Optimizing Black-box Metrics with Iterative Example Weighting

Gaurush Hiranandani[†1] **Jatin Mathur**[1] **Harikrishna Narasimhan**[2] **Mahdi Milani Fard**[2] **Oluwasanmi Koyejo**[3 1]

Abstract

We consider learning to optimize a classification metric defined by a black-box function of the confusion matrix. Such black-box learning settings are ubiquitous, for example, when the learner only has query access to the metric of interest, or in noisy-label and domain adaptation applications where the learner must evaluate the metric via performance evaluation using a small validation sample. Our approach is to adaptively learn example weights on the training dataset such that the resulting weighted objective best approximates the metric on the validation sample. We show how to model and estimate the example weights and use them to iteratively post-shift a pre-trained class probability estimator to construct a classifier. We also analyze the resulting procedure's statistical properties. Experiments on various label noise, domain shift, and fair classification setups confirm that our proposal compares favorably to the state-of-the-art baselines for each application.

1. Introduction

In many real-world machine learning tasks, the evaluation metric one seeks to optimize is not explicitly available in closed-form. This is true for metrics that are evaluated through live experiments or by querying human users (Tamburrelli & Margara, 2014; Hiranandani et al., 2019a), or that require access to private or legally protected data (Awasthi et al., 2021), and hence cannot be written as an explicit training objective. This is also the case when the learner only has access to data with skewed training distribution or labels with heteroscedastic noise (Huang et al., 2019; Jiang et al., 2020), and hence cannot directly optimize the metric on the training set despite knowing its mathematical form.

These problems can be framed as black-box learning tasks, where the goal is to optimize an unknown classification metric on a large (possibly noisy) training data, given access to evaluations of the metric on a small, clean validation sample (Jiang et al., 2020). Our high-level approach to these learning tasks is to adaptively assigns weights to the training examples, so that the resulting weighted training objective closely approximates the black-box metric on the validation sample. We then construct a classifier by using the example weights to post-shift a class-probability estimator pre-trained on the training set. This results in an efficient, iterative approach that does not require any re-training.

Indeed, example weighting strategies have been widely used to both optimize metrics and to correct for distribution shift, but prior works either handle specialized forms of metric or data noise (Sugiyama et al., 2008; Natarajan et al., 2013; Patrini et al., 2017), formulate the example-weight learning task as a difficult non-convex problem that is hard to analyze (Ren et al., 2018; Zhao et al., 2019), or employ an expensive surrogate re-weighting strategy that comes with limited statistical guarantees (Jiang et al., 2020). In contrast, we propose a simple and effective approach to optimize a general black-box metric (that is a function of the confusion matrix) and provide a rigorous statistical analysis.

A key element of our approach is eliciting the weight coefficients by probing the black-box metric at few select classifiers and solving a system of linear equations matching the weighted training errors to the validation metric. We choose the "probing" classifiers so that the linear system is well-conditioned, for which we provide both theoretically-grounded options and practically efficient variants. This weight elicitation procedure is then used as a subroutine to iteratively construct the final plug-in classifier.

Contributions: (i) We provide a method for eliciting example weights for linear black-box metrics (Section 3). (ii) We use this procedure to iteratively learn a plug-in classifier for general black-box metrics (Section 4). (iii) We provide theoretical guarantees for metrics that are concave functions of the confusion matrix under distributional assumptions (Section 5). (iv) We experimentally show that our approach is competitive with (or better than) the state-of-the-art methods for tackling label noise in CIFAR-10 (Krizhevsky et al., 2009) and domain shift in Adience (Eidinger et al., 2014), and optimizing with proxy labels and a black-box fairness

[†]Part of this work was done when G.H. was an intern at Google Research, USA. [1]University of Illinois at Urbana-Champaign, Illinois, USA [2]Google Research, USA [3]Google Research, Accra. Correspondence to: Gaurush Hiranandani <gaurush2@illinois.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

metric on Adult (Dua & Graff, 2017) (Section 7).

Notations: Δ_m denotes the $(m-1)$-dimensional simplex. $[m] = \{1, \ldots, m\}$ represents an index set. onehot$(j) \in \{0,1\}^m$ returns the one-hot encoding of $j \in [m]$. The ℓ_2 norm a vector is denoted by $\|\cdot\|$.

2. Problem Setup

We consider a standard multiclass setup with an instance space $\mathcal{X} \subseteq \mathbb{R}^d$ and a label space $\mathcal{Y} = [m]$. We wish to learn a randomized multiclass classifier $h : \mathcal{X} \to \Delta_m$ that for any input $x \in \mathcal{X}$ predicts a distribution $h(x) \in \Delta_m$ over the m classes. We will also consider deterministic classifiers $h : \mathcal{X} \to [m]$ which map an instance x to one of m classes.

Evaluation Metrics. Let D denote the underlying data distribution over $\mathcal{X} \times \mathcal{Y}$. We will evaluate the performance of a classifier h on D using an evaluation metric $\mathcal{E}^D[h]$, with higher values indicating better performance. Our goal is to learn a classifier h that maximizes this evaluation measure:

$$\max_h \mathcal{E}^D[h]. \tag{1}$$

We will focus on metrics $\mathcal{E}^D$ that can be written in terms of classifier's confusion matrix $\mathbf{C}[h] \in [0,1]^{m \times m}$, where the i,j-th entry is the probability that the true label is i and the randomized classifier h predicts j:

$$C^D_{ij}[h] = \mathbf{E}_{(x,y)\sim D}\left[\mathbf{1}(y=i)h_j(x)\right].$$

The performance of the classifier can then be evaluated using a (possibly unknown) function $\psi : [0,1]^{m \times m} \to \mathbb{R}_+$ of the confusion matrix:

$$\mathcal{E}^D[h] = \psi(\mathbf{C}^D[h]). \tag{2}$$

Several common classification metrics take this form, including typical linear metrics $\psi(\mathbf{C}) = \sum_{ij} L_{ij} C_{ij}$ for some reward matrix $\mathbf{L} \in \mathbb{R}_+^{m \times m}$, the F-measure $\psi(\mathbf{C}) = \sum_i \frac{2C_{ii}}{\sum_j C_{ij} + \sum_j C_{ji}}$ (Lewis, 1995), and the G-mean $\psi(\mathbf{C}) = \left(\prod_i \left(C_{ii} / \sum_j C_{ij}\right)\right)^{1/m}$ (Daskalaki et al., 2006).

We consider settings where the learner has query-access to the evaluation metric $\mathcal{E}^D$, i.e., can evaluate the metric for any given classifier h but cannot directly write out the metric as an explicit mathematical objective. This happens when the metric is truly a black-box function, i.e., ψ is unknown, or when ψ is known, but we have access to only a noisy version of the distribution D needed to compute the metric.

Noisy Training Distribution. For learning a classifier, we assume access to a large sample S^{tr} of n^{tr} examples drawn from a distribution μ, which we will refer to as the "training" distribution. The training distribution μ may be the same as the true distribution D, or may differ from the true distribution D in the feature distribution $\mathbf{P}(x)$, the conditional label distribution $\mathbf{P}(y|x)$, or both. We also assume access to a smaller sample S^{val} of n^{val} examples drawn from the true distribution D. We will refer to the sample S^{tr} as the "training" sample, and the smaller sample S^{val} as the "validation" sample. We seek to solve (1) using both these samples.

The following are some examples of noisy training distributions in the literature:

Example 1 (Independent label noise (ILN) (Natarajan et al., 2013; Patrini et al., 2017))**.** *The distribution μ draws an example (x,y) from D, and randomly flips y to $\widetilde{y}$ with probability $\mathbf{P}(\widetilde{y}|y)$, independent of the instance x.*

Example 2 (Cluster-dependent label noise (CDLN) (Wang et al., 2020a))**.** *Suppose each x belongs to one of k disjoint clusters $g(x) \in [k]$. The distribution μ draws (x,y) from D and randomly flips y to $\widetilde{y}$ with probability $\mathbf{P}(\widetilde{y}|y, g(x))$.*

Example 3 (Instance-dependent label noise (IDLN) (Menon et al., 2018))**.** *μ draws (x,y) from D and randomly flips y to $\widetilde{y}$ with probability $\mathbf{P}(\widetilde{y}|y,x)$, which may depend on x.*

Example 4 (Domain shift (DS) (Sugiyama et al., 2008))**.** *μ draws $\widetilde{x}$ according to a distribution $\mathbf{P}^{\mu}(x)$ different from $\mathbf{P}^D(x)$, but draws y from the true conditional $\mathbf{P}^D(y|\widetilde{x})$.*

Our approach is to learn example weights on the training sample S^{tr}, so that the resulting weighted empirical objective (locally, if not globally) approximates an estimate of the metric $\mathcal{E}^D$ on the validation sample S^{val}. For ease of presentation, we will assume that the metrics only depend on the diagonal entries of the confusion matrix, i.e., C_{ii}'s. In Appendix A, we elaborate how our ideas can be extended to handle metrics that depend on the entire confusion matrix.

