Fast Implicit Constrained Optimization of Non-decomposable Objectives for Deep Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider a popular family of constrained optimization problems in machine learning that involve optimizing a non-decomposable objective while constraining another. Different from the previous approach that expresses the classifier thresholds as a function of all model parameters, we consider an alternative strategy where the thresholds are expressed as a function of only a subset of the model parameters, i.e., the last layer of the neural network. We propose new training procedures that optimize for the bottom and last layers separately, and solve them using standard gradient based methods. Experiments on a benchmark dataset demonstrate our proposed method achieves performance comparable to the existing approach while being computationally efficient.

1 Introduction

Many machine learning applications involve optimizing for *non-decomposable metrics*, such as false positive (FPR) or precision, while constraining another, such as false negative rate (FNR) or recall. For these metrics, the loss on a set of data points cannot be expressed as the sum of losses on individual data points (e.g., FPR involves a ratio that depends on the total false-positive and true-negative examples). Such problems arise in several applications. For example, many fairness objectives can be expressed as rate constraints, including the popular *equal opportunity* and *equalized odds* fairness criteria [1, 4, 11, 12]. The typical baseline for solving these problems is first to train the model using regular cross-entropy (CE) loss, followed by tuning the thresholds imposed on the model predictions to classify between positive versus negative classes while the constraint is satisfied. However, the post hoc approach often yields sub-optimal results as the model parameters are trained on a different objective than the evaluation metric. To improve upon the baseline approach, *Implicit Constrained Optimization* (ICO) [6] reformulates the prediction threshold as a function of the model parameters and transforms the constraint optimization problem into an unconstrained one to jointly optimize for the model parameters and the thresholds. ICO solves the problem by decomposing the gradient into a sum of disjoint terms, which consist of the gradient of the loss function and the constraint with respect to the model parameters and the thresholds. Such a decomposition is computationally more expensive than standard training since it requires additional forward-backward passes to compute the gradients with respect to each component. In particular, ICO becomes more expensive when there are multiple constraints, e.g., when optimizing for the area under the ROC Curve (AUC). To reduce the computational cost, ICO resorts to approximations, e.g., intermittently updating the thresholds – Sec 3.3 in [6]. Thus, it is desirable to close this computation gap while retaining the simplicity of the method as well as potentially improved performance on the metrics.

Motivation of This Work: Constrained Optimization of Only the Last Layer Standard neural network architectures can be decoupled into a *feature extractor* sub-network at the bottom, followed by a single fully-connected *classifier* layer to predict the class probabilities at the top (see Figure 1). Several recent studies have shown that the feature extractor network can be trained reasonably well

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

even when the training data is noisy [7] or imbalanced [9, 5]. These studies show the significance of retraining the last layer weights on a clean or balanced validation set in obtaining good performance based on the feature representation extracted by the bottom sub-network. While it is natural to impose the constraint on all the model parameters in our problem, the results from the earlier work raise the question of whether constraining the last layer weights is sufficient for obtaining desired performance [6].

To this end, we propose to make the prediction thresholds a function only of the last layer parameters. This simplification leaves the parameters of the earlier layers free to be optimized with any surrogate loss in an unconstrained manner. In the meantime, the constraint is imposed on the last layer parameters. The major benefit of our approach is avoiding additional forward-backward passes to calculate the gradient of the constraint(s) with respect to the weights of the bottom sub-network.

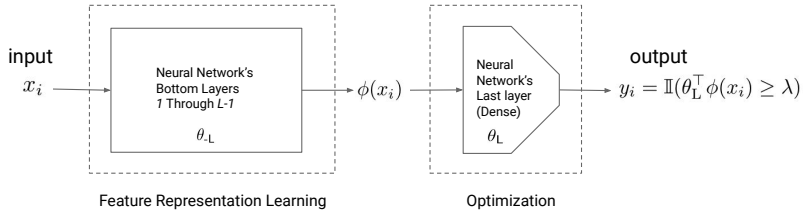

Figure 1: Demonstration of the bottom layer weight $\theta_{\text{-L}}$ and top/last layer weights θ_{L} of a neural network.

