Efficient Generalization with Distributionally Robust Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Distributionally robust learning (DRL) is increasingly seen as a viable method to train machine learning models for improved model generalization. These min-max formulations, however, are more difficult to solve. We provide a new stochastic gradient descent algorithm to efficiently solve this DRL formulation. Our approach applies gradient descent to the outer minimization formulation and estimates the gradient of the inner maximization based on a sample average approximation. The latter uses a subset of the data sampled without replacement in each iteration, progressively increasing the subset size to ensure convergence. We rigorously establish convergence to a near-optimal solution under standard regularity assumptions and, for strongly convex losses, match the best known $O(\epsilon^{-1})$ rate of convergence up to a known threshold. Empirical results demonstrate the significant benefits of our approach over previous work in improving learning for model generalization.

1 Introduction

Consider a general formulation of the distributionally robust learning (DRL) problem of active interest. Let $\mathbb{X}$ denote a sample space, P denote a probability distribution on $\mathbb{X}$, and $\Theta \subseteq \mathbb{R}^d$ denote a parameter space. Let us define $L_P(\theta) := \mathbb{E}_P[l(\theta, \xi)]$ to be the expectation with respect to (w.r.t.) P of a loss function $l : \Theta \times \mathbb{X} \to \mathbb{R}$ representing the estimation error for a learning model with parameters $\theta \in \Theta$ over data $\xi \in \mathbb{X}$. Further define the worst-case expected loss function $R(\theta) := \mathbb{E}_{P^*(\theta)}[l(\theta, \xi)] = \sup_{P \in \mathcal{P}}\{L_P(\theta)\}$, which maximizes the loss L_P over a well-defined set of measures $\mathcal{P}$. Letting P_b denote a base distribution, this set often takes the form $\mathcal{P} = \{P \,|\, D(P, P_b) \leq \rho, \int dP(\xi) = 1, P(\xi) \geq 0\}$ where $D(\cdot, \cdot)$ is a distance function on the space of probability distributions on $\mathbb{X}$ and the constraints limit the feasible candidates to be within a distance of ρ from P_b. We seek to find parameters $\theta \in \Theta$ that solve the DRL problem formulated as

$$R(\theta^*_{rob}) \;=\; \min_{\theta \in \Theta} \Big\{ R(\theta) \Big\} \;=\; \min_{\theta \in \Theta} \Big\{ \sup_{P \in \mathcal{P}} \{L_P(\theta)\} \Big\}, \tag{1}$$

for a given $\mathbb{X}$ and $\mathcal{P}$. In practice, training with DRL amounts to dynamically reweighing the data using $P^*(\theta)$ as the parameters θ range over the space Θ. The inner maximization problem sets these weights to emphasize data that experience high loss at θ. This reweighing approach arises from solid theoretical foundations and can provide strong guarantees on model generalization.

DRL and Model Generalization: In machine learning, optimal values for the model parameters θ are calculated from a finite training dataset (of size N) and this model is then used for inference over other test datasets, all of which are typically assumed to be identically distributed [33, 34]. The equal-weight *empirical* distribution $U_N = \{1/N\}$ over the finite training dataset is the non-parametric maximum likelihood estimator [23] of the (unknown) distribution underlying the datasets. Let θ^*_{erm} denote the minimizer of the empirical loss $L_{U_N}(\cdot)$ over Θ. In real-world applications, any

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

two finite datasets sampled from the same underlying distribution can violate the identical distribution assumption, leading to poor *generalization* when using θ^*_{erm} over other datasets [25]. Popular model selection techniques, such as cross-validation [32], seek to improve the estimation error between training and testing datasets, but they can be computationally prohibitive and lack rigorous guarantees.

With roots in non-parametric statistics [23] and optimization [28, 7], several studies [20, 2, 7, 21, 19] have proposed as an alternative approach the DRL formulation (1) using the empirical distribution U_N over the finite training dataset as the base distribution P_b. This alternative approach explicitly treats the ambiguity in the identity of the true (stationary but unknown) distribution, denoted by P_0. In general, we know that U_N is not equal to P_0 and it is highly likely that, at θ^*_{rob}, the worst-case distribution $P^*(\theta^*_{rob})$ is not equal to P_0. However, for Wasserstein distance metrics and an appropriately chosen value of ρ in the set of measures $\mathcal{P}$, Blanchet et al. [2] show that P_0 is in $\mathcal{P}$ with high probability. Moreover, a broad guideline is provided in [2, 21] such that these high probability guarantees are achieved by setting $\rho = O(\sqrt{d/N})$ for binary classification with logistic models. The DRL approach thus holds great promise and a general theory is actively being pursued.

Goal: Our main objective is to solve the DRL problem (1) in an efficient manner to ensure it is a viable alternative approach for model generalization. The primary difficulty is the min-max formulation, particularly the inner maximization problem. While its solution may be explicitly available in some cases – e.g., constraining $\mathcal{P}$ by certain Wasserstein distance metrics admits an explicit characterization of the robust objective function $\mathbb{E}_{P^*(\theta)}[l(\theta, \xi)]$ [2, 30, 8, 4, 5] – these reductions do not hold in general for all interesting Wasserstein distance metrics and they require solving a convex non-linear program [5]. Namkoong and Duchi [21] show that the inner maximization can be efficiently solved (see Section 2) under χ^2-divergence constraints. Hence, we focus on the entire general class of ϕ-divergence distance functions: $D_\phi(P, P_b) = \mathbb{E}_{P_b}[\phi(\frac{dP}{dP_b})]$, where $\phi(s)$ is a non-negative convex function having the value 0 only at $s = 1$. Members of this class include the modified χ^2 divergence, with $\phi(s) = (s-1)^2$, and the Kullback-Leibler (KL) divergence, with $\phi(s) = s \log s - s + 1$. Defining the vector $P := (p_n)$ of dimension N and setting the base P_b to the uniform empirical distribution U_N, we then have that the loss function and constraint set $\mathcal{P}$ are given by $L_P(\theta) = \sum_{n=1}^N p_n l(\theta, \xi_n)$ and $\mathcal{P} = \{P \mid \sum_{n=1}^N p_n = 1, p_n \geq 0, \forall n, D_\phi(P, U_N) = \frac{1}{N}\sum_{n=1}^N \phi(Np_n) \leq \rho\}$.

Related Work: Assuming convex loss functions $l(\cdot, \xi_n)$, the DRL problem (1) is convex in θ. Ben-Tal et al. [1] consider the typical Lagrangian dual algorithm used for this convex-concave case; see (5) in the supplement. Such a reformulation renders a standard stochastic optimization problem, which they solve by applying classical stochastic gradient descent (SGD) methods. However, when l is non-convex, this approach cannot guarantee convergence to global optima. Moreover, Namkoong and Duchi [20] observe that the characteristics of certain dual variables can cause instability in SGD.

