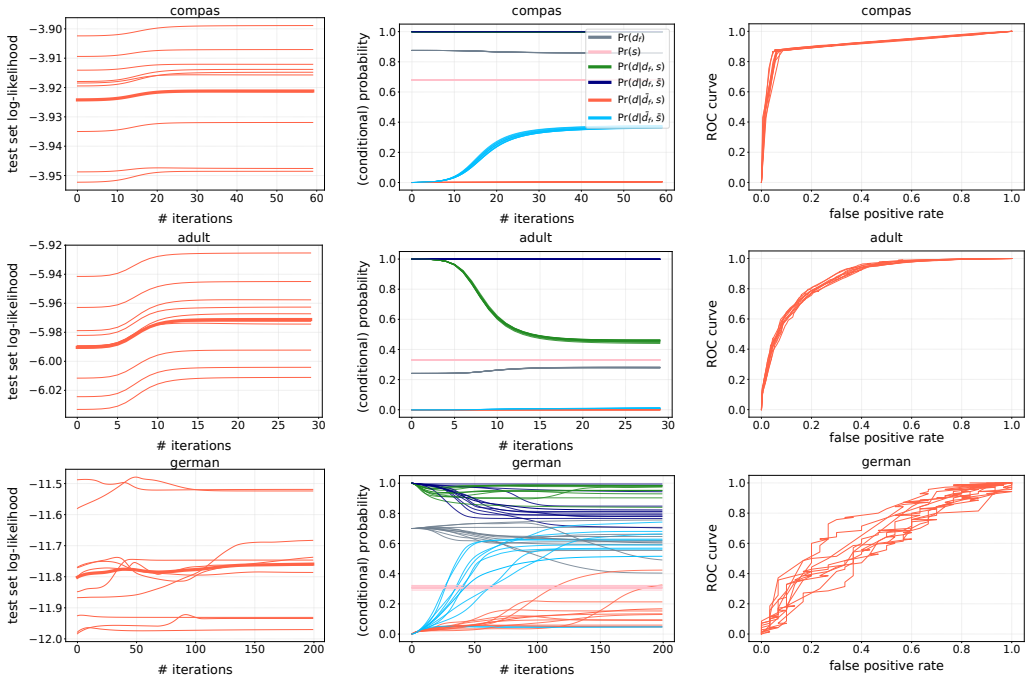

Figure 6: Training curves and ROC curves of FAIRPC

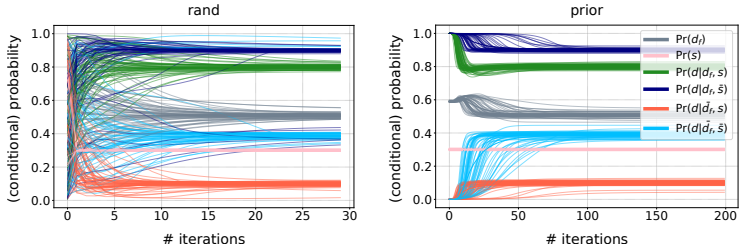

Figure 7: Compare FAIRPC initialization methods

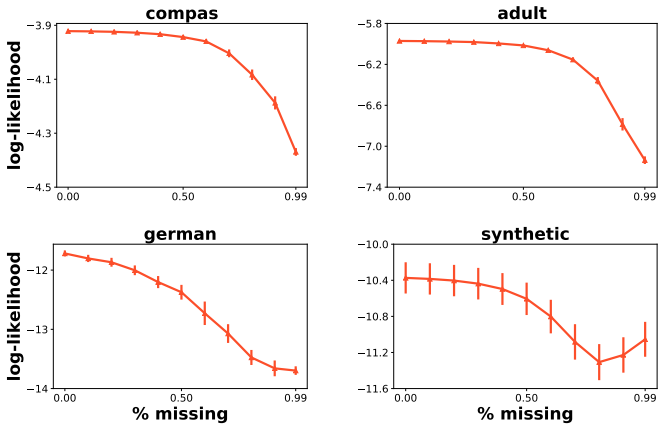

Figure 8: Test log-likelihood under different missingness percentages on real world and synthetic datasets.

