Optimizing Black-box Metrics with Adaptive Surrogates

Qijia Jiang [1 *] **Olaoluwa Adigun** [2 *] **Harikrishna Narasimhan** [3] **Mahdi Milani Fard** [3] **Maya Gupta** [3]

Abstract

We address the problem of training models with black-box and hard-to-optimize metrics by expressing the metric as a monotonic function of a small number of easy-to-optimize surrogates. We pose the training problem as an optimization over a relaxed surrogate space, which we solve by estimating local gradients for the metric and performing inexact convex projections. We analyze gradient estimates based on finite differences and local linear interpolations, and show convergence of our approach under smoothness assumptions with respect to the surrogates. Experimental results on classification and ranking problems verify the proposal performs on par with methods that know the mathematical formulation, and adds notable value when the form of the metric is unknown.

1. Introduction

We consider the problem of training a machine learning model when the true evaluation metric is difficult to optimize on the training set. This general problem arises with many flavors and in different scenarios. For example, we may have a black-box metric whose mathematical expression is unknown or difficult to approximate with a convex training loss. This is the case with non-decomposable evaluation metrics, such as the F-measure or ranking metrics like Precision@K, where it is not straight-forward to construct a smooth objective that closely approximates the metric.

Another example is when the training labels are only a proxy for the true label. This arises in problems where one has access to cheap-to-acquire noisy labels, such as clicks, but wishes to optimize for a more expensive label, such as whether users rate a result as good. If we have access to a small auxiliary validation set with true labels, how can this information be used to influence the training loss? Similar examples also arise when the training data has noisy features and we have a small validation set with clean features, or in machine learning fairness problems where the training data contains group-dependent noise, but we may have access to a small set of auxiliary clean data.

In many of the above scenarios, one wishes to optimize a black-box metric M over d model parameters, but does not have access to explicit gradients for M, nor is it practical to obtain reliable gradient estimates when d is large. We provide a general solution to this problem by choosing $K \ll d$ convex surrogate losses, and expressing M as an *unknown* monotonic function $\psi : \mathbb{R}_+^K \to \mathbb{R}$ of the K surrogates. We then reformulate the original problem as an optimization of ψ over the K-dimensional surrogate space. The choice of surrogates can be as simple as the hinge losses on positive and negative samples, which should work well for metrics like the F-measure, or the surrogates can be chosen to be a family of different convex losses to handle robustness to training noise given a small set of clean validation samples.

Our strategy is to estimate gradients for the unknown function ψ with respect to its K inputs by measuring changes in the metric M and the K surrogates for different perturbations on the model, and use the estimates for $\nabla\psi$ to perform *projected gradient descent* over the K-dimensional surrogate space. We show how the projection step can be implemented inexactly but with convergence guarantees by solving a convex problem in the original d parameters. We are thus able to *adaptively* combine the K surrogates to align well with the target metric M.

The main contributions of this paper include:

1. A novel formulation that poses the problem of optimizing a black-box metric as a lower-dimensional problem in a surrogate space.
2. A projected gradient descent based training algorithm using finite-differences and local linear interpolations to estimate gradients.
3. Theoretical results showing convergence to a stationary point under smoothness assumptions on ψ.
4. Experiments showing that the proposed approach works as well as methods that take advantage of the form of the metric if known, and can give substantial gains when the metric truly is a black-box.

[*]Equal contribution [1]Stanford University, USA [2]University of Southern California, USA [3]Google Research USA. Correspondence to: Harikrishna Narasimhan <hnarasimhan@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

2. Related Work

There has been much work on directly optimizing specialized classes of evaluation metrics during training. These include approaches that relax the metric using convex surrogates (Joachims et al., 2005; Kar et al., 2014; Narasimhan et al., 2015a; Kar et al., 2016), plug-in or post-shift methods that tune a threshold on estimates of class probabilities (Ye et al., 2012; Koyejo et al., 2014; Narasimhan et al., 2014; Yan et al., 2018), reduction approaches that formulate a sequence of cost-sensitive learning tasks (Parambath et al., 2014; Narasimhan et al., 2015b; Alabi et al., 2018; Narasimhan, 2018), and approaches that use constrained optimization and game-based formulations (Eban et al., 2017; Narasimhan et al., 2019).

However, all the above approaches require the evaluation metric to be available in closed-form. Of these, the closest to ours is the approach of Narasimhan et al. (2015b), which reformulates the learning problem as an optimization problem over the space of confusion matrices. To ensure the constraint set is convex, this approach requires the use of stochastic classifiers, and the theoretical guarantees assume that the metrics are convex or pseudo-convex in the confusion matrix. In contrast, we do not require stochastic classifiers, and can handle general metrics.

Recently, there has been some work on optimizing evaluation metrics that are only available as a black-box. Zhao et al. (2019b) approximate black-box metrics with a weighted training loss where the weighting function acts on a low-dimensional embedding of each example, and a validation set is used to estimate the parameters of the example-weighting function. A related approach by Ren et al. (2018) uses meta-gradient descent to re-weight the training examples to handle training set biases and label noise. In contrast we model the unknown metric as a function of surrogate losses, and directly estimate the metric gradients, rather than estimating a weighting function on each example.

Huang et al. (2019) also propose jointly adaptively learning a metric with the model training. They use a parametric form for their learned metric, whereas we nonparametrically estimate the metric gradients. They use reinforcement learning to align the training objective's optimum with that of the true metric, whereas we use gradient descent over a surrogate space. They do not provide any theoretical guarantees.

Grabocka et al. (2019) express the metric as a set function that maps each prediction to an embedding and maps the average embedding across all examples to the predicted metric. They jointly optimize the parameters for the loss and the model. This approach is similar to ours in that it expresses the metric as a function on surrogate losses, and attempts to learn that function. However, our approach is different in two key points. First, we take as *given* known-useful surrogate losses, whereas they *learn* decomposable surrogate mappings from scratch. Second, they parameterize their surrogate functions and final mapping as neural networks, whereas we nonparametrically adaptively estimate the local gradients. They provide limited theoretical guarantees.

Similar to Grabocka et al. (2019), the work of Wu et al. (2018) also learns a parameterized metric (e.g. as a neural network). An auxiliary parametric "teacher" model is used to adaptively learn the parameters for the metric that will maximize performance on a validation set. They do not provide theoretical guarantees.

3. Problem Setup and High-level Approach

Let $\mathcal{X}$ be some instance space and $\mathcal{Y}$ be the label space. Let $f_\theta : \mathcal{X} \to \mathbb{R}$ be a model parametrized by $\theta \in \mathbb{R}^d$ that outputs a score $f_\theta(\mathbf{x})$ for instance $\mathbf{x} \in \mathcal{X}$. One can use this score to make a prediction; e.g. for binary classification problems, one predicts $\text{sign}(f_\theta(\mathbf{x}))$. We measure performance w.r.t. a test distribution D over $\mathcal{X} \times \mathcal{Y}$. We consider two scenarios, one where we are provided a training sample S of n examples directly drawn from D, and the other where the training sample S is drawn from a noisy distribution, and we are provided a smaller clean validation set from D.

