Boosted CVaR Classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many modern machine learning tasks require models with high tail performance, i.e. high performance over the worst-off samples in the dataset. This problem has been widely studied in fields such as algorithmic fairness, class imbalance, and risk-sensitive decision making. A common approach to maximize the model's tail performance is by minimizing the CVaR (Conditional Value at Risk) loss, which computes the average risk over the tails of the loss. However, for classification tasks using the zero-one loss, we can show that minimizing the CVaR loss is equivalent to minimizing the average loss for any deterministic classifier, so that it might seem that CVaR is not useful for classification tasks without additional assumptions on the testing distribution. In this work, we circumvent this negative result by minimizing the CVaR loss over randomized classifier, for which the minimizers of the average loss and the CVaR loss are no longer the same. To learn such randomized classifiers, we propose the Boosted CVaR Classification framework which is motivated by a direct relationship between CVaR and LPBoost, and implement a boosting algorithm named α-AdaLPBoost based on this framework. We empirically evaluate the proposed method on four real datasets and find that it can improve the tail performance by a large margin compared to deterministic model training methods.

1 Introduction

As machine learning continues to find broader usage, there is an increasing understanding of the importance of *tail performance* of models in addition to their average performance. For instance, for datasets with highly imbalanced classes, the tail performance is the accuracy over minority classes which have much fewer samples than the others; in the field of algorithmic fairness where a dataset contains several demographic groups, the tail performance is the accuracy over the minority groups which normal machine learning models often neglect. Thus a model with high tail performance does well across all parts of the data domain instead of just on average. Maximizing the tail performance is also sometimes referred to as learning under *subpopulation shift*, in the sense that the testing distribution could consist of just a subpopulation of the training distribution.

Previous work on subpopulation shift fall into two categories. In the first, also referred to as the *domain-aware* setting, it is assumed that the dataset can be divided into predefined groups, and the goal is to maximize the *worst-group accuracy*, that is the minimum accuracy over all the groups. A number of methods, such as static group reweighting [Shi00] and Group DRO [SKHL20, SRKL20], have been proposed for domain-aware subpopulation shift. However, the domain-aware setting is not always applicable, either because groups are hard to define, or because group labels are not available. Thus, in the second category of work on subpopulation shift, also referred to as the *domain-oblivious* setting [HSNL18], there are no pre-defined groups, and the goal is to maximize the model's performance over the worst-off samples in the dataset. Most previous work on domain-oblivious subpopulation shift [HSNL18, DN18, HNSS18, LBC$^+$20, MHN21] measure the tail performance

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

using the Distributionally Robust Optimization (DRO) loss, which is defined as the model's maximum loss over all distributions within a divergence ball around the training distribution. A popular instance is the α-CVaR loss, which is defined as the model's average loss over the worst $\alpha \in (0, 1)$ fraction of the samples incurring the highest losses in the dataset.

It may seem to many that it is a good way to maximize a model's tail performance by directly minimizing the DRO loss or the α-CVaR loss, which has been used by many previous work [HSNL18, DN18, XDKR20]. However, [HNSS18] proved the negative result that for classification tasks, in which models are evaluated by the zero-one loss, empirical risk minimization (ERM) achieves the lowest possible DRO loss given that the model is deterministic and the DRO loss is defined by some f-divergence function. We extend their result to the α-CVaR loss which can be written as the limit of Rényi-divergence DRO losses (with a much more direct and simpler proof due to the specialized case of CVaR). This is a very pessimistic result since it entails that there is no hope to get a better estimator than ERM so long as models are evaluated with respect to a DRO (or CVaR) loss. So some previous work [LBC+20, MHN21] proposed to avoid this issue by making extra assumptions on the testing distribution (specifically, that the worst-off α fraction of the samples can be learned by a parametric model), and changing the evaluation metric correspondingly to something other than a f-divergence DRO loss.

In this work, we take a different approach and show that no extra assumption is needed provided that *we use randomized models*. While the case of general DRO is more complicated, the reason why CVaR loss minimization yields the same solution as ERM is very simple: there exists a linear relationship between the CVaR zero-one loss and the average zero-one loss, so the former monotonically increases with the latter. For randomized models, however, such a monotonic relationship no longer exists. Note that for any single test sample, the zero-one loss of the deterministic model is either 0 or 1, while the expected zero-one loss of the randomized model is a real number in $[0, 1]$, so that the randomized model can typically achieve lower α-CVaR loss than the deterministic model. In fact, we can prove that if the two models have the same average accuracy, then the α-CVaR loss of the randomized model is *consistently* lower than the deterministic one.

Motivated by the above analysis, we propose the framework of Boosted CVaR Classification to train ensemble models via Boosting. We first rewrite the α-CVaR loss in terms of an objective we call α-LPBoost, as a subpopulation-performance counterpart of a classical boosting variant known as LPBoost [DBST02]. Thus training with respect to the α-CVaR loss can be related to a goal of *boosting an unfair learner*, which always produces a model with low average loss on any training distribution, to obtain a fair ensemble classifier with low α-CVaR loss for a fixed α. Note that this is in contrast to the classical boosting, which boosts a weak learner to produce an ensemble model with better average performance. We can thus show that α-CVaR training is equivalent to a sequential min-max game between the training algorithm and an unfair learner, in which our algorithm provides the sample weights and the unfair learner provides base models with low average loss with respect to these weights. After all base models are trained, we compute the optimal model weights. At inference time, we first randomly sample a base model according to the model weights, and then predict with the model. Thus, the final ensemble model is a linear combination of all the base models.