Table 1: Comparison on real world datasets

Name	# Features	# Samples	Method	Log-likelihood	Accuracy	F1-score	Discrimination
COMPAS	7	60843	FAIRPC	**-3.919**	**0.881**	0.868	**0.009**
			LATNB	-4.210	**0.881**	**0.897**	0.036
			NLATPC	-3.922	0.877	0.723	0.024
			2NB	-4.228	0.879	0.808	0.057
			FAIRLR	N/A	0.878↑	0.699↑	0.007↓
			REDUCTION	N/A	0.882↓	0.769↑	0.011↑
			REWEIGHT	N/A	0.882↓	0.764↑	0.049↑
Adult	13	32561	FAIRPC	**-5.962**	0.822	0.674	**0.028**
			LATNB	-6.515	0.634	**0.761**	0.132
			NLATPC	-5.980	**0.831**	0.649	0.084
			2NB	-6.764	0.823	0.725	0.205
			FAIRLR	N/A	0.786↑	0.377↑	0.009↓
			REDUCTION	N/A	0.821↑	0.625↑	0.019↓
			REWEIGHT	N/A	0.803↑	0.709↓	0.007↓
German	21	1000	FAIRPC	**-11.422**	0.647	0.641	0.056
			LATNB	-11.999	0.499	0.476	0.081
			NLATPC	-11.454	0.680	0.663	**0.050**
			2NB	-12.207	**0.683**	**0.665**	0.064
			FAIRLR	N/A	0.711↓	0.684↓	0.034↓
			REDUCTION	N/A	0.706↓	0.679↓	0.071↑
			REWEIGHT	N/A	0.715↓	0.684↓	0.038↓

Table 2: Pairwise Wilcoxon test p-values for test log-likelihoods

	2NB LATNB	2NB NLATPC	2NB FAIRPC	LATNB NLATPC	LATNB FAIRPC	NLATPC FAIRPC
COMPAS	**9.08E-02**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.42E-302
Adult	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
German	5.53E-08	4.27E-37	1.53E-36	6.05E-19	9.03E-23	**3.34E-01**

Table 3: Pairwise McNemar's test p-values for test prediction accuracy

	2NB LATNB	2NB NLATPC	2NB FAIRPC	2NB REDUCTION	2NB REWEIGHT	2NB FAIRLR	LATNB NLATPC
COMPAS	**7.031E-02**	**3.205E-01**	**7.049E-02**	**6.489E-02**	**7.339E-02**	**5.207E-01**	**3.497E-02**
Adult	0.000E+00	1.438E-05	**3.830E-01**	**2.099E-01**	2.516E-30	2.431E-50	0.000E+00
German	1.810E-17	**8.299E-01**	2.444E-06	**3.347E-02**	4.678E-03	8.730E-03	3.158E-15
	LATNB FAIRPC	LATNB REDUCTION	LATNB REWEIGHT	LATNB FAIRLR	NLATPC FAIRPC	NLATPC REDUCTION	NLATPC REWEIGHT
COMPAS	**9.547E-01**	**4.921E-01**	**5.628E-01**	**8.014E-02**	**2.060E-02**	6.854E-05	1.009E-04
Adult	0.000E+00	0.000E+00	0.000E+00	0.000E+00	2.775E-20	1.486E-21	2.101E-60
German	1.904E-08	8.699E-19	1.094E-19	2.806E-20	1.688E-06	**4.108E-02**	7.096E-03
	NLATPC FAIRLR	FAIRPC REDUCTION	FAIRPC REWEIGHT	FAIRPC FAIRLR	REDUCTION REWEIGHT	REDUCTION FAIRLR	REWEIGHT FAIRLR
COMPAS	**7.791E-01**	**4.710E-01**	**5.478E-01**	**6.730E-02**	**6.392E-01**	7.083E-04	8.349E-04
Adult	1.017E-104	**5.240E-01**	8.664E-34	1.560E-59	5.095E-29	1.574E-65	1.338E-11
German	**1.911E-02**	4.913E-09	5.139E-10	6.094E-10	**1.060E-01**	**4.458E-01**	**6.115E-01**

Table 4: Comparison of log-likelihoods on synthetic datasets

	# non-sensitive variables										
	10	11	12	13	14	15	16	17	18	19	20
True	-6.975	-7.468	-8.091	-8.573	-9.149	-9.808	-10.210	-11.119	-11.296	-11.718	-12.476
LATNB	-7.361	-8.068	-8.614	-9.146	-9.832	-10.677	-10.989	-11.626	-12.524	-13.003	-13.592
NLATPC	-7.059	-7.612	-8.224	-8.717	-9.392	-10.057	-10.537	-11.408	-11.680	-12.207	-12.971
FAIRPC-rand	-7.024	-7.540	-8.166	-8.644	-9.281	-9.920	-10.407	-11.269	-11.513	-11.973	-12.755
FAIRPC-prior	-7.022	-7.540	-8.163	-8.644	-9.276	-9.920	-10.405	-11.269	-11.513	-11.973	-12.755