The performance of f_θ is evaluated by a metric $M : \mathbb{R}^d \to [0, 1]$ computed on D, where M may be as simple as the error rate $M_{err}(\theta) = \mathbf{E}_{(\mathbf{x},y)\sim D}\left[y f_\theta(\mathbf{x}) > 0\right]$ (or an estimate), or M may be a complex, non-decomposable metric such as Precision@K that depends on the scores and the distribution in a more intricate manner. We consider settings where the form of M is unknown, and the metric is available only as a black-box, i.e., for a given $\theta \in \mathbb{R}^d$, we can evaluate $M(\theta)$. The goal is to learn a good f_θ by solving:

$$\min_{\theta\in\mathbb{R}^d} M(\theta). \tag{1}$$

3.1. Reformulation with Surrogates

To optimize (1), one could directly estimate gradients of M with respect to the d parameters, but d is usually too large for that to be practical. To relax (1) to a more tractable problem, we take as given K *convex* surrogate loss functions $\ell_1, \ldots, \ell_K : \mathbb{R}^d \to \mathbb{R}_+$ where $K \ll d$, and express M as an *unknown* non-decreasing function of the K surrogates, with an *unknown* slack:

$$M(\theta) = \psi(\ell_1(\theta), \ldots, \ell_K(\theta)) + \epsilon(\theta),$$

where $\psi : \mathbb{R}_+^K \to [0, 1]$ is *monotonic* but possibly non-convex, and the slack $\epsilon : \mathbb{R}^d \to [-1, 1]$ determines how well the metric can be approximated by the K surrogates. Note that this decomposition of M is not unique. Our results hold for any such decomposition, but to enable a tighter analysis we consider a ψ for which the associated worst-case slack over all θ, i.e., $\max_{\theta\in\mathbb{R}^d} |\epsilon(\theta)|$ is the minimum.

Here are examples of target metrics and convex surrogates.

Example 1 (Classification Metrics). Consider the task of minimizing the G-mean metric given by $1 - \sqrt{\text{TPR} \times \text{TNR}}$, where TPR is the true positive rate and TNR is the true negative rate. This metric promotes high accuracies on both the positive and negative class and is popular for classification tasks where there is class imbalance (Daskalaki et al., 2006). Possible surrogates for this metric include the average logistic or hinge losses on the positive and negatives examples as these serve as proxies for the TPR and TNR. It is reasonable to assume monotonic ψ here, since lower surrogate values tend to produce better TPR and TNR values, and in turn lower G-means. The F-measure is another popular metric that can be written as a monotonic function of the TPR and TNR (Koyejo et al., 2014), and there again the average positive and negative losses would make good surrogates.

Example 2 (Misaligned Training Data). Consider minimizing a metric using a training dataset that is noisy or misaligned with the test distribution, but we have access to a small validation set with clean data. The metric M here is evaluated on the clean validation set, and the surrogates $\ell_1, \ldots, \ell_K$ might be convex l_p losses on the training data with different values of $p > 1$ to tune the noise robustness. In this case, the precise mathematical relationship ψ between the validation metric and the surrogates is unknown.

Example 3 (ML Fairness Problems). For blackbox ML fairness metrics, good surrogates might be logistic losses on the positive and negative samples for different groups.

Example 4 (Ranking Metrics). Consider optimizing a ranking metric such as precision@K. While there are different convex surrogates available for this metric (Joachims, 2005; Narasimhan et al., 2015a), the surrogate that performs the best can vary with the application. We have also observed in practice that sometimes setting a different value of K in the training loss produces a better precision@K during evaluation time. The proposed set-up gives us a way to combine multiple available ranking surrogates (possibly with different K values) to align well with the test metric.

3.2. High-level Approach

Let $\mathcal{L} := \{(\ell_1(\theta), \ldots, \ell_K(\theta)) \,|\, \theta \in \mathbb{R}^d\}$ be the set of feasible surrogate profiles. We then seek to approximate (1) by ignoring the slack ϵ and posing the problem as an optimization of ψ over the K-dimensional set $\mathcal{L}$:

$$\min_{\boldsymbol{\ell} \in \mathcal{L}} \psi(\boldsymbol{\ell}). \tag{2}$$

Our high-level idea is to solve this re-formulated problem by applying *projected gradient descent* over $\mathcal{L}$.

However, there are many challenges in implementing this idea. First, while each ℓ_k is convex, the space of feasible surrogates $\mathcal{L}$ is not necessarily a convex set. Second, the function ψ is unknown to us, and therefore we need to estimate gradients for ψ with only access to the metric M and the surrogates $\boldsymbol{\ell}$. Third, we would need to implement projections onto the K-dimensional surrogate space without explicitly constructing this set.

4. Surrogate Projected Gradient Descent

We now explain how we address the above challenges.

4.1. Convexifying the Surrogate Space

To turn (2) into a problem over a convex domain, we define the *epigraph* of the convex surrogate function profiles:

$$\mathcal{U} := \{\mathbf{u} \in \mathbb{R}_+^K \mid \mathbf{u} \geq \boldsymbol{\ell}(\theta) \text{ for some } \theta \in \mathbb{R}^d\}.$$

Observation 1. *$\mathcal{U}$ is a convex superset of $\mathcal{L}$.*

We then optimize ψ over this K-dimensional convex set:

$$\min_{\mathbf{u} \in \mathcal{U}} \psi(\mathbf{u}). \tag{3}$$

This relaxation preserves the optimizer for (2) because ψ is monotonic and $\mathcal{U}$ consists of upper bounds on surrogate profiles in $\mathcal{L}$:

Observation 2. *For any $\mathbf{u}^* \in \operatorname{argmin}_{\mathbf{u} \in \mathcal{U}} \psi(\mathbf{u})$, there exists $\boldsymbol{\ell}^* \in \mathcal{L}$, $\boldsymbol{\ell}^* \leq \mathbf{u}^*$, such that $\psi(\boldsymbol{\ell}^*) = \psi(\mathbf{u}^*)$.*

4.2. Projected Gradient Descent over $\mathcal{U}$

We then perform projected gradient descent over $\mathcal{U}$. We maintain iterates $\mathbf{u}^t$ in $\mathcal{U}$, and at each step, (i) estimate the gradient of ψ w.r.t. the K-dimensional point $\mathbf{u}^t$, (ii) perform a descent step: $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^t - \eta \nabla \psi(\mathbf{u}^t)$, for some $\eta > 0$, and (iii) project $\tilde{\mathbf{u}}^{t+1}$ onto $\mathcal{U}$ to get the next iterate $\mathbf{u}^{t+1}$.

In order to implement these steps without knowing ψ, or having direct access to the set $\mathcal{U}$, we simultaneously maintain iterates θ^t in the original parameter space that map to iterates $\mathbf{u}^t \in \mathcal{U}$, i.e., for which $\mathbf{u}^t = \boldsymbol{\ell}(\theta^t)$.