This paper is organized as follows: In Section 2, we provide the necessary background of subpopulation shift and CVaR, and show that CVaR loss minimization for deterministic models is equivalent to ERM in classification tasks. In Section 3 we introduce the framework of Boosted CVaR Classification: We first show that minimizing the CVaR loss is equivalent to maximizing the LPBoost objective in Section 3.1; Based on this result, we propose the Boosted CVaR Classification framework and implement an algorithm that uses LPBoost for CVaR classification in Section 3.2; Then, in order to improve computation efficiency, we implement another algorithm called α-AdaLPBoost in Section 3.3. Finally, in Section 4 we empirically evaluate the proposed method on four real datasets.

1.1 Related Work

Caveats of DRO. DRO was first applied to domain-oblivious subpopulation shift tasks in [HSNL18], in which the authors proved that the DRO loss is an upper bound of the worst-case loss over K groups (group CVaR) for fairness problems. [DN18] analyzed the convergence rate of DRO for the Cressie-Read family of f-divergence. However, as noted in the introduction, [HNSS18] showed that minimizing f-divergence DRO with respect to the zero-one classification loss yields the same minimizer as that of ERM, provided the former has loss less than one. [SKHL20] further note

that for highly flexible classifiers (such as many modern neural models) that achieve zero error on each training sample, both empirical average risks and DRO risks are zero, so that the model is not specifically focusing on population DRO objective. [SRKL20] make the related point that the Group DRO objective with respect to the zero one loss is prone to overfit.

Boosting. Boosting is a classic algorithm in machine learning dating back to AdaBoost proposed in [Sch90]. See [Sch03] for a survey of the early works. Motivated by the success of AdaBoost, many later works proposed variants of Boosting, such as AdaBoost$_v^*$ [RWST05], AdaBoost$_{\ell_1}$ [SL09], LPBoost [DBST02], SoftBoost [RWG07] and entropy regularized LPBoost [WGV08]. There are some previous papers applying ensemble methods to tasks related to subpopulation shift: see [GFB+11] for a survey of Bagging, Boosting and hybrid methods for the class imbalance problem, while [IN19, BHL19] use AdaBoost to improve algorithmic fairness.

2 Preliminaries

In this section we provide the necessary background of subpopulation shift, DRO and CVaR, in the context of classification, which is the focus of the paper.

Let the input space be denoted by $\mathcal{X}$, the label space by $\mathcal{Y}$, and the data domain by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. We are given a dataset $\{z_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n$ that is sampled i.i.d. from some underlying data distribution. We consider the separable setting, so that any input $\boldsymbol{x}$ has only one true label y, i.e. for any $\boldsymbol{x} \in \mathcal{X}$ there exists $y \in \mathcal{Y}$ such that $P(y \mid \boldsymbol{x}) = 1$. Given a family of classifiers $\mathcal{F}$, the goal of a subpopulation shift task is to train a classifier $F : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ with high tail performance. Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, denote the average loss of model F over the dataset by

$$\text{ERM}^\ell(F) = \frac{1}{n} \sum_{i=1}^n \ell(F(\boldsymbol{x}_i), y_i). \tag{1}$$

Empirical risk minimization (ERM) is a training algorithm that minimizes the average loss. Denote the set of minimizers of the average loss by $F^*_{\text{ERM}^\ell} = \arg\min_{F \in \mathcal{F}} \text{ERM}^\ell(F)$. For classification tasks, the model performance is evaluated by the zero-one loss $\ell_{0/1}(\hat{y}, y) = \mathbf{1}_{\{\hat{y} \neq y\}}$ at test time, though we usually choose ℓ to be some surrogate loss function during training.

For a fixed $\alpha \in (0, 1)$, the tail performance of model F is measured by the α-CVaR (Conditional Value at Risk) loss defined as

$$\text{CVaR}_\alpha^\ell(F) = \max_{\boldsymbol{w} \in \Delta_n, \boldsymbol{w} \preccurlyeq \frac{1}{\alpha n}} \sum_{i=1}^n w_i \ell(F(\boldsymbol{x}_i), y_i) \tag{2}$$

where $\Delta_n = \{(x_1, \cdots, x_n) : x_i \leq 0, x_1 + \cdots + x_n = 1\}$ is the unit simplex in $\mathbb{R}^n$. The α-CVaR loss measures how well a model performs over the worst α fraction of the dataset. For instance, if $m = \alpha n$ is an integer, then the α-CVaR loss is the average loss over the m samples that incur the highest losses. We use *CVaR classification* to refer to the family of classification tasks where models are evaluated by the CVaR loss. Denote the set of minimizers of the α-CVaR loss by $F^*_{\text{CVaR}_\alpha^\ell} = \arg\min_{F \in \mathcal{F}} \text{CVaR}_\alpha^\ell(F)$.

