Adaptive Sampling for Stochastic Risk-Averse Learning

Sebastian Curi
Dept. of Computer Science
ETH Zurich
scuri@inf.ethz.ch

Kfir Y. Levy
Faculty of Electrical Engineering
Technion
kfirylevy@technion.ac.il

Stefanie Jegelka
CSAIL
MIT
stefje@mit.edu

Andreas Krause
Dept. of Computer Science
ETH Zurich
krausea@inf.ethz.ch

Abstract

In high-stakes machine learning applications, it is crucial to not only perform well *on average*, but also when restricted to *difficult* examples. To address this, we consider the problem of training models in a risk-averse manner. We propose an adaptive sampling algorithm for stochastically optimizing the *Conditional Value-at-Risk (CVaR)* of a loss distribution, which measures its performance on the α fraction of most difficult examples. We use a distributionally robust formulation of the CVaR to phrase the problem as a zero-sum game between two players, and solve it efficiently using regret minimization. Our approach relies on sampling from structured Determinantal Point Processes (DPPs), which enables scaling it to large data sets. Finally, we empirically demonstrate its effectiveness on large-scale convex and non-convex learning tasks.

1 Introduction

Machine learning systems are increasingly deployed in high-stakes applications. This imposes reliability requirements that are in stark discrepancy with how we currently train and evaluate these systems. Usually, we optimize *expected performance* both in training and evaluation via empirical risk minimization (Vapnik, 1992). Thus, we sacrifice occasional large losses on "difficult" examples in order to perform well on average. In this work, we instead consider a *risk-averse* optimization criterion, namely the *Conditional Value-at-Risk* (CVaR), also known as the Expected Shortfall. In short, the α-CVaR of a loss distribution is the average of the losses in the α-tail of the distribution.

Optimizing the CVaR is well-understood in the *convex* setting, where duality enables a reduction to standard empirical risk minimization using a modified, truncated loss function from Rockafellar et al. (2000). Unfortunately, this approach fails when *stochastically* optimizing the CVaR – especially on non-convex problems, such as training deep neural network models. A likely reason for this failure is that mini-batch estimates of gradients of the CVaR suffer from high variance.

To address this issue, we propose a novel *adaptive sampling algorithm* – ADA-CVAR (Section 4). Our algorithm initially optimizes the mean of the losses but gradually adjusts its sampling distribution to increasingly sample tail events (difficult examples), until it eventually minimizes the CVaR (Section 4.1). Our approach naturally enables the use of standard stochastic optimizers (Section 4.2). We provide convergence guarantees of the algorithm (Section 4.3) and an efficient implementation (Appendix A). Finally, we demonstrate the performance of our algorithm in a suite of experiments (Section 5).

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Related Work

Risk Measures Risk aversion is a well-studied human behavior, in which agents assign more weight to adverse events than to positive ones (Pratt, 1978). Approaches for modeling risk include using utility functions that emphasize larger losses (Rabin, 2013); prospect theory that re-scales the probability of events (Kahneman and Tversky, 2013); or direct optimization of coherent risk-measures (Artzner et al., 1999). Rockafellar et al. (2000) introduce the CVaR as a particular instance of the latter class. The CVaR has found many applications, such as portfolio optimization (Krokhmal et al., 2002) or supply chain management (Carneiro et al., 2010), as it does not rely on specific utility or weighing functions, which offers great flexibility.

CVaR in Machine Learning The ν-SVM algorithm by Schölkopf et al. (2000) can be interpreted as optimizing the CVaR of the loss, as shown by Gotoh and Takeda (2016). Also related, Shalev-Shwartz and Wexler (2016) propose to minimize the *maximal loss* among all samples. The maximal loss is the limiting case of the CVaR when $\alpha \to 0$. Fan et al. (2017) generalize this work to the top-k average loss. Although they do not mention the relationship to the CVaR, their learning criterion is equal to the CVaR for empirical measures. For optimization, they use an algorithm proposed by Ogryczak and Tamir (2003) to optimize the maximum of the sum of k functions; this algorithm is the same as the "truncated" algorithm of Rockafellar et al. (2000) to optimize the CVaR. Recent applications of the CVaR in ML include risk-averse bandits (Sani et al., 2012), risk-averse reinforcement learning (Chow et al., 2017), and fairness (Williamson and Menon, 2019). All these use the original "truncated" formulation of Rockafellar et al. (2000) to optimize the CVaR. One of the major shortcomings of this formulation is that mini-batch gradient estimates have high variance. In this work, we address this via a method based on adaptive sampling, inspired by Shalev-Shwartz and Wexler (2016), that allows us to handle large datasets and complex (deep neural network) models.

