Integrating Constraints via Probabilistic Circuits

Soroush Ghandi[1] **Benjamin Quost**[2] **Cassio de Campos**[1]

[1]Eindhoven University of Technology, Eindhoven, Netherlands.
[2]University of Technology of Compiègne, Compiègne, France.

Abstract

One of the recent advances in the domain of Probabilistic Circuits (PCs) is the introduction of methodologies for incorporating constraints into the represented distributions, thereby enabling the integration of external sources of information. In this paper, we investigate the extension of such paradigms to other classes of probabilistic models. In particular, we consider four representative models: continuous mixtures of tractable probabilistic models, Bayesian networks, Chow-Liu trees, and decision trees. We show that principled extensions of the techniques developed for PCs can be effectively applied to these models, thereby facilitating constrained optimization within a broader class of probabilistic frameworks.

1 INTRODUCTION

Probabilistic Circuits (PCs) Vergari et al. [2020] are a family of density representations that facilitate many exact and efficient inference routines. While rarely explored in some other domains, recent work in the domain of PCs introduces methods that can integrate constraints into the distribution represented by a PC [Ghandi et al., 2024, 2025]. In this paper, we aim to investigate whether such methods can be extended to other domains of probabilistic models, so that these domains can benefit from integrating constraints without domain-specific approaches. Recently proposed methods assume an initial PC encoding $p(\mathbf{X})$ and a set of constraints (e.g. probabilistic propositional logic (PPL) constraints of the form $\sum_i \tau_i \cdot p(F_i) \leq \alpha$, where τ_i, α are scalars and F_i is a propositional formula), and the goal is to find another PC encoding $q^*(\mathbf{X})$, such that it is as close as possible to $p(\mathbf{X})$ while respecting the constraints:

$$q^*(\mathbf{X}) = \operatorname*{argmin}_{q(\mathbf{X})} \mathrm{KL}(p(\mathbf{X}), q(\mathbf{X}))$$
$$\text{s.t.} \quad \forall c : \sum_{i_c} \tau_{i_c} \cdot q(F_{i_c}) \leq \alpha_c, \quad q(\mathbf{X}) \in \mathcal{P}(\mathbf{X}) \,.$$

where $\mathcal{P}$ has PCs with same structure as $p(\mathbf{X})$. For the sake of space and to dive quickly into the connections among models, we assume the reader knows about such methods in previous work [Ghandi et al., 2024, 2025].

We propose a natural and easy-to-implement generalization to other domains of probabilistic models by performing a 3-step process: i) convert the probabilistic model to a PC; ii) apply constraints to the equivalent PC using the methods proposed in [Ghandi et al., 2024, 2025]; and iii) convert the resulting PC back to the original probabilistic model. This way, we can enable other domains to benefit from constraints, by adapting the existing methods, and not by devising a specific method catered towards each domain. Hence, in this framework, we refer to the PC domain as the *intermediary domain* as well. Each one of the approaches explored in this paper can be considered as an instantiation of the above-mentioned process, and hence, we present this process as the core concept behind our discussions in this paper. Our main focus will be on 4 different types of models, namely continuous mixtures of tractable probabilistic models (CM-TPMs) [Correia et al., 2023], Bayesian networks (BNs), Chow-Liu trees (CLTs), and decision trees (DTs). For the rest of this section, we will briefly introduce each one of the model types we intend to investigate.

A Bayesian network (BN) represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). The joint probability distribution of a Bayesian network is computed as the product of its conditional probability distributions, i.e. $p(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \mathbf{X}_{\pi(i)})$.

Where X_i are the variables of the BN, and $\mathbf{X}_{\pi(i)}$ are the parents of X_i in the DAG. Chow-Liu trees (CLTs) [Chow and Liu, 1968] can be considered as a specific type of Bayesian

networks that represent a joint probability distribution structured as a *tree*, that is, all variables have one parent (except for the one called root which has no parents).