Namkoong and Duchi [20] therefore propose an alternative approach, for convex losses $l(\theta, \xi)$, that interleaves stochastic mirror-descent steps in each of the θ and P variables. This requires randomized coordinate descent in P along coordinates sampled from the (non-uniform) probability mass function (pmf) P iterates. Such sampling is computationally demanding, where the fastest methods to sample an N-sized non-uniform pmf require $O(N)$ computational effort, which is only slightly smaller than the effort needed to solve the inner maximization problem *exactly* (see below). Coupled with the iterations in a composite dimension $d + N$, this results in slow convergence, as observed in our empirical results in the supplement. To rectify this, Namkoong and Duchi [21] propose to directly obtain the optimal $P^*(\theta)$ that defines $R(\theta)$, namely solving (1) as a large deterministic problem that is well-defined for finite N. For χ^2-divergence, they show that the inner maximization can be reduced to two one-dimensional root-finding problems, which can be solved via bisection search, thus requiring an $O(N \log N)$ computational effort (see Proposition 2) at each iteration which can be expensive for large datasets. They also establish a guarantee that the deterministic algorithm for convex losses l converges to an ϵ-optimal solution with $O(N\epsilon^{-2})$ computational effort.

Recent work [15, 10] seeks to reduce the computational effort of this costly full-gradient method by developing an algorithm to efficiently estimate the gradient of the inner maximization problem through a multi-level Monte Carlo randomization procedure introduced by Giles [11]. Although there are important differences in the theoretical arguments supporting the algorithm and its parameters, in theory the approach holds great promise especially for large datasets; in fact, a computational effort complexity of $O(\epsilon^{-3})$ is established for convex losses l in [15]. Despite this, our experiments expose two adverse factors. Each iteration is more expensive because it solves multiple inner maximization problems. At the same time, the algorithm inherently produces an increase in the variance of the

gradient estimator of the robust loss as a result of the added randomization of the Giles estimator; this can negatively impact both the computational effort and solution quality of the overall approach.

Our Contributions: We consider in Section 2 a new SGD algorithm tailored to efficiently solving large DRL problems. Our Algorithm 1 significantly reduces the expense of solving the inner maximization over the full training dataset by *subsampling* the support of P from the (finite) training dataset in the iterates of the algorithm and estimating the robust loss via a sample average approximation. Subsampling (mini-batching) typically works well with SGD because the gradient estimates are unbiased. However, in the DRL context, our Theorem 3 shows that subsampling produces a bias in the estimation of the robust loss gradient because, with fixed mini-batch subsampling, chances are high that critical data that suffer high loss will be missed, leading to an optimistic estimation of the robust loss; empirical results (in the supplement) illustrate the important impact of this bias. Namkoong and Duchi [21] bypass this issue by assembling the full-data gradient, whereas the Giles estimator approach [15, 10] randomizes the choice of mini-batch size over the entire dataset. Although Levy et al. [15] provide an analysis of the bias induced by mini-batching, our algorithm is fundamentally different in that we assemble the mini-batch by sampling *without replacement* from the training dataset, whereas sampling *with replacement* is employed in [20, 21, 15]. While our theory is harder to develop because the individual samples in the mini-batch are not distributionally independent, the estimation is significantly improved for the same computational effort by only considering unique samples ξ and their losses $l(\theta, \xi)$ in approximating $R(\theta)$. Indeed, an M-sized set sampled with replacement has on average only about $N(1 - e^{-M/N})$ unique support points (see supplement); this issue leads Levy et al. [15] to show that the variance of their $R(\theta)$ estimator may not vanish with increasing batch size, thus contributing to their worse computational effort complexity.

Algorithm 1 reduces the identified inherent bias by progressively increasing the subsample size with the iterates up to the maximum size N. We establish in Theorem 4 that convergence to a close neighborhood of a local optimal solution is assured, even for non-convex losses l, provided that the subsamples increase at a certain minimal rate. We further extend our analysis for strongly convex losses l in Theorem 6 to show that the algorithm takes $O(\epsilon^{-1})$ effort to reduce the optimality gap to ϵ, matching the best SGD rates known for stochastic optimization [22]. The dependence on N is via poly-log terms. This holds for all $\epsilon \geq \bar{E}^* = O(N^{-(1-\delta)})$ where δ is a small positive value. Hence, we match the best rates of standard SGD w.r.t. ϵ, like [21] (without their $O(N)$ dependence), with only poly-log N dependence, like [15] (without their worse ϵ order), for strongly convex losses. Additional theoretical results and technical details, and all proofs can be found in the supplement.

Our empirical results in Section 3 consider convex DRL formulations of binary classification problems. We compare the computation and generalization performance of our algorithm with the corresponding performance under the algorithms proposed in [20, 21, 15, 10] and under a regularized ERM formulation tuned via k-fold cross validation (CV), the standard practical approach for model generalization. Our results show that Algorithm 1 attains equivalent or (often) superior generalization performance as k-fold CV and all other DRL methods. Meanwhile, our algorithm provides significant improvements in the computational effort required by the other DRL methods and orders of magnitude reductions in the computational effort required by k-fold CV. Furthermore, the key parameters of our DRL algorithm do not require fine tuning and are set based on our theoretical results, whereas the key parameters of all other DRL methods require fine tuning while also impacting both the solution quality and computation effort of the Giles approach. Additional empirical results, including the impact of key algorithmic parameters, and related technical details can be found in the supplement.

