Certifying Fairness of Probabilistic Circuits

Nikil Roashan Selvam[1], **Guy Van den Broeck**[1], **YooJung Choi**[2*]

[1] Computer Science Department, University of California, Los Angeles
[2] School of Computing and Augmented Intelligence, Arizona State University
nikilrselvam@cs.ucla.edu, guyvdb@cs.ucla.edu, yj.choi@asu.edu

Abstract

With the increased use of machine learning systems for decision making, questions about the fairness properties of such systems start to take center stage. Most existing work on algorithmic fairness assume complete observation of features at prediction time, as is the case for popular notions like statistical parity and equal opportunity. However, this is not sufficient for models that can make predictions with partial observation as we could miss patterns of bias and incorrectly certify a model to be fair. To address this, a recently introduced notion of fairness asks whether the model exhibits any *discrimination pattern*, in which an individual—characterized by (partial) feature observations—receives vastly different decisions merely by disclosing one or more sensitive attributes such as gender and race. By explicitly accounting for partial observations, this provides a much more fine-grained notion of fairness. In this paper, we propose an algorithm to search for discrimination patterns in a general class of probabilistic models, namely probabilistic circuits. Previously, such algorithms were limited to naive Bayes classifiers which make strong independence assumptions; by contrast, probabilistic circuits provide a unifying framework for a wide range of tractable probabilistic models and can even be compiled from certain classes of Bayesian networks and probabilistic programs, making our method much more broadly applicable. Furthermore, for an unfair model, it may be useful to quickly find discrimination patterns and distill them for better interpretability. As such, we also propose a sampling-based approach to more efficiently mine discrimination patterns, and introduce new classes of patterns such as minimal, maximal, and Pareto optimal patterns that can effectively summarize exponentially many discrimination patterns.

1 Introduction

Machine learning systems are increasingly being used for critical decision making in a variety of areas ranging from education and health care, to financial lending and recidivism prediction (Chouldechova 2017; Berk et al. 2018; Datta, Tschantz, and Datta 2015; Henderson et al. 2015). Consequently, there has been growing interest and concern about the fairness properties of these methods. In particular, biases in the training data and model architecture can result in certain individuals or groups receiving unfavorable treatment based on some sensitive attributes such as gender and race. Naturally, various notions of fairness and ways to enforce them have been proposed (Barocas and Selbst 2016; Dwork et al. 2012; Hardt, Price, and Srebro 2016; Nabi and Shpitser 2018; Madras et al. 2018; Salimi et al. 2019).

In this paper, we investigate the fairness properties of probabilistic models that represent joint distributions over the decision/prediction variable as well as the features. Such models are ubiquitous in decision-making systems for various real-world applications (Koller and Friedman 2009; Sonnenberg and Beck 1993; Griffiths et al. 2010). In particular, they can be used to make classifications by inferring the probability of the class given some observations. Thus, by handling classifications as inference tasks, they can naturally handle missing features at prediction time.

While many existing work on algorithmic fairness assume that predictions are always made with complete observations of features, this fails to analyze the behavior of a model—in particular, its fairness—when making decisions with missing features. On the other hand, the notion of discrimination pattern (Choi et al. 2020) explicitly aims to address fairness of decisions made with partial information. Specifically, it refers to an individual or a group of individuals, characterized by some partial assignments to the features, who may see a significant discrepancy in the prediction after additionally disclosing some sensitive attributes. While the complete observation case is also covered, it is merely a special case of discrimination patterns which in fact take into account all possible partial observation of features. As we show later, a model that is deemed fair according to popular notions such as disparate impact can still exhibit hundreds of discrimination patterns, when considering missing features.

This fine-grained notion leads to new challenges in certifying fairness, as there are now exponentially many patterns of unfairness to check and analyze. Thus, our first key contribution is to introduce special classes of discrimination patterns—minimal, maximal, and Pareto optimal patterns—which can "summarize" a large number of patterns. The set of these summary patterns are often far smaller than the set of all discrimination patterns, making them great targets to find in a model in order to discover and understand its unfairness.

The next contributions we make are algorithms to find discrimination patterns. The existing algorithm is limited to naive Bayes models, which make strong assumptions that

[*] This work was performed while YC was at UCLA.

may not suit real-world data. On the other hand, our proposed methods can be applied to a more general class of models, namely probabilistic circuits (PCs) (Choi, Vergari, and Van den Broeck 2020), which have demonstrated competitive performance in various density estimation tasks (Dang, Vergari, and Van den Broeck 2022; Liu, Mandt, and Van den Broeck 2022; Peharz et al. 2020). These are a family of probabilistic models that support tractable inference, encompassing arithmetic circuits (Darwiche 2003), sum-product networks (Poon and Domingos 2011), and-or graphs (Dechter and Mateescu 2007), probabilistic sentential decision diagrams (Kisa et al. 2014), and more. Some classical graphical models—such as those with bounded treewidth (Chow and Liu 1968) and their mixtures (Meila and Jordan 2000)—as well as certain kinds of probabilistic programs (Holtzen, Van den Broeck, and Millstein 2020) can even be compiled into PCs, and thus can benefit from our proposed methods.

We propose a search-based algorithm that can find all discrimination patterns (or a special subset of them) in a PC or otherwise certify that there exists none and thus the PC is fair. Moreover, we introduce a sampling-based method, which can no longer prove the non-existence of discrimination patterns, but can far more efficiently find many of them in a PC, provided that they exist. Through empirical evaluation on three benchmark datasets, we demonstrate that our search algorithm is able to find all discrimination patterns while only traversing a fraction of the space of possible patterns. Furthermore, we show that the sampling-based approach is indeed significantly faster in pattern mining, while still returning similar summary patterns as the exact approach.

2 Discrimination Patterns

Notation We denote random variables by uppercase letters (X) and their assignments by lowercase letters (x). We use bold uppercase ($\mathbf{X}$) and lowercase letters ($\mathbf{x}$) for sets of variables and their assignments, respectively. The set of possible values of $\mathbf{X}$ is denoted by $\mathsf{val}(\mathbf{X})$. D denotes a binary decision variable, and we use d to refer to the assignment to D that represents a favorable decision (e.g. a loan approval). We assume that a set of discrete (categorical) variables $\mathbf{Z}$ are used to make decisions. Furthermore, a subset of variables $\mathbf{S} \subset \mathbf{Z}$ are designated as *sensitive attributes*, which are protected characteristics such as race or gender.

The notion of a discrimination pattern was introduced to study fairness of decisions made given partial observations of features. They are particularly relevant for probabilistic models which can naturally handle missing value predictions, treating each prediction as an inference problem to compute the probability of a decision given some observations.

Definition 1 (Discrimination patterns (Choi et al. 2020))**.** Let P be a probability distribution over $D \cup \mathbf{Z}$, and $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$, respectively. For some threshold $\delta \in [0, 1]$, we say $\mathbf{x}$ and $\mathbf{y}$ form a *discrimination pattern* w.r.t. P and δ if:

$$|P(d \mid \mathbf{x}, \mathbf{y}) - P(d \mid \mathbf{y})| > \delta.$$

We refer to the LHS as the *discrimination score* of pattern $\mathbf{x}, \mathbf{y}$, denoted by $\Delta(\mathbf{x}, \mathbf{y})$.