Continuous mixtures of tractable probabilistic models utilize a hybrid of tractable models and continuous models, in order to get the best of both worlds. More specifically, a continuous mixture of tractable models is written as $p(\mathbf{x}) = \mathbb{E}_{p(z)}[p(\mathbf{x}|\phi(z))] \approx \int p(\mathbf{x}|\phi(z))p(z)dz$, where $p(z)$ is a continuous prior, usually an isotropic Gaussian, $\phi(z)$ is a decoder (e.g. a neural network) that maps the latent space to the parameters of a tractable probabilistic model (e.g. a PC in form of a factorized distribution), and $p(\mathbf{x}|\phi(z))$ is a tractable model with parameters determined by z.

Decision Trees (DTs) are one of the most widely known non-linear machine learning tools. A DT has a hierarchical tree structure, which consists of a root node, branches, internal nodes and leaf nodes. The root node of the structure represents the entire dataset that the DT is trained on, each internal node represents a split/partition on the data based on the values of one or more features, and the branches represent each distinct outcome of the partitioning. Finally, leaf nodes are the terminal nodes of the structure that represent the final outcomes or predictions of the DT.

2 CONTINUOUS MIXTURES OF TRACTABLE PROBABILISTIC MODELS

Using numerical methods to approximate the joint probability distribution of a CM-TPM will result in a finite mixture of tractable models, i.e. $p(\mathbf{x}) = \sum_{i=1}^{N} w(z_i)p(\mathbf{x}|\phi(z_i))$. This means that if the tractable model of choice $p(\mathbf{x}|\phi(z))$ is a PC, each approximation would be a finite mixture of PCs, which is a PC itself. This essentially means that the proposed methods in Ghandi et al. [2024, 2025] to apply constraints to PCs can be applied to any numerical approximation of CM-TPMs without any necessary modifications.

Contrary to the convex approach [Ghandi et al., 2024], sample-based methods [Ghandi et al., 2024] allow for propagating the gradient information through the decoder, which means that we can train the decoder to respect fairness. In this scenario, instead of learning a constrained approximation, we will end up with a constrained continuous model, which can generalize the constraints to any arbitrary numerical approximation during the test time. As a result, we will use the sample-based approach in our experiments.

The general scheme of our approach for training constrained CM-TPMs is shown in Figure 1. We consider tractable probabilistic models of the CM-TPM to be in the form of PCs, and as a result, we will refer to such models as CM-PCs. At each iteration of training, a numerical approximation of the CM-PC is calculated (i.e. a discrete mixture of PCs which is also a PC), which is trained using a regularized objective function, and its gradient information is then propagated back through the decoder of the structure, which leads to a constrained CM-PC.

We consider tractable probabilistic models of the CM-TPM to be fully factorized distributions. As for the constraints, we consider applying fairness in the form of distribution statistical parity (DSP), sample-based statistical parity (SBSP), and sample-based equalized odds (SBEO), in two scenarios where classification is performed by sampling and argmax. For a brief introduction and formulation of DSP, SBSP, and SBEO, please refer to [Ghandi et al., 2024, 2025], and Appendix A respectively. We use the Adult dataset [Le Quy et al., 2022] for this experiment. The results are averaged over 5 random initializations, and are shown in Table 1. The bold number in each row indicates the targeted fairness measure in each of the experiments.

Table 1: Performance of constrained CM-TPMs.

Train Target	Test LL	DSP	Sampling			Argmax		
			Acc.	SBSP	SBEO	Acc.	SBSP	SBEO
No Target	-13.425	0.177	0.760	0.190	0.237	0.824	0.167	0.215
DSP	-13.440	**0.007**	0.749	0.090	0.115	0.819	0.066	0.154
SBSP - Sampling	-13.447	0.019	0.723	**0.019**	0.098	0.816	0.078	0.105
SBEO - Sampling	-13.455	0.047	0.696	0.051	**0.057**	0.817	0.108	0.109
SBSP - Argmax	-13.429	0.11	0.752	0.118	0.121	0.810	**0.013**	0.264
SBEO - Argmax	-13.398	0.160	0.755	0.165	0.182	0.823	0.107	**0.089**

Each row corresponds to optimizing a particular constraint; for instance, "**SBSP - Sampling**" refers to the case where we aim to enforce sample-based statistical parity assuming the classification is done via sampling. The correspondence in columns follows a similar sense. As the results clearly suggest, applying various forms of constraints turns out to be impactful in CM-TPMs; in all cases, applying constraints effectively mitigates the targeted bias in models, without significantly harming the performance. However, based on the reported results, we can see that applying DSP to models is more effective than applying sample-based constraints. DSP directly targets the distribution, and does not rely on the samples to integrate fairness; as a result, its optimization does not have the challenges that come with sample-based (regularized) integration of constraints.