The CVaR loss can be written as the limit of DRO (Distributional Robust Optimization) losses. For some divergence function D between distributions, the DRO loss measures the model's performance over the worst-case distribution $Q \ll P$[1] within a ball w.r.t. divergence D around the training distribution P. Formally, the DRO loss of model F is defined as

$$\text{DRO}_{D,\rho}^\ell(F) = \sup_{Q \ll P} \{\mathbb{E}_Q[\ell(F(\boldsymbol{x}), y)] : D(Q \parallel P) \leq \rho\} \tag{3}$$

If we denote the Rényi-divergence by $D_\beta(P \parallel Q) = \frac{1}{\beta - 1} \log \int (\frac{dP}{dQ})^\beta dQ$, then the α-CVAR loss is equal to the limit of $\text{DRO}_{D_\beta, -\log \alpha}$ as $\beta \to \infty$. Many previous work proposed to train a model with high tail performance by minimizing the CVaR loss or the DRO loss. However, the following result

[1] $Q \ll P$ if for any event A, $P(A) = 0 \Rightarrow Q(A) = 0$.

shows that for classification tasks, any model in $F^*_{\text{ERM}^{\ell_{0/1}}}$ is also the minimizer of the CVaR loss if $\mathcal{F}$ only contains deterministic models, i.e. every $F \in \mathcal{F}$ is a deterministic mapping $F : \mathcal{X} \mapsto \mathcal{Y}$ (all proofs can be found in Appendix A):

Proposition 1. *If $\mathcal{F}$ only contains deterministic models, then for any model $F \in \mathcal{F}$ and any $F^* \in F^*_{\text{ERM}^{\ell_{0/1}}}$, $\text{CVaR}_\alpha^{\ell_{0/1}}(F) \geq \text{CVaR}_\alpha^{\ell_{0/1}}(F^*)$. Moreover, if $\min_{F\in\mathcal{F}} \text{ERM}^{\ell_{0/1}}(F) < \alpha$, then we have $F^*_{\text{ERM}^{\ell_{0/1}}} = F^*_{\text{CVaR}_\alpha^{\ell_{0/1}}}$.*

[HNSS18] showed that a counterpart of the above result holds for any f-divergence DRO loss. As noted earlier, we can write the α-CVaR loss as the limit of the Rényi family of f-divergence DRO losses. However our proof is much more direct and simple, and proceeds by showing the following simple relationship between the average zero-one loss and the α-CVaR zero-one loss: $\text{CVaR}_\alpha^{\ell_{0/1}}(F) = \min\left\{1, \frac{1}{\alpha}\text{ERM}^{\ell_{0/1}}(F)\right\}$, so the α-CVaR loss monotonically increases with the ERM loss. This is a very pessimistic result, since it entails that there is no hope to obtain a better model than ERM no matter what learning algorithm we use for CVaR classification.

For the DRO context, some previous papers [HNSS18, LBC+20, MHN21] propose to avoid this issue by making extra assumptions on the testing distribution, so as to change the evaluation metric to some function other than the f-divergence DRO loss. In this work, however, we take a completely different approach: we show that the above difficulty can be circumvented without any extra assumptions by using randomized models[2]. For a randomized model, the monotonic relationship in Proposition 1 does not exist, so that the minimizers of the average loss and the α-CVaR loss are no longer the same. For example, consider two deterministic models, one with average accuracy 90% and one 50%, and every sample is correctly classified by at least one of the models. The 10%-CVaR loss of the 90%-accuracy model is 1. However, the 10%-CVaR loss of the average of the two models is 0.5, while its average accuracy is only 70%. Thus, the randomized model achieves a lower CVaR loss though its average loss is higher. Furthermore, we can prove the following result:

Proposition 2. *Let F be a deterministic model, and F' be any randomized model whose average zero-one loss is the same as that of F. Then, for any $\alpha \in (0, 1)$, $\text{CVaR}_\alpha^{\ell_{0/1}}(F') \leq \text{CVaR}_\alpha^{\ell_{0/1}}(F)$.*

This result implies that the tail performance of a randomized model is consistently higher than a deterministic model with the same average performance. In a nutshell, we have proved that for CVaR classification, ERM is the best deterministic model learning algorithm, and randomized models can achieve higher performance than deterministic models.

3 Boosted CVaR Classification

In this section, we propose the framework of Boosted CVaR Classification, which learns ensemble models with high tail performance via Boosting. An ensemble model consists of T base models $f^1, \cdots, f^T : \mathcal{X} \to \mathcal{Y}$ and a distribution $\boldsymbol{\lambda} = (\lambda^1, \cdots, \lambda^T) \in \Delta_T$ over the models. $\boldsymbol{\lambda}$ is called the model weight vector. At inference time, we first sample a model f^t according to $\boldsymbol{\lambda}$, and then predict with the model. Denote the zero-one loss of base model f^t over sample z_i by ℓ_i^t. Then, the α-CVaR zero-one loss of the ensemble model $F = (f^1, \cdots, f^T, \boldsymbol{\lambda})$ is

$$\text{CVaR}_\alpha^{\ell_{0/1}}(F) = \text{CVaR}_\alpha^{\ell_{0/1}}(f^1, \cdots, f^T, \boldsymbol{\lambda}) = \max_{\boldsymbol{w}\in\Delta_n, \boldsymbol{w}\preccurlyeq\frac{1}{\alpha n}} \sum_{i=1}^n w_i \sum_{t=1}^T \lambda_t \ell_i^t \tag{4}$$