Intuitively, a discrimination pattern corresponds to an individual (or a group of individuals sharing the same attributes) who would see a significant difference in the probability of getting a favorable decision, just by disclosing some sensitive information. Clearly, we want to avoid such scenarios; thus, the model is said to be δ-fair iff it exhibits no discrimination patterns with respect to δ.

2.1 Relation to Other Fairness Notions

Before we describe our main contributions, let us briefly discuss how discrimination patterns relate to some other notions of fairness in the literature. Many prominent fairness definitions—such as statistical/demographic parity (SP), disparate impact (DI), and equalized odds (EO)—fall under the notion of group fairness, which aims to equalize certain quantities across demographic groups (Feldman et al. 2015; Hardt, Price, and Srebro 2016). While these definitions assume that predictions are made given complete observation of features, discrimination patterns account for fairness of decisions made with partial information, providing a more fine-grained notion of fairness. For instance, by definition a discrimination pattern of the form $(\mathbf{s}, \emptyset)$ means that $|P(d \mid \mathbf{s}) - P(d)| > \delta$, which implies a violation of statistical parity. That is, a δ-fair model satisfies statistical parity for some threshold that depends on δ, while the converse does not hold. On the other hand, discrimination patterns formed by complete assignments can be interpreted as a violation of individual fairness (Dwork et al. 2012); we refer to Choi et al. (2020) for detailed examples.

Various methods have been proposed to verify different notions of fairness (Galhotra, Brun, and Meliou 2017; Bellamy et al. 2018), including those that utilize probability distributions (Albarghouthi et al. 2017; Bastani, Zhang, and Solar-Lezama 2019; Ghosh, Basu, and Meel 2021, 2022). However, they often make simplifying assumptions such as conditional independence between attributes, making it challenging to scale these methods to probabilistic circuits which are more general as we will show later. More importantly, verifying properties such as DI and SP is not sufficient if we wish to certify fairness according to discrimination patterns.

To illustrate this on real-world data, we learn PCs on the COMPAS dataset before and after fair data repair by Feldman et al. (2015). We use different degrees of data repair (controlled by parameter λ), yielding PCs with varying levels of fairness according to the above-mentioned standard notions (Table 1). While the learned PCs would be certified as fair according to verifiers using metrics such as DI, SP, and EO, they may still exhibit discrimination patterns. For instance, many learned models in Table 1 satisfy SP with $\epsilon < 0.02$ and EO with $\epsilon = 0$, but still exhibit hundreds of discrimination patterns with scores $\delta > 0.05$, some as high as 0.22.

That is, discrimination patterns enable a more fine-grained auditing of fairness. However, even a simple classifier could exhibit a large number of discrimination patterns, thereby

[1] DI $= 1 - \frac{\min_{\mathbf{s}} \mathbb{P}(d|\mathbf{s})}{\max_{\mathbf{s}} \mathbb{P}(d|\mathbf{s})}$, SP$= \max_{\mathbf{s}} \mathbb{P}(d \mid \mathbf{s}) - \min_{\mathbf{s}} \mathbb{P}(d \mid \mathbf{s})$, SP (1 variable) $= \max_{s} \mathbb{P}(d \mid s) - \min_{s} \mathbb{P}(d \mid s)$ for a single s, EO $= \max\{\max_{\mathbf{s}} \mathbb{P}(\hat{d} \mid \mathbf{s}d) - \min_{\mathbf{s}} \mathbb{P}(\hat{d} \mid \mathbf{s}d), \max_{\mathbf{s}} \mathbb{P}(\hat{d} \mid \mathbf{s}\bar{d}) - \min_{\mathbf{s}} \mathbb{P}(\hat{d} \mid \mathbf{s}\bar{d})\}$.

Table 1: Different metrics of fairness[1] for PCs on the COMPAS dataset before and after fair data repair (Feldman et al. 2015).

	Original	$\lambda=.5$	$\lambda=.9$	$\lambda=.95$	$\lambda=.99$	$\lambda=1$
DI	0.187	0.063	0.017	0.020	0.015	0.023
SP	0.183	0.061	0.016	0.019	0.014	0.022
SP (1 variable)	0.055	0.015	0.002	0.001	0.001	0.001
EO	0.752	0.000	0.000	0.000	0.000	0.000
# Disc. Patt. (0.05)	3866	1320	488	659	894	578
# Disc. Patt. (0.1)	1761	311	11	64	74	34
Highest Disc. Score	0.372	0.208	0.112	0.225	0.176	0.123

making it hard for domain experts and users to examine them effectively. For instance, a naive Bayes classifier with 7 features for the COMPAS data[2] was shown to have more than 2,000 discrimination patterns (Choi et al. 2020); this is clearly not scalable for interpretation. Thus, we propose new classes of discrimination patterns that can be used as representatives for a large number of patterns, thereby being more amenable for interpretations.

2.2 Summarizing Patterns

A natural way to choose the most "interesting" patterns may be by ranking them by their discrimination scores, and focusing on a few instances that are the most discriminatory. While it may be useful to study the most problematic patterns and address them, they do not necessarily provide insight into other discrimination patterns that exist. Instead, we propose the notion of maximal and minimal patterns that can summarize groups of patterns, namely their extensions and contractions. An extension of a pattern $(\mathbf{x}, \mathbf{y})$, denoted by $(\mathbf{x}', \mathbf{y}') \supset (\mathbf{x}, \mathbf{y})$, can be generated by adding an assignment to the pattern: that is, $\mathbf{x} \subseteq \mathbf{x}'$, $\mathbf{y} \subseteq \mathbf{y}'$, and either $\mathbf{x} \subset \mathbf{x}'$ or $\mathbf{y} \subset \mathbf{y}'$. Conversely, $(\mathbf{x}, \mathbf{y})$ is called a contraction of $(\mathbf{x}', \mathbf{y}')$.

Definition 2 (Maximal patterns)**.** Let Σ denote a set of discrimination patterns w.r.t. a distribution P and threshold δ. The set of *maximal patterns* $\Sigma_{\text{max}} \subseteq \Sigma$ consists of all patterns $(\mathbf{x}, \mathbf{y}) \in \Sigma$ that are not a complete assignment (i.e. $\mathbf{Z} \setminus (\mathbf{X} \cup \mathbf{Y}) \neq \emptyset$) and:

$$\forall (\mathbf{x}', \mathbf{y}') \supset (\mathbf{x}, \mathbf{y}),\ (\mathbf{x}', \mathbf{y}') \notin \Sigma.$$

In other words, a maximal pattern is a discrimination pattern such that none of its extensions are discrimination patterns. As the name suggests, an extension of a maximal pattern cannot also be maximal, because by definition it will not be a discrimination pattern. Hence, an individual with attributes $\mathbf{x}$ and $\mathbf{y}$ who may see a discrimination in the decision by disclosing their sensitive information would no longer receive such treatment if they additionally share other features, whatever their values may be. This notion is nicely complemented by the following notion of minimal patterns.