While our approach uses randomized classifiers, in practice one can replace them with similarly performing deterministic classifiers using, e.g., the techniques of (Cotter et al., 2019a). In what follows, we will need the empirical confusion matrix on the validation set $\widehat{\mathbf{C}}^{\text{val}}[h]$, where $\widehat{C}^{\text{val}}_{ij}[h] = \frac{1}{n^{\text{val}}} \sum_{(x,y)\in S^{\text{val}}} \mathbf{1}(y=i)h_j(x)$.

3. Example Weighting for Linear Metrics

We first describe our example weighting strategy for linear functions of the diagonal entries of the confusion matrix, which is given by:

$$\mathcal{E}^D[h] = \sum_i \beta_i C^D_{ii}[h] \tag{3}$$

for some (unknown) weights $\beta_1, \ldots, \beta_m$. In the next section, we will discuss how to use this procedure as a subroutine to handle more complex metrics.

3.1. Modeling Example Weights

We define an example weighting function $\mathbf{W} : \mathcal{X} \to \mathbb{R}_+^m$ which associates m *correction weights* $[W_i(x)]_{i=1}^m$ with each example x so that:

$$\mathbf{E}_{(x,y)\sim\mu}\left[\sum_i W_i(x)\,\mathbf{1}(y=i)h_i(x)\right] \approx \mathcal{E}^D[h], \forall h. \tag{4}$$

Table 1: Example weights $\mathbf{W} : \mathcal{X} \to \mathbb{R}_+^{m \times m}$ for linear metric $\mathcal{E}^D[h] = \langle \mathbf{L}, \mathbf{C}^D[h] \rangle$ under the noise models in Exmp. 1–4, where $W_{ij}(x)$ is the weight on entry C_{ij}. In Sec. 3–4, we consider metrics that are functions of the diagonal confusion entries alone (i.e. $\mathbf{L}$ and $\mathbf{T}$ are diagonal), and handle general metrics in Appendix A.

Model	Noise Transition Matrix	Correction Weights
ILN	$T_{ij} = \mathbf{P}(\widetilde{y} = j \vert y = i)$	$\mathbf{W}(x) = \mathbf{L} \odot \mathbf{T}^{-1}$
CDLN	$T_{ij}^{[k]} = \mathbf{P}(\widetilde{y} = j \vert y = i, g(x) = k)$	$\mathbf{W}(x) = \mathbf{L} \odot (\mathbf{T}^{[g(x)]})^{-1}$
IDLN	$T_{ij}(x) = \mathbf{P}(\widetilde{y} = j \vert y = i, x)$	$\mathbf{W}(x) = \mathbf{L} \odot (\mathbf{T}(x))^{-1}$
DS	-	$W_{ij}(x) = \mathbf{P}^D(x)/\mathbf{P}^\mu(x), \forall i, j$

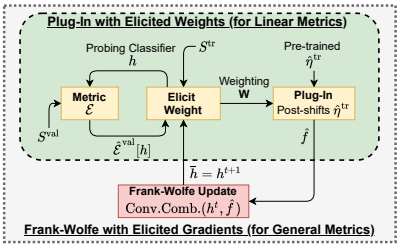


Figure 1: Overview of our approach.

Indeed for the noise models in Examples 1–4, there exist weighting functions $\mathbf{W}$ for which the above holds with equality. Table 1 shows the form of the weighting function for general linear metrics.

Ideally, the weighting function $\mathbf{W}$ assigns m independent weights for each example $x \in \mathcal{X}$. However, in practice, we estimate $\mathcal{E}^D$ using a small validation sample $S^{\text{val}} \sim D$. So to avoid having the example weights over-fit to the validation sample, we restrict the flexibility of $\mathbf{W}$ and set it to a weighted sum of L basis functions $\phi^\ell : \mathcal{X} \to [0, 1]$:

$$W_i(x) \;=\; \textstyle\sum_{\ell=1}^{L} \alpha_i^\ell \phi^\ell(x), \tag{5}$$

where $\alpha_i^\ell \in \mathbb{R}$ is the coefficient associated with basis function ϕ^ℓ and diagonal confusion entry (i, i).

In practice, the basis functions can be as simple as a partitioning of the instance space into L clusters, i.e.,:

$$\phi^\ell(x) = \mathbf{1}(g(x) = \ell), \tag{6}$$

for a clustering function $g : \mathcal{X} \to [L]$, or may define a more complicated soft clustering using, e.g., radial basis functions (Sugiyama et al., 2008) with centers x^ℓ and width σ:

$$\phi^\ell(x) = \exp\left(-\|x - x^\ell\|/2\sigma^2\right). \tag{7}$$

3.2. ϕ-transformed Confusions

Expanding the weighting function in (4) gives us:

$$\sum_{\ell=1}^{L}\sum_{i=1}^{m} \alpha_i^\ell \underbrace{\mathbf{E}_{(x,y)\sim\mu}\left[\phi^\ell(x)\,\mathbf{1}(y = i)h_i(x)\right]}_{\Phi_i^{\mu,\ell}[h]} \;\approx\; \mathcal{E}^D[h], \forall h,$$

where $\mathbf{\Phi}^{\mu,\ell}[h] \in [0, 1]^m$ can be seen as a ϕ-transformed confusion matrix for the training distribution μ. For example, if one had only one basis function $\phi^1(x) = 1, \forall x$, then $\Phi_i^{\mu,1}[h] = \mathbf{E}_{(x,y)\sim\mu}\left[\mathbf{1}(y = i)h_i(x)\right]$ gives the standard confusion entries for the training distribution. If the basis functions divides the data into L clusters, as in (6), then $\Phi_i^{\mu,\ell}[h] = \mathbf{E}_{(x,y)\sim\mu}\left[\mathbf{1}(g(x) = \ell, y = i)h_i(x)\right]$ gives the training confusion entries evaluated on examples from cluster ℓ. We can thus re-write equation (4) as a weighted combination of the Φ-confusion entries:

$$\sum_{\ell=1}^{L}\sum_{i=1}^{m} \alpha_i^\ell \Phi_i^{\mu,\ell}[h] \;\approx\; \mathcal{E}^D[h], \forall h. \tag{8}$$

3.3. Eliciting Weight Coefficients $\boldsymbol{\alpha}$

We next discuss how to estimate the weighting function coefficients α_i^ℓ's from the training sample S^{tr} and validation sample S^{val}. Notice that (8) gives a relationship between statistics $\mathbf{\Phi}^{\mu,\ell}$'s computed on the training distribution μ, and the evaluation metric of interest computed on the true distribution D. Moreover, for a fixed classifier h, the left-hand side is *linear* in the unknown coefficients $\boldsymbol{\alpha} = [\alpha_1^1, \ldots, \alpha_1^L, \ldots, \alpha_m^1, \ldots, \alpha_m^L] \in \mathbb{R}^{Lm}$.

We therefore probe the metric $\widehat{\mathcal{E}}^{\text{val}}$ at Lm different classifiers $h^{1,1}, \ldots, h^{1,m}, \ldots, h^{L,1}, \ldots, h^{L,m}$, which results in a set of Lm linear equations of the form in (8):

$$\begin{aligned} \textstyle\sum_{\ell,i} \alpha_i^\ell\, \widehat{\Phi}_i^{\text{tr},\ell}[h^{1,1}] &= \widehat{\mathcal{E}}^{\text{val}}[h^{1,1}], \\ &\vdots \\ \textstyle\sum_{\ell,i} \alpha_i^\ell\, \widehat{\Phi}_i^{\text{tr},\ell}[h^{L,m}] &= \widehat{\mathcal{E}}^{\text{val}}[h^{L,m}], \end{aligned} \tag{9}$$

where $\widehat{\Phi}_i^{\text{tr},\ell}[h] = \frac{1}{n^{\text{tr}}} \sum_{(x,y)\in S^{\text{tr}}} \phi^\ell(x)\,\mathbf{1}(y = i)h_i(x)$ is evaluated on the training sample and the metric $\widehat{\mathcal{E}}^{\text{val}}[h] = \sum_i \beta_i \widehat{C}_{ii}^{\text{val}}[h]$ is evaluated on the validation sample.