Distributionally Robust Optimization The CVaR also has a natural *distributionally robust optimization* (DRO) interpretation (Shapiro et al., 2009, Section 6.3), which we exploit in this paper. For example, Ahmadi-Javid (2012) introduces the entropic value-at-risk by considering a different DRO set. Duchi et al. (2016); Namkoong and Duchi (2017); Esfahani and Kuhn (2018); Kirschner et al. (2020) address related DRO problems, but with different uncertainty sets. We use the DRO formulation of the CVaR to phrase its optimization as a game. To solve the game, we propose an adaptive algorithm for the learning problem. Our algorithm is most related to (Namkoong and Duchi, 2016), who develop an algorithm for DRO sets induced by Cressie-Read f-divergences. Instead, we use a different DRO set that arises in common data sets (Mehrabi et al., 2019) and we provide an *efficient* algorithm to solve the DRO problem in large-scale datasets.

3 Problem Statement

We consider supervised learning with a *risk-averse learner*. The learner has a data set comprised of i.i.d. samples from an unknown distribution, i.e., $D = \{(x_1, y_1), \dots (x_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y})^N \sim \mathcal{D}^N$, and her goal is to learn a function $h_\theta : \mathcal{X} \to \mathcal{R}$ that is parametrized by $\theta \in \Theta \subset \mathbb{R}^d$. The performance of h_θ at a data point is measured by a *loss function* $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to [0, 1]$. We write the random variable $L_i(\theta) = l(\theta; x_i, y_i)$. The learner's goal is to minimize the ***CVaR of the losses*** on the (unknown) distribution $\mathcal{D}$ w.r.t. the parameters θ.

CVaR properties The CVaR of a random variable $L \sim P$ is defined as $\mathbb{C}^\alpha[L] = \mathbb{E}_P[L | L \geq \ell^\alpha]$, where ℓ^α is the $1-\alpha$ quantile of the distribution, also called the *Value-at-Risk* (VaR). We illustrate the mean, VaR and CVaR of a typical loss distribution in Figure 1. It can be shown that the CVaR of a random variable has a natural *distributionally robust optimization* (DRO) formulation, namely as the expected value of the same random variable under a *different* law. This law arises from the following optimization problem (Shapiro et al., 2009, Sec. 6.3):

$$\mathbb{C}^\alpha[L] = \max_{Q \in \mathcal{Q}^\alpha} \mathbb{E}_Q[L], \tag{1}$$

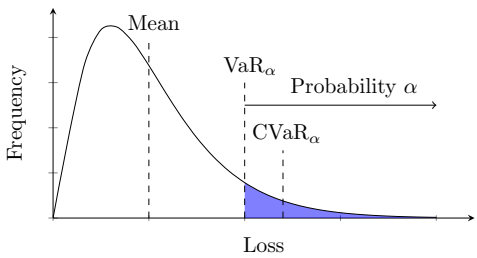

Figure 1: CVaR of a loss distribution

where $\mathcal{Q}^\alpha = \left\{ Q \ll P, \frac{dQ}{dP} \leq \frac{1}{\alpha} \right\}$. Here, $Q \ll P$ means that Q is absolutely continuous w.r.t. P. The distribution $Q^\star$ that solves Problem (1) places all the mass uniformly in the tail, i.e., the blue shaded region of Figure 1. Thus, optimizing the CVaR can be viewed as *guarding against a particular kind of distribution shift*, which reweighs arbitrary parts of the data up to a certain amount $\frac{1}{\alpha}$. Rockafellar et al. (2000) prove strong duality for Problem (1). The dual objective is:

$$\mathbb{C}^\alpha[L] = \min_{\ell \in \mathbb{R}} \ell + \frac{1}{\alpha} \mathbb{E}_P \left[\max \{0, L - \ell\} \right]. \tag{2}$$

Learning with the CVaR Problem (2) can be used to estimate the CVaR of a random variable by replacing the expectation $\mathbb{E}_P$ by the empirical mean $\hat{\mathbb{E}}$, yielding

$$\min_{\ell \in \mathbb{R}, \theta \in \Theta} \ell + \frac{1}{\alpha N} \sum\nolimits_{i=1}^{N} \left[\max \{0, L_i(\theta) - \ell\} \right]. \tag{3}$$

For convex L_i, Problem (3) has computable subgradients, and hence lends itself to subgradient-based optimization. Furthermore, in this case Problem (3) is jointly convex in (ℓ, θ). We refer to this standard approach as TRUNC-CVAR, as it effectively optimizes a modified loss, truncated at ℓ.

Problem (3) is indeed a sensible learning objective in the sense that the empirical CVaR concentrates around the population CVaR uniformly for all functions $h \in \mathcal{H}$.

Proposition 1. *Let $h : \mathcal{X} \to \mathcal{Y}$ be a finite function class $|\mathcal{H}|$. Let $L(h) : \mathcal{H} \to [0, 1]$ be a random variable. Then, for any $0 < \alpha \leq 1$, with probability at least $1 - \delta$,*

$$\mathbb{E}\left[\sup_{h \in \mathcal{H}} \left| \widehat{\mathbb{C}}^\alpha[L(h)] - \mathbb{C}^\alpha[L(h)] \right| \right] \leq \frac{1}{\alpha} \sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}.$$

Proof. See Appendix B.1. □

The result above is easily extended to classes $\mathcal{H}$ with finite VC (pseudo-)dimension. Concurrent to this work, Lee et al. (2020), Soma and Yoshida (2020), and Mhammedi et al. (2020) present similar statistical rates based on different assumptions.