The motivation of this framework comes from a direct relationship between the CVaR loss and the objective of a variant of Boosting we call α-LPBoost, which shows that training with respect to the α-CVaR loss can be related to the goal of *boosting an unfair learner* (Section 3.1). Thus in Section 3.2, we present the Boosted CVaR Classification framework which formulates the training process as a sequential min-max game between the training algorithm and the unfair learner, and implement an algorithm that uses (Regularized) LPBoost for CVaR classification. Finally, in Section 3.3 we implement another algorithm which we name α-AdaLPBoost that is computationally more efficient.

[2] If F is a randomized model, then $P(F(x) = y)$ is a real number in $[0, 1]$ instead of binary.

3.1 α-LPBoost

Suppose we have already obtained t base models $\{f^s\}_{s\in[t]}$, and the s-th function f^s incurs losses $\{\ell_i^s\}_{i\in[n]}$ on the n samples $\{z_i\}_{i\in[n]}$. Consider the following primal/dual linear programs:

Dual:

$$\begin{aligned} \min_{\boldsymbol{w},\gamma} \quad & \gamma \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\ell}^s \rangle \geq 1-\gamma;\ \forall s \in [t] \\ & \boldsymbol{w} \in \Delta_n, \boldsymbol{w} \preccurlyeq \frac{1}{\alpha n} \end{aligned} \tag{5}$$

Primal:

$$\begin{aligned} \max_{\boldsymbol{\lambda},\rho} \quad & \rho - \frac{1}{\alpha n}\sum_{i=1}^{n}(\rho - 1 + \sum_{s=1}^{t}\lambda^s \ell_i^s)_+ \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \Delta_t \end{aligned} \tag{6}$$

where $(x)_+ = \max\{x, 0\}$. Let us denote the optimal dual objective by γ_*^t and the optimal primal objective by ρ_*^t. For this linear program, strong duality holds, i.e. $\rho_*^t = \gamma_*^t$. We refer to these primal/dual linear programs as α-LPBoost, since it can be seen as subpopulation-performance counterpart of a classical variant of Boosting called LPBoost [DBST02].

The following result shows the connection between α-LPBoost and α-CVaR:

Proposition 3. *For any $f^1, \cdots, f^t$, we have the following relationship between the α-LPBoost objective and the α-CVaR zero-one loss:*

$$\rho_*^t = \gamma_*^t = 1 - \min_{\boldsymbol{\lambda}\in\Delta_t} \text{CVaR}_\alpha^{\ell_{0/1}}(f^1, \cdots, f^t, \boldsymbol{\lambda}) \tag{7}$$

This result shows that minimizing the α-CVaR loss is equivalent to maximizing the α-LPBoost objective. Now let us focus on the dual problem of LPBoost. The dual problem solves for a sample weight vector $\boldsymbol{w}$ such that each of $f^1, \cdots, f^t$ achieves at least $1-\gamma$ weighted average loss w.r.t. $\boldsymbol{w}$. If we can train a new model f^{t+1} whose weighted average loss w.r.t. $\boldsymbol{w}$ is small, then γ_*^{t+1} will be greater than γ_*^t, so the ensemble model will perform better with respect to the α-LPBoost objective (and hence α-CVaR).

Motivated by the above analysis, we assume access to an unfair learner $\mathcal{L}$ that takes a sample weight vector $\boldsymbol{w}$ as input and always outputs a base model with weighted average zero-one loss at most $g \in (0,1)$ w.r.t. $\boldsymbol{w}$. g is called the *guarantee* of the learner $\mathcal{L}$. By feeding $\mathcal{L}$ with the $\boldsymbol{w}$ produced by the dual α-LPBoost problem, we can boost this unfair learner to obtain fair ensemble models with low α-CVaR zero-one loss for any fixed α. Moreover, the primal α-LPBoost problem solves for the optimal $\boldsymbol{\lambda}$, so it also provides a way to pick the optimal model weight vector.