3 BAYESIAN NETWORKS

Similar to the discussion in Section 1, we apply constraints to BNs in a 3-step process, where we first take the initial BN and convert it to a PC, then apply the constraints to the resulting PC, and finally, we convert the constrained PC back to the BN.

We consider applying distribution statistical parity (DSP) to a given BN learned from the binarized version of the Adult dataset [Ghandi et al., 2024]. We consider CM-TPMs as the intermediary PC structures in our 3-step pipeline.

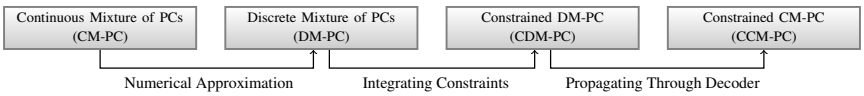

Figure 1: General Scheme for integrating constraints into CM-PCs.

As our method for converting a BN to a PC, we use the learned BN to generate synthetic samples, and then the produced samples will be further used to train the CM-TPM/intermediary PC. Then, we apply DSP to the intermediary CM-TPM, based on the sample-based method proposed in [Ghandi et al., 2025]. In order to convert the resulting PC back to the BN domain, we simply query the parameters of each CPD $P(X_i|\mathbf{X}_{\pi(i)})$ in the initial BN from the constrained PC. The same process is also used for integrating constraints into CLTs as well. A general scheme of this approach is shown in Figure 2.

We expect that the process of converting a BN to a PC and back to a BN, even without applying any particular constraints to the intermediary PC, to lead to a performance loss in the BN domain. To differentiate between the performance loss of conversion and that of applying constraints, we perform two different experiments, one with and one without applying constraints, which we denote by "**DSP**" and "**No Target**" respectively. The results of these experiments, averaged over 5 random initializations, are outlined in Table 2.

Table 2: Distribution Statistical Parity in BNs.

Test Log-Likelihood			
Train Target	**BN**	**PC**	**PC2BN**
No Target	-7.515	-7.761	-7.732
DSP	-7.515	-7.780	-7.747
Statistical Parity			
Train Target	**BN**	**PC**	**PC2BN**
No Target	0.178	0.203	0.180
DSP	0.178	0.009	0.114
Test Accuracy			
Train Target	**BN**	**PC**	**PC2BN**
No Target	0.8388	0.8297	0.8325
DSP	0.8388	0.8210	0.8349

The column denoted by "**BN**" refers to the results of the initial/given Bayesian network; the column denoted by "**PC**" refers to the results of the (constrained) conversion to PC domain; and the column denoted by "**PC2BN**" refers to the Bayesian network resulting from the conversion of the (constrained) PC back to the BN domain.

We can see that the proposed 3-step process successfully improves the behavior of BNs towards respecting the constraints. First, we would like to note that while the accuracy of the models remains comparable to one another, the process of converting a BN to a PC (even without integrating any constraints) results in a notable loss of performance (in terms of test log-likelihood) when compared to the original BN. Although this performance loss can be mitigated in various ways, it comes at the cost of higher computation and training time, and it will still be a trade-off between having better likelihood and applying constraints. As a result, we consider our method to be better suited when the accuracy of predictions takes precedence over the likelihood of the models.

On another note, we see that although our model mitigates the bias towards the sensitive variable in the final BN, it fails to completely nullify it. This is mainly because of the fact that after applying the constraints, the resulting PC may not be fully explained with a structure similar to the original BN. In other words, after applying the constraints to the converted PC, the interactions and dependencies between the variables may change, and in the case that some dependency between variables does not exist in the structure of the initial BN in the form of one or more edges of the graph, then its information will be lost during the conversion of the PC back to the BN. As a result, while our method can guarantee improvement, it cannot guarantee that the model completely respects the constraints. Theoretically, this can be addressed by adding extra constraints that correspond to maintaining the original structure of the BN; however, in practical scenarios, this translates to applying numerous structural constraints, which makes the cost function hard to properly optimize.