2 Problem Formulation

Basic Setup In this paper, we consider a binary classification setting with feature space X and binary label $\{0, 1\}$. Our goal is to learn a *binary threshold scoring rule* $s_{\theta|\lambda} : X \to \{0, 1\}^m$ parameterized by model parameter $\theta \in \mathbb{R}^p$ and *threshold* values $\lambda \in \mathbb{R}^m$, such that for any given input $x \in X$, $[s_{\theta|\lambda}(x)]_i = \mathbb{1}(s_\theta(x) \geq \lambda_i)$ where $s_\theta(x) : X \to \mathbb{R}$ is a scoring model parameterized by the model parameters $\theta \in \mathbb{R}^p$ and $\mathbb{1}(\cdot)$ is the indicator function.

We assume the scoring model s_θ is an L-layer neural network whose parameters can be divided into two disjoint parts $\theta = [\theta_{\text{-L}}, \theta_{\text{L}}] \in \mathbb{R}^p$: the parameters of the last dense layer $\theta_{\text{L}} \in \mathbb{R}^{p_{\text{L}}}$, and the weights from the rest of the network, which we will also refer to as the bottom layers: $\theta_{\text{-L}} \in \mathbb{R}^{p_{\text{-L}}}$, where $p = p_{\text{-L}} + p_{\text{L}}$. The bottom layers transform the input x into a feature representation $\phi(x)$. The last dense layer then sets $s_\theta(x) = \theta_{\text{L}}^\top \phi(x)$. In this work, we explore ideas involving faster training of deep networks using non-decomposable objectives by applying different strategies for training the last layer weights θ_{L} and the rest of the weights $\theta_{\text{-L}}$.

Objective Function In this paper, we consider a general form of constrained optimization problem with m constraints:

$$\min_{\theta \in \mathbb{R}^p} f(\theta, \lambda) \quad \text{s.t.} \quad g(\theta, \lambda) = \mathbf{0}\,, \tag{1}$$

where $f : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^m$ are non-decomposable functions that cannot be expressed as the sum of losses on individual data points, $\theta \in \mathbb{R}^p$ is the model parameter, and $\lambda \in \mathbb{R}^m$ are the thresholds. Similar to [6], throughout the training process, we assume that θ stays in the feasible region, namely $\forall\theta, \exists\lambda$ s.t. $g(\theta, \lambda) = \mathbf{0}$.

Here, we provide one example that satisfies this general constraint optimization form, which is also going to be the main task of our experimental demonstration in Section 4:

Example 1 (Minimizing FNR subject to a fixed FPR)**.** Consider a setting where we want to minimize the false negative rate (FNR) at the threshold $\lambda \in \mathbb{R}$ at which its false positive rate (FPR) is a fixed value $\beta \in [0, 1]$:

$$f(\theta, \lambda) = \text{FNR}(s_{\theta|\lambda}) = \frac{\text{FN}(s_{\theta|\lambda})}{\text{TN}(s_{\theta|\lambda}) + \text{FP}(s_{\theta|\lambda})},\ g(\theta, \lambda) = \text{FPR}(s_{\theta|\lambda}) = \frac{\text{FP}(s_{\theta|\lambda})}{\text{TP}(s_{\theta|\lambda}) + \text{FN}(s_{\theta|\lambda})} - \beta\,,$$

where $\text{FN}(s_{\theta|\lambda})$, $\text{TP}(s_{\theta|\lambda})$, and $\text{TN}(s_{\theta|\lambda})$ denote the true positive, false positive, and false negative rates, respectively, for the threshold scoring rule.