Definition 3 (Minimal patterns)**.** Let Σ denote a set of discrimination patterns w.r.t. a distribution P and threshold δ. The set of *minimal patterns* $\Sigma_{\text{min}} \subseteq \Sigma$ consists of all patterns $(\mathbf{x}, \mathbf{y}) \in \Sigma$ such that:

$$\forall (\mathbf{x}', \mathbf{y}') \supset (\mathbf{x}, \mathbf{y}),\ (\mathbf{x}', \mathbf{y}') \in \Sigma$$
$$\text{and} \quad \forall (\mathbf{x}'', \mathbf{y}'') \subset (\mathbf{x}, \mathbf{y}) \text{ s.t. } \mathbf{x}'' \neq \emptyset,\ (\mathbf{x}'', \mathbf{y}'') \notin \Sigma.$$

That is, we say a discrimination pattern is minimal if all of its extensions and none of its contractions are discriminatory. Thus, a single minimal pattern can be interpreted as representing a large set of discrimination patterns—more precisely, whose size is exponential in the number of unobserved features. For example, suppose $\mathbf{Z} = \{X, Y, U\}$ are binary features with $\mathbf{S} = \{X\}$, and let $(\{X=1\}, \{Y=0\})$ be a minimal pattern. Then its valid contraction—i.e. $(\{X=1\}, \{\})$—is not a discrimination pattern; while all of its extensions—i.e. $(\{X=1\}, \{Y=0, U=0\})$ and $(\{X=1\}, \{Y=0, U=1\})$—are discriminatory. Note that by definition a minimal pattern cannot be a maximal pattern, and vice versa.

As a case study, we train a probabilistic model[3] on the COMPAS dataset, which exhibits 7445, 2338, and 1164 discrimination patterns for $\delta = 0.01, 0.05$, and 0.1 respectively. On the other hand, this model has 170, 74, and 0 maximal patterns, and only 103, 10, and 1 minimal patterns, for respective values of threshold δ. Interestingly, we observe that among the 74 maximal patterns for $\delta = 0.05$, none of them includes an assignment to the variable regarding 'supervision level', suggesting that there are many instances where an individual would not see an unfair prediction if the supervision level is additionally known. Moreover, remarkably, the single minimal pattern in the case of $\delta = 0.1$ can represent 512 patterns (its extensions) out of 1164 total discrimination patterns in the model.[4]

Another important consideration when studying discrimination patterns is their probability. Recall that each pattern $(\mathbf{x}, \mathbf{y})$ represents a group of people sharing the attributes $\mathbf{x}$ and $\mathbf{y}$. Then the probability $P(\mathbf{x}, \mathbf{y})$ corresponds to the proportion of the population that could be affected. Even though any unfairness is equally undesirable for minority groups (i.e. lower probability) as it is for majority groups, patterns that are so specific and have exceedingly low probability would not be as insightful or even relevant when auditing a model for real-world fairness concerns. To address this, patterns can be ranked by their divergence score, which takes into account the probability of a pattern as well as its discrimination score.

Definition 4 (Divergence score (Choi et al. 2020))**.** Let P be a probability distribution over $D \cup \mathbf{Z}$ and δ some threshold in $[0.1]$. Further suppose $\mathbf{x}$ and $\mathbf{y}$ are joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$, respectively. Then the *divergence score* of $(\mathbf{x}, \mathbf{y})$ is:

$$\min_{Q} \mathrm{D}_{\mathrm{KL}}(P \parallel Q) \text{ s.t. } \Delta(\mathbf{x}, \mathbf{y}) \leq \delta,$$
$$P(d, \mathbf{z}) = Q(d, \mathbf{z}),\ \forall\, \mathbf{z} \not\supset \mathbf{x} \cup \mathbf{y}$$

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

Informally, it aims to quantify how much the distribution P needs to be changed in order to remove the discrimination

[2] https://github.com/propublica/compas-analysis

[3] See Section 5 for details of the trained model.

[4] The minimal pattern is where $\mathbf{x} = \{$Not Married$\}$ and $\mathbf{y} = \{$Low–Medium Supervision Level$\}$.

pattern $(\mathbf{x}, \mathbf{y})$. Thus, the patterns with highest divergence scores would tend to have both high discrimination score as well as high probability. In fact, we could summarize all existing discrimination patterns in a model by explicitly constructing a set of patterns with the following behavior: one cannot increase the discrimination score without decreasing the probability, and vice versa.

Definition 5 (Pareto optimal patterns)**.** Let Σ denote a set of discrimination patterns w.r.t. a distribution P and threshold δ. The set of *Pareto optimal patterns* $\Sigma_{\text{PO}} \subseteq \Sigma$ consists of the patterns $(\mathbf{x}, \mathbf{y}) \in \Sigma$ such that:

$$\forall (\mathbf{x}', \mathbf{y}') \in \Sigma \setminus \{(\mathbf{x}, \mathbf{y})\},$$
$$\Delta(\mathbf{x}, \mathbf{y}) > \Delta(\mathbf{x}', \mathbf{y}') \text{ or } P(\mathbf{x}, \mathbf{y}) > P(\mathbf{x}', \mathbf{y}').$$

Pareto optimal patterns can be a very effective way to study fairness of a probabilistic model, as it significantly reduces the number of discrimination patterns one would examine. For example, the model trained on the COMPAS with 2388 and 1164 discrimination patterns w.r.t. $\delta = 0.05$ and 0.1, respectively, has only 38 and 28 Pareto optimal patterns. That is, for $\delta = 0.1$, each of the $1164 - 28 = 1136$ patterns has discrimination score and probability that are both dominated by those of some Pareto optimal pattern.

In the following sections, we discuss how to find discrimination patterns in probabilistic circuits, as well as generating the summaries in the form of maximal, minimal, and Pareto optimal patterns.

3 Finding Discrimination Patterns in Probabilistic Circuits

We now describe our algorithm to find discrimination patterns, if there exists any, or certify that there are none. As discussed in Section 2.2, 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, the existing algorithm to discover discrimination patterns assumes naive Bayes classifiers, which make strong independence assumptions and are generally too restrictive to fit real-world distributions. We instead consider a more expressive type of probabilistic models, called probabilistic circuits.

3.1 Probabilistic Circuits

A probabilistic circuit is a directed acyclic graph (DAG) with parameters, in which inner nodes can either be sum or product nodes, and input nodes are associated with simple univariate distributions (often indicator functions in the case of discrete variables). Moreover, each edge (n, c) between a sum node n and its child c is associated with a parameter $\theta_{n,c} > 0$. A PC $\mathcal{C}$ over random variables $\mathbf{X}$ recursively defines a probability distribution over $\mathbf{X}$ as follows:

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

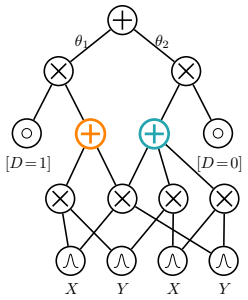


Figure 1: A PC over variables $\{D, X, Y\}$

Here, $\mathsf{ch}(n)$ denotes the set of children of an inner node n. Then the distribution P defined by a PC is exactly $P(\mathbf{x}) = n(\mathbf{x})$ where n is the root of the PC.

A key strength of PCs is that they allow tractable inference of certain probabilistic queries, based on which structural properties are satisfied by the circuit. The first inference task we need is computing conditional probabilities, which is required to get the discrimination score (Definition 1) of a pattern. Moreover, we would also like to compute probabilities of patterns—that is, marginal probabilities given some partial observations. PCs support efficient marginal and conditional inference if they satisfy two structural properties called smoothness and decomposability. A PC is *smooth* if for every sum node its children include exactly the same set of variables, and it is *decomposable* if for every product node its children depend on disjoint sets of variables (Darwiche and Marquis 2002). Given these properties, computing any marginal probability can be done through a single feedforward evaluation of the PC, thus taking linear time in the size of the circuit. Hence, we will assume smooth and decomposable PCs throughout this paper.