Now to estimate gradients without direct access to ψ, we measure changes in the K surrogates $\boldsymbol{\ell}(\cdot)$ and changes in the metric $M(\cdot)$ at different perturbations of θ^t and compute estimates of $\nabla \psi(\mathbf{u}^t))$ based on finite-differences or local linear interpolations. To compute projections without direct access to $\mathcal{U}$, we formulate a convex optimization problem over the original parameters θ, and show that this results in an over-constrained projection onto $\mathcal{U}$.

Thus we maintain iterates $(\mathbf{u}^t, \theta^t)$ such that $\mathbf{u}^t = \boldsymbol{\ell}(\theta^t)$, and execute the following at every iteration:

$$\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^t - \eta \, \text{gradient}_\psi(\theta^t; M, \boldsymbol{\ell})$$
$$(\mathbf{u}^{t+1}, \theta^{t+1}) = \text{project}_{\mathcal{U}}(\tilde{\mathbf{u}}^{t+1}; \boldsymbol{\ell}).$$

Figure 1 gives a schematic description of the updates. The gradient computation takes the current θ^t as input and

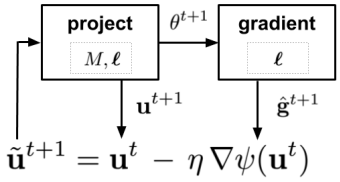


Figure 1. **PGD over K-dimensional set $\mathcal{U}$.** 'project' performs an over-constrained projection onto $\mathcal{U}$. 'gradient' probes M and ℓ returns an estimate $\hat{\mathbf{g}}^{t+1} \in \mathbb{R}^K$ for $\nabla\psi$.

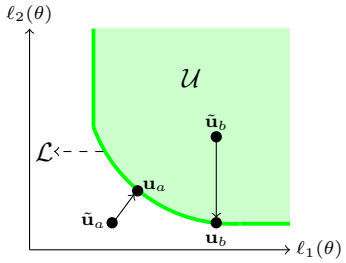


Figure 2. **Over-constrained projection.** The space of surrogate profiles $\mathcal{L} = \{(\ell_1(\theta), \ell_2(\theta)) \,|\, \theta \in \mathbb{R}^d\}$ is a non-convex set (solid line), and its epigraph $\mathcal{U} = \{\mathbf{u} \geq \boldsymbol{\ell} \,|\, \boldsymbol{\ell} \in \mathcal{L}\}$ is convex (shaded region). For the point $\tilde{\mathbf{u}}_a$ outside $\mathcal{U}$, the solution $\mathbf{u}_a$ to (4) is the same as the exact projection $\Pi(\tilde{\mathbf{u}}_a)$ onto $\mathcal{U}$. For the point $\tilde{\mathbf{u}}_b$ inside the set, the projection $\Pi(\tilde{\mathbf{u}}_b)$ is the same point $\tilde{\mathbf{u}}_b$, whereas the the solution to (4) is any point $\mathbf{u}_b$ on the boundary that is lesser or equal to the exact projection in each coordinate, i.e., $\mathbf{u}_b \leq \Pi(\tilde{\mathbf{u}}_b)$.

probes M and $\boldsymbol{\ell}$ to return an estimate of $\nabla\psi(\mathbf{u}^t)$. We elaborate on how we estimate gradients in Section 5. The projection computation takes the updated $\tilde{\mathbf{u}}^{t+1}$ as input and returns a point $\mathbf{u}^{t+1}$ in $\mathcal{U}$ and an associated θ^{t+1} such that $\mathbf{u}^{t+1} = \boldsymbol{\ell}(\theta^{t+1})$. We explain this next.

4.3. Over-constrained Projection

To implement the projection without explicit access to $\mathcal{U}$, we set up an optimization over θ by penalizing a clipped L_2-distance between the surrogate profile $\boldsymbol{\ell}(\theta)$ and $\tilde{\mathbf{u}}^{t+1}$:

$$\begin{aligned} \theta^{t+1} &\in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \| \left(\boldsymbol{\ell}(\theta) - \tilde{\mathbf{u}}^{t+1}\right)_+ \|^2 \\ \mathbf{u}^{t+1} &= \boldsymbol{\ell}(\theta^{t+1}), \end{aligned} \tag{4}$$

where $(z)_+ := \max\{0, z\}$ is applied element-wise and $\|\cdot\|$ is the L_2-norm. Note that we penalize errors in only one direction (i.e., the errors where $\ell_k(\theta) \geq \tilde{u}_k^{t+1}$). This has the advantage of the optimization problem being convex. Moreover, as we show below, (4) results in an over-constrained projection: any solution $\mathbf{u}^{t+1}$ to (4) is feasible (i.e., is in $\mathcal{U}$), and for a monotonic ψ, yields a ψ-value that is no worse than what we would get with an exact projection.

Lemma 1. *Let $\mathbf{u}^+$ be the exact projection of $\tilde{\mathbf{u}}^{t+1} \in \mathbb{R}_+^K$ onto $\mathcal{U}$. For any solution $\mathbf{u}^{t+1}$ to (4), we have $\mathbf{u}^{t+1} \in \mathcal{U}$, $\mathbf{u}^{t+1} \leq \mathbf{u}^+$, and for a monotonic ψ, $\psi(\mathbf{u}^{t+1}) \leq \psi(\mathbf{u}^+)$.*

Algorithm 1 Surrogate Projected Gradient Descent

1: **Input:** Black-box metric M, surrogate loss functions $\ell_1, \ldots, \ell_K : \mathbb{R}^d \to \mathbb{R}_+^K$, hyper-parameters: T, η
2: Initialize $\theta^1 \in \mathbb{R}^d$, $\mathbf{u}^1 = \boldsymbol{\ell}(\theta^1)$
3: **for** $t = 1$ **to** T **do**
4: **Gradient estimate:** Obtain an estimate $\hat{\mathbf{g}}^t$ for gradient $\nabla\psi(\mathbf{u}^t)$ by invoking Algorithms 2 or 3 with inputs θ^t, M and $\ell_1, \ldots, \ell_K$
5: **Gradient update:** $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^t - \eta\, \hat{\mathbf{g}}^t$
6: **Over-constrained projection:** Solve:

$$\theta^{t+1} \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \| \left(\boldsymbol{\ell}(\theta) - \tilde{\mathbf{u}}^{t+1}\right)_+ \|^2$$

to accuracy $\mathcal{O}\left(\frac{1}{\beta^2 T}\right)$ and set $\mathbf{u}^{t+1} = \boldsymbol{\ell}(\theta^{t+1})$
7: **end for**

Problem (4) may not have a unique solution. For example, when $\tilde{\mathbf{u}}^{t+1}$ is in the interior of $\mathcal{U}$, the exact projection $\mathbf{u}^+$ is the same as $\tilde{\mathbf{u}}^{t+1}$, whereas the solutions to (4) are the points $\mathbf{u}$ on the boundary of $\mathcal{U}$ with $\mathbf{u} \leq \mathbf{u}^+$ (see Figure 2). As ψ is monotonic, picking any of these solutions for the next iterate is sufficient for the algorithm's convergence.