Surrogate Losses It is possible that the optimization objectives and constraints described above are non-differentiable. To make the training viable using gradient descent based methods, we replace f and g with their corresponding *smooth surrogate losses*, $\tilde{f}$ and $\tilde{g}$, and relax Eq. (1) into[1]:

$$\min_{\theta \in \mathbb{R}^p} \tilde{f}(\theta, \lambda) \quad \text{s.t.} \quad \tilde{g}(\theta, \lambda) = \mathbf{0}\,. \tag{2}$$

3 Methods

We first briefly discuss the previous methods and introduce our efficient alternative.

i) Previous Approach: Implicit Constraint Optimization (ICO) on All Weights Different from using traditional Lagrangian function based method [], ICO [6] avoids explicitly solving the constrained problem in Eq. (2) by formulating an equivalent unconstrained problem in which they express the threshold λ as an implicit function of the model parameters θ (within a neighborhood around θ). Specifically, the implicit function theorem [10] implies: suppose (θ_0, λ_0) satisfies the constraints, namely $\tilde{g}(\theta_0, \lambda_0) = 0$. Then, we can express the threshold as $\lambda_0 = \tilde{h}(\theta_0)$ in the neighborhood of θ_0 where $\tilde{h}$ is some implicit function that depends on problem setup and the local structure of (θ_0, λ_0). See the detailed description of the theorem (Theorem 1) in the appendix.

The benefit of using Theorem 1 is that we can change the constrained optimization problem to be an unconstrained optimization problem with only one free variable θ:

$$\min_{\theta} \tilde{f}(\theta, \lambda) \quad \text{s.t.} \quad \tilde{g}(\theta, \lambda) = \mathbf{0} \overset{\text{Theorem 1}}{\Longrightarrow} \min_{\theta} \tilde{f}(\theta, \tilde{h}(\theta)) \tag{3}$$

A differentiable function $\tilde{h}(\theta)$ that provide us $\tilde{g}(\theta, \tilde{h}(\theta)) = \mathbf{0}$ for a particular threshold that we care about might not always exist. However, [6] identifies some conditions under which the resulting Eq. (3) is convex in θ.[2] We also provide detailed derivations in the appendix.

ii) Proposed Approach: Constraining the last layer weights θ_{L} only In this work, we propose making the threshold λ a function of only the last layer's weight θ_{L}, and performing different training procedure for the bottom layer weight $\theta_{\text{-L}}$ and last layer weight θ_{L} separately.

Implicit Function Theorem on θ_{L} First, we propose to make the threshold λ in the constraint $\tilde{g}(\theta, \lambda) = \mathbf{0}$ a function of only θ_{L}, which gives us the following modified form of the implicit function theorem (see the appendix). The modified theorem implies that given $\phi(x)$, the classifier weights θ_{L} control the value of the threshold λ in a feasible region.

New Objective Form: Separate Training for $\theta_{\text{-L}}$ and θ_{L} We propose a new optimization procedure for training a neural network that optimize its bottom layer weight $\theta_{\text{-L}}$ and last layer weight θ_{L} separately using different training objectives:

$$\min_{\theta_{\text{-L}}} \tilde{r}(\theta_{\text{-L}} | \theta_{\text{L}}, \lambda) \quad \text{and} \quad \min_{\theta_{\text{L}}} \tilde{\ell}(\theta_{\text{L}}, \lambda | \theta_{\text{-L}}) \quad \text{s.t.} \quad \tilde{g}(\theta_{\text{L}}, \lambda | \theta_{\text{-L}}) = \mathbf{0}\,. \tag{4}$$

where $\tilde{r} : \mathbb{R}^{p_{\text{-L}}} \times \mathbb{R}^m \to \mathbb{R}^m$ and $\tilde{\ell} : \mathbb{R}^{p_{\text{L}}} \times \mathbb{R}^m \to \mathbb{R}^m$ are differential objective function for $\theta_{\text{-L}}$ and θ_{L} separately. For the objective function $\tilde{r}$, the notation $(\theta_{\text{-L}} | \theta_{\text{L}}, \lambda)$ means we optimize over $\theta_{\text{-L}}$ while fixing θ_{L} and λ. The notation holds similarly for $\tilde{\ell}$ and $\tilde{g}$. Using this convention, we can write $\tilde{f}(\theta, \lambda) = \tilde{r}(\theta_{\text{-L}} | \theta_{\text{L}}, \lambda) + \tilde{\ell}(\theta_{\text{L}}, \lambda | \theta_{\text{-L}})$.