4 CHOW-LIU TREES

Since Chow-Liu trees (CLTs) can be considered as a special case of Bayesian networks, the experiment setup for the CLTs is quite similar to that of generic BNs. Nevertheless, applying fairness to a CLT is counter-intuitive; this process aims to adjust the model in a way that the class and sensitive variables become independent of one another. In a CLT, however, the only way to adjust the model to respect fairness is by removing one of the edges along the path between the class and sensitive variables, which breaks the structure of the CLT and turns it into a forest, which is not the intended result.

Therefore, throughout our experiments on CLTs, instead of fairness, we assume matching the marginals of the distribution as the task at hand [Ghandi et al., 2024, 2025]. We will work with the binarized version of the Adult dataset.

We assume that we only have a subsample of 100 datapoints from the dataset (to simulate a lack of information which can be compensated for with the information encoded in the marginals, just as if an expert would have given that information to us). We also assume that we have access to

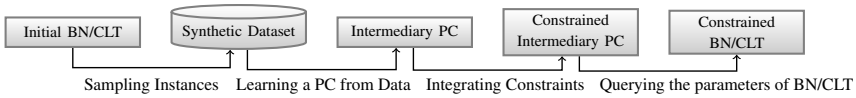

Figure 2: General Scheme for integrating constraints into BNs/CLTs.

the information on the marginals of the test data (i.e. we have $p(X_i) = \alpha_i$ for all the variables, empirically calculated on test data, as a source of extra information to be incorporated in the learning process), and we want to use this information to improve the distribution represented by a CLT.

First, using the 100 training samples, we learn a CLT to represent our initial/given model. Next, we generate synthetic samples from the learned CLT and use them to train the constrained intermediary PC; finally, we convert the constrained PC back to the CLT by querying the parameters of each CPD from the constrained PC. The results of these experiments are summarized in Table 3. In order to highlight the impact of applying marginal constraints in our experimental setup, we also show the results of performing our 3-step process without applying any constraints to the intermediary PC, which are shown throughout the results labeled as "**No Target**". The column denoted by "**CLT**" refers to the results of the initial/given Chow-Liu tree; the column denoted by "**PC**" refers to the results of the (constrained) conversion to the PC domain; and the column denoted by "**PC2CLT**" refers to the CLT resulting from the conversion of the (constrained) PC back to the CLT domain.

Table 3: Matching marginals in CLTs.

	Test Log-Likelihood		
Train Target	**CLT**	**PC**	**PC2CLT**
No Target	-8.816	-8.722	-8.730
Marginals	-8.816	-8.570	-8.538

Based on the results, we see that our 3-step process manages to improve the performance of the initial CLT in terms of test log-likelihood, which highlights the effectiveness of our approach in the domain of CLTs.

5 DECISION TREES

For the experiments on DTs, once again we assume integrating fairness in the form of statistical parity into the models. Given a learned DT and the dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, Correia et al. [2020] propose a method to obtain a corresponding generative model, which we use as our method to convert a DT into a PC. Next, we use convex optimization method proposed in [Ghandi et al., 2024] to integrate fairness into the resulting PC, which can be considered as an attempt to find the *closest* constrained/fair PC to the original DT. The general scheme of this approach is outlined in Figure 3.

The experiments in this section are performed on Adult, German Credit, Bank Marketing, Dutch Census, Credit Cards Clients, and Law School datasets, and the results are outlined in Table 4.

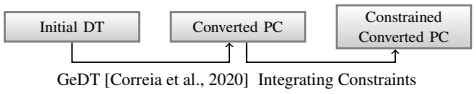

Figure 3: Integrating constraints into DTs.

Table 4: Convex optimization [Ghandi et al., 2024] for integrating fairness into decision trees.