An outline of the projected gradient descent with this inexact projection is presented in Algorithm 1. One can interpret the algorithm as *adaptively* combining the K surrogates ℓ_k's to optimize the metric M (see Appendix C for the details).

4.4. Convergence Guarantee

We show convergence of Algorithm 1 to a stationary point of $\psi(\boldsymbol{\ell}(\cdot))$. Since we probe M to estimate gradients for ψ, the errors in the estimate would depend on how closely $\psi(\boldsymbol{\ell}(\cdot))$ approximates M, and in turn on the magnitude of the slack term ϵ. We assume here that the gradient estimation error $\mathbf{E}\left[\|\hat{\mathbf{g}}^t - \nabla\psi(\boldsymbol{\ell}(\theta^t))\|^2\right]$ at each step t is bounded by κ_ϵ that depends on the slack ϵ. In Section 5, we present gradient estimates that satisfy this condition.

Theorem 2 (Convergence of Algorithm 1)**.** *Let $M(\theta) = \psi(\boldsymbol{\ell}(\theta)) + \epsilon(\theta)$, for a ψ that is monotonic, β-smooth and L-Lipschitz, and the worst-case slack $\max_{\theta \in \mathbb{R}^d} |\epsilon(\theta)|$ is the minimum among all such decompositions of M.*

Suppose each ℓ_k is γ-smooth and Φ-Lipschitz in θ with $\|\boldsymbol{\ell}(\theta)\| \leq G$, $\forall\theta$. Suppose the gradient estimates $\hat{\mathbf{g}}^t$ satisfy $\mathbf{E}\left[\|\hat{\mathbf{g}}^t - \nabla\psi(\boldsymbol{\ell}(\theta^t))\|^2\right] \leq \kappa_\epsilon$, $\forall t \in [T]$ and the projection step satisfies $\|(\boldsymbol{\ell}(\theta^{t+1}) - \tilde{\mathbf{u}}^t)_+\|^2 \leq \min_{\theta \in \mathbb{R}^d} \|(\boldsymbol{\ell}(\theta) - \tilde{\mathbf{u}}^t)_+\|^2 + \mathcal{O}(\frac{1}{\beta^2 T})$, $\forall t \in [T]$. Set stepsize $\eta = \frac{1}{\beta^2}$.

Then Algorithm 1 converges to an approximate stationary point of $\psi(\boldsymbol{\ell}(\cdot))$:

$$\min_{1 \leq t \leq T} \mathbf{E}\left[\|\nabla\psi(\boldsymbol{\ell}(\theta^t))\|^2\right] \leq C\left(\frac{\beta}{\sqrt{T}} + \sqrt{\kappa_\epsilon} + \sqrt{L}\kappa_\epsilon^{1/4}\right),$$

where the expectation is over the randomness in the gradient estimates, and $C = \mathcal{O}\left(KL\left(\gamma\left(G + \frac{L}{\beta^2}\right) + \Phi^2\right)\right)$.

Remark 1 (**Stationary point of M**). When the gradient estimation error κ_ϵ is small and the number of steps $T \to \infty$, the algorithm reaches a model θ with a small gradient norm $\|\nabla\psi(\boldsymbol{\ell}(\cdot))\|$. If additionally the slack term ϵ is Lipschitz in θ, then this implies that the algorithm also converges to an approximate stationary point of the metric M.

The proof of Theorem 2 proceeds in two parts. We first show that the algorithm converges to an approximate stationary point of ψ over $\mathcal{U}$. For this, we extend recent results of Ghadimi et al. (2016) on convergence of projected gradient descent for smooth non-convex objectives. We then exploit the smoothness of the surrogates $\boldsymbol{\ell}$ to show that this result translates to the algorithm converging to an approximate stationary point of $\psi(\boldsymbol{\ell}(\cdot))$ w.r.t. θ.

Remark 2 (**Prior convergence results**). A key difference between our analysis and prior works on zeroth-order gradient methods (Duchi et al., 2015; Ghadimi et al., 2016; Nesterov and Spokoiny, 2017) is that we do not directly optimize the given objective over the space of parameters θ, and instead perform an optimization over a relaxed surrogate space $\mathcal{U}$ that is not directly specified, and do so using inexact projections and approximate gradient estimates.

5. Gradient Estimation Techniques

We now address the issue of estimating the gradient $\hat{\mathbf{g}}^t$ of ψ at a given $\boldsymbol{\ell}(\theta^t)$ without explicit access to ψ. We provide an algorithm based on finite-differences, and another based on local linear interpolations. We also show error bounds for these algorithms, i.e., bound the errors κ_ϵ in Theorem 2.

5.1. Finite Differences

We first consider the case where both the surrogates $\boldsymbol{\ell}$ and metric M are evaluated on the same sample. Let $\mathbf{f}_\theta := [f_\theta(\mathbf{x}_1), \ldots, f_\theta(\mathbf{x}_n)]^\top \in \mathbb{R}^n$ denote the scores of the model θ computed on the n training examples. We overload notation and use $M(\mathbf{f}_\theta, \mathbf{y})$ to denote the value of the evaluation metric M on the model scores $\mathbf{f}_\theta \in \mathbb{R}^n$ and labels $\mathbf{y} \in \mathcal{Y}^n$. Similarly, we use $\ell_k(\mathbf{f}_\theta, \mathbf{y})$ to denote the value of surrogate loss ℓ_k on $\mathbf{f}_\theta$ and $\mathbf{y}$.

We present our method in Algorithm 2. We adopt a standard finite-difference gradient estimate (Nesterov and Spokoiny, 2017), which requires us to perturb the surrogates $\boldsymbol{\ell}$ with random Gaussian vectors $Z^1, \ldots, Z^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$, evaluate ψ at the perturbed surrogate profiles, and calculate

$$\frac{1}{m}\sum_{j=1}^{m} \frac{\psi(\boldsymbol{\ell}(\mathbf{f}_\theta, \mathbf{y}) + \sigma Z^j) - \psi(\boldsymbol{\ell}(\mathbf{f}_\theta, \mathbf{y}))}{\sigma},$$

for $\sigma > 0$. In our case, we cannot directly perturb the surrogates $\boldsymbol{\ell}$ and evaluate changes in ψ. Instead, we perturb the scores $\mathbf{f}_\theta$ so that the corresponding changes in $\boldsymbol{\ell}$ follows a Gaussian distribution, and evaluate the difference between the metric M at the original and perturbed scores. This is possible, for example, when each $\ell_k(\mathbf{f}_\theta, \mathbf{y})$ is an average of point-wise losses $\phi_k(y_i f_\theta(x_i))$ on different subsets of the data, for some invertible function $\phi_k : \mathbb{R} \to \mathbb{R}$, in which case, it is easy to compute the right amount of perturbation to the scores $\mathbf{f}_\theta$ to produce the desired perturbation in ℓ_k.