2 Algorithm and Analysis

Algorithm 1 presents our progressively sampled subgradient descent algorithm that follows SGD-like iterations for the outer minimization problem in (1) according to

$$\theta_{t+1} \;=\; \theta_t - \gamma_t \nabla_\theta \hat{R}_t(\theta_t) \;=\; \theta_t - \gamma_t G_t, \tag{2}$$

where γ_t is the step size (or gain sequence or learning rate), $\hat{R}_t(\cdot)$ is a sample-average approximation of the robust loss $R(\cdot)$ from the inner maximization formulation over D_ϕ-constrained $\mathcal{P}$, and $G_t := \nabla_\theta \hat{R}_t(\theta_t)$. Our algorithm uses progressively increasing sample sizes M_t to estimate G_t. This view of (1) permits us to depart from the convex-concave formulations of Ben-Tal et al. [1] and consider non-convex losses l, as long as the subgradient $\nabla_\theta \hat{R}_t(\cdot)$ approximates the gradient $\nabla_\theta R(\cdot)$ sufficiently

well. We first establish that the gradient $\nabla_\theta R(\cdot)$ exists, recalling that $R(\theta)$ is the optimal value of the inner maximization problem. Let us define the set $\Theta_\varnothing := \{\theta : l(\theta, \xi_{n_1}) = l(\theta, \xi_{n_2}),\ \forall n_1, n_2\}$, and for a small $\varsigma > 0$ define the set $\Theta_{\varnothing,\varsigma} := \cup_{\theta_o \in \Theta_\varnothing} \{\theta : |\theta - \theta_o| < \varsigma\}$ to be the ς-neighborhood of $\Theta_\varnothing$. We assume in Proposition 1 that the learning model precludes this neighborhood set in order to avoid a degenerate case in which the inner maximization objective function does not depend on the decision variables p_n and the entire feasible set is optimal. The existence and form of $\nabla_\theta R(\theta)$ is then derived in part by exploiting Danskin's Theorem [29, Theorem 7.21].

Proposition 1. *Let the feasible region Θ be compact and assume $\Theta \subseteq \Theta^c_{\varnothing,\varsigma}$, for a small $\varsigma > 0$. Further suppose ϕ in the D_ϕ-constraint has strictly convex level sets, and let $\rho < \bar{\rho}(N, \phi)$ with $\bar{\rho}(N, \phi)$ defined in the supplement. Then: (i) the optimal solution P^* of $R(\theta) = \sup_{P \in \mathcal{P}}\{L_P(\theta)\}$ is unique, and the gradient is given by $\nabla_\theta R(\theta) = \sum_{n \in \mathcal{N}} p^*_n(\theta)\, \nabla_\theta l(\theta, \xi_n)$; and (ii) for all ρ, the gradient $\nabla_\theta R(\theta)$ is a sub-gradient of $R(\theta)$.*

Algorithm 1 Progressively Sampled Subgradient Descent

Given: Step size γ; Sample size sequence $\{M_t\}_{t=1}^{\mathcal{T}}$, $M_{\mathcal{T}} = N$; Initial iterate θ_0.

```
for t = 1, 2, ..., T do
    Sample M_t indices without replacement uniformly from {1, ..., N}, and gather them in M_t
    Solve inner maximization problem to obtain optimal solution P*_t        {use Procedure 1}
    Set G_t ← Σ_{m∈M_t} p*_{t,m} ∇_θ l(θ_t, ξ_m)
    Set θ_{t+1} ← θ_t − γ G_t                                                {subgradient descent step}
    Increment t ← t + 1
end for
```

Next, we construct the estimate $\hat{R}_t$ from the inner maximization problem restricted only to a relatively small subset $\mathcal{M}_t$ of the full dataset with size $|\mathcal{M}_t| = M_t$. Define $P := (p_m)$ of dimension M_t and define the objective coefficients $z_m := l(\theta, \xi_m)$. We then have

$$\hat{R}_t(\theta) \;=\; \max_{P=(p_m)} \sum_{m \in \mathcal{M}_t} p_m z_m \qquad \text{s.t.} \sum_{m \in \mathcal{M}_t} \phi(M_t p_m) \le M_t \rho_t, \sum_{m \in \mathcal{M}_t} p_m = 1,\ p_m \ge 0. \quad (3)$$

The uncertainty radius $\rho_t = \rho + \eta_t$ now changes with the iteration t where η_t represents a small additional inflation; Theorem 3 below motivates why this is used. Now suppose $P^*_t(\theta) = (p^*_{t,m}(\theta))$ is an optimal solution to (3). Then a valid subgradient for $\hat{R}_t(\theta_t)$ is obtained as an expression analogous to that in Proposition 1(i) under appropriate substitutions w.r.t. θ_t, P^*_t and $\mathcal{M}_t$. Writing the Lagrangian objective of (3) as $\mathcal{L}(\alpha, \lambda, P)$, with an explicit expression given in the supplement, we then have the optimal objective value $\hat{R}^*_t(\theta) = \min_{\alpha \ge 0, \lambda} \max_{\hat{p}_m \ge 0} \mathcal{L}(\alpha, \lambda, P)$; refer to [18]. The optimal primal and dual variables can then be obtained for various ϕ functions by the general **Procedure 1** in the supplement to solve the Lagrangian formulations. Importantly, a worst-case bound on the computational effort required to obtain an ϵ-optimal solution to (3) can be obtained.

Proposition 2. *For any ϕ-divergence, **Procedure 1** finds a feasible primal-dual solution $(\tilde{P}^*_t, \tilde{\alpha}^*, \tilde{\lambda}^*)$ to (3) with an objective function value $\tilde{R}^*_t$ such that $|\hat{R}^*_t(\theta) - \tilde{R}^*_t| < \epsilon$ with a worst-case computational effort bounded by $O(M_t \log M_t + (\log \frac{1}{\epsilon})^2)$, where ϵ is a small precision parameter.*

The machine-precision ϵ does not relate to any formulation or algorithm parameters (e.g., M_t, N, ρ), and it is required because **Procedure 1** solves two one-dimension bisection searches in sequence. In the sequel we assume that ϵ is a fixed small value and **Procedure 1** returns the exact unique solution $(P^*_t, \alpha^*, \lambda^*)$ to problem (3), and that the computational effort is bounded by $O(M_t \log M_t)$.

2.1 Small-Sample Approximation of $\nabla_\theta R(\theta)$

In Algorithm 1, the gradient $\nabla_\theta R$ is approximated using the gradient expression in Proposition 1(ii) with the optimal P^*_t for a small sample size M_t. This estimate is expected to suffer a bias because a subsample of the full dataset might miss data where loss is high when subsampling the dataset, and this leads to optimistic estimation of the robust loss $R(\theta)$ and its gradient. Let the mass vector $P^* = (p^*_1, \ldots, p^*_N)$ be the optimal solution to the full-data version of (3), i.e., with $M_t = N$, and let the mass vector $P^*_t = (p^*_1, \ldots, p^*_{M_t})$ be the optimal solution when restricted to any subset $\mathcal{M}_t$.

Assumption 1. *The ϕ-divergence satisfies uniformly for all s and $\zeta < \zeta_0$ the continuity condition (for constants $\zeta_0, \kappa_1, \kappa_2 > 0$): $|\phi(s(1+\zeta)) - \phi(s)| \le \kappa_1 \zeta \phi(s) + \kappa_2 \zeta$.*

The above condition, as described in [29], only allows for (local) linear growth in ϕ, and it can be verified for many common ϕ-divergences of interest including the modified χ^2- and KL-divergences. Let $\mathbb{E}_t$ and $\mathbb{P}_t$ respectively denote expectation and probability w.r.t. the random set $\mathcal{M}_t$.