Dataset	**DT Acc.**	**DTSP**	**PC Acc.**	**PCSP**	**PCDSP**
Adult	0.835	0.163	0.804	0.035	$\leq 10^{-6}$
German Credit	0.683	0.054	0.671	0.063	$\leq 10^{-6}$
Bank Marketing	0.878	0.028	0.8775	0.008	$\leq 10^{-6}$
Dutch Census	0.800	0.169	0.736	0.007	$\leq 10^{-6}$
Credit Cards Clients	0.720	0.027	0.560	0.034	$\leq 10^{-6}$
Law School	0.865	0.314	0.894	0.013	$\leq 10^{-6}$

In Table 4, **DTSP** refers to the statistical parity of the DT computed on the test data (since a DT is not a generative model, we cannot directly calculate distribution statistical parity (DSP)), **PCSP** refers to the statistical parity of the constrained/fair PC on test data, and **PCDSP** refers to the distribution statistical parity of the PC. As the results in Table 4 would suggest, the resulting PC manages to successfully integrate fairness (defined here as statistical parity) without significant performance loss in 5 out of 6 datasets, which is in line with the results reported in [Ghandi et al., 2024]. This essentially means that our method can be used to apply constraints (in this case, fairness) to DTs, as long as we can be lenient with the generative nature of the PCs. In a practical sense, this is a very mild assumption, since deterministic inference in PCs (e.g. classification with argmax) is particularly similar to treating a PC as a DT and then performing the classification.

6 CONCLUSION

In this paper, we discussed the possibility of integrating constraints into various probabilistic models, by extending the ideas proposed in [Ghandi et al., 2024, 2025]. For each specific model of interest (CM-TPMs, BNs, CLTs, and DTs), we explained a natural approach to adapt the PC-specific methods, and show that our adaptations can integrate constraints into these probabilistic models, albeit with their own caveats. These experiments might open doors for further research on PCs as intermediate models in other applications.

7 ACKNOWLEDGMENTS

The authors thank the support from (i) Eindhoven Artificial Intelligence Systems Institute, (ii) Department of Mathematics and Computer Science of TU Eindhoven, and (iii) Dutch Research Council (NWO) via project NGF.1609.242.024.

References

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

Alvaro Correia, Robert Peharz, and Cassio P de Campos. Joints in random forests. In *Advances in neural information processing systems*, volume 33, pages 11404–11415, 2020.

Alvaro HC Correia, Gennaro Gala, Erik Quaeghebeur, Cassio de Campos, and Robert Peharz. Continuous mixtures of tractable probabilistic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7244–7252, 2023.

Soroush Ghandi, Benjamin Quost, and Cassio de Campos. Probabilistic circuits with constraints via convex optimization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2024, Vilnius, Lithuania, September 9-13, 2024, Proceedings, Part II 10*, pages 161–177. Springer, 2024.

Soroush Ghandi, Benjamin Quost, and Cassio de Campos. Imposing constraints in probabilistic circuits via gradient optimization. In *International Symposium on Intelligent Data Analysis*, pages 209–220. Springer, 2025.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Tai Le Quy, Arjun Roy, Vasileios Iosifidis, Wenbin Zhang, and Eirini Ntoutsi. A survey on datasets for fairness-aware machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(3):e1452, 2022.

A. Vergari, YJ. Choi, R. Peharz, and G Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. `http://starai.cs.ucla.edu/slides/AAAI20.pdf`, 2020. Tutorial at AAAI 2020.

Integrating Constraints via Probabilistic Circuits

Soroush Ghandi[1] **Benjamin Quost**[2] **Cassio de Campos**[1]

[1]Eindhoven University of Technology, Eindhoven, Netherlands.
[2]University of Technology of Compiègne, Compiègne, France.

A SAMPLE-BASED EQUALIZED ODDS

In this section, we will define sample-based equalized odds (SBEO) as a constraint in the overall loss function of the gradient descent, similar to the way sample-based statistical parity (SBSP) is defined in [Ghandi et al., 2024]. Nonetheless, we reiterate the notations for the sake of completeness.

We assume that we have a dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where each $\mathbf{x}_i$ is defined over the scope $\mathbf{X}$ with $X' \in \mathbf{X}$ representing a binary protected attribute. Y represents the binary class/target variable. We also assume having an arbitrary probabilistic model $q(X, Y)$ that represents the joint distribution over X and Y. Additionally, we define two different approaches for classification, which results in predicted label $\hat{\mathbf{y}}$ based on the learned probabilistic model q.