Algorithm 2 Finite-difference Gradient Estimate

Input: $\theta' \in \mathbb{R}^d$, M, $\ell_1, \ldots, \ell_K$
Hyper-parameters: Num of perturbations m, σ
Draw $Z^1, \ldots, Z^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$
Find $\Delta^j \in \mathbb{R}^n$ s.t. $\boldsymbol{\ell}(\mathbf{f}_{\theta'} + \Delta^j, \mathbf{y}) = \boldsymbol{\ell}(\mathbf{f}_{\theta'}, \mathbf{y}) + \sigma Z^j$, for $j = 1, \ldots, m$
$\hat{\mathbf{g}} = \frac{1}{m}\sum_{j=1}^{m} \frac{M(\mathbf{f}_{\theta'} + \Delta^j, \mathbf{y}) - M(\mathbf{f}_{\theta'}, \mathbf{y})}{\sigma} Z^j$
Output: $\hat{\mathbf{g}}$

Algorithm 3 Linear Interpolation Gradient Estimate

Input: $\theta' \in \mathbb{R}^d$, M, $\ell_1, \ldots, \ell_K$
Hyper-parameters: Num of perturbations m, σ
Draw $Z_1^1, \ldots, Z_1^m, Z_2^1, \ldots, Z_2^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
$\mathbf{H}_{j,:} = \boldsymbol{\ell}(\theta' + \sigma Z_1^j) - \boldsymbol{\ell}(\theta' + \sigma Z_2^j),\ j = 1, \ldots, m$
$\mathbf{M}_{j,:} = M(\theta' + \sigma Z_1^j) - M(\theta' + \sigma Z_2^j),\ j = 1, \ldots, m$
$\hat{\mathbf{g}} \in \operatorname{argmin}_{\hat{\mathbf{g}} \in \mathbb{R}^K} \|\mathbf{H}\hat{\mathbf{g}} - \mathbf{M}\|^2$
Output: $\hat{\mathbf{g}}$

Lemma 3 (Finite difference estimate). *Let M be as defined in Theorem 2 and $|\epsilon(\theta)| \le \bar{\epsilon}, \forall\theta$. Let $\hat{\mathbf{g}}$ be returned by Algorithm 2 for a given θ', m perturbations and $\sigma = \frac{\sqrt{\bar{\epsilon}}}{\sqrt{K}\beta^2}$.*

$$\mathbf{E}\left[\|\hat{\mathbf{g}} - \nabla\psi(\boldsymbol{\ell}(\theta'))\|^2\right] \le \mathcal{O}\left(\frac{L^2 K}{m} + \bar{\epsilon} K^2 \beta^2\right),$$

where the expectation is over the random perturbations.

This gives a bound on κ_ϵ in Theorem 2 when Algorithm 2 is used for gradient estimates. Note the error depends on the slack magnitude $\bar{\epsilon}$, and decreases with more perturbations.

5.2. Local Linear Interpolations

The finite difference approach is not applicable to settings where the metric is evaluated on a validation sample but the surrogates are evaluated on training examples (as in Example 2), or where finding the right amount of perturbation on the scores is difficult. For such cases we present a local linear interpolation based approach in Algorithm 3, where we perturb the model parameters θ instead of the scores.

We use the fact that a smooth function ψ can be locally approximated by a linear function, and estimate the gradient of ψ of at $\boldsymbol{\ell}(\theta)$ by perturbing θ, measuring the corresponding differences in the surrogates $\boldsymbol{\ell}$ and the metric M, and fitting a linear function from the surrogate differences to

the metric differences. Specifically, for d model parameters, we draw two independent sets of d-dimensional Gaussian perturbations $Z_1^1, \ldots, Z_1^m, Z_2^1, \ldots, Z_2^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, and return a linear fit from $\mathbf{H} = [\boldsymbol{\ell}(\theta + \sigma Z_1^j) - \boldsymbol{\ell}(\theta + \sigma Z_2^j)]_{j=1}^m$ to $\mathbf{M} = [M(\theta + \sigma Z_1^j) - M(\theta + \sigma Z_2^j)]_{j=1}^m$.

Lemma 4 (Linear interpolation estimate)**.** *Let M be defined as in Theorem 2 and $|\epsilon(\theta)| \leq \bar{\epsilon}, \forall \theta$. Assume each ℓ_k is Φ-Lipschitz in θ w.r.t. the L_∞-norm, and $\|\boldsymbol{\ell}(\theta)\| \leq G \ \forall \theta$. Suppose for a given θ', σ and perturbation count m, the expected covariance matrix for the left-hand-side of the linear system $\mathbf{H}$ is well-conditioned, and has the smallest singlular value $\lambda_{\min}(\sum_{i=1}^m \mathbf{E}[\mathbf{H}_i \mathbf{H}_i^\top]) = \mathcal{O}(m\sigma^2\Phi^2)$. Then setting $\sigma = \tilde{\mathcal{O}}\left(\frac{G^{1/3}\bar{\epsilon}^{1/3}}{\Phi K^{3/2}\beta^{1/3}}\right)$ and $m = \tilde{\mathcal{O}}\left(\frac{G^4 K^9 \beta^2}{\bar{\epsilon}^2}\right)$, Algorithm 3 returns w.h.p. (over draws of random perturbations) a gradient estimate $\hat{\mathbf{g}}$ that satisfies:*

$$\|\hat{\mathbf{g}} - \nabla\psi(\boldsymbol{\ell}(\theta'))\|^2 \leq \tilde{\mathcal{O}}\left(G^{1/3}\bar{\epsilon}^{1/3}K^3\beta^{2/3}\right).$$

We show in Appendix A.5 how this high probability statement can then be used to derive a bound on the expected errors κ_ϵ in Theorem 2. Prior works provide error bounds on a similar gradient estimate under an assumption that the perturbation matrix $\mathbf{H}$ can be chosen to be invertible (Conn et al., 2008; 2009; Berahas et al., 2019). In our case, however, $\mathbf{H}$ is not chosen explicitly, but instead contains measurements of changes in surrogates for random perturbations on θ. Hence to show an error bound, we need a slightly subtle condition on the correlation structure of the surrogates (that essentially says the variance of the perturbed surrogates are large enough and the rates are not strongly correlated with each other), which we express as a condition on the smallest singular value of the covariance of $\mathbf{H}$.