Challenges for Stochastic Optimization In the common case that a variant of SGD is used to optimize the learning problem (3), the expectation is approximated with a mini-batch of data. But, when this batch is sampled uniformly at random from the data, only a fraction α of points will contain gradient information. The gradient of the remaining points gets truncated to zero by the $\max\{\cdot\}$ non-linearity. Furthermore, the gradient of the examples that *do* contain information is scaled by $1/\alpha$, leading to exploding gradients. These facts make stochastic optimization of Problem (3) extremely challenging, as we demonstrate in Section 5.

Our key observation is that the root of the problem lies in the *mismatch* between the sampling distribution P and the unknown distribution $Q^\star$, from which we would ideally like to sample. In fact, Problem (3) can be interpreted as a form of rejection sampling – samples with losses smaller than ℓ are rejected. It is well known that Monte Carlo estimation of rare events suffers from high variance (Rubino and Tuffin, 2009). To address this issue, we propose a novel sampling algorithm that *adaptively* learns to sample events from the distribution $Q^\star$ while optimizing the model parameters θ.

4 ADA-CVAR: Adaptive Sampling for CVaR Optimization

We directly address the DRO problem (1) on the empirical measure $\hat{P}$ for learning. The DRO set is $\mathcal{Q}^\alpha = \left\{ q \in \mathbb{R}^N \mid 0 \leq q_i \leq \frac{1}{k}, \sum_i q_i = 1 \right\}$ with $k = \lfloor \alpha N \rfloor$. The learning problem becomes:

$$\min_{\theta \in \Theta} \max_{q \in \mathcal{Q}^\alpha} \mathbb{E}_q[L_i(\theta)] = \min_{\theta \in \Theta} \max_{q \in \mathcal{Q}^\alpha} q^\top L(\theta), \tag{4}$$

where $L(\theta) \in \mathbb{R}^N$ has i-th index $L_i(\theta)$. The learning problem (4) is a minimax game between a θ-player (the learner), whose goal is to *minimize* the objective function by selecting $\theta \in \Theta$, against a q-player (the sampler), whose goal is to *maximize* the objective function by selecting $q \in \mathcal{Q}^\alpha$.

To solve the game (4), we use techniques from regret minimization with partial (bandit) feedback. In particular, we exploit that one can solve minimax games by viewing both players as online learners

that compete, and by equipping each of them with no-regret algorithms (Freund and Schapire, 1999). With the partial (bandit) feedback model, we only need to consider a *small* subset of the data in each iteration of the optimization algorithm. In contrast, full-information feedback would require a full pass over the data per iteration, invalidating all benefits of stochastic optimization.

Next, we describe and analyze an online learning algorithm for each of the two players and prove guarantees with respect to the DRO problem (4). We outline the final algorithm, which we call ADA-CVAR, in Algorithm 1, where we use an adaptive sampling scheme for the q-player and SGD for the θ-player. Initially, the q-player (sampler) plays the uniform distribution and the θ-player (learner) selects any parameter in the set Θ. In iteration t, the sampler samples a data point (or a mini-batch) with respect to the distribution q_t. Then, the learner performs an SGD step on the sample(s) selected by the sampler player[1]. Finally, the q-player adapts the distribution to favor examples with higher loss and thus maximize the objective in (4).

Algorithm 1: ADA-CVAR

input Learning rates η_s, η_l.
1: **Sampler:** Initialize k-DPP $w_1 = \mathbf{1}_N$.
2: **Learner:** Initialize parameters $\theta_0 \in \Theta$.
3: **for** $t = 1, \ldots, T$ **do**
4: **Sampler:** Sample data point $i_t \sim q_t = \frac{1}{k}\mathbb{P}_{w_t}(i)$.
5: **Learner:** $\theta_t = \theta_{t-1} - \eta_l \nabla L_{i_t}(\theta_{t-1})$.
6: **Sampler**: Build estimate $\hat{L}_t = \frac{L_{i_t}(\theta_t)}{q_{t,i_t}}[[i == i_t]]$.
7: **Sampler**: Update k-DPP $w_{t+1,i} = w_{t,i} e^{\eta_s \hat{L}_{t,i}}$.
8: **end for**
output $\bar{\theta}, \bar{q} \sim_{u.a.r} \{(\theta_t, q_t)\}_{t=1}^T$