We assume the conditions in Theorem 2 hold, which means that we can re-write the threshold λ as a function of the last layer weight θ_{L} (i.e., $\lambda = \tilde{h}_{\text{L}}(\theta_{\text{L}})$). The specification of the function form $\tilde{h}_{\text{L}}(\cdot)$ holds with respect to the modified setting.

The main focus of this work is to explore different training objectives and training strategies that takes in the form of Eq. (4) to speed up the training procedure and get better prediction outcomes.

• **Method 1: minimizing CE for $\theta_{\text{-L}}$ and FNR for θ_{L}** Our first proposed method is to use cross-entropy (CE) loss as the objective function for the bottom network $\theta_{\text{-L}}$, and use false negative rate (FNR) only for updating the gradient for θ_{L}. This corresponds to specify $\tilde{r}$, $\tilde{\ell}$, and $\tilde{g}$ in Eq. (4) as follows:

$$\tilde{r}(\theta_{\text{-L}} | \theta_{\text{L}}, \lambda) = \text{CE}(\theta_{\text{-L}} | \theta_{\text{L}}), \quad \tilde{\ell} = \widetilde{\text{FNR}}, \quad \tilde{g} = \widetilde{\text{FPR}} - \beta$$

where $\beta \in [0, 1]$ is the target FPR rate for the constraint.

[1] We provide an example of the surrogate loss for our particular setting involves in $f = \text{FNR}$ in the appendix.

[2] See Proposition 1 in Section 3 in [6].

Table 1: Minimizing false negative rate (FNR) at a given false positive rate (FPR) for **celebA**. The mean FNR (in %) are reported over five random trails for different methods including (CE/TFCO/ICO) [6]/CE+FNR/FPR Regularizer on θ_{L}/FPR Regularizer on θ, respectively. Among the three proposed methods, FPR Regularizer on θ_{L} outperforms the other two. **Bold** indicates the best performance among the three newly proposed methods. The lower value the better.

FPR	High-cheekbones	Smiling	Wearing-lipstick
1%	(53.5/49.0/46.9) 51.3/**46.4**/48.2	(37.4/35.9/33.7) 37.5/**32.7**/37.1	(44.0/42.6/37.5) 42.40/**38.8**/40.5
2%	(44.8/40.9/39.8) 45.1/**40.0**/41.8	(29.4/27.8/26.1) 30.2/**26.5**/27.5	(32.7/30.4/26.7) 31.74/29.7/**28.3**
5%	(32.9/30.1/28.5) 31.9/**28.9**/29.4	(18.7/17.0/16.9) 18.9/**16.4**/16.6	(16.3/14.9/13.1) 15.4/15.4/**15.1**
10%	(22.9/20.4/19.7) 22.8/**20.2**/**20.2**	(11.7/10.7/10.2) 12.1/**10.7**/11.0	(6.6/5.9/4.7) 7.0/**5.3**/5.5

• **Method 2: minimizing FNR with regularization** Our second proposed method involves minimizing FNR for the whole network while imposing FPR constraints on the last layer weight. In addition, we impose a constraint regularizer on $\theta_{\text{-L}}$. This corresponds to specify $\tilde{r}$, $\tilde{\ell}$, and $\tilde{g}$ in Eq. (4) as follows:

$$\tilde{r} = \widetilde{\mathrm{FNR}} + \eta \left|\widetilde{\mathrm{FPR}} - \beta\right|, \tilde{\ell} = \widetilde{\mathrm{FNR}}, \tilde{g} = \widetilde{\mathrm{FPR}} - \beta, \tag{5}$$

where $\beta \in [0, 1]$ is the target FPR rate for the constraint and $\eta > 0$ balances between minimizing the FNR loss and the regularization term. The intuition behind adding this regularization term is to avoid trivially minimizing FNR by always predicting one at the output for any given input. Also, the regularized objective form in Eq. (5) resembles the Lagrangian method proposed in [3, 2, 8] for solving constrained optimization problems.