More formally, let $\widehat{\mathbf{\Sigma}} \in \mathbb{R}^{Lm \times Lm}$ and $\widehat{\boldsymbol{\mathcal{E}}} \in \mathbb{R}^{Lm}$ denote the left-hand and right-hand side observations in (9), i.e.,:

$$\begin{aligned} \widehat{\Sigma}_{(\ell,i),(\ell',i')} \;&=\; \frac{1}{n^{\text{tr}}} \sum_{(x,y)\in S^{\text{tr}}} \phi^{\ell'}(x)\mathbf{1}(y = i')h_{i'}^{\ell,i}(x), \\ \widehat{\mathcal{E}}_{(\ell,i)} \;&=\; \widehat{\mathcal{E}}^{\text{val}}[h^{\ell,i}]. \end{aligned} \tag{10}$$

Then the weight coefficients are given by $\widehat{\boldsymbol{\alpha}} = \widehat{\mathbf{\Sigma}}^{-1}\widehat{\boldsymbol{\mathcal{E}}}$.

3.4. Choosing the Probing Classifiers $h^{1,1}, \ldots, h^{L,m}$

We will have to choose the Lm probing classifiers so that $\widehat{\mathbf{\Sigma}}$ is well-conditioned. One way to do this is to choose the classifiers so that $\widehat{\mathbf{\Sigma}}$ has a high value on the diagonal entries and a low value on the off-diagonals, i.e. choose each classifier $h^{\ell,i}$ to evaluate to a high value on $\widehat{\Phi}_i^{\text{tr},\ell}[h]$ and a low value on $\widehat{\Phi}_{i'}^{\text{tr},\ell'}[h]$, $\forall(\ell', i') \neq (\ell, i)$. This can be framed as the following constraint satisfaction problem on S^{tr}:

$$\begin{aligned} &\text{For } h^{\ell,i} \text{ pick } h \in \mathcal{H} \text{ such that:} \\ &\widehat{\Phi}_i^{\text{tr},\ell}[h] \geq \gamma, \text{ and } \widehat{\Phi}_{i'}^{\text{tr},\ell'}[h] \leq \omega, \forall(\ell', i') \neq (\ell, i), \end{aligned} \tag{11}$$

Algorithm 1: **ElicitWeights** for Diagonal Linear Metrics

Input: $\widehat{\mathcal{E}}^{\text{val}}$, Basis functions $\phi^1, \ldots, \phi^L : \mathcal{X} \to [0,1]$, Training set $S^{\text{tr}} \sim \mu$, Val. set $S^{\text{val}} \sim D$, $\bar{h}$, ϵ, $\mathcal{H}$, γ, ω
If *fixed classifier*:
  Choose $h^{\ell,i}(x) = \epsilon\phi^{\ell}(x)\, e^i(x) + (1 - \epsilon\phi^{\ell}(x))\, \bar{h}(x)$
Else:
  $\bar{\mathcal{H}} = \{\tau h + (1-\tau)\bar{h} \,|\, h \in \mathcal{H}, \tau \in [0, \epsilon]\}$
  Pick $h^{\ell,i} \in \bar{\mathcal{H}}$ to satisfy (11) with slack $\gamma, \omega, \forall(\ell, i)$
Compute $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\boldsymbol{\mathcal{E}}}$ using (10) with metric $\widehat{\mathcal{E}}^{\text{val}}$
Output: $\widehat{\boldsymbol{\alpha}} = \widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mathcal{E}}}$

Algorithm 2: **P**lug-**i**n with **E**licited **W**eights (**PI-EW**) for Diagonal Linear Metrics

Input: $\widehat{\mathcal{E}}^{\text{val}}$, Basis functions $\phi^1, \ldots, \phi^L : \mathcal{X} \to [0,1]$, Class probability model $\widehat{\eta}^{\text{tr}} : \mathcal{X} \to \Delta_m$ for μ, Training set $S^{\text{tr}} \sim \mu$, Validation set $S^{\text{val}} \sim D$, $\bar{h}$, ϵ
$\widehat{\boldsymbol{\alpha}} = \textbf{ElicitWeights}(\widehat{\mathcal{E}}^{\text{val}}, \phi^1, \ldots, \phi^L, S^{\text{tr}}, S^{\text{val}}, \bar{h}, \epsilon)$
Example-weights: $\widehat{W}_i(x) = \sum_{\ell=1}^{L} \widehat{\alpha}_i^{\ell}\phi_i^{\ell}(x)$
Plug-in: $\widehat{h}(x) \in \text{argmax}_{i \in [m]}\, \widehat{W}_i(x)\widehat{\eta}_i^{\text{tr}}(x)$
Output: $\widehat{h}$

for some $\gamma > \omega > 0$ and a sufficiently flexible hypothesis class $\mathcal{H}$ for which the constraints are feasible. These problems can generally be solved by formulating a constrained classification problem (Cotter et al., 2019b; Narasimhan, 2018). We show in Appendix G that this problem is feasible and can be efficiently solved for a range of settings.

In practice, we do not explicitly solve (11) over a hypothesis class $\mathcal{H}$. Instead, a simpler and surprisingly effective strategy is to set the probing classifiers to *trivial* classifiers that predict the same class on all (or a subset of) examples. To build intuition for why this is a good idea, consider a simple setting with only one basis function $\phi^1(x) = 1, \forall x$, where the ϕ-confusions $\widehat{\Phi}_i^{\text{tr},1}[h] = \frac{1}{n^{\text{tr}}}\sum_{(x,y)\in S^{\text{tr}}} \mathbf{1}(y = i)h_i(x)$ are the standard confusion entries on the training set. In this case, a trivial classifier $e^i(x) = \text{onehot}(i), \forall x$, which predicts class i on all examples, yields the highest value for $\widehat{\Phi}_i^{\text{tr},1}$ and 0 for all other $\widehat{\Phi}_j^{\text{tr},1}, \forall j \neq i$. In fact, in our experiments, we set the probing classifier $h^{1,i}$ to a randomized combination of e^i and some fixed base classifier $\bar{h}$:

$$h^{1,i}(x) = \epsilon e^i(x) + (1-\epsilon)\bar{h}(x),$$

for large enough ϵ so that $\widehat{\boldsymbol{\Sigma}}$ is well-conditioned.

Similarly, if the basis functions divide the data into L clusters (as in (6)), then we can randomize between $\bar{h}$ and a trivial classifier that predicts a particular class i on all examples assigned to the cluster $\ell \in [L]$. The confusion matrix for the resulting classifiers will have higher values than $\bar{h}$ on the (ℓ, i)-th diagonal entry and a lower value on other

Algorithm 3: **F**rank-**W**olfe with **E**licited **G**radients (**FW-EG**) for General Diagonal Metrics (also depicted in Fig. 1)

Input: $\widehat{\mathcal{E}}^{\text{val}}$, Basis functions $\phi^1, \ldots, \phi^L : \mathcal{X} \to [0,1]$, Pre-trained $\widehat{\eta}^{\text{tr}} : \mathcal{X} \to \Delta_m$, $S^{\text{tr}} \sim \mu$, $S^{\text{val}} \sim D$, T, ϵ
Initialize classifier h^0 and $\mathbf{c}^0 = \text{diag}(\widehat{\mathbf{C}}^{\text{val}}[h^0])$
For $t = 0$ **to** $T-1$ **do**
  if $\mathcal{E}^D[h] = \psi(C_{11}^D[h], \ldots, C_{mm}^D[h])$ for known ψ:
    $\boldsymbol{\beta}^t = \nabla\psi(\mathbf{c}^t)$
    $\widehat{\mathcal{E}}^{\text{lin}}[h] = \sum_i \beta_i^t \widehat{C}_{ii}^{\text{val}}[h]$
  else
    $\widehat{\mathcal{E}}^{\text{lin}}[h] = \widehat{\mathcal{E}}^{\text{val}}[h]$ {small ϵ recommendeded}
  $\widehat{f} = \textbf{PI-EW}(\widehat{\mathcal{E}}^{\text{lin}}, \phi^1, ..., \phi^L, \widehat{\eta}^{\text{tr}}, S^{\text{tr}}, S^{\text{val}}, h^t, \epsilon)$
  $\widetilde{\mathbf{c}} = \text{diag}(\widehat{\mathbf{C}}^{\text{val}}[\widehat{f}])$
  $h^{t+1} = \left(1 - \frac{2}{t+1}\right)h^t + \frac{2}{t+1}\text{onehot}(\widehat{f})$
  $\mathbf{c}^{t+1} = \left(1 - \frac{2}{t+1}\right)\mathbf{c}^t + \frac{2}{t+1}\widetilde{\mathbf{c}}$
End For
Output: $\widehat{h} = h^T$

entries. These classifiers can be succinctly written as:

$$h^{\ell,i}(x) = \epsilon\phi^{\ell}(x)e^i(x) + (1 - \epsilon\phi^{\ell}(x))\bar{h} \qquad (12)$$

where we again tune ϵ to make sure that the resulting $\widehat{\boldsymbol{\Sigma}}$ is well-conditioned. This choice of the probing classifiers also works well in practice for general basis functions ϕ^{ℓ}'s.