Let us quickly remark on the wide applicability of probabilistic circuits and subsequently our pattern mining algorithm which takes PCs and inputs. First, both the structure and parameters of PCs can be learned to fit the data, and in a wide range of tasks, they were shown to achieve competitive and state-of-the-art performance (Dang, Vergari, and Van den Broeck 2022; Liu, Mandt, and Van den Broeck 2022; Peharz et al. 2020; Li et al. 2021). In addition, as mentioned previously, they can be compiled efficiently from bounded-treewidth graphical models, and thus we can apply the following search algorithm on such models as well.

3.2 Search Algorithm

To certify whether a distribution defined by a probabilistic circuit is δ-fair, we search for discrimination patterns in the PC. If the search concludes without finding any pattern, then we know that the PC is δ-fair; otherwise, we return all or some of the discrimination patterns selected according to one of the criteria discussed in Section 2.

More precisely, we adopt a branch-and-bound search approach. Algorithm 1 outlines the pseudocode of our search algorithm. At each search step, it checks whether the current assignments form a discrimination pattern, and explores extensions by recursively adding variable assignments. Note

Algorithm 1 SEARCH-DISC-PATT$(\mathbf{x}, \mathbf{y}, \mathbf{E})$

Input: a PC $\mathcal{C}$ over variables $D \cup \mathbf{Z}$ and a threshold δ
Output: a set of discrimination patterns Σ
Data: current pattern $(\mathbf{x}, \mathbf{y}) \leftarrow (\{\}, \{\})$; excluded variables $\mathbf{E} \leftarrow \{\}$

$\Sigma \leftarrow \{\}$
for each $z \in \mathsf{val}(Z)$ for some $Z \in \mathbf{Z} \setminus (\mathbf{X} \cup \mathbf{Y} \cup \mathbf{E})$ **do**
  if $Z \in \mathbf{S}$ **then**
    if $\Delta(\mathbf{x} \cup \{z\}, \mathbf{y}) > \delta$ **then**
      $\Sigma \leftarrow \Sigma \cup \{(\mathbf{x} \cup \{z\}, \mathbf{y})\}$
    if $UB(\mathbf{x} \cup \{z\}, \mathbf{y}, \mathbf{E}) > \delta$ **then**
      $\Sigma \leftarrow \Sigma \cup$ SEARCH-DISC-PATT$(\mathbf{x} \cup \{\mathbf{z}\}, \mathbf{y}, \mathbf{E})$
  if $\Delta(\mathbf{x}, \mathbf{y} \cup \{z\}) > \delta$ **then** $\Sigma \leftarrow \Sigma \cup \{(\mathbf{x}, \mathbf{y} \cup \{z\})\}$
  if $UB(\mathbf{x}, \mathbf{y} \cup \{z\}, \mathbf{E}) > \delta$ **then**
    $\Sigma \leftarrow \Sigma \cup$ SEARCH-DISC-PATT$(\mathbf{x}, \mathbf{y} \cup \{z\}, \mathbf{E})$
if $UB(\mathbf{x}, \mathbf{y}, \mathbf{E} \cup \{Z\}) > \delta$ **then**
  $\Sigma \leftarrow \Sigma \cup$ SEARCH-DISC-PATT$(\mathbf{x}, \mathbf{y}, \mathbf{E} \cup \{Z\})$
return Σ

that while we mainly present the algorithm that returns all discrimination patterns, we can easily tweak it to return the top-k most discriminating patterns: by keeping a running list of top-k patterns and using the k-th highest score as the threshold instead of δ.

As there are exponentially many potential patterns, we rely on a good upper bound to effectively prune the search tree. In particular, we use the following as our bound $UB(\mathbf{x}, \mathbf{y}, \mathbf{E})$:

$$\max\{ \left| \max_{\mathbf{u}} P(d \mid \mathbf{x}, \mathbf{y}, \mathbf{u}) - \min_{\mathbf{u}} P(d \mid \mathbf{y}, \mathbf{u}) \right|, \quad (1)$$

$$\left| \min_{\mathbf{u}} P(d \mid \mathbf{x}, \mathbf{y}, \mathbf{u}) - \max_{\mathbf{u}} P(d \mid \mathbf{y}, \mathbf{u}) \right| \} \quad (2)$$

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 $P(d \mid \mathbf{y}, \mathbf{u})$ over the values of some $\mathbf{U}$ for a given $\mathbf{y}$. 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.[5]

Lemma 1. Let P be a distribution over $D \cup \mathbf{Z}$ and $\mathbf{x}$ 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:

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

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.

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

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

[5]Complete proofs of lemmas can be found in the appendix.

Combining these observations, we see that the upper bound in Equation (1) can be computed easily if we can efficiently maximize and minimize quantities of the form $P(\mathbf{x}, \mathbf{u} \mid d)/P(\mathbf{x}, \mathbf{u} \mid \overline{d})$. 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. Deferring the algorithmic details and proof of correctness to the appendix, here we instead provide high-level insights to these additional tractability conditions. First, Vergari et al. (2021) shows the necessary structural conditions (called compatibility and determinism) such that the quotient of two PCs can be computed tractably and represented as another circuit representation that allows for linear-time optimization (Choi and Darwiche 2017). Then all we need is to represent the conditional distributions $P(\mathbf{Z} \mid d)$ and $P(\mathbf{Z} \mid \overline{d})$ as two PCs satisfying those conditions. If the decision variable D appears at the top of the PC over $D \cup \mathbf{Z}$ as illustrated in Figure 1, then the two subcircuits rooted at each child of the root node exactly corresponds to the conditional distributions given D. For instance, the PCs rooted at the orange and green sum nodes in Figure 1 correspond to the conditional distributions $P(X, Y \mid D{=}1)$ and $P(X, Y \mid D{=}0)$, respectively.

Furthermore, we can similarly search for top-k patterns ranked by their divergence scores. Choi et al. (2020) provides an upper bound on divergence score that once again requires efficiently maximizing/minimizing 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.

Lastly, we briefly discuss how to obtain the special types of discrimination patterns introduced in Section 2.2. We can make a small tweak to the search algorithm to keep track of the Pareto front of discrimination patterns found so far in each search step. Concretely, we maintain an ordered container storing the probability and discrimination score in increasing order of the former and decreasing order of the latter. Similarly, finding the set of maximal patterns is almost identical to searching for all discrimination patterns. In particular, when exploring a pattern in the search tree, we declare it to be maximal if no extension of it can be a discrimination pattern, determined by the quadratic-time upper bound on discrimination score. For minimal patterns, we derive a sub-quadratic time algorithm to examine a set of patterns and recover the minimal ones; see appendix for details.