5.3. Handling Non-smooth Metrics

For ψ that is non-smooth and Lipschitz, we extend the finite difference gradient estimate in Section 5.1 with a two-step perturbation. We draw two sets of Gaussian random vectors $Z_1^1, \ldots, Z_1^m, Z_2^1, \ldots, Z_2^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ and calculate

$$\frac{1}{m}\sum_{j=1}^m \frac{\psi(\boldsymbol{\ell}(\mathbf{f}_\theta, \mathbf{y}) + \sigma_1 Z_1^j + \sigma_2 Z_2^j) - \psi(\boldsymbol{\ell}(\mathbf{f}_\theta, \mathbf{y}) + \sigma_1 Z_1^j)}{\sigma_2}$$

for $\sigma_1, \sigma_2 > 0$, where we perturb $\boldsymbol{\ell}$ by perturbing the scores $\mathbf{f}_\theta$. This approach computes a finite-difference gradient estimate for a smooth approximation to the original ψ, given by $\psi_{\sigma_1}(\mathbf{u}) := \mathbf{E}[\psi(\mathbf{u} + \sigma_1 Z_1)]$, where $Z_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$. We provide error bounds in Appendix B by building on recent work by Duchi et al. (2015), and discuss asymptotic convergence of Algorithm 1 as $\sigma_1 \to 0$.

6. Experiments

We present experiments to show the proposed approach, Algorithm 1, is able to perform as well as methods that take advantage of a metric's form where available, and is also able to provide gains for metrics that are truly a black-box. We consider a simulated classification task, a fair classification task with noisy features, a ranking task and classification tasks with proxy and noisy labels. The datasets we use are listed in Table 1.

Table 1. Datasets used in our experiments.

Dataset	#instances	#features	Groups
Simulated	5000	2	-
COMPAS	4073	31	M/F
Adult	32561	122	M/F
Credit	30000	89	M/F
Business	11560	36	C/NC
KDD Cup 08	102294	117	-

Table 2. Test G-mean on sim. data. *Lower* is better. Due to the class imbalance, logistic regression learns to always predict the majority negative class and yields a poor G-mean. Post-shift tunes a threshold on the logistic regression model to optimize G-mean.

	LogReg	PostShift	Proposed
Simulated	1.000	0.498	**0.344**

We use the linear interpolation approach in Algorithm 3 for estimating gradients, as this is the most practical among the proposed estimation methods, and applicable when the surrogates and metrics are evaluated on different samples. We train linear models in all experiments, and tune hyper-parameters such as step sizes and the perturbation parameter σ for gradient estimation using a held-out validation set. We run the projected gradient descent with 250 outer iterations and 1000 perturbations. For the projection step, we run 100 iterations of Adagrad. See Appendix D for more details and a discussion on perturbations. TensorFlow code has been made available.[1]

6.1. Optimizing G-mean on Simulated Data

As a sanity check, we first apply our approach to maximize a non-black box evaluation metric: G-mean $= 1 - \sqrt{\text{TPR} \times \text{TNR}}$, described in Example 1. We consider a simulated 2-dimensional binary classification task, containing 10% positives and 90% negatives. The positive examples are drawn from a Gaussian with mean $[0, 0]$ and covariance matrix $0.2 \times \mathbf{I}_2$. The negative examples are drawn from a mixture of two Gaussians centered at $[-1, -1]$ and $[1, 1]$, with equal priors, and with a covariance matrix of $0.1 \times \mathbf{I}_2$.

We apply our method with two surrogate functions: the average hinge losses on the positive and negative examples. We compare with the plug-in or *post-shift* approach, a common baseline for the G-mean metric (Narasimhan et al., 2014). This method trains a logistic regression model and tunes

[1] https://github.com/google-research/google-research/tree/master/adaptive_surrogates

Table 3. Average test macro F-measure across groups with clean features. *Higher* is better. Despite having only black-box access to the metric, our approach performs comparable to methods that take advantage of the form of the metric.

	LogReg	PostShift	RelaxedFM	GenRates	Proposed
Business	0.793	0.789	0.794	0.793	**0.796**
COMPAS	0.560	**0.631**	0.614	0.620	0.629
Adult	**0.668**	0.664	0.665	0.654	0.665
Default	0.467	**0.536**	0.525	0.532	0.533

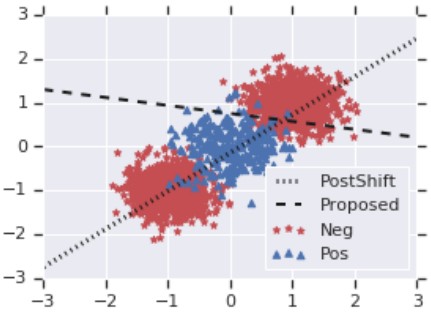


Figure 3. Hyperplanes learned by proposed method and PostShift on simulated data.

a threshold on the model to optimize G-mean. As shown in Table 2, the proposed approach yields the best G-mean. It is clear from the resulting decision boundaries shown in Figure 3 that our method learns the better linear separator.

6.2. Macro F-measure with Noisy Features

For this experiment we consider training a classifier with fairness goals defined on binary protected attributes. We seek to maximize the average F-measure across the groups:

$$\text{Macro } F_1 = \frac{1}{2} \sum_{G \in \{0,1\}} \frac{2 \times \text{Precision}_G \times \text{Recall}_G}{\text{Precision}_G + \text{Recall}_G},$$

where Precision_G and Recall_G are the precision and recall on protected group G. Optimizing a sum of F-measures is harder than optimizing the binary F-measure because the summation destroys its pseudo-convexity property (Narasimhan et al., 2019).

We use four fairness datasets: (1) *COMPAS*, where the goal is to predict recidivism with *gender* as the protected attribute (Angwin et al., 2016); (2) *Adult*, where the goal is to predict if a person's income is more than 50K/year, and we take *gender* as the protected group (Blake and Merz, 1998); (3) *Credit Default*, where the task is to predict whether a customer would default on his/her credit card payment, and we take *gender* as the protected group (Blake and Merz, 1998); (4) *Business Entity Resolution*, a proprietary dataset from a large internet services company, where the goal is to predict whether a pair of business descriptions refer to identical businesses, and we consider *non-chain* businesses as protected. In each case, we split the data into train-validation-test sets in the ratio $4/9 : 2/9 : 1/3$.