4.1 Sampler (q-Player) Algorithm

In every iteration t, the learner player sets a vector of losses through θ_t. We denote by $L(\theta_t; x_i, y_i) = L_{t,i}$ the loss at time t for example i and by $L(\theta_t) = L_t$ the vector of losses. The sampler player chooses an index i_t (or a mini-batch) and a vector q_t. Then, only L_{t,i_t} is revealed and she suffers a cost $q_t^\top L_t$. In such setting, the best the player can aim to do is to minimize its *regret*:

$$\mathrm{SR}_T \coloneqq \max_{q \in \mathcal{Q}^\alpha} \sum_{t=1}^T q^\top L_t - \sum_{t=1}^T q_t^\top L_t. \tag{5}$$

The regret measures how good the sequence of actions of the sampler is, compared to the best single action (i.e., distribution over the data) in hindsight, after seeing the sequence of iterates L_t. The sampler player problem is a linear adversarial bandit. Exploration and sampling in this setting are hard (Bubeck et al., 2012). Our efficient implementation exploits the specific combinatorial structure.

In particular, the DRO set $\mathcal{Q}^\alpha$ is a polytope with $\binom{N}{k}$ vertices, each corresponding to a different subset I of size k of the ground set $2^{[N]}$. As the inner optimization problem over q in (4) is a linear program, the optimal solution $q^\star$ is a vertex. Thus, the sampler problem can be reduced to a *best subset selection* problem: find the best set among all size-k subsets $\mathcal{I}_k = \left\{I \subseteq 2^{[N]} \mid |I| = k\right\}$. Here, the value of a set I at time t is simply the average of the losses $(1/k)\sum_{i \in I} L_i(\theta_t)$. The problem of maximizing the value over time t can be viewed as a *combinatorial bandit* problem, as we have a combinatorial set of "arms", one per $I \in \mathcal{I}_k$ (Lattimore and Szepesvári, 2018, Chapter 30). Building on Alatur et al. (2020), we develop an efficient algorithm for the sampler.

Starting Point: EXP.3 A well known no-regret bandit algorithm is the celebrated EXP.3 algorithm (Auer et al., 2002). Let $\Delta_I \coloneqq \left\{\tilde{W} \in \mathbb{R}^{\binom{N}{k}} | \sum_I \tilde{W}_I = 1, \tilde{W}_I \geq 0\right\}$ be the simplex of distributions over the $\binom{N}{k}$ subsets. Finding the best distribution $W_I^\star \in \Delta_I$ is equivalent to finding the best subset $I^\star \in \mathcal{I}_k$. By transitivity, this is equivalent to finding the best $q^\star \in \mathcal{Q}^\alpha$. To do this, EXP.3 maintains a vector $W_{I,t} \in \Delta_I$, samples an element $I_t \sim W_{I,t}$ and observes a loss associated with element I_t. Finally, it updates the distribution using multiplicative weights. Unfortunately, EXP.3 is *intractable in two ways*: Sampling a k-subset I_t would require evaluating the losses of $k = \lfloor \alpha N \rfloor$ data points, which is impractical. Furthermore, the naive EXP.3 algorithm is intractable because the dimension of $W_{I,t}$ is *exponential* in k. In turn, the regret of this algorithm also depends on the dimension of $W_{I,t}$.

Efficiency through Structure The crucial insight is that we can *exploit the combinatorial structure of the problem and additivity of the loss* to exponentially improve efficiency. First, we exploit that

[1]Note that we do *not* use any importance sampling correction.

weights of individual elements and sets of them are related by $W_{t,I} = \sum_{i \in I} w_{t,i}$. Thus, instead of observing the loss L_{I_t}, we let the q-player sample only a *single element* i_t uniformly at random from the set $I_t \sim W_{I,t}$, observe its loss L_{i_t}, and use it to update a weight vector $w_{t,i}$. The single element i_t sampled by the algorithm provides information about the loss of all $\binom{N-1}{k-1}$ *sets* that contain i_t. This allows us to obtain regret guarantees that are sub-linear in N (rather than in N^k). Second, we *exponentially improve computational cost* by developing an algorithm that maintains a vector $w \in \mathbb{R}^N$ and uses *k-Determinantal Point Processes* to map it to distributions over subsets of size k.

Definition 4.1 (k-DPP, Kulesza et al. (2012))**.** A k-Determinantal Point Process over a ground set N is a distribution over all subsets of size k s.t. the probability of a set is $\mathbb{P}(I) \propto \det(K_I)$, where K is a positive definite kernel matrix and K_I is the submatrix of K indexed by I. ■

In particular, we consider k-DPPs with *diagonal* kernel matrices $K = \operatorname{diag} w$, with $w \in \mathbb{R}^N_{\geq 0}$ and at least k strictly positive elements. This family of distributions is sufficient to contain, for example, the uniform distribution over the $\binom{N}{k}$ subsets and all the vertices of $\mathcal{Q}^\alpha$. We use such k-DPPs to *efficiently* map a vector of size N to a distribution over $\binom{N}{k}$ subsets. We also denote the marginal probability of element i by $\mathbb{P}_w(i)$. It is easy to verify that the vector of marginals $\frac{1}{k}\mathbb{P}_w(i) \in \mathcal{Q}^\alpha$. Hence, we directly use the k-DPP marginals as the sampler's decision variables.