4 Discrimination Pattern Mining by Sampling

Certifying that there exists no discrimination pattern, among exponentially many possible assignments, is a very hard problem. In real-world settings, one may simply be interested in quickly studying examples of unfairness that may be present in the model. In fact, this is the goal for many existing fairness auditing tools (Saleiro et al. 2018): to find patterns of bias (potentially using different fairness definitions) for the developer or user to examine. Hence, we introduce efficient sampling-based methods to mine discrimination patterns in PCs. While these methods cannot necessarily certify a model to be δ-fair, they can very quickly find a large number of

Algorithm 2 SAMPLE-DISC-PATTERNS$(\mathcal{C}, \mathbf{Z})$

Input: a PC $\mathcal{C}$ over variables $D \cup \mathbf{Z}$ and a threshold δ
Output: a set of sampled discrimination patterns Σ

$\Sigma \leftarrow \{\}$
repeat ▷ generate samples until timeout
  $(\mathbf{x}, \mathbf{y}) \leftarrow (\{\}, \{\})$
  while $|\mathbf{x}| + |\mathbf{y}| < n$ **do**
    for $(\mathbf{x}', \mathbf{y}') \in \mathsf{extensions}(\mathbf{x}, \mathbf{y}))$ **do**
      ▷ each extension by a single variable
      if $\Delta(\mathbf{x}', \mathbf{y}') > \delta$ **then** $\Sigma \leftarrow \Sigma \cup \{(\mathbf{x}', \mathbf{y}')\}$
    $(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{sample}_{\mathsf{weight}:\Delta(\mathbf{x}'\mathbf{y}')}(\mathsf{extensions}(\mathbf{x}, \mathbf{y}))$
until timeout
return Σ

patterns, as we will later show empirically.

Our proposed approach is summarized in Algorithm 2. At a high level, each run of the sampling algorithm starts from an empty assignment $\mathbf{xy}$ and incrementally adds one attribute at a time until a complete assignment is obtained. The attribute to be added (or more precisely the immediate extension to be explored) at each step is sampled at random with a likelihood proportional to the discrimination score of the resulting assignment. Any assignment explored along the way with a sufficiently high discrimination score is added to our set of patterns. Intuitively, at any assignment, we use the discrimination score of its immediate extension as a heuristic for its extensions being discrimination patterns.

Algorithm 2 is the base of our proposed sampling algorithm. We also derive a more sophisticated algorithm with memoization between samples and control over exploration versus exploitation at different stages of sampling. At a high level, we maintain an estimator at each assignment corresponding to the expected discrimination score of an extension and backtrack at the end of each sampling run to update the estimates before the next run. We refer the reader to the appendix for details regarding the same. Observe that the sampling algorithm is computationally inexpensive overall as the only circuit evaluations that need to be performed are a few feed-forward evaluations (linear time) to compute conditionals at each assignment explored. Lastly, it is worth noting that after the discrimination patterns have been sampled, we can utilize similar techniques as described in Section 3.2 to efficiently summarize them through minimal, maximal, and Pareto optimal patterns.

5 Empirical Evaluation

We evaluate our discrimination pattern mining algorithms on three datasets: *COMPAS* which is used for recidivism prediction and the *Adult* (Dua and Graff 2017) and *Income* (Ding et al. 2021) datasets for predicting income levels. As preprocessing, we remove redundant features and features with unique values, and discretize numerical values. We learn a PC from each dataset using the STRUDEL algorithm (Dang, Vergari, and Van den Broeck 2022), which returns deterministic and structured decomposable PCs as required by our search algorithm.[6] All experiments were run on an Intel(R) Xeon(R) CPU E5-2640 (2.40GHz).

Table 2: Dataset statistics (number of examples, number of sensitive features S, non-sensitive features N) and speedup of top-k search v.s. naive enumeration, in terms of the fraction of search space explored.

Dataset	Size	S	N	k	Disc. $\delta=0.1$	Divergence $\delta=0.01$	Divergence $\delta=0.05$	Divergence $\delta=0.10$
COMPAS	48834	4	3	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	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	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

With regards to empirical comparison with existing work in literature, it is worth emphasizing that efficiently mining discrimination patterns has only been possible so far for naive Bayes (Choi et al. 2020) which makes strong assumptions, and our method is (to the best of our knowledge) the first method to extend this to a much more general class of models, namely PCs. Furthermore, other existing fairness verifiers in literature as discussed in Section 2.1 are not able to find discrimination patterns: they only verify a weaker fairness properties such as statistical parity. Hence, in this section, we evaluate our exact and approximate methods against the only available baseline of naive enumeration, and against each other to clearly test comparative efficiency.

5.1 Exact Search

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 that does not require naive Bayes assumption, 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 2 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 computes 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

[6] Link to pre-processed data, trained models, and code: https://github.com/UCLA-StarAI/PC-DiscriminationPatterns.

Table 3: Average speedup of sampling v.s. naive enumeration to find top 1 pattern (in terms of proportion of search space explored).

Dataset	Discrimination Avg	Discrimination StdDev	Divergence Avg	Divergence StdDev
Compas	26x	54x	29x	15x
Income	17x	14x	143x	6x
Adult	48x	53x	49480x	2113x

Table 4: Number of discrimination patterns and highest score found by exact search and sampling.

Dataset	Time	δ	# Patterns found Exact	# Patterns found Sampling	Highest score Exact	Highest score Sampling
COMPAS	3s	0.05	347	**751**	**0.2236**	0.2230
		0.10	210	**347**	**0.2236**	0.2230
Income	5s	0.05	209	**1090**	0.1076	**0.1658**
		0.10	3	**225**	0.1076	**0.1659**
Adult	600s	0.05	37167	**113763**	0.6725	**0.6871**
		0.10	30982	**99578**	0.6725	**0.6844**

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.

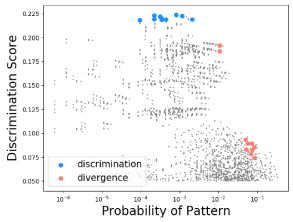
5.2 Sampling

The primary motivation for the sampling algorithm is to not only quickly audit a model to check if it exhibits any discrimination patterns, but in particular to quickly analyze the most interesting patterns. To that end, we first evaluate how efficiently the sampling algorithm is able to find the most discriminatory/divergent pattern. We first find the highest discrimination and divergence score using exact search, then run the sampling algorithm until it finds the top-1 pattern (using $\delta = 0.01$ for the divergence scores). Table 3 details the speedup relative to naive enumeration, in terms of the number of assignments explored to find the top-1 pattern; each result is the average over 10 independent random trials.

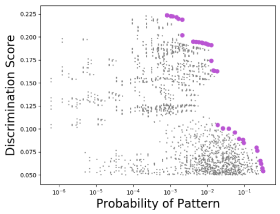
Recall that each sampling instance merely requires a few feed-forward evaluations of the circuit (linear time) for computing marginals. Thus, from the table it is clear that the sampling algorithm is much quicker than exact search in finding the patterns with highest scores (for instance, as much as 49480x fewer patterns explored compared to naive search to find the most divergent pattern in Adult).

For a more direct comparison, we run the exact search algorithm and the sampling method with a specified timeout. We report both the number of patterns found and the highest score in different settings (Table 4). We observe that the sampling algorithm consistently outperforms the exact search in both number of patterns found the scores of patterns found. Thus, we can reliably use the sampling approach to quickly mine many discrimination patterns of significance.

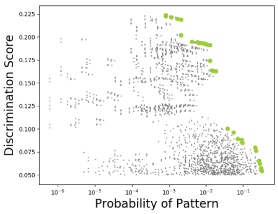
Lastly, we are interested in whether the patterns returned by the sampling algorithm are "interesting". To that end, we analyze the top 10 discrimination patterns produced by our sampling algorithm on COMPAS with a 3 second timeout



(a) Top 10 discrimination and divergence patterns found by sampling.