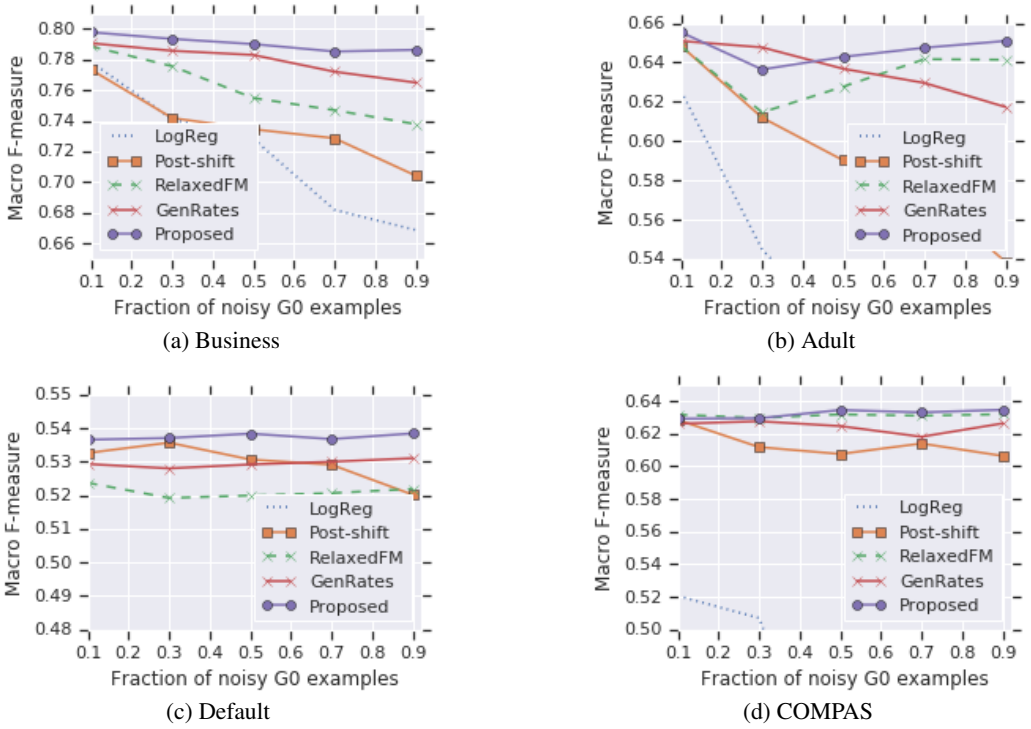
Training with no noise. The first set of experiments tests if the proposed approach is able to match the performance of existing methods that are customized to optimize the macro F-measure. We compare against (i) plain logistic regression method, (ii) a plug-in or post-shift method that tunes a threshold on the logistic regression model to maximize the F-measure (Koyejo et al., 2014; Narasimhan et al., 2014), (iii) an approach that optimizes a continuous relaxation to the F-measure that replaces the indicators with the hinge loss (see e.g. Zhao et al. (2019a)), and (iv) the recent "generalized rates" approach of Narasimhan et al. (2019) for optimizing metrics that are a sum of ratios. We apply our approach using four surrogate losses, each one is the hinge loss averaged over either the positive or negative examples, calculated separately for each of the two groups. As seen in Table 3, despite having only black-box access to the metric, the proposed approach is competitive with the other methods that are directly tailored to optimize the macro F-measure.

Training with noisy features. The second set of experiments evaluates the performance of these methods when the training set has noisy features for just one of the groups, while the smaller validation set contains clean features. We use our approach to adaptively combine the same four surrogate losses computed on the noisy training set to best optimize the macro F-measure on the clean validation set.

We chose a certain fraction of the examples at random from one of the groups, which we refer to as group 0, and for these examples, we add Gaussian noise to the real features (with mean 0 and the same standard deviation as the feature), and flip the binary features with probability 0.9. Figure 4 shows the test F-measure for the different methods with varying fraction of noisy examples in group 0. Except for logistic regression, all other methods have access to the validation set: post-shift uses the validation set to tune a threshold on the logistic regression model; the RelaxedFM and GenRates method optimize their loss on the training set, but pick the best model iterate using the validation set. The proposed approach is able to make the best use of the validation set, and consistently performs the best across most noise levels.

6.3. Ranking to Optimize PRBEP

We next consider a ranking task, where the goal is to learn a scoring function f that maximizes the precision-recall break-even point (PRBEP), i.e. yields maximum precision at the



(a) Business (b) Adult

(c) Default (d) COMPAS

Figure 4. Test macro F-measure across groups for varying noise levels, averaged over 5 trials. *Higher* is better.

Table 4. PRBEP on KDD Cup 2008 data. *Higher* is better.

	Kar et al. (2015)	Proposed
Train	0.473	**0.546**
Test	0.441	**0.480**

threshold where precision and recall are equal. PRBEP is a special case of Precision@K when K is set to the number of positive examples in the dataset. For this task, we experiment with the KDD Cup 2008 breast cancer detection data set (Rao et al., 2008) popularly used in this literature (Kar et al., 2015; Mackey et al., 2018). We randomly split this dataset 60/20/20 for training, validation, and test.

Since the break-even point for a dataset is not known beforehand, we use surrogates that approximate precision at different recall thresholds τ. We use the quantile-based surrogate losses of Mackey et al. (2018) with $\tau = 0.25, 0.5, 0.75$. As a comparison, we optimize the avg-precision@K surrogate provided by Kar et al. (2015). As seen in Table 4, the proposed approach is able to learn a better training loss by combining the three quantile surrogates, and yields the best PRBEP on the both the training and test sets.

6.4. Classification with Proxy Labels

Next, we consider classification tasks where the training labels are proxies for the true labels, but the validation data has the true labels. We seek to minimize the classification error on the validation set by combining hinge loss surrogates evaluated separately on the positive and negative training examples. While the theory requires convex losses, we experiment with also running the algorithm with non-convex sigmoid losses as surrogates.

Table 5. Test classification error where the training labels are only proxy labels with unknown relationship to the true labels. The proposed method was run with both hinge and sigmoid surrogates. *Lower* is better.

	LogReg	PostShift	Hinge	Sigmoid
Adult	0.333	0.322	**0.314**	**0.314**
Business	0.340	0.251	0.256	**0.236**

For the Adult data, we predict whether a candidate's gender is female, and take the marital-status-wife feature as the proxy label. For the Business Entity Resolution data, we predict whether a pair of business descriptions refer to the same business, and use the has-same-phone-number feature as a proxy label.

We compare with a logistic regression model trained with the proxy labels and a post-shift method that corrects the logistic regression threshold to minimize classification error on the validation data. In the results shown in 5, as expected logistic regression yields the highest test error. On Adult, both variants of the proposed method are better than PostShift. On Business, the proposed method performs slightly worse than PostShift when run with hinge surrogates, but yields notable improvements when run with sigmoid surrogates, which are tighter relaxations to the true errors.

Table 6. Validation and test G-mean on Adult dataset with noisy training labels, averaged over 5 trials. The proposed method was run with both hinge and sigmoid surrogates. *Lower* is better.

	LogReg	PostShift	MOEW	Hinge	Sigmoid
Vali	0.482	0.178	0.199	0.187	**0.165**
Test	0.437	0.184	0.223	0.186	**0.176**

6.5. Optimizing G-mean with Noisy Labels

We finally consider classification tasks where the training labels are noisy, but we have access to a validation sample with true labels. We experiment with the Adult dataset, with the goal of predicting if a person's income is more than 50K/year. We hold out 1% of the training data for validation, and in remaining data, we randomly pick 30% of the positive labels and flip them to negative. We seek to minimize the G-mean metric given by $1 - \sqrt{\text{TPR} \times \text{TNR}}$ on the validation set. We apply our approach using *ten* surrogate losses, each one is the hinge loss averaged over either the positive or negative examples, calculated separately for five groups of examples: male, female, black, white and private-workforce. We also applied our approach using sigmoid surrogates averaged over the same ten subsets.