Theorem 3. *Suppose Assumption 1 and the assumptions of Proposition 1 hold, and further suppose the gradient $\nabla \hat{R}_t(\theta)$ is the optimal solution to* (3) *over a subsample $\mathcal{M}_t$ of size M_t sampled uniformly without replacement from a set of size N. Define $\eta_t = c(\frac{1}{M_t} - \frac{1}{N})^{(1-\delta)/2}$ for small constants $c, \delta > 0$, and set the D_ϕ-target in* (3) *to be $\rho_t = \rho + \eta_t$. Then, there exists a small positive M' defined in the supplement and of order $o(N)$ such that, for all $M_t \geq M'$, the subgradient $\nabla_\theta \hat{R}_t(\theta)$ and full-gradient $\nabla_\theta R(\theta)$ satisfy for any $C < \infty$ and $1 - \bar{\tau}_t = O(\eta_t^{2\delta/(1-\delta)})$:*
$\mathbb{P}_t(\eta_t^{-2} \|\nabla_\theta \hat{R}_t(\theta) - \nabla_\theta R(\theta)\|_2^2 \leq C) \geq \bar{\tau}_t$.

Corollary 1. *If the conditions for Theorem 3 are satisfied, then $\|\mathbb{E}_t[\nabla_\theta \hat{R}_t(\theta)] - \nabla_\theta R(\theta)\|_2^2 = O(\eta_t^2)$.*

The proof of Theorem 3 (in the supplement) includes exact expressions for the corresponding constants such as $\bar{\tau}_t$, whose dependence on the constant C and on the magnitude of $\nabla_\theta R(\theta)$ is made explicit in Lemma 9. Theorem 3 provides a bound on the squared bias in approximating the true gradient $\nabla_\theta R(\theta)$ with the subsample estimate $\nabla_\theta \hat{R}(\theta)$ as a function of the sample size M_t. Levy et al. [15] describe an analogous bias result for the case when the subsample $\mathcal{M}_t$ is gathered by sampling support points ξ_n *with replacement* in strong contrast to our sampling $\mathcal{M}_t$ *without replacement*. As previously motivated, the latter is more practical in terms of avoiding repeated computation with sample points that get sampled repeatedly. Unlike the with-replacement case, the samples within our $\mathcal{M}_t$ are no longer independent. Consequently, our analysis will primarily use tools of probability related to sampling finite sets without replacement; see the start of Appendix A.3. Motivating the form of η_t, note that sampling a subset of size M uniformly without replacement from a larger finite set of size N induces variance terms of the form $(\frac{1}{M} - \frac{1}{N})$ in place of the standard $\frac{1}{M}$.

To provide an outline of the proof, it starts by constructing $\tilde{P}^*$, a restriction of the (unique) P^* onto the (random) subset $\mathcal{M}_t$ in the restricted problem (3), where $\tilde{p}_m^* \propto p_m^*, \ \forall m \in \mathcal{M}_t$. The assumptions ensure with probability at least $\bar{\tau}_t$ that $\tilde{p}_m^* \neq \mathbf{0}$. With the same high probability, $\tilde{P}^*$ is also a feasible solution to (3) when ρ_t is inflated as assumed. Next, we establish that its objective function value is within η_t of the optimum with high probability, which yields the desired result. The Corollary 1, which is the main form of Theorem 3 used in the sequel, follows from Theorem 17.4 in [13].

Note that the bias identified in Theorem 3 vanishes only with $M_t \nearrow N$ as $t \to \infty$. Since fixed bias in gradients violates a basic requirement for SGD [3, Section 4.3] that the gradient estimator $\mathbb{E}[\nabla_\theta \hat{R}_t(\theta)] = \Theta(\nabla_\theta R(\theta))$ (using standard time complexity notation [31]), then the convergence of (2) cannot be guaranteed when $M_t = M$ for all t, where $M < N$. To address this bias issue, in strong contrast with using the full-data gradient $\nabla_\theta R(\theta)$ [21] or randomizing the subset size M_t over the entire dataset [15, 10], our algorithm chooses to grow the mini-batch size M_t progressively with t. Bottou et al. [3, Section 5.1] show that even with unbiased gradient estimation, keeping step sizes γ_t fixed and directly increasing the mini-batch size M_t is more advantageous because (informally) the larger M_t makes the gradient estimates more accurate. In our case, we grow the sample size in the SGD algorithm (2) to eliminate bias. Since this also provides a decrease in stochastic error as a consequence, it is no longer necessary to diminish the step size γ_t; indeed, doing so negates the benefits of the extra work in computing gradients using a larger M_t. Algorithm 1 therefore takes fixed-length steps γ. The maximum size N is reached after a (large) finite number of iterations $\mathcal{T}$, at which point our algorithm reduces to a deterministic full-gradient optimization approach.

2.2 Convergence of Algorithm 1

Seen in the above light, for a finite N, the strictly increasing M_t sequence will eventually end at an iterate $\mathcal{T} < \infty$ where $M_\mathcal{T} = N$, that is $\mathcal{T} := \inf_t\{M_t \geq N\}$. Hence, Algorithm 1 is not guaranteed to converge by the $\mathcal{T}$th iteration. We can nevertheless provide a guarantee on the performance of the method over any loss function with Lipschitz gradients.

Theorem 4. *Suppose the constant step size $\gamma_t = \gamma$ satisfies $\gamma \leq \frac{1}{2L}$, a lower bound $R_{\inf}$ exists for the robust loss function $R(\theta)$, $\forall \theta \in \Theta$, and the conditions of Theorem 3 hold. Then, the variance of the estimate $\nabla \hat{R}_t(\theta)$ calculated over the sampled-without-replacement $\mathcal{M}_t$ with subsample size M_t obeys $\mathbb{E}[\|\nabla \hat{R}_t(\theta) - \mathbb{E}[\nabla \hat{R}(\theta)]\|_2^2] \leq C(\frac{1}{M_t} - \frac{1}{N})$. Further assume the gradient $\nabla_\theta R(\theta)$ is*

L-Lipschitz. Then, at termination,

$$\sum_{t=1}^{\mathcal{T}} \|\nabla_\theta R(\theta_t)\|_2^2 \leq \frac{R(\theta_0) - R_{\inf}}{\frac{\gamma}{2}(2 - L\gamma)} + C\frac{L\gamma + 1}{2 - L\gamma}\sum_{t=1}^{\mathcal{T}} \eta_t^2. \tag{4}$$

The first result in Theorem 4 establishes that the variance of $\nabla \hat{R}_t(\theta)$ obeys a bound which is as to be expected for sampling without replacement from any finite set [35]. Combining this with the bias (see Corollary 1) allows us to progressively decrease the mean squared error. This leads to the result (4) in Theorem 4, which shows that the sum of the gradients of $R(\theta_t)$ at iterates visited by the algorithm is bounded above, in particular, by $\sum_{t=1}^{\mathcal{T}}(\frac{1}{M_t} - \frac{1}{N})^{(1-\delta)}$. If this summation remains finite as $\mathcal{T} \to \infty$, then the upper bound of (4) remains finite, and thus the gradients $\|\nabla_\theta R(\theta_t)\|_2^2$ at the iterates converge to 0; in other words, the algorithm converges to a local optimal solution. The summation can converge for M_t increasing moderately, such as at a polynomial rate.