(b) Exact Pareto front



(c) Sampled Pareto front

Figure 2: Discrimination score and probability of all patterns (grey) and different summary patterns (color) for COMPAS.

and find that they indeed correspond to some of the most discriminating patterns in the model (Figure 2a). Furthermore, we compare the Pareto optimal patterns from the sampling algorithm (Figure 2c) to the true Pareto front obtained through exact search (Figure 2b) and find that the sampling algorithm contains most of the same patterns in its Pareto front despite its short timeout. More concretely, the Pareto front from the sampling approach contains 30 patterns, of which 27 are in the true set of Pareto optimal patterns (there are 38 total).

6 Conclusions, Limitations, and Discussion

This paper studies fairness of probabilistic classifiers through the lens of discrimination patterns. With the goal of efficient and interpretable auditing for fairness, we introduce new classes of patterns, such as maximal, minimal, and Pareto optimal patterns, and propose efficient exact and approximate algorithms for mining these patterns in probabilistic circuits, an expressive class of tractable models. We empirically demonstrate the effectiveness of our methods in mining instances to help analyze a model's fairness properties.

Our method can be used by domain experts and ML practitioners not only for auditing in the deployment phase, but also as a subroutine when learning fair models. For example, one could use our approach to discover discrimination patterns, then enforce their elimination as a constraint during learning to obtain fair classifiers, which is a promising future direction. Although we demonstrate the efficiency of our sampling-based mining algorithm empirically, we currently lack theoretical guarantees for the same, which we also leave as future work. Lastly, we re-iterate that our method provides an efficient fairness auditing tool focusing on one specific notion out of many, and it remains the responsibility of domain experts and developers to interpret any pattern of unfairness in the context of the application.

7 Acknowledgments

We thank the reviewers for their thoughtful feedback towards improving this paper. This work was funded in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, Samsung, CISCO, a Sloan Fellowship, and a UCLA Samueli Fellowship.

References

Albarghouthi, A.; D'Antoni, L.; Drews, S.; and Nori, A. V. 2017. FairSquare: Probabilistic Verification of Program Fairness. 1(OOPSLA).

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

Bastani, O.; Zhang, X.; and Solar-Lezama, A. 2019. Probabilistic Verification of Fairness Properties via Concentration. 3(OOPSLA).

Bellamy, R. K. E.; Dey, K.; Hind, M.; Hoffman, S. C.; Houde, S.; Kannan, K.; Lohia, P.; Martino, J.; Mehta, S.; Mojsilovic, A.; Nagar, S.; Ramamurthy, K. N.; Richards, J.; Saha, D.; Sattigeri, P.; Singh, M.; Varshney, K. R.; and Zhang, Y. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias.

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.

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 AAAI Conference on Artificial Intelligence*, volume 34, 10077–10084.

Choi, Y.; Vergari, A.; and Van den Broeck, G. 2020. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models.

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*.

Dang, M.; Vergari, A.; and Van den Broeck, G. 2022. Strudel: A Fast and Accurate Learner of Structured-Decomposable Probabilistic Circuits. *International Journal of Approximate Reasoning*, 140: 92–115.

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

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.

Dechter, R.; and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artif. Intell.*, 171(2-3): 73–106.

Ding, F.; Hardt, M.; Miller, J.; and Schmidt, L. 2021. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

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*, KDD '15, 259–268. New York, NY, USA: Association for Computing Machinery. ISBN 9781450336642.

Galhotra, S.; Brun, Y.; and Meliou, A. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, 498–510. New York, NY, USA: Association for Computing Machinery. ISBN 9781450351058.

Ghosh, B.; Basu, D.; and Meel, K. S. 2021. Justicia: A Stochastic SAT Approach to Formally Verify Fairness. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9): 7554–7563.

Ghosh, B.; Basu, D.; and Meel, K. S. 2022. Algorithmic Fairness Verification with Graphical Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9539–9548.

Griffiths, T. L.; Chater, N.; Kemp, C.; Perfors, A.; and Tenenbaum, J. B. 2010. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8): 357 – 364.

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

Henderson, L.; Herring, C.; Horton, H. D.; and Thomas, M. 2015. 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.

Holtzen, S.; Van den Broeck, G.; and Millstein, T. 2020. Scaling Exact Inference for Discrete Probabilistic Programs. *Proc. ACM Program. Lang. (OOPSLA)*.

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*. MIT press.

Li, W.; Zeng, Z.; Vergari, A.; and Van den Broeck, G. 2021. Tractable Computation of Expected Kernels. In *Proceedings*

of the 37th Conference on Uncertainty in Aritifical Intelligence (UAI).

Liu, A.; Mandt, S.; and Van den Broeck, G. 2022. Lossless Compression with Probabilistic Circuits. In *International Conference on Learning Representations (ICLR)*.

Madras, D.; Creager, E.; Pitassi, T.; and Zemel, R. 2018. Fairness Through Causal Awareness: Learning Latent-Variable Models for Biased Data. *arXiv preprint arXiv:1809.02519*.

Meila, M.; and Jordan, M. I. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1(Oct): 1–48.

Nabi, R.; and Shpitser, I. 2018. Fair inference on outcomes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Van den Broeck, G.; Kersting, K.; and Ghahramani, Z. 2020. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

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.

Saleiro, P.; Kuester, B.; Hinkson, L.; London, J.; Stevens, A.; Anisfeld, A.; Rodolfa, K. T.; and Ghani, R. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*.

Salimi, B.; Rodriguez, L.; Howe, B.; and Suciu, D. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *Proceedings of the 2019 International Conference on Management of Data*, 793–810.

Sonnenberg, F. A.; and Beck, J. R. 1993. Markov models in medical decision making: a practical guide. *Medical decision making*, 13(4): 322–338.

Vergari, A.; Choi, Y.; Liu, A.; Teso, S.; and Van den Broeck, G. 2021. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

A Proofs of Lemmas

Lemma 1. Let P be a distribution over $D \cup \mathbf{Z}$ and $\mathbf{x}$ 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:

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

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

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

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 $P(d \mid \mathbf{x}, \mathbf{u}) \leq P(d \mid \mathbf{x}, \mathbf{u}, w)$ for any $\mathbf{u}$, we will show that $P(d \mid \mathbf{x}, \mathbf{u}) > P(d \mid \mathbf{x}, \mathbf{u}, w_i)$ for $i = 1, \ldots, n-1$ implies that $P(d \mid \mathbf{x}, \mathbf{u}) < P(d \mid \mathbf{x}, \mathbf{u}, w_n)$. First, for all $i \leq n-1$ we have:

$$\begin{aligned} &P(d \mid \mathbf{x}, \mathbf{u}) > P(d \mid \mathbf{x}, \mathbf{u}, w_i) \\ &\implies P(d, \mathbf{x}, \mathbf{u}) P(\mathbf{x}, \mathbf{u}, w_i) > P(\mathbf{x}, \mathbf{u}) P(d, \mathbf{x}, \mathbf{u}, w_i). \end{aligned}$$

By taking the sum of both sides of the above inequality, we get:

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

□

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

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

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

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

□

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

Input: deterministic and compatible PCs n and m over $\mathbf{Z}$; an assignment $\mathbf{x} \in \mathsf{val}(\mathbf{X})$ for $\mathbf{X} \subset \mathbf{Z}$

Output: $\max_{\mathbf{u} \in \mathsf{val}(\mathbf{U})} n(\mathbf{x}, \mathbf{u}) / m(\mathbf{x}, \mathbf{u})$ 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 $n(\mathbf{x}) \neq 0, m(\mathbf{x}) \neq 0$ **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: $\mathrm{BR}(n, m) \leftarrow \max_{n_i \in \mathsf{ch}(n), m_j \in \mathsf{ch}(m)} \frac{\theta_i}{\theta_j} \mathrm{BR}(n_i, m_j)$

B Computing Upper Bounds

B.1 Discrimination Score

Here we describe our algorithm to compute the upper bound on discrimination score (Section 3.2). Recall that we need to maximize or minimize quantities of the form $P(\mathbf{x}, \mathbf{u} \mid d) / P(\mathbf{x}, \mathbf{u} \mid \overline{d})$ over values of $\mathbf{U} = \mathbf{Z} \setminus \mathbf{X}$ for some given evidence $\mathbf{x} \in \mathsf{val}(\mathbf{X})$. The pseudocode of our algorithm to maximize such ratio is given in Algorithm 3. Again, we assume that the root of the PC is effectively a decision node on D, and thus its children represent the conditional distributions $P(\mathbf{z} \mid d)$ and $P(\mathbf{z} \mid \overline{d})$. Hence we can run Algorithm 3 by giving those two children nodes as inputs. 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.