Algorithm 1 summarizes the weight elicitation procedure, where the probing classifiers are either constructed by solving the constrained satisfaction problem (11) or set to the "fixed" classifiers in (12). In both cases, the algorithm takes a base classifier $\bar{h}$ and the parameter ϵ as input, where ϵ controls the extent to which $\bar{h}$ is perturbed to construct the probing classifiers. This radius parameter ϵ restricts the probing classifiers to a neighborhood around $\bar{h}$ and will prove handy in the algorithm we develop in Section 4.2.

4. Plug-in Based Algorithms

Having elicited the weight coefficients $\boldsymbol{\alpha}$, we now seek to learn a classifier that optimizes the left hand side of (8). We do this via the *plug-in* approach: first *pre-train* a model $\widehat{\eta}^{\text{tr}} : \mathcal{X} \to \Delta_m$ on the noisy training distribution μ to estimate the conditional class probabilities $\widehat{\eta}_i^{\text{tr}}(x) \approx \mathbf{P}^{\mu}(y = i|x)$, and then apply the correction weights to *post-shift* $\widehat{\eta}^{\text{tr}}$.

4.1. Plug-in Algorithm for Linear Metrics

We first describe our approach for (diagonal) linear metrics $\mathcal{E}^D[h] = \sum_i \beta_i C_{ii}^D[h]$ in Algorithm 2. Given the correction weights $\widehat{\mathbf{W}} : \mathcal{X} \to \mathbb{R}_+^m$, we seek to maximize the following weighted objective on the training distribution:

$$\max_h \mathbf{E}_{(x,y)\sim\mu}\left[\sum_i \widehat{W}_i(x)\,\mathbf{1}(y = i)h_i(x)\right].$$

This is a standard example-weighted learning problem, for which the following plug-in (post-shift) classifier is a consistent estimator (Narasimhan et al., 2015b; Yang et al., 2020):

$$\widehat{h}(x) \in \operatorname{argmax}_{i\in[m]} \widehat{W}_i(x)\,\widehat{\eta}^{\text{tr}}_i(x).$$

4.2. Iterative Algorithm for General Metrics

To optimize generic non-linear metrics of the form $\mathcal{E}^D[h] = \psi(C^D_{11}[h], \ldots, C^D_{mm}[h])$ for $\psi : [0,1]^m \to \mathbb{R}_+$, we apply Algorithm 2 iteratively. We consider both cases where ψ is unknown, and where ψ is known, but needs to be optimized using the noisy distribution μ. The idea is to first elicit local linear approximations to ψ and to then learn plug-in classifiers for the resulting linear metrics in each iteration.

Specifically, following Narasimhan et al. (2015b), we derive our algorithm from the classical Frank-Wolfe method (Jaggi, 2013) for maximizing a smooth concave function $\psi(\mathbf{c})$ over a convex set $\mathcal{C} \subseteq \mathbb{R}^m$. In our case, $\mathcal{C}$ is the set of confusion matrices $\mathbf{C}^D[h]$ achieved by any classifier h, and is convex when we allow randomized classifiers (see Lemma 10, Appendix B.3). The algorithm maintains iterates $\mathbf{c}^t$, and at each step, maximizes a linear approximation to ψ at $\mathbf{c}^t$: $\widetilde{\mathbf{c}} \in \operatorname{argmax}_{\mathbf{c}\in\mathcal{C}} \langle \nabla\psi(\mathbf{c}^t), \mathbf{c}\rangle$. The next iterate $\mathbf{c}^{t+1}$ is then a convex combination of $\mathbf{c}^t$ and the current solution $\widetilde{\mathbf{c}}$.

In Algorithm 3, we outline an adaptation of this Frank-Wolfe algorithm to our setting, where we maintain a classifier h^t and an estimate of the diagonal confusion entries $\mathbf{c}^t$ from the validation sample S^{val}. At each step, we linearize ψ using $\widehat{\mathcal{E}}^{\text{lin}}[h] = \sum_i \beta^t_i \widehat{C}^{\text{val}}_{ii}[h]$, where $\boldsymbol{\beta}^t = \nabla\psi(\mathbf{c}^t)$, and invoke the plug-in method in Algorithm 2 to optimize the linear approximation $\widehat{\mathcal{E}}^{\text{lin}}$. When the mathematical form of ψ is known, one can directly compute the gradient $\boldsymbol{\beta}^t$. When it is not known, we can simply set $\widehat{\mathcal{E}}^{\text{lin}}[h] = \widehat{\mathcal{E}}^{\text{val}}[h]$, but restrict the weight elicitation routine (Algorithm 1) to choose its probing classifiers $h^{\ell,i}$'s from a small neighborhood around the current classifier h^t (in which ψ is effectively linear). This can be done by passing $\bar{h} = h^t$ to the weight elicitation routine, and setting the radius ϵ to a small value.

Each call to Algorithm 2 uses the training and validation set to elicit example weights for a local linear approximation to ψ, and uses the weights to construct a plug-in classifier. The final output is a randomized combination of the plug-in classifiers from each step. Note that Algorithm 3 runs efficiently for reasonable values of L and m. Indeed the runtime is almost always dominated by the pre-training of the base model $\widehat{\eta}^{\text{tr}}$, with the time taken to elicit the weights (e.g. using (12)) being relatively inexpensive (see App. E).

5. Theoretical Guarantees

We provide theoretical guarantees for the weight elicitation procedure and the plug-in methods in Algorithms 1–3.

Assumption 1. *The distributions D and μ are such that for any linear metric $\mathcal{E}^D[h] = \sum_i \beta_i C_{ii}[h]$, with $\|\boldsymbol{\beta}\| \le 1$, $\exists \bar{\boldsymbol{\alpha}} \in \mathbb{R}^{Lm}$ s.t. $\left|\sum_{\ell,i} \bar{\alpha}^\ell_i \Phi^{\mu,\ell}_i[h] - \mathcal{E}^D[h]\right| \le \nu, \forall h$ and $\|\bar{\boldsymbol{\alpha}}\|_1 \le B$, for some $\nu \in [0,1)$ and $B > 0$.*

The assumption states that our choice of basis functions $\phi^1, \ldots, \phi^L$ are such that, any linear metric on D can be approximated (up to a slack ν) by a weighting $W_i(x) = \sum_\ell \bar{\alpha}^\ell_i \phi^\ell(x)$ of the training examples from μ. The existence of such a weighting function depends on how well the basis functions capture the underlying distribution shift. Indeed, the assumption holds for some common settings in Table 1, e.g., when the noise transition $\mathbf{T}$ is diagonal (Appendix A handles a general $\mathbf{T}$), and the basis functions are set to $\phi^1(x) = 1, \forall x$, for the IDLN setting, and $\phi^\ell(x) = \mathbf{1}(g(x) = \ell), \forall x$, for the CDLN setting.

We analyze the coefficients $\widehat{\boldsymbol{\alpha}}$ elicited by Algorithm 1 when the probing classifiers $h^{\ell,i}$ are chosen to satisfy (11). In Appendix C, we provide an analysis when the probing classifiers $h^{\ell,i}$ are set to the fixed choices in (12).