Theorem 4 assumes that the gradient $\nabla_\theta R(\theta)$ of the robust loss is Lipschitz continuous. The gradients of such extreme value functions are in general not Lipschitz even if the objective function is Lipschitz. For example, a linear objective $l(\theta, \xi_n) = \theta^t \xi_i$ leads to $R(\theta) = \max_p \sum_i p_i \theta^t \xi_i$ and when maximized over a *polyhedral* constraint set it will not preserve the 0-Lipschitzness of the objective functions, because the optimal solutions P^* are picked from the discrete set of vertices of the polyhedron and thus $\nabla_\theta R(\theta)$ is piecewise discontinuous. Our Proposition 1 assumptions yield an inner maximization with a non-zero linear objective over a *strictly convex* feasible set. The desired smoothness can then be obtained with some additional conditions on the loss functions $l(\theta, \xi_i)$. Proposition 5 provides one such condition where the Lipschitzness of $\nabla_\theta R(\theta)$ follows from the Hessian of $R(\theta)$ being bounded in norm, which is often satisfied by common statistical learning losses, e.g., log-logistic and squared losses of linear models over compact spaces.

Proposition 5. *Assume the conditions in Proposition 1 hold. Further suppose that the Hessians $\nabla_\theta^2 l(\theta, \xi_n)$ exist, $\forall \theta$ and each ξ_n, and are bounded in Frobenious norm $\|\nabla_\theta^2 l(\theta, \xi_n)\|_F \leq L$, $\forall \theta, n$. Then, the robust loss also follows $\|\nabla_\theta^2 R(\theta)\|_F \leq M$ for some positive $M < \infty$.*

A key consideration then is to obtain $\theta_{\mathcal{T}}$ as close as possible to the minimizer θ_{rob} that attains $R_{\inf}$. The tradeoff in (4) suggests that increasing M_t aggressively will lead to smaller gradients at termination, but this will also increase the computational effort in each iteration. In the remainder of the section, we study this tradeoff under an additional assumption that the loss functions $l(\cdot, \xi)$ are strongly convex for each ξ, which we will show in Theorem 6 yields the strong convexity of $R(\theta)$. Consequently, it also provides a unique minimizer θ_{rob} for (1) that satisfies $R_{\inf} := R(\theta_{\text{rob}})$. We therefore seek a lower bound on how close the value of $R(\theta_{\mathcal{T}})$ and its gradient at termination of our progressively sampled Algorithm 1 can get to the unique optimal solution as a function of the sample growth sequence $\{M_t\}$, and also seek to establish convergence guarantees for the computational effort needed by our algorithm to attain a desired optimality gap larger than this lower bound.

Our notion of efficiency will be developed w.r.t. the total computational effort W_t that is expended up until iterate t, which is the sum of the individual work w_s in each iterate $s \leq t$. From the discussion following Proposition 2, we assume the relation $w_t = k_e M_t \log M_t$ holds for some constant k_e. Defining the ratio $\nu_t := M_{t+1}/M_t$ as the *growth factor* of the sequence $\{M_t\}$, we consider two important cases: (i) *Diminishing*-factor growth with $\nu_t \searrow 1$ as $t \uparrow$, e.g., the polynomial growth of $\nu_t = 1 + \frac{1}{t}$; and (ii) *Constant*-factor growth with $\nu_t = \nu > 1,\ \forall t$. Assuming we always start with the smallest size of $M_0 = 1$, the algorithm is employed to a maximum of $\mathcal{T}$ iterations where in this notation $\mathcal{T} = \inf\{t : N = \prod_{s=1}^{t} \nu_s\}$. For constant growth sequences, we have $\mathcal{T} = \log(N)/\log \nu$, and we use the notation $\mathcal{T}(\nu)$ and $W_{\mathcal{T}}(\nu)$ to denote the total iterations and total computational effort as functions of ν. Our final result characterizes guarantees on the rate at which the expected optimality gap $E_{\mathcal{T}} := \mathbb{E}_{\mathcal{T}-1}[R(\theta_{\mathcal{T}})] - R(\theta_{\text{rob}})$ decreases w.r.t. $W_{\mathcal{T}}$. Recall δ defines the parameter η_t in Theorem 3, and denote by E_0 the optimality gap at the starting iterate.

Theorem 6. *Suppose all the conditions of Theorem 4 are satisfied and the loss functions $l(\theta, \xi_n)$ are c-strongly convex. Then the function $R(\theta)$ is c-strongly convex. Further suppose that $\gamma \leq \min\{\frac{1}{4L}, 4c\}$ is fixed. Then: (i) For diminishing growth sequences M_t, we have that $E_{\mathcal{T}} W_{\mathcal{T}} \to \infty$ as $\nu \to 1^+$; and (ii) For constant growth sequences, we have that the total effort $W_{\mathcal{T}}(\nu)$ is a decreasing function over $\nu \in (1, \infty)$, with $\lim_{\nu \to 1^+}(\nu - 1)W_{\mathcal{T}}(\nu) = k_e(N \log N - (N - 1))$. Moreover, $E_{\mathcal{T}}(\nu) \leq \bar{E}_{\mathcal{T}}(\nu)$ where $\bar{E}_{\mathcal{T}}(\nu)$ is an increasing function over $\nu \in (1, \infty)$ with its infimum*

$\bar{E}^* = \inf_\nu \bar{E}_\mathcal{T}(\nu) = (1/N^{1-\delta})(E_0 + 8CLc^2))$ *attained as* $\nu \to 1^+$. *Finally, we have* $\lim_{\nu \to 1^+} W_\mathcal{T}(\nu) \;\; (\bar{E}_\mathcal{T}(\nu) - \bar{E}^*) = 8CLc^2(1-\delta)(1-\frac{\gamma}{4c}) \;\cdot\; k_e \left(N^\delta \log N - (N^\delta - N^{-(1-\delta)})\right)$.