• **Regularization on All Layers Weights** For comparison, we also try imposing the constraint regularizer on all weights. This is equivalent of modifying $\tilde{\ell}$, the objective function for θ_{L} in Eq. (5) by adding a similar regularization term as in $\tilde{r}$:

$$\tilde{\ell}(\theta_{\mathrm{L}}, \lambda|\theta_{\text{-L}}) = \widetilde{\mathrm{FNR}}(\theta_{\mathrm{L}}, \lambda|\theta_{\text{-L}}) + \eta \left|\widetilde{\mathrm{FPR}}(\theta_{\mathrm{L}}, \lambda|\theta_{\text{-L}}) - \beta\right| \tag{6}$$

Interestingly, we observe that regulating all layers leads to a drop in performance compared to regulating only $\theta_{\text{-L}}$. We provide a detailed comparison in Eq. (4) (see Table 1).

4 Experiments and Conclusions

We evaluate our approach on an image classification task. In particular, we use the **celebA** dataset.We consider the task of minimizing false negative rate (FNR) at a given false positive rate (FPR) of β. Please refer to the appendix for more details.

Experimental Results We present our experimental results in Table 1. Overall our proposed method 2 (which we refer to as "FPR Regularizer on θ_{L}") achieves comparable performance compared to ICO, and we observe a consistent improvement of both our proposed method 2 and the ablation method (which we refer to as "FPR Regularizer on θ") compared to the other two baseline methods (CE and TFCO). Since our method involves in training θ_{L} and $\theta_{\text{-L}}$ separately while fixing the other parameters and making the thresholds as a function of only θ_{L}, it is significantly cheaper to train compared to the baseline method ICO , which involves in training all parameters together and making the thresholds as a function of all the parameters. This observation demonstrates that our proposed methods can successfully preserve the performance while reducing the cost of training compared to previous method. Interestingly, among the two proposed methods, imposing the FPR regularizer only on θ_{L} (FPR Regularizer on θ_{L}) leads to a better performance compared to imposing FPR-regularizer on all weights θ (FPR Regularizer on θ). This supports our hypothesis that $\theta_{\text{-L}}$ and θ_{L} take up different roles during training, that the last layer θ_{L} plays a more significant role for the optimization step.

Conclusion and Future Work We propose new training techniques for optimizing a popular constrained optimization problem that involves non-decomposable rate metrics. Compared to the previous approach where the classifier thresholds are expressed as a function of all model parameters, we consider an alternative setting where the thresholds are expressed as a function of only the last layer of the neural networks. Our empirical results shows that among all proposed methods, adding a regularizer on the rest layer's weight achieve a comparable performance to the previous approach, supporting our hypothesis on the important role of last layer training in deep neural networks.

References

[1] Yatong Chen, Reilly Raab, Jialu Wang, and Yang Liu. Fairness transferability subject to bounded distribution shift. *arXiv preprint arXiv:2206.00129*, 2022.

[2] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332. PMLR, 2019.

[3] Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Ryan Rifkin, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial intelligence and statistics*, pages 832–840. PMLR, 2017.

[4] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[5] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations, 2022.

[6] Abhishek Kumar, Harikrishna Narasimhan, and Andrew Cotter. Implicit rate-constrained optimization of non-decomposable objectives. pages 5861–5871, 2021.

[7] Jingling Li, Mozhi Zhang, Keyulu Xu, John P Dickerson, and Jimmy Ba. Noisy labels can induce good representations. *arXiv preprint arXiv:2012.12896*, 2020.

[8] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization, 2022.

[10] Loring W Tu. Manifolds. In *An Introduction to Manifolds*, pages 47–83. Springer, 2011.

[11] Jimmy Wu, Yatong Chen, and Yang Liu. Metric-fair classifier derandomization. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022.

[12] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.