Theorem 1 (Error bound on elicited weights). *Let $\gamma, \omega > 0$ be such that the constraints in (11) are feasible for hypothesis class $\bar{\mathcal{H}}$, for all ℓ, i. Suppose Algorithm 1 chooses each classifier $h^{\ell,i}$ to satisfy (11), with $\mathcal{E}^D[h^{\ell,i}] \in [c, 1], \forall \ell, i$, for some $c > 0$. Let $\bar{\alpha}$ be defined as in Assumption 1. Suppose $\gamma > 2\sqrt{2}Lm\omega$ and $n^{\text{tr}} \ge \frac{L^2 m \log(Lm|\mathcal{H}|/\delta)}{(\frac{\gamma}{2} - \sqrt{2}Lm\omega)^2}$. Fix $\delta \in (0,1)$. Then w.p. $\ge 1 - \delta$ over draws of S^{tr} and S^{val} from μ and D resp., the coefficients $\widehat{\boldsymbol{\alpha}}$ output by Algorithm 1 satisfies:*

$$\|\widehat{\boldsymbol{\alpha}} - \bar{\boldsymbol{\alpha}}\| \le$$
$$\mathcal{O}\left(\frac{Lm}{\gamma^2}\left(\sqrt{\frac{L\log(\frac{Lm|\mathcal{H}|}{\delta})}{n^{\text{tr}}}} + \sqrt{\frac{L\log(\frac{Lm}{\delta})}{c^2 n^{\text{val}}}}\right) + \frac{\nu\sqrt{Lm}}{\gamma}\right),$$

where the term $|\mathcal{H}|$ can be replaced by a measure of capacity of the hypothesis class $\mathcal{H}$.

Because the probing classifiers are chosen using the training set alone, it is only the sampling errors from the training set that depend on the complexity of $\mathcal{H}$, and not those from the validation set. This suggests robustness of our approach to a small validation set as long as the training set is sufficiently large and the number of basis functions is reasonably small.

For the iterative plug-in method in Algorithm 3, we bound the gap between the metric value $\mathcal{E}^D[\widehat{h}]$ for the output classifier $\widehat{h}$ on the true distribution D, and the optimal value. We handle the case where the function ψ is *known* and its gradient $\nabla\psi$ can be computed in closed-form. The more general case of an unknown ψ is handled in Appendix D. The above bound depends on the gap between the estimated class probabilities $\widehat{\eta}^{\text{tr}}_i(x)$ for the training distribution and true class probabilities $\eta^{\text{tr}}_i(x) = \mathbf{P}(y = i|x)$, as well as the quality of the coefficients $\widehat{\boldsymbol{\alpha}}$ provided by the weight estima-

tion subroutine, as measured by $\kappa(\cdot)$. One can substitute $\kappa(\cdot)$ with, e.g., the error bound provided in Theorem 1.

Theorem 2 (Error Bound for FW-EG). *Let $\mathcal{E}^D[h] = \psi(C^D_{11}[h], \ldots, C^D_{mm}[h])$ for a* known *concave function $\psi : [0,1]^m \to \mathbb{R}_+$, which is Q-Lipschitz and λ-smooth. Fix $\delta \in (0,1)$. Suppose Assumption 1 holds, and for any linear metric $\sum_i \beta_i C^D_{ii}[h]$, whose associated weight coefficients is $\bar{\boldsymbol{\alpha}}$ with $\|\bar{\boldsymbol{\alpha}}\| \leq B$, w.p. $\geq 1-\delta$ over draw of S^{tr} and S^{val}, the weight estimation routine in Alg. 1 outputs coefficients $\widehat{\boldsymbol{\alpha}}$ with $\|\widehat{\boldsymbol{\alpha}} - \bar{\boldsymbol{\alpha}}\| \leq \kappa(\delta, n^{\text{tr}}, n^{\text{val}})$, for some function $\kappa(\cdot) > 0$. Let $B' = B + \sqrt{Lm}\,\kappa(\delta/T, n^{\text{tr}}, n^{\text{val}})$. Then w.p. $\geq 1-\delta$ over draws of S^{tr} and S^{val} from D and μ resp., the classifier $\widehat{h}$ output by Algorithm 3 after T iterations satisfies:*

$$\max_h \mathcal{E}^D[h] - \mathcal{E}^D[\widehat{h}] \leq$$
$$2QB'\mathbf{E}_x\left[\|\eta^{\text{tr}}(x) - \widehat{\eta}^{\text{tr}}(x)\|_1\right] + 4Q\sqrt{Lm}\,\kappa(\tfrac{\delta}{T}, n^{\text{tr}}, n^{\text{val}}) +$$
$$\mathcal{O}\left(\lambda m \sqrt{\frac{m\log(m)\log(n^{\text{val}}) + \log(m/\delta)}{n^{\text{val}}}} + \frac{\lambda}{T} + Q\nu\right).$$

The proof in turn derives an error bound for the plug-in classifier in Algorithm 2 for linear metrics (see App. B.2).

6. Related Work

Methods for closed-form metrics. There has been a variety of work on optimizing complex evaluation metrics, including both plug-in type algorithms (Ye et al., 2012; Narasimhan et al., 2014; Koyejo et al., 2014; Narasimhan et al., 2015b; Yan et al., 2018), and those that use convex surrogates for the metric (Joachims, 2005; Kar et al., 2014; 2016; Narasimhan et al., 2015a; Eban et al., 2017; Narasimhan et al., 2019; Hiranandani et al., 2020). These methods rely on the test metric having a specific closed-form structure and do not handle black-box metrics.

Methods for black-box metrics. Among recent black-box metric learning works, the closest to ours is Jiang et al. (2020), who learn a weighted combination of surrogate losses to approximate the metric on a validation set. Like us, they probe the metric at multiple classifiers, but their approach has several drawbacks on both practical and theoretical fronts. Firstly, Jiang et al. (2020) require retraining the model in each iteration, which can be time-intensive, whereas we only post-shift a pre-trained model. Secondly, the procedure they prescribe for eliciting gradients requires perturbing the model parameters multiple times, which can be very expensive for large deep networks, whereas we only require perturbing the predictions from the model. Moreover, the number of perturbations they need grows *polynomially* with the precision with which they need to estimate the loss coefficients, whereas we only require a *constant* number of them. Lastly, their approach does not come with strong statistical guarantees, whereas ours does. Besides these benefits over (Jiang et al., 2020), we will also see in Section 7 that our method yields better accuracies. Other related black-box learning methods include Zhao et al. (2019), Ren et al. (2018), and Huang et al. (2019), who learn a (weighted) loss to approximate the metric, but do so using computationally expensive procedures (e.g. meta-gradient descent or RL) that often require retraining the model from scratch, and come with limited theoretical analysis.

Methods for distribution shift. The literature on distribution shift is vast, and so we cover a few representative papers; see (Frénay & Verleysen, 2013; Csurka, 2017) for a comprehensive discussion. For the *independent label noise* setting (Natarajan et al., 2013), Patrini et al. (2017) propose a loss correction approach that first trains a model with noisy label, use its predictions to estimate the noise transition matrix, and then re-trains model with the corrected loss. This approach is however tailored to optimize linear metrics; whereas, we can handle more complex metrics as well without re-training the underlying model. A plethora of approaches exist for tackling *domain shift*, including classical importance weighting (IW) strategies (Sugiyama et al., 2008; Shimodaira, 2000; Kanamori et al., 2009; Lipton et al., 2018) that work in two steps: estimate the density ratios and train a model with the resulting weighted loss. One such approach is Kernel Mean Matching (Huang et al., 2006), which matches covariate distributions between training and test sets in a high dimensional RKHS feature space. These IW approaches are however prone to over-fitting when used with deep networks (Byrd & Lipton, 2019). More recent iterative variants seek to remedy this (Fang et al., 2020).

7. Experiments

We run experiments on four classification tasks, with both known and black-box metrics, and under different label noise and domain shift settings. All our experiments use a large training sample, which is either noisy or contains missing attributes, and a smaller clean (and complete) validation sample. We always optimize the cross-entropy loss for learning $\widehat{\eta}^{\text{tr}}(x) \approx \mathbf{P}^{\mu}(Y|x)$ using the training set (or $\widehat{\eta}^{\text{val}}(x) \approx \mathbf{P}^{D}(Y|x)$ for some baselines), where the models are varied across experiments. For monitoring the quality of $\widehat{\eta}^{\text{tr}}$ and $\widehat{\eta}^{\text{val}}$, we sample small subsets *hyper-train* and *hyper-val* data from the original training and validation data, respectively. We repeat our experiments over 5 random train-vali-test splits, and report the mean and standard deviation for each metric. We will use *, **, and *** to denote that the differences between our method and the closest baseline are statistically significant (using Welch's t-test) at a confidence level of 90%, 95%, and 99%, respectively. Table 6 in App. H summarizes the datasets used. The source code (along with random seeds) is provided on the link below.[1]

Common baselines: We use representative baselines from

[1] https://github.com/koyejolab/fweg/

Table 2: Test accuracy for noisy label experiment on CIFAR-10.