The algorithm assumes PCs that satisfy two structural constraints: determinism and compatibility. 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 $\mathbf{z}$, at most one of the children 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.

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 $n(\mathbf{z}) = \prod_i n_i(\mathbf{z}_i)$ and $m(\mathbf{z}) = \prod_i m_i(\mathbf{z}_i)$, 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:

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

leading to Line 7 in Algorithm 3.

Finally, consider two deterministic sum nodes. Then for any $\mathbf{z}$, 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. $n(\mathbf{z}) = \sum_i n_i(\mathbf{z}) = \max_i n_i(\mathbf{z})$. Moreover, among all pairs of children n_i, m_j, the ratio $n_i(\mathbf{z})/m_j(\mathbf{z})$ for any fixed $\mathbf{z}$ would be non-zero for at most one pair (again, we treat the ratio that is undefined as 0). Therefore, we have:

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

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

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

□

B.2 Divergence Score

Choi et al. (2020) gives the following upper bound on the divergence scores of extensions of an assignment $(\mathbf{x}, \mathbf{y})$:

$$P(d, \mathbf{x}, \mathbf{y}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(d \mid \mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(d \mid \mathbf{z})} + P(\overline{d}\mathbf{xy}) \log \frac{\max_{\mathbf{z} \models \mathbf{xy}} P(\overline{d} \mid \mathbf{z})}{\min_{\mathbf{z} \models \mathbf{y}} P(\overline{d} \mid \mathbf{z})},$$

where $\mathbf{z} \models \mathbf{xy}$ denotes a complete assignment to $\mathbf{Z}$ that agrees with $\mathbf{x}$ and $\mathbf{y}$ on their assignments to variables in $\mathbf{X}$ and $\mathbf{Y}$, respectively.

Observe that this upper bound once again requires efficient maximization and minimization of conditional probability of extensions. Thus, Algorithm 3 allows us to leverage this upper bound and straightforwardly extend our exact search algorithm to mine divergence patterns as well.

B.3 Relative Discrimination Score

We define discrimination patterns using an absolute difference in conditional probabilities, but one may wish to characterize discrimination using a quantity that is proportional to the initial prediction probability. For instance, a prediction that goes from 0.15 to 0.05 after disclosing the sensitive attributes could be seen as more problematic than one that goes from 0.8 to 0.7, but they would have the same discrimination score. We can alternatively define a score based on relative difference as the following.

Definition 6 (Relative Degree of Discrimination)**.** Let P be a probability distribution over $D \cup \mathbf{Z}$, and $\mathbf{x}$ and $\mathbf{y}$ be joint assignments to $\mathbf{X} \subseteq \mathbf{S}$ and $\mathbf{Y} \subseteq \mathbf{Z} \setminus \mathbf{X}$, respectively. The *relative discrimination score* of pattern $\mathbf{x}, \mathbf{y}$, defined as $\Delta'(\mathbf{x}, \mathbf{y}) = \frac{P(d|\mathbf{x},\mathbf{y})}{P(d|\mathbf{y})}$.

We can mine discrimination patterns under this notion as well, by using a similar approach to derive the upper bound. That is, to get an upper bound on the relative discrimination score, we independently minimize/maximize $P(d \mid \mathbf{x}, \mathbf{y})$ and $P(d \mid \mathbf{y})$ over extensions, as described in Section 3.2 and Appendix B.1.

C Discussion of Summary Patterns

The summary patterns are motivated theoretically as a means to capture the most interesting discrimination patterns to ML practitioners for performing quick and efficient audits. For instance, maximal and minimal patterns are intended to be a succinct way of explaining away exponentially many patterns or the lack thereof. Pareto optimal patterns attempt to account for probability of a pattern in addition to its discrimination score.

To help the reader appreciate the quality of these patterns, in Section 2.2 we provide concrete examples of maximal and minimal patterns for model on the COMPAS dataset. For instance, a single minimal pattern ($\delta = 0.1$) was able to represent and hence explain away 512 patterns (its extensions) out of 1164 total discrimination patterns in the model. In this case, the minimal pattern was $\mathbf{x}$ = {Not Married} and $\mathbf{y}$ = {Low–Medium Supervision Level}. This tells us that no matter what other features are observed, the corresponding group of people would experience unfair treatment from the model; that is, minimal patterns identify the root of discrimination by representing exponentially many patterns. This sort of succinct information is clearly helpful to an ML practitioner auditing the model for fairness. As another example, recall that if $\mathbf{xy}$ is a maximal pattern, then no extension of it is a pattern. Hence, an individual with attributes $\mathbf{x}$ and $\mathbf{y}$ who may see a discrimination in the decision by disclosing their sensitive information would no longer receive such treatment if they additionally share other features. On a PC trained on COMPAS, we observe that among the 74 maximal patterns for $\delta = 0.05$, none of them includes an assignment to the variable regarding 'supervision level', suggesting that there are many instances where an individual would not see an unfair prediction if the supervision level is additionally known.

Our proposed summary patterns are a first attempt to capture a small subset of patterns that would be most helpful to practitioners, and we leave it to future work to explore other possible types of summaries.

D Recovering Minimal Patterns

Suppose we are given a set of patterns Σ. While a trivial algorithm to extract minimal patterns is quadratic in the number of patterns, we can recover the minimal patterns in time $\mathcal{O}\left(P \cdot N^2 + PNlog(PN)\right)$, where P is the number of potential patterns and $N = |\mathbf{Z}|$. Note that this is particularly of interest when the number of patterns is very large.

First, we pre-process the patterns to identify candidate minimal patterns, which are patterns all of whose extensions are also patterns. Note that we get this for for free in the case of exact search with minor modifications. Consider the poset of all possible assignments $\mathbf{xy}$ ordered by inclusion. We traverse this graph in level order, while maintaining a queue of nodes to visit and a set of assignments that we do not want to visit S. At the beginning of each level, we expand the nodes in S. Then, for each node in the queue for the current level that is not in S, if it is a candidate minimal pattern, we mark it as minimal, and add all its children to S. Otherwise, we add every child not in S to the queue for the next level.

E Sampling with Memoization

At a high level, there are key two additions to Algorithm 2 in Algorithm 4.