3.2 The Boosted CVaR Classification Framework

Now let us introduce the Boosted CVaR Classification framework that formulates the Boosting training process outlined in the previous section as a sequential min-max game between the training algorithm and the unfair learner $\mathcal{L}$. For each round, our algorithm picks a sample weight vector $\boldsymbol{w}^t = (w_1^t, \cdots, w_n^t)$ and feed it to the learner $\mathcal{L}$, which will then output a base model f^t whose average loss w.r.t. $\boldsymbol{w}^t$ is at most g. The training process goes as the following:

- For $t = 1, \cdots, T$,
  - Pick a sample weight vector $\boldsymbol{w}^t = (w_1^t, \cdots, w_n^t) \in \Delta_n$ and feed it to $\mathcal{L}$
  - Receive a base model f^t from $\mathcal{L}$ such that $\sum_{i=1}^n w_i^t \ell_i^t \leq g$
- At the end of training, pick a model weight vector $\boldsymbol{\lambda} \in \Delta_T$ and return $F = (f^1, \cdots, f^T, \boldsymbol{\lambda})$

To study the worst-case performance of our algorithm, we can view $\mathcal{L}$ as an adversary: The algorithm picks $\boldsymbol{w}^t$ and $\boldsymbol{\lambda}$ in order to minimize (4), while $\mathcal{L}$ picks $\boldsymbol{\ell}^t = (\ell_1^t, \cdots, \ell_n^t)$ under the constraint $\langle \boldsymbol{w}^t, \boldsymbol{\ell}^t \rangle \leq g$ in order to maximize (4).

Based on this framework, we implement Algorithm 1, which uses LPBoost for picking sample weights and model weights. For solving linear programs, there are a number of convex optimization solvers available. To show that Algorithm 1 converges, we need to prove that with a sufficiently large T, the α-CVaR loss of the ensemble model can be as close to g as we want[3]. Ideally we would like T

[3]The α-CVaR loss is lower bounded by g, because $\mathcal{L}$ can always output a model whose average loss over the uniform distribution of $z_1, \cdots, z_n$ is at least g, so that the average loss of any ensemble model is at least g.

Algorithm 1 (Regularized) α-LPBoost for CVaR Classification

Input: Density of the test subpopulation α, regularization coefficient β

Initialization: $\boldsymbol{w}^1 = (\frac{1}{n}, \cdots, \frac{1}{n})$
for $t = 1, \cdots, T$ **do**
    Run learner $\mathcal{L}$ with weight vector $\boldsymbol{w}^t$
    $\mathcal{L}$ outputs model f^t with sample losses $\boldsymbol{\ell}^t = (\ell_1^t, \cdots, \ell_n^t)$
    Solve the dual α-LPBoost problem (5) or its regularized version (8) over the training set to get the sample weights $\boldsymbol{w}^{t+1}$
Solve the primal α-LPBoost problem (6) over the validation set to get the optimal $\boldsymbol{\lambda}$
return $F = (f^1, \cdots, f^T, \boldsymbol{\lambda})$

to be in the order of $\log \frac{1}{\alpha}$. However, [RWG07] presented a counterexample in its Theorem 1 where α-LPBoost requires $T = \Omega(\frac{1}{\alpha})$ base models to converge. To make T logarithmic in $\frac{1}{\alpha}$, [WGV08] proposed (Entropy) Regularized α-LPBoost, which adds regularization to the dual problem. At each iteration it solves the following convex problem for some regularization coefficient $\beta > 0$ to pick $\boldsymbol{w}$:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \gamma - \frac{1}{\beta} H(\boldsymbol{w}) \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\ell}^s \rangle \leq 1 - \gamma \ (s \in [t]), \ \boldsymbol{w} \in \Delta_n, \ \boldsymbol{w} \preccurlyeq \frac{1}{\alpha n} \end{aligned} \tag{8}$$

where $H(\boldsymbol{w}) = -\sum_{i=1}^n w_i \log w_i$ is the entropy function. With the regularization, we can prove the following theorem:

Theorem 4 (Theorem 1 in [WGV08])**.** *For any $\delta > 0$, if we run Regularized α-LPBoost with $\beta = \max(\frac{2}{\delta} \log \frac{1}{\alpha}, \frac{1}{2})$, then we have* $\mathrm{CVaR}_\alpha^{\ell_{0/1}}(F) \leq g + \delta$ *with T base models where*

$$T = \max \left\{ \frac{32}{\delta^2} \log \frac{1}{\alpha}, \frac{8}{\delta} \right\} \tag{9}$$

Connection to AdaBoost. [SL09] proved that classical Regularized LPBoost (i.e. α-LPBoost with $\alpha = 1/n$) is equivalent to AdaBoost[FS97] with ℓ_1 regularization (see its Section 3.2).

3.3 α-AdaLPBoost

We have proved that the α-CVaR loss of Regularized α-LPBoost can be arbitrarily close to g with $T = O(\log \frac{1}{\alpha})$. However, one disadvantage of Algorithm 1 is that it needs to train a different set of T base models for every α. It would be more efficient if we could only train one set of T base models for all α, and by adjusting the model weight vector $\boldsymbol{\lambda}$ for different α we could still achieve tail performance comparable to LPBoost.

To this end, we first use the classical regularized LPBoost (such that $\alpha = 1/n$) to pick $\boldsymbol{w}^t$ to train the base models, since the resulting base models would also be suitable for more general $\alpha > 1/n$. Following [SL09], we solve this step using a variant of AdaBoost. We then pick the model weights by solving the α-LPBoost primal problem after all base models are trained. The method, which we denote by α-AdaLPBoost, is listed in Algorithm 2. The difference between Algorithm 2 and the original AdaBoost is that AdaBoost picks $w_i^{t+1} \propto w_i^t \beta_t^{\ell_i^t}$ where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ and $\epsilon_t = \sum_{i=1}^n w_i^t \ell_i^t$ is the weighted average loss, whereas Algorithm 2 picks $w_i^{t+1} \propto w_i^t \exp(\eta \ell_i^t)$ for a constant η, which we find achieves better performance than AdaBoost in our experiments.