Several key properties of our Algorithm 1 are apparent from Theorem 6 for the strongly-convex $R(\theta)$ case. To understand these results, first note (referring to the supplement for details) that if the full batch-gradient method is applied in each iteration ($M_t = N$), a strongly-convex objective R would enjoy a linear (i.e., constant factor) reduction of size $(1 - \gamma/4c)$ in the error $R(\theta_t) - R(\theta_{\text{rob}})$ for a step-size γ chosen to satisfy the conditions of Theorem 6. The average optimality gap can be written as a sum of this deterministic error and an additional term representing the stochastic error induced by the subsampling of the support. Theorem 6(i) establishes that any general diminishing-factor growth of M_t will lead to the stochastic error decreasing to zero much slower than the geometric drop in the deterministic error, and thus the stochastic error dominates. Consequently, there is a suboptimal reduction in the optimality gap $E_\mathcal{T}$ w.r.t. the total computational effort $W_\mathcal{T}$.

Constant factor sequences, however, can achieve a balance in the tradeoff between the rate of reduction in stochastic error and the drop in deterministic error. Specifically, although the total iterations $\mathcal{T}$ and total computational effort $W_\mathcal{T}$ fall as the growth factor ν grows (an exact expression for $W_\mathcal{T}(\nu)$ is provided in the supplement for any $\nu \in (1, \infty)$), this also leads to a poorer guarantee $\bar{E}_\mathcal{T}(\nu)$ on the attained optimality gap and thus $\nu \to 1^+$ is required to improve the guarantees on $E_\mathcal{T}$. Theorem 6 characterizes an absolute limit in $\bar{E}^* = O(N^{-(1-\delta)})$ on how much optimality gap reduction can be guaranteed in this regime, independent of chosen step length γ, and more importantly shows that, for any desired gap $\bar{E}^* + \epsilon$, our algorithm attains this result with $O(\epsilon^{-1})$ effort. Moreover, since δ is an arbitrarily small chosen constant, the dependence on N is only through poly-log terms. Hence, for such an ϵ, our progressively subsampled algorithm for the DRL formulation matches the best results for the SGD family of algorithms for strongly convex stochastic optimization formulations.

3 Empirical Results

Numerous experiments were conducted to empirically evaluate our progressively sampled subgradient descent (PSSG) Algorithm 1 and assess the efficacy of the DRO approach as a viable alternative to regularized ERM in producing good model generalization. In particular, we compare the performance of PSSG with that of competing algorithms to solve the DRO formulation (1), namely the full-support gradient (FG) algorithm of Namkoong and Duchi [21] and the multi-level Monte Carlo (Giles) algorithm of Levy et al. [15] and Ghosh and Squillante [10]. We also include in supplement B consideration of the standard SGD (fixed minibatch $M_t = M$) method to gauge the impact of the bias shown in Theorem 3; and further include consideration of the SGD method of Namkoong and Duchi [20] for a small dataset to demonstrate that the method was not competitive. These methods were all re-implemented since full source code was not made available with the corresponding publications.

Following Namkoong and Duchi [21], all examples use a logistic binary classification loss $l(\theta, (x, y)) = \log(1 + \exp(-y\theta^t x))$ where x represents samples with d features and y represents the class labels ± 1. All algorithms sampled the initial θ_0 uniformly from the hypercube $[-1, 1]^d$. All experiments were implemented in Python 3.7 and run on a 16-core 2.6GHz Intel Xeon processor with 128GB memory. Additional empirical results and technical details can be found in the supplement.

Generalization. We investigate whether the DRO formulation (1) can improve the generalization of learning models against an ERM-trained model that is regularized via 10-fold CV. Experiments were conducted over 13 public-domain datasets as detailed in Table 1 with sizes ranging from $O(10^2)$ to $O(10^6)$. We include `MNIST`, a non-convex neural-network model, where we follow [15] and admit only the last logistic classification layer into the DRL formulation. Table 1 presents a comparison of the test misclassification produced by the DRO algorithms and the regularized ERM algorithm at termination for the 13 datasets. (The parameter settings for each algorithm are detailed below). The 95% confidence intervals (CIs) are calculated over 10 permutations of the datasets into training (80%) and testing (20%) sets. We highlight in bold the methods that produce a generalization error (within its CIs) that is clearly the best. As evident, at least one DRO method – always including PSSG – produces models of equal or better quality as the regularized ERM formulation for all datasets.

Recall that the DRO methods provide this level of performance by solving a single instance of the DRO formulation (1), thus avoiding the burdensome 10-fold CV enumeration. Table 1 provides the average CPU time in seconds recorded over the 10 permutations. The average time taken by the

Dataset	$\sqrt{\frac{d}{N}}$	Test Misclassified (%)				CPU Time (secs)			
		FG	Giles	PSSG	ERM	FG	Giles	PSSG	ERM
adult*	0.051	17.1±0.0	16.7±0.1	16.6±0.1	16.7±0.1	45	214	36	2542
fabert†	0.312	**9.1±0.2**	9.9±0.2	**9.2±0.1**	9.9±0.1	32	61	20	4128
gina_agnostic†	0.529	**13.2±0.1**	15.4±0.4	**13.1±0.2**	15.9±0.7	33	21	13	1765
gina_prior†	0.475	13.7±0.4	14.3±1.0	**12.7±0.3**	14.6±0.7	34	38	31	1147
guillermo†	0.463	**30.7±0.2**	34.1±0.3	**30.7±0.2**	32.0±0.4	908	505	116	31547
hiv1*	0.166	5.9±0.1	6.3±0.2	**5.8±0.0**	6.1±0.1	41	45	35	1012
IMDB.drama †	0.091	36.1±0.1	37.0±0.1	36.2±0.0	36.2±0.1	176	865	89	19436
la1s.wc†	2.029	9.3±0.0	**8.3±0.3**	**8.2±0.1**	9.0±0.1	17	47	12	2456
MNIST⊕	1.342	2.0±0.1	1.7±0.1	**1.5±0.0**	1.8±0.1	39	28	20	984
OVA_Breast†	2.660	3.2±0.1	3.8±0.4	**3.0±0.1**	3.4±0.2	140	23	37	4310
rcv1‡	0.242	5.7±0.0	5.3±0.0	**5.1±0.0**	6.3±0.0	2628	1271	543	701843
riccardo†	0.463	4.9±0.1	2.0±0.1	**1.5±0.1**	1.7±0.1	259	201	120	86575
tr31.wc†	3.305	2.7±0.3	2.7±0.4	2.7±0.2	2.9±0.3	6	14	6	987

Table 1: Comparison of the DRO and regularized ERM formulations over 13 publicly available machine learning (ML) datasets, from UCI* [17], OpenML† [6], MNIST⊕ [14] and SKLearn‡ [16].The first set of four columns provides a 95% confidence interval of the percentage misclassified over withheld test datasets. The second set of four columns provides the average CPU time taken.