Cross-entropy [train]	0.582 ± 0.007
Cross-entropy [val]	0.386 ± 0.031
Learn-to-reweight	0.651 ± 0.017
Plug-in [train-val]	0.733 ± 0.044
Forward Correction	0.757 ± 0.005
Fine-tuning	0.769 ± 0.005
PI-EW	$\mathbf{0.781 \pm 0.019}$

the black-box learning (Jiang et al., 2020), iterative re-weighting (Ren et al., 2018), label noise correction (Patrini et al., 2017), and importance weighting (Huang et al., 2006) literatures. First, we list the ones common to all experiments.

1. **Cross-entropy [train]:** Maximizes accuracy on the training set and predicts: $\widehat{h}(x) \in \operatorname{argmax}_{i\in[m]} \widehat{\eta}_i^{\text{tr}}(x)$.
2. **Cross-entropy [val]:** Maximizes accuracy on the validation set and predicts: $\widehat{h}(x) \in \operatorname{argmax}_{i\in[m]} \widehat{\eta}_i^{\text{val}}(x)$.
3. **Fine-tuning:** Fine-tunes the pre-trained $\widehat{\eta}^{\text{tr}}$ using the validation data, monitoring the cross-entropy loss on the hyper-val data for early stopping.
4. **Opt-metric [val]:** For metrics $\psi(\mathbf{C}^D[h])$, for which ψ is *known*, trains a model to directly maximize the metric on the small *validation* set using the Frank-Wolfe based algorithm of (Narasimhan et al., 2015b).
5. **Learn-to-reweight** (Ren et al., 2018): Jointly learns example weights, with the model, to maximize accuracy on the validation set; does not handle specialized metrics.
6. **Plug-in [train-val]:** Constructs a classifier $\widehat{h}(x) \in \operatorname{argmax}_i w_i \widehat{\eta}_i^{\text{val}}(x)$, where the weights $w_i \in \mathbb{R}$ are tuned to maximize the given metric on the validation set, using a coordinate-wise line search (details in Appendix F).
7. **Adaptive Surrogates** (Jiang et al., 2020): Learns a weighted combination of surrogate losses (evaluated on clusters of examples) to approximate the metric on the validation set. Since this method is not directly amenable for use with large neural networks (see Section 6), we compare with it only when using linear models, and present additional comparisons in App. H (Table 7).

Hyper-parameters: The learning rate for Fine-tuning is chosen from $1\text{e}^{\{-6,\ldots,-4\}}$. For PI-EW and FW-EG, we tune the parameter ϵ from $\{1, 0.4, 1\text{e}^{-\{4,3,2,1\}}\}$. The line search for Plug-in is performed with a spacing of 1e^{-4}. The only hyper-parameters the other baselines have are those for training $\widehat{\eta}^{\text{tr}}$ and $\widehat{\eta}^{\text{val}}$, which we state in the individual tasks.

7.1. Maximizing Accuracy under Label Noise

In our first task, we train a 10-class image classifier for the CIFAR-10 dataset (Krizhevsky et al., 2009), replicating the independent (asymmetric) label noise setup from (Patrini et al., 2017). The evaluation metric we use is accuracy. We take 2% of original training data as validation data and flip labels in the remaining training set based on the following transition matrix: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ PLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG, with a flip probability of 0.6. For $\widehat{\eta}^{\text{tr}}$ and $\widehat{\eta}^{\text{val}}$, we use the same ResNet-14 architecture as (Patrini et al., 2017), trained using SGD with momentum 0.9, weight decay 1e^{-4}, and learning rate 0.01, which we divide by 10 after 40 and 80 epochs (120 in total).

We additionally compare with the *Forward Correction* method of (Patrini et al., 2017), a specialized method for correcting independent label noise, which estimates the noise transition matrix $\mathbf{T}$ using predictions from $\widehat{\eta}^{\text{tr}}$ on the training set, and retrains it with the corrected loss, thus training the ResNet twice. We saw a notable drop with this method when we used the (small) validation set to estimate $\mathbf{T}$.

We apply the proposed PI-EW method for linear metrics, using a weighting function $\mathbf{W}$ defined with one of two choices for the basis functions (chosen via cross-validation): (i) a default basis function that clusters all the points together $\phi^{\text{def}}(x) = 1\,\forall x$, and (ii) ten basis functions $\phi^1, \ldots, \phi^{10}$, each one being the average of the RBF kernels (see (7)) centered at validation points belonging to a true class. The RBF kernels are computed with width 2 on UMAP-reduced 50-dimensional image embeddings (McInnes et al., 2018).

As shown in Table 2, PI-EW achieves significantly better test accuracies than all the baselines. The results for Forward Correction matches those in (Patrini et al., 2017); unlike this method, we train the ResNet only once, but achieve 2.4% higher accuracy. Cross-entropy [val] over-fits badly, and yields the least test accuracy. Surprisingly, the simple fine-tuning yields the second-best accuracy. A possible reason is that the pre-trained model learns a good feature representation, and the fine-tuning step adapts well to the domain change. We also observed that PI-EW achieves better accuracy during cross-validation with ten basis functions, highlighting the benefit of the underlying modeling in PI-EW. Lastly, in Figure 2(a), we show the elicited (class) weights with the default basis function ($\phi^{\text{def}}(x) = 1\,\forall x$), where e.g. because BIRD $\rightarrow$ PLANE, the weight on BIRD is upweighted and that on PLANE is down-weighted.

Table 3: Test G-mean for proxy label experiment on Adult.

Cross-entropy [train]	0.654 ± 0.002
Cross-entropy [val]	0.394 ± 0.064
Opt-metric [val]	0.652 ± 0.027
Learn-to-reweight	0.668 ± 0.003
Plug-in [train-val]	0.672 ± 0.013
Forward Correction	0.214 ± 0.004
Fine-tuning	0.631 ± 0.017
Importance Weights	0.662 ± 0.024
Adaptive Surrogates	0.682 ± 0.002
FW-EG [unknown ψ]	$\mathbf{0.685 \pm 0.002}^{**}$
FW-EG [known ψ]	$\mathbf{0.685 \pm 0.001}^{*}$

7.2. Maximizing G-mean with Proxy Labels

Our next experiment borrows the "proxy label" setup from Jiang et al. (2020) on the Adult dataset (Dua & Graff, 2017). The task is to predict whether a candidate's gender is male,

Table 4: Test F-measure for domain shift experiment on Adience.

Cross-entropy [train]	0.760 ± 0.014
Cross-entropy [val]	0.708 ± 0.022
Opt-metric [val]	0.760 ± 0.014
Plug-in [train-val]	0.759 ± 0.014
Importance Weights [KMM]	0.760 ± 0.013
Learn-to-reweight	0.773 ± 0.009
Fine-tuning	0.781 ± 0.014
FW-EG [unknown ψ]	$\mathbf{0.815 \pm 0.013}^{***}$
FW-EG [known ψ]	$\mathbf{0.804 \pm 0.015}^{***}$

but the training set contains only a proxy for the true label. We sample 1% validation data from the original training data, and replace the labels in the remaining sample with the feature 'relationship-husband'. The label noise here is instance-dependent (see Example 3), and we seek to maximize the G-mean metric: $\psi(\mathbf{C}) = \left(\prod_i \left(C_{ii}/\sum_j C_{ij}\right)\right)^{1/m}$.

We train $\widehat{\eta}^{\text{tr}}$ and $\widehat{\eta}^{\text{val}}$ using linear logistic regression using SGD with a learning rate of 0.01. As additional baselines, we include the Adaptive Surrogates method of (Jiang et al., 2020) and *Forward Correction* (Patrini et al., 2017). The inner and outer learning rates for Adaptive Surrogates are each cross-validated in $\{0.1, 1.0\}$. We also compare with a simple Importance Weighting strategy, where we first train a logistic regression model f to predict if an example (x, y) belongs to the validation data, and train a gender classifier with the training examples weighted by $f(x, y)/(1-f(x, y))$.