The advantage of using our two-step approach is that when α changes, we only need to adjust the model weight vector $\boldsymbol{\lambda}$ without training an entirely new set of base models. We next show that the two-step approach is not just intuitively reasonable, but comes with strong theoretical guarantees. To begin with, we consider AdaBoost + Average, where we train the based models with AdaBoost (as in Algorithm 2) and output the average of the base models (such that $\boldsymbol{\lambda} = (\frac{1}{T}, \cdots, \frac{1}{T})$). For AdaBoost + Average, we have the following result:

Theorem 5. *For any $\delta > 0$, with a proper η, we can make the training α-CVaR zero-one loss of the ensemble model given by AdaBoost + Average at most $g + \delta$ with $T = O(\frac{\log n}{\delta^2})$ base models.*

Algorithm 2 α-AdaLPBoost

Input: Step size η, density of the test subpopulation α

Initialization: $\boldsymbol{w}^1 = (\frac{1}{n}, \cdots, \frac{1}{n})$
for $t = 1, \cdots, T$ **do**
    Run learner $\mathcal{L}$ with weight vector $\boldsymbol{w}^t$
    $\mathcal{L}$ outputs model f^t with sample losses $\boldsymbol{\ell}^t = (\ell_1^t, \cdots, \ell_n^t)$
    Pick $\boldsymbol{w}^{t+1} \in \Delta_n$ such that $w_i^{t+1} \propto \exp(\eta \sum_{s=1}^{t} \ell_i^s)$
Solve the α-LPBoost primal problem (6) over the validation set to get the optimal $\boldsymbol{\lambda}$
return $f^1, \cdots, f^T$ and $\boldsymbol{\lambda}$

The theorem guarantees that AdaBoost + Average can achieve low α-CVaR zero-one loss. Next, note that the α-CVaR zero-one loss of α-AdaLPBoost is upper bounded by the minimum of those of ERM and AdaBoost + Average, because α-LPBoost ensures that the $\boldsymbol{\lambda}$ it picks achieves the lowest α-CVaR zero-one loss, while ERM corresponds to $\boldsymbol{\lambda} = (1, 0, \cdots, 0)$ and AdaBoost + Average corresponds to $\boldsymbol{\lambda} = (\frac{1}{T}, \frac{1}{T}, \cdots, \frac{1}{T})$. In other words, the theorem ensures that the tail performance of AdaBoost + Average is high, and α-AdaLPBoost can achieve even higher tail performance.

Connection to (Domain-Aware) Group DRO. Our algorithm has a strong connection to Group DRO proposed in [SKHL20] for domain-aware subpopulation tasks. Group DRO assumes that the dataset is divided into K groups and minimizes the model's maximum loss over the K groups with AdaBoost. We can obtain Group DRO from AdaBoost + Average by choosing $n = K$, $m = 1$ and defining ℓ_i^t as the loss of model f^t over group i. Our algorithm suggests that we can improve Group DRO if we choose the model weights with LPBoost.

4 Experiments

4.1 Setup

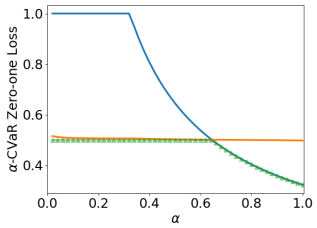
Datasets. We conduct our experiments on four datasets: COMPAS [LMKA16], CelebA [LLWT15], Cifar-10 and Cifar-100 [KH+09]. The first, COMPAS is a tabular recidivism dataset widely used in fairness research. The second, CelebA, with the target label as whether a person's hair is blond or not, was used in [SKHL20] to evaluate their proposed Group DRO algorithm; we too adopt this task in our experiments. And finally, Cifar-10 and Cifar-100 are two widely used image datasets. Both COMPAS and CelebA have binary classification tasks, while Cifar-10 is 10-class and Cifar-100 is 100-class classification. On COMPAS we use the training set as the validation set because the dataset is very small. CelebA has its official train-validation split. On Cifar-10 and Cifar-100 we take out 10% of the training samples and use them for validation. Our selection of datasets covers different types, complexity, and both binary and multi-class classification.

Learner $\mathcal{L}$. We implement the unfair learner $\mathcal{L}$ as follows: given a sample weight vector $\boldsymbol{w} = (w_1, \cdots, w_n) \in \Delta_n$, for each iteration we sample a minibatch with replacement according to the probability distribution $\boldsymbol{w}$, and then update the model parameters with ERM over the minibatch. The learner stops and returns the model after a fixed number of iterations, with the learning rate decayed twice during training. See our code for training details.