We compare with a logistic regression model trained with the noisy labels and a post-shift method that corrects the logistic regression threshold to minimize G-mean on the validation data. We also compare with a grid-search-based variant of the metric-optimized example weights (MOEW) approach of Zhao et al. (2019b). MOEW optimizes a black-box metric by learning a weighted training objective, where the weights on the individual examples are trained to minimize a given metric on the validation set. We model the example weights as a linear function of a 2-dimensional feature embedding and the labels, and tune the parameters of the weighting function using an exhaustive grid search.

The results are shown in Table 6. The proposed method with sigmoid surrogates yields the lowest G-mean on the validation and test sets, with post-shift coming in second. The sigmoid surrogates, being tighter approximation to the training errors, once again yield better results than the hinge surrogates. In Appendix D, we provide further details and an additional experiment with simulated data.

7. Discussion

There is currently a lot of interest in training models with better alignment with evaluation metrics. Here, we have investigated a simple method that directly estimates only the needed gradients for gradient descent training, and does not require assuming a parametric form. This simplicity enabled us to provide rigorous theoretical guarantees.

Experimentally, our approach was as good as strategies that take advantage of a metric's known mathematical form. In our experiments imitating real use cases for our method, we obtained notable gains over existing baselines for classification with group-dependent noise, and for black-box ranking. For the experiments with proxy and noisy labels, however, we found the post-shifting approach to be competitive, with the proposed method yielding gains over post-shift when run with tight surrogate approximations to classification errors.

Post-shift is a strong baseline – in theory, for many metrics it is optimal to simply post-shift the Bayes class probability model $\mathbf{P}(y = 1|x)$ with a suitable threshold β (Koyejo et al., 2014; Yan et al., 2018). Post-shift only has one degree of freedom, which limits it, but also enables choosing β to directly optimize the *true* metric. In contrast, our method acts through surrogate losses to optimize the target metric. We argue that post-shift should be a required baseline for experiments on custom metric optimization.

We look forward to seeing further theoretical analysis for handling black-box metrics, and further experimentation comparing methods with fewer but smarter parameters to those with more flexible modeling.

Acknowledgements

We thank Andrew Cotter for several helpful discussions.

References

Alabi, D., Immorlica, N., and Kalai, A. (2018). Unleashing linear optimizers for group-fair learning and optimization. In *COLT*.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias. *ProPublica, May*, 23.

Berahas, A. S., Cao, L., Choromanski, K., and Scheinberg, K. (2019). A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *arXiv preprint arXiv:1905.01332*.

Blake, C. and Merz, C. J. (1998). UCI repository of machine learning databases.

Conn, A. R., Scheinberg, K., and Vicente, L. N. (2008). Geometry of interpolation sets in derivative free optimization. *Mathematical programming*, 111(1-2):141–172.

Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009). *Introduction to derivative-free optimization*, volume 8. Siam.

Daskalaki, S., Kopanas, I., and Avouris, N. (2006). Evaluation of classifiers for an uneven class distribution problem. *Applied Artificial Intelligence*, 20:381–417.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono, A. (2015). Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806.

Eban, E., Schain, M., Mackey, A., Gordon, A., Saurous, R. A., and Elidan, G. (2017). Scalable learning of non-decomposable objectives. In *AISTATS*.

Garmanjani, R. and Vicente, L. N. (2013). Smoothing and worst-case complexity for direct-search methods in nonsmooth optimization. *IMA Journal of Numerical Analysis*, 33(3):1008–1028.

Ghadimi, S., Lan, G., and Zhang, H. (2016). Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1):267–305.

Grabocka, J., Scholz, R., and Schmidt-Thieme, L. (2019). Learning surrogate losses. *arXiv preprint arXiv:1905.10108*.

Huang, C., Zhai, S., Talbott, W., Bautista, M. A., Sun, S.-Y., Guestrin, C., and Susskind, J. (2019). Addressing the loss-metric mismatch with adaptive loss alignment. In *ICML*.

Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., and Jordan, M. I. (2019). A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint arXiv:1902.03736*.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML*.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*.

Kar, P., Li, S., Narasimhan, H., Chawla, S., and Sebastiani, F. (2016). Online optimization methods for the quantification problem. In *KDD*.

Kar, P., Narasimhan, H., and Jain, P. (2014). Online and stochastic gradient methods for non-decomposable loss functions. In *NIPS*.

Kar, P., Narasimhan, H., and Jain, P. (2015). Surrogate functions for maximizing precision at the top. In *ICML*.

Koyejo, O., Natarajan, N., Ravikumar, P., and Dhillon, I. (2014). Consistent binary classification with generalized performance metrics. In *NIPS*.

Mackey, A., Luo, X., and Eban, E. (2018). Constrained classification and ranking via quantiles. *arXiv preprint arXiv:1803.00067*.

Narasimhan, H. (2018). Learning with complex loss functions and constraints. In *AISTATS*.

Narasimhan, H., Cotter, A., and Gupta, M. (2019). Optimizing generalized rate metrics through game equilibrium. In *NeurIPS*.

Narasimhan, H., Kar, P., and Jain, P. (2015a). Optimizing non-decomposable performance measures: A tale of two classes. In *ICML*.

Narasimhan, H., Ramaswamy, H., Saha, A., and Agarwal, S. (2015b). Consistent multiclass algorithms for complex performance measures. In *ICML*.

Narasimhan, H., Vaish, R., and Agarwal, S. (2014). On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *NIPS*.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Found. Comput. Math.*, 17(2):527–566.

Parambath, S., Usunier, N., and Grandvalet, Y. (2014). Optimizing F-measures by cost-sensitive classification. In *NIPS*.

Rao, R. B., Yakhnenko, O., and Krishnapuram, B. (2008). Kdd cup 2008 and the workshop on mining medical data. *ACM SIGKDD Explorations Newsletter*, 10(2):34–38.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to reweight examples for robust deep learning. In *ICML*.

Tropp, J. A. (2015). An introduction to matrix concentration inequalities. *Found. Trends Mach. Learn.*, 8(1-2):1–230.

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press.

Wu, L., Tian, F., Xia, Y., Fan, Y., Qin, T., Jian-Huang, L., and Liu, T.-Y. (2018). Learning to teach with dynamic loss functions. In *NeurIPS*, pages 6466–6477.

Yan, B., Koyejo, O., Zhong, K., and Ravikumar, P. (2018). Binary classification with karmic, threshold-quasi-concave metrics. In *ICML*.

Ye, N., Chai, K., Lee, W., and Chieu, H. (2012). Optimizing F-measures: A tale of two approaches. In *ICML*.

Zhao, K., Gao, S., Wang, W., and Cheng, M.-M. (2019a). Optimizing the F-measure for threshold-free salient object detection. In *ICCV*.

Zhao, S., Milani Fard, M., Narasimhan, H., and Gupta, M. R. (2019b). Metric-optimized example weights. In *ICML*.