We can finally describe the sampler algorithm. We initialize the k-DPP kernel with the uniform distribution $w_1 = \mathbf{1}_N$. In iteration t, the sampler plays the distribution $q_t = \frac{1}{k}\mathbb{P}_{w_t}(\cdot) \in \mathcal{Q}^\alpha$ and samples an element $i_t \sim q_t$. The loss at index i_t, L_{t,i_t}, is revealed to the sampler and only the index i_t of w_t is updated according to the multiplicative update $w_{t+1,i_t} = w_{t+1,i_t} e^{kL_{t,i_t}/q_{t,i_t}}$.

This approach addresses the disadvantages of the EXP.3 algorithm. Computationally, it only requires $O(N)$ memory. After sampling every element i_t, the distribution over the $\binom{N-1}{k-1}$ sets that contain i_t are updated. This yield rates that depend sub-linearly on the data set size which we prove next.

Lemma 1. *Let the sampler player play the* ADA-CVAR *Algorithm with* $\eta_s = \sqrt{\frac{\log N}{NT}}$*. Then, for any sequence of losses she suffers a regret* (5) *of at most* $O(\sqrt{TN \log N})$.

Proof sketch. For a detailed proof please refer to Appendix B.2. First, we prove in Proposition 3 that the iterates of the algorithm are effectively in $\mathcal{Q}^\alpha$. Next, we prove in Proposition 4 that the comparator in the regret of Alatur et al. (2020) and in the sampler regret (5) have the same value (scaled by k). Finally, the result follows as a corollary from these propositions and Alatur et al. (2020, Lemma 1). □

4.2 Learner (θ-Player) Algorithm

Analogous to the sampler player, the learner player seeks to minimize its regret

$$\mathrm{LR}_T := \sum_{t=1}^T q_t^\top L(\theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^T q_t^\top L(\theta). \tag{6}$$

Crucially, the learner can choose θ_t *after* the sampler selects q_t. Thus, the learner can play the Be-The-Leader (BTL) algorithm:

$$\theta_t = \arg\min_{\theta \in \Theta} \sum_{\tau=1}^t q_\tau^\top L(\theta) = \arg\min_{\theta \in \Theta} \bar{q}_t^\top L(\theta), \tag{7}$$

where $\bar{q}_t = \frac{1}{t}\sum_{\tau=1}^t q_\tau$ is the average distribution (up to time t) that the sampler player proposes.

Instead of assuming access to an exact optimizer, we assume to have an ERM oracle available.

Assumption 1 (ϵ_{oracle}-correct ERM Oracle)**.** *The learner has access to an ERM oracle that takes a distribution q over the dataset as input and outputs $\hat{\theta}$, such that*

$$q^\top L(\hat{\theta}) \leq \min_{\theta \in \Theta} q^\top L(\theta) + \epsilon_{\text{oracle}}.$$

To implement the ERM Oracle, the learner player must solve a *weighted* empirical loss minimization in Problem (7) in every round. For non-convex problems, this is in general NP-hard (Murty and Kabadi, 1987), so obtaining efficient and provably no-regret guarantees in the non-convex setting seems unrealistic in general.

Despite this hardness, the success of deep learning empirically demonstrates that stochastic optimization algorithms such as SGD are able to find very good (even if not necessarily optimal) solutions for the ERM-like non-convex problems. Furthermore, SGD on the sequence of samples $\{i_\tau \sim q_\tau\}_{\tau=1}^{t}$ approximately solves the BTL problem. To see why, we note that such sequence of samples is an unbiased estimator of $\bar{q}_t$ from the BTL algorithm (7). Then, for the freshly sampled $i_t \sim q_t$, a learner that chooses $\theta_t \coloneqq \theta_{t-1} - \eta_l \nabla L_{i_t}(\theta_{t-1})$ is (approximately) solving the BTL algorithm with SGD.

Lemma 2. *A learner player that plays the BTL algorithm with access to an ERM oracle as in Assumption 1, achieves at most $\epsilon_{\text{oracle}} T$ regret.*

Proof. See Appendix B.3. □

For convex problems, we know that it is not necessary to solve the BTL problem (7), and algorithms such as online projected gradient descent (Zinkevich, 2003) achieve no-regret guarantees. As shown in Appendix C, the learner suffers $\text{LR}_T = O(\sqrt{T})$ regret by playing SGD in convex problems.