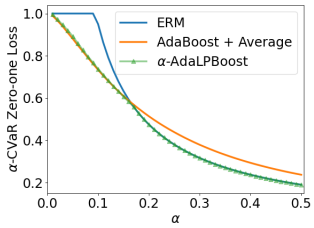
Training. We use a three-layer feed-forward neural network with ReLU activations on COMPAS, a ResNet-18 [HZRS16] on CelebA, a WRN-28-1 [ZK16] on Cifar-10 and a WRN-28-10 on Cifar-100. On each dataset, we first warmup the model with a few epochs of ERM, and then train $T = 100$ base models from the warmup model with the sample weights given by the boosting algorithms. We train our models with CPU on COMPAS and with one Nvidia GTX 1080ti on other datasets. We solve linear programs with the CVXPY package [DB16, AVDB18], which at its core invokes MOSEK [ApS21] and ECOS [DCB13] for optimization.[4]

On each dataset, we run ERM and α-AdaLPBoost with different values of α. Here ERM refers to the deterministic version, in which a single base model is trained and used as the final model. We

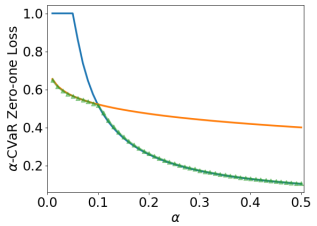
[4] Please refer to their official websites for license information.



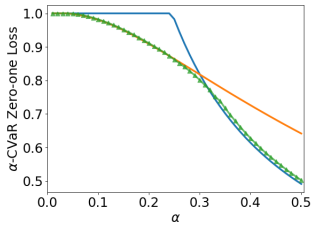
(a) COMPAS



(b) Cifar-10

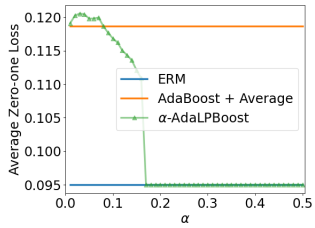


(c) CelebA

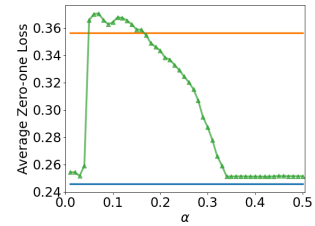


(d) Cifar-100

Figure 1: Test α-CVaR zero-one loss.



(a) Cifar-10



(b) Cifar-100

Figure 2: Test average loss.

also compare α-AdaLPBoost with AdaBoost + Average in order to demonstrate the effectiveness of selecting $\boldsymbol{\lambda}$ with LPBoost. To simultaneously compare the performances under different α, we plot the α-CVaR zero-one loss vs α curve for each method on every dataset.

4.2 Results

We plot the experimental results in Figure 1. It is remarkable that on every dataset, the α-CVaR loss curve of α-AdaLPBoost almost overlaps with the maximum of ERM and AdaBoost + Average. From the plots we can make the following observations:

- When α is large, the α-CVaR loss is close to the average loss, and we can see that the α-CVaR loss of α-AdaLPBoost is very close to that of ERM. We also plot the test average loss of the three methods on Cifar-10 and Cifar-100 in Figure 2, from which we can see that there is a gap between the ERM line and AdaBoost + Average line, and the average loss curve of α-AdaLPBoost mostly lies between them.
- When α is small, the α-CVaR loss of α-AdaLPBoost is close to that of AdaBoost + Average, which is much lower than that of ERM. Recall our initial theoretical results on the equivalence of minimizers of the average loss and the CVaR loss for deterministic models, ERM achieves the lowest α-CVaR loss among all deterministic models, so the results demonstrate that the ensemble model achieves higher tail performance than any deterministic model.

Overall, we find that α-AdaLPBoost can automatically choose between high average performance (when α is big) and high tail performance (when α is small).

One might ask why α-AdaLPBoost does not achieve lower average loss than ERM in our experiments, given that the model class of the ensemble models is larger than that of the deterministic models. The reason is that the model class we use for deterministic models (e.g. ResNet) is already very complex, so the average performance of ERM is high enough, and its loss mainly comes from the generalization gap instead of insufficient capacity of representation. In fact, we find that when α is big, the model weight vector produced by LPBoost is very close to $(1, 0, 0, \cdots, 0)$, i.e. almost all the weight is put on the first ERM model. That is why the α-AdaLPBoost curve and the ERM curve overlap when α is big.

Comparison with Regularized LPBoost. Now we empirically show that α-AdaLPBoost can achieve performance comparable to Regularized LPBoost. In Figure 3 we plot the α-CVaR loss of

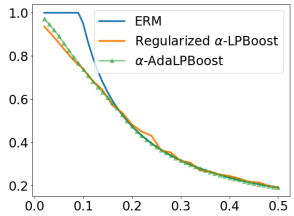
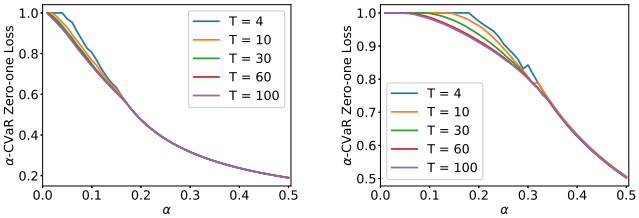


Figure 3: Comparison between α-AdaLPBoost and Regularized α-LPBoost on Cifar-10.