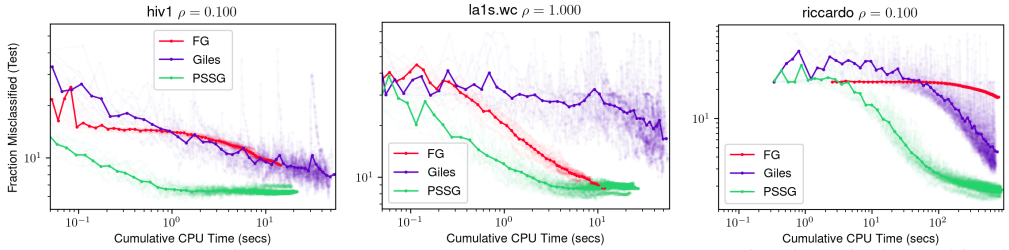

Figure 1: Comparison of PSSG (green), FG (red) and Giles (blue) on the fraction of misclassification in testing (y-axis) versus cumulative CPU time (log-scale x-axis) over the `dilbert`, `la1s.wc` and `riccardo` datasets, with ρ set as noted.

ERM 10-fold regularization in solving its formulation multiple times in a serial computing mode to identify the best regularization parameter (parameter λ below) exceeds that taken on average to solve the DRO formulation by one to two orders of magnitude. The computation time of a single PSSG DRO run is of the same order as that of a single ERM run. This indicates a significant computational savings in using DRO because of the elimination of the expensive hyper-parameter tuning step.

The main parameter of the DRO formulation is the uncertainty radius ρ for the set of measures $\mathcal{P}$, and following [2, 21] it is set to the same (base-10) order of magnitude as the value $\sqrt{d/N}$; Table 1 records this quantity for each dataset. We provide in supplement B detailed comparisons of the three DRO methods over multiple ρ values (and other key parameter values) for several datasets. These results illustrate similar performance trends and insensitivities across the datasets relative to the $O(\sqrt{d/N})$ guideline for FG and PSSG (which are independent of its growth factor). The Giles method, however, exhibits complex interactions between ρ and its minimum mini-batch size parameter that significantly impact both the solution quality and the computation time. Thus, fine tuning of these two key parameters of the Giles method is required to achieve the best quality solution and the smallest computational effort. In strong contrast, the interactions between ρ and the key parameters of PSSG are relatively weak with the growth factor only impacting the computational effort (and not the solution quality). Hence, it is sufficient to run PSSG with key parameters that are set based on our theoretical results without any fine tuning. This lack of fine tuning requirements is another important advantage of PSSG, in addition to its solution quality and computational advantages.

Among the DRO methods, PSSG takes the least time to solve the problem for all datasets (with one exception), often significantly so, while providing equal or better performance, including significant performance improvement over ERM in several cases. As expected, FG takes more time to solve the problem than PSSG for all datasets (comparable time for `tr31.wc`). For all but one dataset, Giles takes significantly more time than PSSG to solve the problem, by an order of magnitude in several

cases. These longer computation times are due in large part to the added variability experienced by the Giles method arising from the Monte Carlo randomization. The effect of this added variability is also evident in the consistently wider CIs of the misclassification errors for Giles, as well as in the wider range of CPU times with an isolated case terminating the fastest (`OVA_Breast`) and several cases terminating the slowest among DRO methods (e.g., `adult`, `fabert`, `IMDB.drama`, `la1s.wc`).

Figure 1 presents three representative comparisons of the empirical DRO runs over time, where the sample paths of all ten runs for each method are plotted along with their average shown in bold; corresponding plots for other datasets are provided in the supplement. The sample paths for PSSG (green) exhibit relatively low variability overall with reductions in variability as the sample size M_t grows with the iterations t. In strong contrast, the sample paths for Giles (blue) exhibit much higher variability across the iterations, and are computationally more expensive, for the reasons explained above. Although FG (red) also exhibits low variability, it is clearly more expensive computationally.

PSSG DRO. Our Algorithm 1 starts with an initial sample size of 1, the fixed step length $\gamma = 0.5$ and, following Theorem 6, the *constant-growth factor* $\nu = 1.001$ is chosen close to 1. Although the parameter δ appears prominently in the inflation of ρ to individual ρ_t as defined in Theorem 3, δ is required only to be a small positive constant. Since the empirical results were found not sensitive to δ, we set $\delta = 0.01$. All DRO algorithms solved the inner-maximization formulation to within ϵ-accuracy where $\epsilon = 10^{-7}$. Each method monitors the performance of the current model in correctly classifying the set-aside test set and stops if the average of the last 20 misclassification fraction values does not improve more than 1% when compared with the average of the previous 80 evaluations.

Theorem 3 provides the fundamental relationship between the bias suffered by a fixed batch size SGD ($M_t = M$) method in solving (1) and the batch size M. The important impact of this bias is illustrated by empirical results presented in the supplement. Under a small growth factor of $\nu = 1.001$, the iterations of PSSG operate with $M_t = 1$ until $t \approx 400$ when it then rises to $M_t = 2$. Hence, PSSG initially enjoys the benefits of fast objective value reduction similar to SGD with fixed batch size, but then eventually eliminates the introduced bias and thus also enjoys fast convergence. PSSG therefore avoids the expense of hyper-parameter tuning the batch size of standard SGD for bias reduction.

We also consider the impact of two key algorithm parameters: sample size growth factor ν and step length γ. Following Theorem 6, these settings are broadly expected to impact the computation times and solution quality. Empirical results (in the supplement), contrasting the performance of PSSG keeping one of the parameters γ and ν fixed while varying the other, show that the performance is insensitive to ν for values smaller than 1.001 and γ larger than 0.5, indicating that an error floor $\bar{E}^*$ has been reached. Hence we use these parameter values for all experiments in Table 1.

FG DRO. The step lengths of the FG algorithm are determined by the LBFGS-B algorithm with a maximum of 0.5 for all experiments. This parameter can have a notable impact on the speed of convergence, and our results in the supplement show this to be the best choice over many datasets.

Giles DRO. The parameters of the Giles algorithm are a minimum mini-batch size of 5 and a stepsize sequence of $\gamma_t = 0.5 * (5000/(5000 + t))$ for all experiments. As noted above, the minimum mini-batch size has a significant impact on the performance of the Giles method. The value 5 was chosen after careful study over multiple datasets as described in the supplement.