4.3 Guarantees for CVaR Optimization

Next, we show that if both players play the no-regret algorithms discussed above, they solve the game (4). Using $J(\theta, q) \coloneqq q^\top L(\theta)$, the minimax equilibrium of the game is the point $(\theta^\star, q^\star)$ such that $\forall \theta \in \Theta, q \in \mathcal{Q}^\alpha;\ J(\theta^\star, q) \leq J(\theta^\star, q^\star) \leq J(\theta, q^\star)$. We assume that this point exists (which is guaranteed, e.g., when the sets $\mathcal{Q}^\alpha$ and Θ are compact). The game regret is

$$\text{GameRegret}_T \coloneqq \sum_{t=1}^{T} J(\theta_t, q^\star) - J(\theta^\star, q_t). \tag{8}$$

Theorem 1 (Game Regret)**.** *Let $L_i(\cdot) : \Theta \to [0, 1]$, $i = \{1, ..., N\}$ be a fixed set of loss functions. If the sampler plays* ADA-CVAR *and the learner plays the oracle-BTL algorithm, then the game has regret $O(\sqrt{TN \log N} + \epsilon_{\text{oracle}} T)$.*

Proof sketch. We bound the Game regret with the sum of the learner and sampler regret and use the results of Lemma 2 and Lemma 1. For a detailed proof please refer to Appendix B.4. □

This immediately implies our main theoretical result, namely a performance guarantee for the solution obtained by ADA-CVAR for the central problem of minimizing the empirical CVaR (4).

Corollary 1 (Online to Batch Conversion)**.** *Let $L_i(\cdot) : \Theta \to [0, 1]$, $i = \{1, ..., N\}$ be a set of loss functions sampled from a distribution $\mathcal{D}$. Let $\theta^\star$ be the minimizer of the CVaR of the empirical distribution $\hat{\mathbb{C}}^\alpha$. Let $\bar{\theta}$ be the output of* ADA-CVAR*, selected uniformly at random from the sequence $\{\theta_t\}_{t=1}^{T}$. Its expected excess CVaR is bounded as:*

$$\mathbb{E}\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] \leq \hat{\mathbb{C}}^\alpha[L(\theta^\star)] + O(\sqrt{N \log N / T}) + \epsilon_{\text{oracle}}$$

where the expectation is taken w.r.t. the randomization in the algorithm, both for the sampling steps and the randomization in choosing $\bar{\theta}$.

Proof sketch. For a detailed proof please refer to Appendix B.5. The excess CVaR is bounded by the duality gap, which in turn is upper-bounded by the average game regret. □

Corollary 2 (Population Guarantee)**.** *Let $L(\cdot) : \Theta \to [0, 1]$ be a Random Variable induced by the data distribution $\mathcal{D}$. Let $\theta^\star$ be the minimizer of the CVaR at level α of such Random Variable. Let $\bar{\theta}$ be the output of* ADA-CVAR*, selected uniformly at random from the sequence $\{\theta_t\}_{t=1}^{T}$. Then, with probability at least δ the expected excess CVaR of $\bar{\theta}$ is bounded as:*

$$\mathbb{E}\mathbb{C}^\alpha[L(\bar{\theta})] \leq \mathbb{C}^\alpha[L(\theta^\star)] + O(\sqrt{N \log N / T}) + \epsilon_{\text{oracle}} + \epsilon_{\text{stat}}$$

where $\epsilon_{\text{stat}} = \tilde{O}(\frac{1}{\alpha}\sqrt{\frac{1}{N}})$ comes from the statistical error and the expectation is taken w.r.t. the randomization in the algorithm, both for the sampling steps and the randomization in choosing $\bar{\theta}$.

Proof sketch. For a detailed proof please refer to Appendix B.6. We bound the statistical error using Proposition 1 and the optimization error is bounded using Corollary 1. □

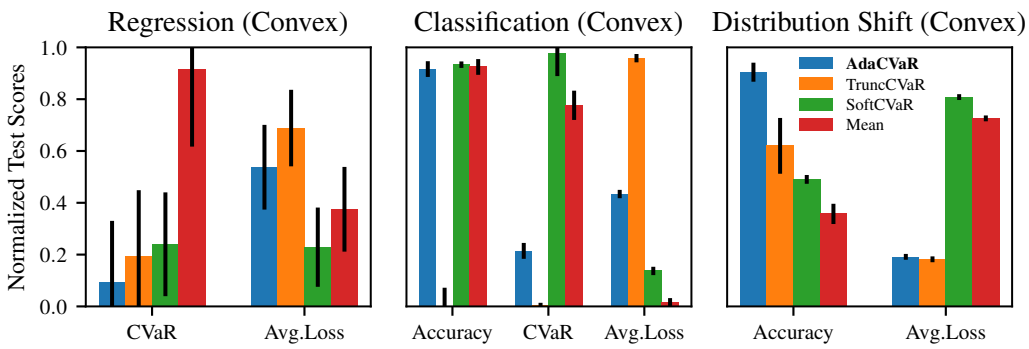

Figure 2: Scores are normalized between 0 and 1 to compare different data sets. *Left*: Linear regression tasks. ADA-CVAR has lower CVaR than benchmarks. *Middle*: Binary classification (logistic regression) tasks. ADA-CVAR obtains the same accuracy as MEAN and SOFT-CVAR with lower CVaR. TRUNC-CVAR outputs an approximately uniform distribution yielding low CVaR but poor predictive accuracy. *Right*: Binary classification (logistic regression) tasks with train/test 90% distribution shift. ADA-CVAR has the highest test accuracy and low average surrogate loss.