(a) Cifar-10

(b) Cifar-100

Figure 4: Test α-CVaR zero-one loss of α-AdaLPBoost under different values of T.

ERM, α-AdaLPBoost and Regularized LPBoost ($\beta = 100$) on Cifar-10. The plot clearly shows that there is almost no gap between the performance of α-AdaLPBoost and Regularized LPBoost, so we can safely replace the latter with the former for computational efficiency.

Convergence. Finally, we empirically examine the convergence rate of α-AdaLPBoost by studying how the α-CVaR loss of the ensemble model changes with T. In Figure 4 we plot the results on Cifar-10 and Cifar-100. The figure shows that α-AdaLPBoost has almost converged with $T = 30$ base models on both datasets. Note that this does not mean that the training time is 30 times more, because we can train each base model for fewer iterations since it is initialized from a warmup model and only needs finetune. In our experiments, the training time of AdaBoost is 3-6 times the training time of ERM if $T = 30$.

5 Conclusion

In this work, we addressed an issue raised by previous work that no deterministic model learning algorithm could be better than ERM for DRO classification (which we show formally also extends to CVaR) by learning randomized models. Specifically we proposed the Boosted CVaR Classification framework which was motivated by the direct relationship between α-CVaR and α-LPBoost, which is a sub-population variant of the classical LPBoost algorithm for classification. To further improve the computational efficiency, we implemented the α-AdaLPBoost algorithm. In our experiments, we showed that the ensemble models produced by α-AdaLPBoost achieved higher tail performance than deterministic models, and the algorithm could automatically choose between high average performance and high tail performance under different values of α.

One future direction is the application of ensemble methods to the fairness without demographics problem, in which the dataset is divided into several groups which are unknown during training, and the goal is to minimize the model's worst-group loss, i.e. its maximum loss over all the groups. The worst-group loss can be upper bounded by the CVaR loss or certain families of DRO loss, but the bound is loose, and the model with the lowest CVaR loss is not guaranteed to achieve the lowest worst-group loss. Due to the difficulty of the problem, some recent work considered the scenario where a small set of samples with group labels is provided after training and before testing. Ensemble methods can be useful in this scenario: we can train T base models, and then solve a linear program to obtain the optimal model weight vector with the provided validation set with no need of training new models. We leave the design of such algorithms to future work.

Social Impact. Subpopulation shift has been widely studied in order to improve the fairness of machine learning, which is of great social importance. Models with high tail performance are considered fair because they perform well on all parts of the data domain. In this work we show how to improve the tail performance by learning ensemble models, which is a great contribution to the area of fair machine learning. However, we also observe that ensemble methods improve the tail performance by lowering the accuracy over samples in the majority group. Such trade-offs are nonetheless inevitable under the assumption that the average accuracy does not increase. It is an interesting sociological question to what extent it is just to improve fairness by sacrificing the majority group.

References

[ApS21] MOSEK ApS. *MOSEK Optimizer API for Python. Version 9.2.44*, 2021.

[AVDB18] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[BHL19] Dheeraj Bhaskaruni, Hui Hu, and Chao Lan. Improving prediction fairness via model ensemble. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1810–1814, 2019.

[DB16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DBST02] Ayhan Demiriz, Kristin P Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254, 2002.

[DCB13] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[DN18] John Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization. *arXiv preprint arXiv:1810.08750*, 2018.

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[GFB+11] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[HNSS18] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR, 2018.

[HSNL18] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In Jennifer Dy and Andreas Krause, editors, *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1929–1938, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IN19] Vasileios Iosifidis and Eirini Ntoutsi. Adafair: Cumulative fairness adaptive boosting. *CoRR*, abs/1909.08982, 2019.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[LBC+20] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed Chi. Fairness without demographics through adversarially reweighted learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[LMKA16] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9(1), 2016.

[MHN21] Paul Michel, Tatsunori Hashimoto, and Graham Neubig. Modeling the second player in distributionally robust optimization. In *International Conference on Learning Representations*, 2021.

[RWG07] Gunnar Rätsch, Manfred K Warmuth, and Karen A Glocer. Boosting algorithms for maximizing the soft margin. *Advances in neural information processing systems*, 20:1585–1592, 2007.

[RWST05] Gunnar Rätsch, Manfred K Warmuth, and John Shawe-Taylor. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(12), 2005.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[Sch03] Robert E Schapire. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pages 149–171, 2003.

[Shi00] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[SKHL20] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020.

[SL09] Chunhua Shen and Hanxi Li. A duality view of boosting algorithms. *CoRR*, abs/0901.3590, 2009.

[SRKL20] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8346–8356. PMLR, 13–18 Jul 2020.

[WGV08] Manfred K Warmuth, Karen A Glocer, and SVN Vishwanathan. Entropy regularized lpboost. In *International Conference on Algorithmic Learning Theory*, pages 256–271. Springer, 2008.

[XDKR20] Ziyu Xu, Chen Dan, Justin Khim, and Pradeep Ravikumar. Class-weighted classification: Trade-offs and robust approaches. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10544–10554. PMLR, 13–18 Jul 2020.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 1.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]