We consider two decision criteria for classification, namely class assignment via argmax:

$$\hat{y} = \text{argmax}_y \; q(Y = y|\mathbf{X} = x), \tag{1}$$

and classification via sampling:

$$\hat{y} \sim q(Y|\mathbf{X} = x)\,. \tag{2}$$

For the most part, equalized odds [Hardt et al., 2016] is similar in definition to statistical parity, except that equalized odds is conditioned on the true label $\mathbf{y}$. In other words, equalized odds is preserved when statistical parity is preserved in groups defined by the true label $\mathbf{y}$:

$$Eq.Odds : Pr(\hat{\mathbf{y}} = 1|X = 0, Y = \mathbf{y}) = Pr(\hat{\mathbf{y}} = 1|X = 1, Y = \mathbf{y}) \tag{3}$$

Based on this definition, we can measure the prevalent bias as:

$$eo = \sum_{\mathbf{y}\in\{0,1\}} |Pr(\hat{\mathbf{y}} = 1|X = 0, Y = \mathbf{y}) - Pr(\hat{\mathbf{y}} = 1|X = 1, Y = \mathbf{y})| \tag{4}$$

Because of its dependence on the true label $\mathbf{y}$, equalized odds can only be considered in a sample-based scenario, where we have access to the true labels of data. Taking a similar frequentist approach to [Ghandi et al., 2025], we can measure equalized odds for an arbitrary model q. Assuming for each data $\mathbf{x}_i \in \mathcal{D}$ the model q predicts a corresponding label $\hat{\mathbf{y}}_i$, we have:

$$eo_q = |sp_q^0| + |sp_q^1| \tag{5}$$

Where,

$$\begin{aligned} sp_q^0 &= \frac{\sum_i \mathbb{1}(\hat{\mathbf{y}}_i = 1, \mathbf{x}'_i = 0, \mathbf{y} = 0)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 0, \mathbf{y} = 0)} - \frac{\sum_i \mathbb{1}(\hat{\mathbf{y}}_i = 1, \mathbf{x}'_i = 1, \mathbf{y} = 0)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 1, \mathbf{y} = 0)} \\ sp_q^1 &= \frac{\sum_i \mathbb{1}(\hat{\mathbf{y}}_i = 1, \mathbf{x}'_i = 0, \mathbf{y} = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 0, \mathbf{y} = 1)} - \frac{\sum_i \mathbb{1}(\hat{\mathbf{y}}_i = 1, \mathbf{x}'_i = 1, \mathbf{y} = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 1, \mathbf{y} = 1)} \end{aligned} \tag{6}$$

We can rewrite the formulations in 6 as,

$$
\begin{aligned}
sp_q^0 &= \frac{\sum_{i;\mathbf{x}'_i=0;\mathbf{y}=0} \mathbb{1}(\hat{\mathbf{y}}_i = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 0, \mathbf{y} = 0)} - \frac{\sum_{i;\mathbf{x}'_i=1;\mathbf{y}=0} \mathbb{1}(\hat{\mathbf{y}}_i = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 1, \mathbf{y} = 0)} \\
sp_q^1 &= \frac{\sum_{i;\mathbf{x}'_i=0;\mathbf{y}=1} \mathbb{1}(\hat{\mathbf{y}}_i = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 0, \mathbf{y} = 1)} - \frac{\sum_{i;\mathbf{x}'_i=1;\mathbf{y}=1} \mathbb{1}(\hat{\mathbf{y}}_i = 1)}{\sum_i \mathbb{1}(\mathbf{x}'_i = 1, \mathbf{y} = 1)} .
\end{aligned}
\tag{7}
$$

Now, we just need to define a differentiable estimator to $\mathbb{1}(\hat{y}_i = 1)$. As discussed in [Ghandi et al., 2025], for the case of classification with sampling, we use $q(Y = 1|\mathbf{X} = x_i)$ as an unbiased estimator of $\mathbb{1}(\hat{y}_i = 1)$; and for the case of classification with *argmax*, we mimic the behavior of $\mathbb{1}(\hat{y}_i = 1)$ by applying a sharp softmax to $q(Y = 1|\mathbf{X} = x_i)$. This way, both of the terms sp_q^0 and sp_q^1 become differentiable, and the equalized odds measure in 5 can be treated as a regularization term, and be optimized through gradient descent. Note that throughout our experiments in Section 2, we do not directly optimize the formulation in 5; in order to improve stability of the learning, we use a similar, albeit smoother curve:

$$
\hat{eo} = (sp_q^0)^2 + (sp_q^1)^2 \tag{8}
$$