It is instructive to consider the special cases $k = 1$ and $k = N$. For $k = N$, q_t remains uniform and ADA-CVAR reduces to SGD. For $k = 1$, the sampler simply plays standard EXP.3 over data points and ADA-CVAR reduces to the algorithm of Shalev-Shwartz and Wexler (2016) for the max loss.

5 Experiments

In our experimental evaluation, we compare ADA-CVAR on both convex (linear regression and classification) and non-convex (deep learning) tasks. In addition to studying how it performs in terms of the CVaR and empirical risk on the training and test set, we also investigate to what extent it can help guard against distribution shifts. In Appendix D, we detail the experimental setup. We provide an open-source implementation of our method, which is available at `http://github.com/sebascuri/adacvar`.

Baseline Algorithms We compare our adaptive sampling algorithm (ADA-CVAR) to three baselines: first, an i.i.d. sampling scheme that optimizes Problem (3) using the truncated loss (TRUNC-CVAR); second, an i.i.d. sampling scheme that uses a smoothing technique to relax the $\sum_i [x_i]_+$ non-linearity (SOFT-CVAR). Tarnopolskaya and Zhu (2010) compare different smoothing techniques for the $\sum_i [x_i]_+$ non-linearity. Of these, we use the relaxation $T \log(\sum_i e^{x_i/T})$ proposed by Nemirovski and Shapiro (2006). In each iteration, we heuristically approximate the population sum with a mini-batch. Third, we also compare a standard i.i.d. sampling ERM scheme that stochastically minimizes the average of the losses (MEAN).

5.1 Convex CVaR Optimization

We first compare the different algorithms in a controlled convex setting, where the classical TRUNC-CVAR algorithm is expected to perform well. We consider three UCI regression data sets, three synthetic regression data sets, and eight different UCI classification data sets (Dua and Graff, 2017). The left and middle plots in Figure 2 present a summary of the results (see Table 1 in Appendix E for a detailed version). We evaluate the CVaR ($\alpha = 0.01$) and average loss for linear regression and the accuracy, and CVaR and average surrogate loss for classification (logistic regression) on the test set. In linear regression, ADA-CVAR performs comparably or better to benchmarks in terms of the CVaR of the loss and is second best in terms of the average loss. In classification, TRUNC-CVAR performs better in terms of the CVaR for the *surrogate loss* but performs *poorly* in terms of accuracy. This is due to the fact that it learns a predictive distribution that is close to uniform. ADA-CVAR has a comparable accuracy to ERM (MEAN algorithm) but a much better CVaR. Hence, it finds a good predictive model while successfully controlling the prediction risk.

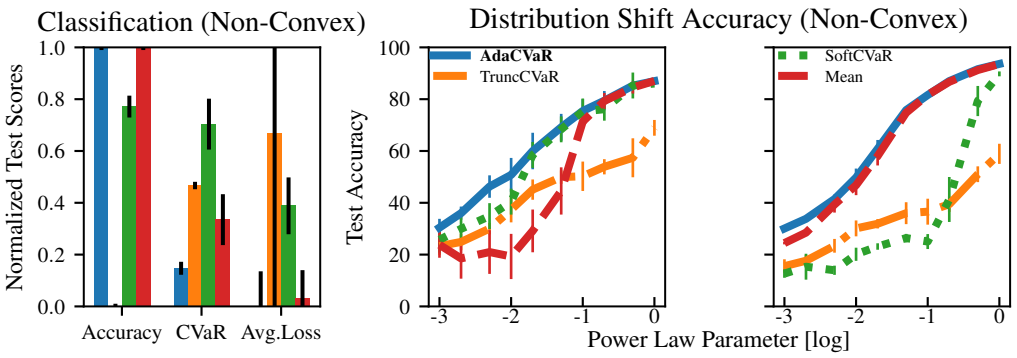

Figure 3: Non Convex Optimization tasks. *Left*: Normalized scores in image classification tasks. ADA-CVAR attains state-of-the-art accuracy and lowest CVaR. *Middle and Right*: Test accuracy under train/test distribution shift on CIFAR-10 for VGG16-BN (middle) and ResNet-18 (right). Lower β indicates larger shift. ADA-CVAR has always better test accuracy than benchmarks.

5.2 Convex CVaR Distributional Robustness

We use the same classification data sets and classifiers as in Section 5.1. To produce the distribution shift, we randomly sub-sample the majority class in the training set, so that the new training set has a 10%/90% class imbalance and the majority/minority classes are inverted. The test set is kept unchanged. Such shifts in class frequencies are common (Mehrabi et al., 2019, Section 3.2).