We choose between three sets of basis functions (using cross-validation): (i) a default basis function $\phi^{\text{def}}(x) = 1\,\forall x$, (ii) $\phi^{\text{def}}, \phi^{\text{pw}}, \phi^{\text{npw}}$, where $\phi^{\text{pw}}(x) = \mathbf{1}(x_{\text{pw}} = 1)$ and $\phi^{\text{npw}}(x) = \mathbf{1}(x_{\text{npw}} = 1)$ use features 'private-workforce' and 'non-private-workforce' to form hard clusters, (iii) ϕ^{def}, $\phi^{\text{pw}}, \phi^{\text{npw}}, \phi^{\text{inc}}$, where $\phi^{\text{inc}}(x) = \mathbf{1}(x_{\text{inc}} = 1)$ uses the binary feature 'income'. These choices are motivated from those used by (Jiang et al., 2020), who compute surrogate losses on the individual clusters. We provide their Adaptive Surrogates method with the same clustering choices.

Table 3 summarizes our results. We apply both variants of our FW-EG method for a non-linear metric ψ, one where ψ is *known* and its gradient is available in closed-form, and the other where ψ is assumed to be *unknown*, and is treated as a general black-box metric. Both variants perform similarly and are better than the baselines. Adaptive Surrogates comes a close second, but underperforms by 0.3% (with results being statistically significant). While the improvement of FW-EG over Adaptive Surrogates is small, the latter is time intensive as, in each iteration, it re-trains a logistic regression model. We verify this empirically in Figure 2(b) by reporting run-times for Adaptive Surrogates and our method FW-EG (including the pre-training time) against the choices of basis functions (clustering features). We see that our approach is $5\times$ faster for this experiment. Lastly, Forward Correction performs poorly, likely because its loss correction is not aligned with this label noise model.

Table 5: Black-box fairness metric on the test set for Adult.

Cross-entropy [train]	0.736 ± 0.005
Cross-entropy [val]	0.610 ± 0.020
Learn-to-reweight	0.729 ± 0.007
Fine-tuning	0.738 ± 0.005
Adaptive Surrogates	0.812 ± 0.004
Plug-in [train-val]	0.812 ± 0.005
FW-EG	$\mathbf{0.822 \pm 0.002}^{***}$

7.3. Maximizing F-measure under Domain Shift

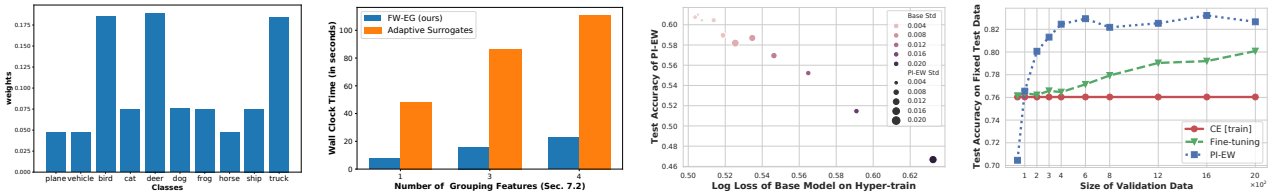
We now move on to a domain shift application (see Example 4). The task is to learn a gender recognizer for the Adience face image dataset (Eidinger et al., 2014), but with the training and test datasets containing images from different age groups (domain shift based on age). We use images belonging to age buckets 1–5 for training (12.2K images), and evaluate on images from age buckets 6–8 (4K images). For the validation set, we sample 20% of the 6–8 age bucket images. Here we aim to maximize the F-measure.

For $\widehat{\eta}^{\text{tr}}$ and $\widehat{\eta}^{\text{val}}$, we use the same ResNet-14 model from the CIFAR-10 experiment, except that the learning rate is divided by 2 after 10 epochs (20 in total). As an additional baseline, we compute importance weights using *Kernel Mean Matching (KMM)* (Huang et al., 2006), and train the same ResNet model with a weighted loss. Since the image size is large for directly applying KMM, we first compute the 2048-dimensional ImageNet embedding (Krizhevsky et al., 2012) for the images and further reduce them to 10-dimensions via UMAP. The KMM weights are learned on the 10-dimensional embedding. For the basis functions, besides the default basis $\phi^{\text{def}}(x) = 1\,\forall x$, we choose from subsets of six RBF basis functions $\phi^1, \ldots, \phi^6$, centered at points from the validation set, each representing one of six age-gender combinations. We use the same UMAP embedding as KMM to compute the RBF kernels.

Table 4 presents the test F-measure values. Both variants of FW-EG algorithm provide statistically significant improvements over the baselines. Both Fine-tuning and Learning-to-reweight improve over plain cross-entropy optimization (train), however only moderately, likely because of the small size of the validation set, and because these methods are not tailored to optimize the F-measure.

7.4. Maximizing Black-box Fairness Metric

We next handle a black-box metric given only query access to its value. We consider a fairness application where the goal is to balance classification performance across multiple protected groups. The groups that one cares about are known, but due to privacy or legal restrictions, the protected attribute for an individual cannot be revealed (Awasthi et al., 2021). Instead, we have access to an oracle that reveals the value of the fairness metric for predictions on a validation sample, with the protected attributes absent from the train-



(a) (b) (c) (d)

Figure 2: (a) Elicited (class) weights for CIFAR-10 by PI-EW for the default basis (Sec. 7.1); (b) Run-time for FW-EG and Adaptive Surrogates (Jiang et al., 2020) vs no. of grouping features on proxy label task (Sec. 7.2); (c) Effect of quality of the base model $\widehat{\eta}^{\text{tr}}$ on Adult (Sec. 7.2): as the base model's quality improves, the test accuracies of PI-EW also improves; (d) Effect of the validation set size on Adience (Sec. 7.3): PI-EW performs better than fine-tuning even for small validation sets, while both improve with larger ones.

ing sample. This setup is different from recent work on learning fair classifiers from incomplete group information (Lahoti et al., 2019; Wang et al., 2020b), in that the focus here is on optimizing *any* given black-box fairness metric.

We use the Adult dataset, and seek to predict whether the candidate's income is greater than \$50K, with *gender* as the protected group. The black-box metric we consider (whose form is unknown to the learner) is the geometric mean of the true-positive (TP) and true-negative (TN) rates, evaluated separately on the male and female examples, which promotes equal performance for both groups and classes:

$$\mathcal{E}^D[h] = \left(\text{TP}^{\text{male}}[h]\,\text{TN}^{\text{male}}[h]\right)\text{TP}^{\text{female}}[h]\,\text{TN}^{\text{female}}[h]\Big)^{1/4}.$$

We train the same logistic regression models as in previous Adult experiment in Section 7.2. Along with the basis functions ϕ^{def}, ϕ^{pw} and ϕ^{npw} we used there, we additionally include two basis ϕ^{hs} and ϕ^{wf} based on features 'relationship-husband' and 'relationship-wife', which we expect to have correlations with gender.[2] We include two baselines that can handle black-box metrics: Plug-in [train-val], which tunes a threshold on $\widehat{\eta}^{\text{tr}}$ by querying the metric on the validation set, and Adaptive Surrogates. The latter is cross-validated on the same set of clustering features (i.e., basis functions in our method) for computing the surrogate losses.

As seen in Table 5, FW-EG yields the highest black-box metric on the test set, Adaptive Surrogates comes in second, and surprisingly the simple plug-in approach fairs better than the other baselines. During cross-validation, we also observed that the performance of FW-EG improves with more basis functions, particularly with the ones that are better correlated with gender. Specifically, FW-EG with basis functions $\{\phi^{\text{def}}, \phi^{\text{pw}}, \phi^{\text{npw}}, \phi^{\text{wf}}, \phi^{\text{hs}}\}$ achieves approximately 1% better performance than both FW-EG with ϕ^{def} basis function and FW-EG with basis functions $\{\phi^{\text{def}}, \phi^{\text{pw}}, \phi^{\text{npw}}\}$.