Regularized ERM. The 10-*fold CV* procedure partitions the full training dataset into 10 equal parts and trains a *regularized* model over each dataset formed by holding out one of the 10 parts as the *validation* dataset. The ERM loss objective is regularized by a $\lambda\|\theta\|_2^2$ term, and a fixed-batch SGD (size 10) is used for each combination of partition and λ values. Enumeration is employed to find the optimal λ from a grid of 20 points in the range $[10^{-6}, 10^6]$, starting with 10^6 and backtracking until the *average* performance over the 10 validation datasets does not improve for three λ enumerations. Note that the computational benefits of PSSG would be even larger if all λ values over all grid points were enumerated. Our results in the supplement show a significant amount of variability in the optimal λ over all the datasets, with no evident pattern relating to dataset characteristics, which highlights the need for ERM computations over a wide range of λ values for each dataset.

Summary. Our empirical results support our theoretical results and show that PSSG produces equal or better quality models as FG, Giles and regularized ERM with significantly less computational effort, often by orders of magnitude, thus providing a strong alternative approach to generalization.

References

[1] A. Ben-Tal, D. Den Hertog, A. De Waegenaere, B. Melenberg, and G. Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2): 341–357, 2013.

[2] J. Blanchet, Y. Kang, and K. Murthy. Robust Wasserstein Profile Inference and Applications to Machine Learning. *ArXiv e-prints*, Oct. 2016.

[3] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *ArXiv e-prints*, June 2016.

[4] R. Chen and I. Paschalidis. A Robust Learning Approach for Regression Models Based on Distributionally Robust Optimization. *Journal of Machine Learning Research*, 19(13):1–48, 2018.

[5] P. Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations. *Mathematical Programming*, 171: 115–166, 2018.

[6] M. Feurer, J. N. van Rijn, A. Kadra, P. Gijsbers, N. Mallik, S. Ravi, A. Mueller, J. Vanschoren, and F. Hutter. Openml-python: an extensible python api for openml. *arXiv*, 1911.02490, 2019. URL `https://arxiv.org/pdf/1911.02490.pdf`.

[7] R. Gao and A. J. Kleywegt. Distributionally Robust Stochastic Optimization with Wasserstein Distance. *arXiv e-prints*, art. arXiv:1604.02199, Apr. 2016.

[8] R. Gao, X. Chen, and A. J. Kleywegt. Wasserstein Distributional Robustness and Regularization in Statistical Learning. *arXiv e-prints*, art. arXiv:1712.06050, Dec. 2017.

[9] S. Ghosh and H. Lam. Robust analysis in stochastic simulation: Computation and performance guarantees. *Operations Research*, pages To–appear, 2018.

[10] S. Ghosh and M. S. Squillante. Unbiased Gradient Estimation for Distributionally Robust Learning. *arXiv e-prints*, art. arXiv:2012.12367, Dec. 2020.

[11] M. B. Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3):607–617, 2008. doi: 10.1287/opre.1070.0496. URL `https://doi.org/10.1287/opre.1070.0496`.

[12] F. S. Hashemi, R. Pasupathy, and M. R. Taaffe. The adaptive sampling gradient method: Optimizing smooth functions with an inexact oracle. 2017. Under review.

[13] J. Jacod and P. Protter. *Probability Essentials*. Universitext. Springer Berlin Heidelberg, 2004. ISBN 9783540438717.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[15] D. Levy, Y. Carmon, J. C. Duchi, and A. Sidford. Large-scale methods for distributionally robust optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*. 2020.

[16] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[17] M. Lichman. Uci machine learning repository, 2013. URL `https://archive.ics.uci.edu/ml/datasets`.

[18] D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, 1969.

[19] F. Luo and S. Mehrotra. Decomposition Algorithm for Distributionally Robust Optimization using Wasserstein Metric. *arXiv e-prints*, art. arXiv:1704.03920, Apr. 2017.

[20] H. Namkoong and J. C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2208–2216. Curran Associates, Inc., 2016.

[21] H. Namkoong and J. C. Duchi. Variance-based regularization with convex objectives. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2971–2980. Curran Associates, Inc., 2017.

[22] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 9780387400655. URL `https://books.google.com/books?id=VbHYoSyelFcC`.

[23] A. B. Owen. *Empirical Likelihood*. Chapman & Hall, 2001.

[24] R. Pasupathy, P. Glynn, S. Ghosh, and F. Hashemi. On sampling rates in simulation-based recursions. *SIAM Journal on Optimization*, 28(1):45–73, 2018. doi: 10.1137/140951679. URL `https://doi.org/10.1137/140951679`.

[25] C. Perlich, F. Provost, and J. S. Simonoff. Tree Induction vs. Logistic Regression: A Learning-Curve Analysis. *Journal of Machine Learning Research*, 4:211–255, 2003.

[26] T. Rognvaldsson, L. You, and D. Garwicz. State of the art prediction of hiv-1 protease cleavage sites. *Bioinformatics*, 31(8):1204–1210, 2014.

[27] A. Shapiro. Second-order derivatives of extremal-value functions and optimality conditions for semi-infinite programs. *Math. of OR*, 10:207–219, 1985. doi: 10.1287/moor.10.2.207.

[28] A. Shapiro. Worst-case distribution analysis of stochastic programs. *Math. Program.*, 107: 91–96, 06 2006. doi: 10.1007/s10107-005-0680-6.

[29] A. Shapiro, D. Dentcheva, and A. Ruszczynski. *Lectures on Stochastic Programming Modeling and Theory*. SIAM, Philadelphia, 2009.

[30] A. Sinha, H. Namkoong, and J. Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. In *ICLR 2018*. Oct. 2017.

[31] M. Sipser. *Introduction to the Theory of Computation*. Course Technology Inc., 2006. ISBN 0-619-21764-2.

[32] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[33] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[34] G. Wahba. Spline models for observational data. society for industrial and applied mathematics, 1990.

[35] S. S. Wilks. *Mathematical Statistics*. John Wiley & Sons, 1962.

Checklist

The checklist follows the references. Please

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? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not anticipate any negative impacts, and hope that improved generalization performance will bring benefits.

(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] They are included in the supplement.
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]
   (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)? [Yes] We include them as confidence intervals.
   (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] We use scikit-learn and Openml and provide a citation for each.
   (b) Did you mention the license of the assets? [N/A] They are very popular commonly used libraries.
   (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] We use only public domain datasets from well-known public sources.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use only public domain datasets from well-known public sources.
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]