First, the weights to sample from immediate extensions are no longer merely their discrimination scores. Instead, we maintain an estimator $\Phi(\mathbf{x}, \mathbf{y})$ at each assignment corresponding to the expected discrimination score of an extension of $(\mathbf{x}, \mathbf{y})$ (not just the immediate extensions by a single variable). $\Phi(\mathbf{x}, \mathbf{y})$ is initialized as the discrimination score for every assignment $\mathbf{x}, \mathbf{y}$. After each sampling run (which refers to the complete extension path taken from the empty assignment $(\{\}, \{\})$ to a complete assignment), we backtrack to update $\Phi(\mathbf{x}, \mathbf{y})$ with our new information about average score of pattern encountered on the path after that assignment. More precisely, one can think of $\Phi(\mathbf{xy})$ as tracking the average discrimination score of a path of extensions from $(\mathbf{x}, \mathbf{y})$, averaged over all extension paths explored so far. Observe that in contrast to Algorithm 2, consecutive sampling runs are no longer independent, and later runs have a more informed heuristics for exploring the search space.

Second, at any particular assignment $(\mathbf{x}, \mathbf{y})$, the extensions are no longer directly sampled in proportion to $\Phi(\mathbf{x}, \mathbf{y})$. Instead, we instead introduce a power factor of $\Gamma(\mathbf{x}, \mathbf{y}) = \left(1 + \frac{|\mathbf{x}|+|\mathbf{y}|}{n}\right)$ as a heuristic for how strongly we wish to adhere to our estimator in picking our path. One can view this as a control for exploration versus exploitation as $\Gamma(\mathbf{x}, \mathbf{y})$ varies from 1 to 2. Intuitively, we are more open to exploration early on in our path as given a target pattern $(\mathbf{x}', \mathbf{y}')$, there are initially exponentially many paths to reach it. However, we prefer to exploit our estimator $\Phi(\mathbf{x}, \mathbf{y})$ as the number of extension paths to $(\mathbf{x}', \mathbf{y}')$ reduces later in the sampling run. We note that this is particularly of significance in settings where the number of variables (and consequently the search space) is large, as for most practical purposes we are interested in quickly finding the most interesting patterns, and not necessarily interested in exploring the search space to extract all possible patterns.

F Time Complexity

- The time complexity of the upper bound routine (Algorithm 3) is $\mathcal{O}\left(|C|^2\right)$ where $|C|$ is the size of the circuit.
- Algorithm 1 is a branch and bound search, with the worst-case time complexity $\mathcal{O}$ (Search Space * Upper Bound Computation) $= \mathcal{O}\left(2^n * |C|^2\right)$, where n is the number of attributes.
- Algorithm 4 (and its basic version Algorithm 2) is a sampling based approach which is run till timeout. Each sample is extremely efficient because for each attribute that we add in our construction, we only need to perform a few feed-forward evaluations of the circuit for computing marginals, which is linear time in the size of the circuit, resulting in a time complexity of $\mathcal{O}(n * |C|)$

G Additional Experimental Results

First, we report the log likelihood and number of nodes in the PCs learnt in our experiments (Table 5). In addition, in Table 6, we report an extended version of the results reported in Table 4. In particular, we compare the number and scores of patterns found by exact search and Algorithm 4 with a fixed timeout in various settings (dataset, type of score, and threshold). We find that Algorithm 4 consistently outperforms exact search in both number of patterns found and score of highest pattern found across different settings.

Table 5: Log likelihood and number of nodes in learnt PCs for various datasets

Dataset	Log likelihood	Number of nodes
Compas	-192194.49	89
Income	-201072.99	119
Adult	-974185.15	291

Algorithm 4 SAMPLE-DISC-PATTERNS$(\mathcal{C}, \mathbf{Z})$

Input: a PC $\mathcal{C}$ over variables $D \cup \mathbf{Z}$ and a threshold δ
Output: a set of sampled discrimination patterns Σ

$\Sigma \leftarrow \{\}$
$\Phi(\mathbf{x}, \mathbf{y}) \leftarrow \Delta(\mathbf{x}, \mathbf{y}) \quad \forall \mathbf{xy}$
$\sigma(\mathbf{x}, \mathbf{y}) \leftarrow 1 \quad \forall \mathbf{xy}$
repeat ▷ generate samples until timeout
  $(\mathbf{x}, \mathbf{y}) \leftarrow (\{\}, \{\})$
  $p \leftarrow []$
  while $|\mathbf{x}| + |\mathbf{y}| < n$ **do**
    for $(\mathbf{x}', \mathbf{y}') \in \mathsf{extensions}(\mathbf{x}, \mathbf{y}))$ **do** ▷ extensions by a single variable
      if $\Delta(\mathbf{x}', \mathbf{y}') > \delta$ **then** $\Sigma \leftarrow \Sigma \cup \{(\mathbf{x}', \mathbf{y}')\}$
    $(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{sample}_{\text{weight:}\Phi(\mathbf{x},\mathbf{y})^{\left(1+\frac{|\mathbf{x}|+|\mathbf{y}|}{n}\right)}}(\mathsf{extensions}(\mathbf{x}, \mathbf{y}))$
  $p \leftarrow p + (\mathbf{x}, \mathbf{y})$
  for $(\mathbf{x}, \mathbf{y}) \in \mathsf{reversed}(p))$ **do** ▷ update estimates
    $t \leftarrow \frac{\sum_{i=|\mathbf{x}|+|\mathbf{y}|}^{n} \Phi(p[i])}{n-|\mathbf{x}|-|\mathbf{y}|}$
    $\sigma(\mathbf{x}, \mathbf{y}) \leftarrow \sigma(\mathbf{x}, \mathbf{y}) + 1$
    $\Phi(\mathbf{x}, \mathbf{y}) \leftarrow \frac{\Phi(\mathbf{x},\mathbf{y}) \cdot (\sigma(\mathbf{x},\mathbf{y})-1)+t}{\sigma(\mathbf{x},\mathbf{y})}$
until timeout
return Σ

Table 6: Number of patterns and highest score of pattern found by exact search and sampling.

Score	Delta	Exact	Sampling	Score	Delta	Exact	Sampling
Discrimination	0.01	584	**980**	Discrimination	0.01	**0.2236**	0.2226
	0.05	347	**751**		0.05	**0.2236**	0.2230
	0.1	210	**347**		0.1	**0.2236**	0.2230
Divergence	0.01	586	**1186**	Divergence	0.01	0.0015	**0.0071**
	0.05	577	**1644**		0.05	0.0002	**0.0009**

(a) COMPAS dataset with a 3 second timeout

Score	Delta	Exact	Sampling	Score	Delta	Exact	Sampling
Discrimination	0.01	953	**2236**	Discrimination	0.01	0.1076	**0.1658**
	0.05	209	**1090**		0.05	0.1076	**0.1658**
	0.1	3	**225**		0.1	0.1076	**0.1658**
Divergence	0.01	840	**2366**	Divergence	0.01	0.0046	**0.0100**
	0.05	818	**2179**		0.05	0.0004	**0.0023**

(b) Income dataset with a 5 second timesout

Score	Delta	Exact	Sampling	Score	Delta	Exact	Sampling
Discrimination	0.01	42467	**127855**	Discrimination	0.01	0.6725	**0.6935**
	0.05	37167	**113763**		0.05	0.6725	**0.6871**
	0.1	30982	**99578**		0.1	0.6725	**0.6844**
Divergence	0.01	35792	**133780**	Divergence	0.01	0.0317	**0.2125**
	0.05	35292	**130232**		0.05	0.0162	**0.1098**

(c) Adult dataset with a 600 second timeout