7.5. Ablation Studies

We close with two sets of experiments. First, we analyze how the performance of PI-EW, while optimizing accuracy for the Adult experiment (Section 7.2), varies with the quality of the base model $\widehat{\eta}^{\text{tr}}$. We save an estimate of $\widehat{\eta}^{\text{tr}}$ after every 50 batches (batch size 32) while training the logistic regression model, and use these estimates as inputs to PI-EW. As shown in Figure 2(c), the test accuracies for PI-EW improves with the quality of $\widehat{\eta}^{\text{tr}}$ (as measured by the log loss on the hyper-train set). This is in accordance with Theorem 2. One can further improve the quality of the estimate η^{tr} by using calibration techniques (Guo et al., 2017), which will likely enhance the performance of PI-EW as well.

Next, we show that PI-EW is robust to changes in the validation set size when trained on the Adience experiment in Section 7.3 to optimize accuracy. We set aside 50% of 6–8 age bucket data for testing, and sample varying sizes of validation data from the rest. As shown in Figure 2(d), PI-EW generally performs better than fine-tuning even for small validation sets, while both improve with larger ones. The only exception is 100-sized validation set (0.8% of training data), where we see overfitting due to small validation size.

8. Conclusion and Discussion

We have proposed the FW-EG method for optimizing black-box metrics given query access to the evaluation metric on a small validation set. Our framework includes common distribution shift settings as special cases, and unlike prior distribution correction strategies, is able to handle general non-linear metrics. A key benefit of our method is that it is agnostic to the choice of $\widehat{\eta}^{\text{tr}}$, and can thus be used to post-shift pre-trained deep networks, without having to re-train them. We showed that the post-shift example weights can be flexibly modeled with various choices of basis functions (e.g., hard clusters, RBF kernels, etc.) and empirically demonstrated their efficacies. We look forward to further improving the results with more nuanced basis functions.

Acknowledgements

We thank the anonymous reviewers for their helpful and constructive feedback. We also thank Google Cloud for supporting this research with cloud computing credits.

[2] The only domain knowledge we use is that the protected group is "gender"; beyond this, the form of the metric is unknown, and importantly, an individual's gender is not available.

References

Awasthi, P., Beutel, A., Kleindessner, M., Morganstern, J., and Wang, X. Evaluating fairness of machine learning models under uncertain and incomplete information. In *FAccT*, 2021.

Byrd, J. and Lipton, Z. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pp. 872–881. PMLR, 2019.

Cotter, A., Gupta, M., and Narasimhan, H. On making stochastic classifiers deterministic. In *Advances in Neural Information Processing Systems*, 2019a.

Cotter, A., Jiang, H., Wang, S., Narayan, T., You, S., Sridharan, K., and Gupta, M. R. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research (JMLR)*, 20(172):1–59, 2019b.

Csurka, G. A comprehensive survey on domain adaptation for visual applications. *Domain adaptation in computer vision applications*, pp. 1–35, 2017.

Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. Multiclass learnability and the erm principle. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 207–232. JMLR Workshop and Conference Proceedings, 2011.

Daniely, A., Sabato, S., Ben-David, S., and Shalev-Shwartz, S. Multiclass learnability and the erm principle. *Journal of Machine Learning Research*, 16:2377–2404, 2015.

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

Demmel, J. W. *Applied numerical linear algebra*. SIAM, 1997.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Eban, E., Schain, M., Mackey, A., Gordon, A., Rifkin, R., and Elidan, G. Scalable learning of non-decomposable objectives. In *Artificial intelligence and statistics*, pp. 832–840. PMLR, 2017.

Eidinger, E., Enbar, R., and Hassner, T. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, 2014.

Fang, T., Lu, N., Niu, G., and Sugiyama, M. Rethinking importance weighting for deep learning under distribution shift. *arXiv preprint arXiv:2006.04662*, 2020.

Frénay, B. and Verleysen, M. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Hiranandani, G., Boodaghians, S., Mehta, R., and Koyejo, O. Performance metric elicitation from pairwise classifier comparisons. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 371–379, 2019a.

Hiranandani, G., Boodaghians, S., Mehta, R., and Koyejo, O. O. Multiclass performance metric elicitation. In *Advances in Neural Information Processing Systems*, pp. 9351–9360, 2019b.

Hiranandani, G., Vijitbenjaronk, W., Koyejo, S., and Jain, P. Optimization and analysis of the pap@ k metric for recommender systems. In *International Conference on Machine Learning*, pp. 4260–4270. PMLR, 2020.

Huang, C., Zhai, S., Talbott, W., Martin, M. B., Sun, S.-Y., Guestrin, C., and Susskind, J. Addressing the loss-metric mismatch with adaptive loss alignment. In *International Conference on Machine Learning*, pp. 2891–2900. PMLR, 2019.

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., and Smola, A. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

Jiang, Q., Adigun, O., Narasimhan, H., Fard, M. M., and Gupta, M. Optimizing black-box metrics with adaptive surrogates. In *ICML*, 2020.

Joachims, T. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pp. 377–384. ACM, 2005.

Kanamori, T., Hido, S., and Sugiyama, M. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.

Kar, P., Narasimhan, H., and Jain, P. Online and stochastic gradient methods for non-decomposable loss functions. *arXiv preprint arXiv:1410.6776*, 2014.

Kar, P., Li, S., Narasimhan, H., Chawla, S., and Sebastiani, F. Online optimization methods for the quantification problem. In *Proceedings of the 22nd ACM SIGKDD*

international conference on knowledge discovery and data mining, pp. 1625–1634, 2016.

Koyejo, O. O., Natarajan, N., Ravikumar, P. K., and Dhillon, I. S. Consistent binary classification with generalized performance metrics. In *NIPS*, pp. 2744–2752, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Lahoti, P., Gummadi, K. P., and Weikum, G. ifair: Learning individually fair data representations for algorithmic decision making. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1334–1345. IEEE, 2019.

Lewis, D. Evaluating and optimizing autonomous text classification systems. In *SIGIR*, 1995.

Lipton, Z., Wang, Y.-X., and Smola, A. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pp. 3122–3130. PMLR, 2018.

McInnes, L., Healy, J., Saul, N., and Großberger, L. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

Menon, A. K., Van Rooyen, B., and Natarajan, N. Learning from binary labels with instance-dependent noise. *Machine Learning*, 107(8-10):1561–1595, 2018.

Narasimhan, H. Learning with complex loss functions and constraints. In *International Conference on Artificial Intelligence and Statistics*, pp. 1646–1654, 2018.

Narasimhan, H., Vaish, R., and Agarwal, S. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Advances in Neural Information Processing Systems*, pp. 1493–1501, 2014.

Narasimhan, H., Kar, P., and Jain, P. Optimizing non-decomposable performance measures: A tale of two classes. In *International Conference on Machine Learning*, pp. 199–208. PMLR, 2015a.

Narasimhan, H., Ramaswamy, H., Saha, A., and Agarwal, S. Consistent multiclass algorithms for complex performance measures. In *ICML*, pp. 2398–2407, 2015b.

Narasimhan, H., Cotter, A., and Gupta, M. Optimizing generalized rate metrics with three players. In *Advances in Neural Information Processing Systems*, pp. 10746–10757, 2019.

Natarajan, B. K. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. *Advances in neural information processing systems*, 26:1196–1204, 2013.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1944–1952, 2017.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pp. 4334–4343. PMLR, 2018.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Stewart, G. W. Perturbation theory for the singular value decomposition. Technical report, 1998.

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., and Kawanabe, M. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

Tamburrelli, G. and Margara, A. Towards automated *A/B* testing. In *International Symposium on Search Based Software Engineering*, pp. 184–198. Springer, 2014.

Wang, J., Liu, Y., and Levy, C. Fair classification with group-dependent label noise. *arXiv preprint arXiv:2011.00379*, 2020a.

Wang, S., Guo, W., Narasimhan, H., Cotter, A., Gupta, M., and Jordan, M. I. Robust optimization for fairness with noisy protected groups. 2020b.

Yan, B., Koyejo, S., Zhong, K., and Ravikumar, P. Binary classification with karmic, threshold-quasi-concave metrics. In *International Conference on Machine Learning*, pp. 5531–5540. PMLR, 2018.

Yang, F., Cisse, M., and Koyejo, S. Fairness with overlapping groups. 2020.

Ye, N., Chai, K. M., Lee, W. S., and Chieu, H. L. Optimizing f-measures: a tale of two approaches. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 289–296. Omnipress, 2012.

Zhao, S., Fard, M. M., Narasimhan, H., and Gupta, M. Metric-optimized example weights. In *International Conference on Machine Learning*, pp. 7533–7542. PMLR, 2019.