We consider $\alpha = 0.1$, which is compatible with the data imbalance. The right plot in Figure 2 shows a summary of the results (See Table 2 in Appendix E for detailed results). ADA-CVAR has higher test accuracy than the benchmarks and is comparable to TRUNC-CVAR on average log-likelihood.

We note that the CVaR provides robustness with respect to worst-case distribution shifts. Such a *worst case* distribution might be too pessimistic to be encountered in practice, however. Instead, ADA-CVAR appears to benefit from the varying distributions during training and protects better against non-adversarial distribution shifts. Other techniques for dealing with imbalanced data might also be useful to address this distribution shift empirically, but are only useful if there is an a-priori knowledge of the class ratio in the test set. Instead, the CVaR optimization guards against any distribution shift. Furthermore, with such a-priori knowledge, such techniques can also be used together with ADA-CVAR. We provide extended experiments analyzing distribution shift in Appendix E.1.

5.3 Non-Convex (Deep Learning) CVaR Optimization

We test our algorithm on common non-convex optimization benchmarks in deep learning (MNIST, Fashion-MNIST, CIFAR-10). As it is common in these setting, we perform data-augmentation on the training set. Thus, the effective training set size is infinite. To address this, we consider a mixture of distributions in a similar spirit as Borsos et al. (2019). Each data point serves as a representative of a distribution over all its possible augmentations. We optimize the CVaR of this mixture of distributions as a surrogate of the CVaR of the infinite data set. The left plot in Figure 3 summarizes the results (See Table 3 in Appendix E). ADA-CVAR reaches the same accuracy as ERM in all cases and has lower CVaR. Only in CIFAR-10 it does not outperform TRUNC-CVAR in terms of the CVaR of the surrogate loss. This is because the TRUNC-CVAR yields a predictive model that is close to uniform. Instead, ADA-CVAR still yields useful predictions while controlling the CVaR.

Gradient Magnitude and Training Time The gradients of TRUNC-CVAR are either 0 or $1/\alpha$ times larger than the gradients of the same point using MEAN. A similar but smoothed phenomenon arises with SOFT-CVAR. This makes training these losses considerably harder due to exploding gradients and noisier gradient estimates. With the same learning rates, these algorithms usually produce numerical overflows and, to stabilize learning, we used considerably smaller learning rates. In turn, this increased the number of iterations required for convergence. ADA-CVAR does not suffer from this as the gradients have the same magnitude as in MEAN. For example, to reach 85 % *train* accuracy

ADA-CVAR requires 7 epochs, MEAN 9, SOFT-CVAR 21, and TRUNC-CVAR never surpassed 70 % train accuracy. There was no significant difference between time per epoch of each of the algorithms.

5.4 Distributional Robustness in Deep Learning through Optimizing the CVaR

Lastly, we demonstrate that optimizing the CVaR yields improved robustness to distribution shifts in deep learning. We simulate distribution shift through mismatching training and test class frequencies. Since we consider multi-class problems, we simulate power-law class frequencies, which are commonly encountered in various applications (Clauset et al., 2009). More specifically, we sub-sample each class of the training set of CIFAR-10 so that the class size follows a power-law distribution $p(|c|) \propto c^{\log \beta}$, where $|c|$ is the size of the c-th class and keep the test set unchanged. In middle and right plots of Figure 3, we show the test accuracy for different values of β for VGG16-BN and ResNet-18 networks. The algorithms do not know a-priori the amount of distribution shift to protect against and consider a fixed $\alpha = 0.1$. For all distribution shifts, ADA-CVAR is superior to the benchmarks.

When a high-capacity network learns a perfectly accurate model, then the average and CVaR of the loss distribution have both zero value. This might explain the similarity between MEAN and ADA-CVAR for ResNet-18. Instead, there is a stark discrepancy between ADA-CVAR and MEAN in VGG16. This shows the advantage of training in a risk averse manner, particularly when the model makes incorrect predictions due to a strong inductive bias.

6 Conclusions

The CVaR is a natural criterion for training ML models in a risk-aware fashion. As we saw, the traditional way of optimizing it via truncated losses fails for modern machine learning tasks due to high variance of the gradient estimates. Our novel adaptive sampling algorithm ADA-CVAR exploits the distributionally robust optimization formulation of the CVaR, and tackles it via regret minimization. It naturally enables the use of standard stochastic optimization approaches (e.g., SGD), applied to the marginal distribution of a certain k-DPP. Finally, we demonstrate in a range of experiments that ADA-CVAR is superior to the TRUNC-CVAR algorithm for regression and classification tasks, both in convex and non-convex learning settings. Furthermore, ADA-CVAR provides higher robustness to (non-adversarial) distribution shifts than TRUNC-CVAR, SOFT-CVAR, or MEAN algorithms.

Broader Impact

Increasing *reliability* is one of the central challenges when deploying machine learning in high-stakes applications. We believe our paper makes important contributions to this endeavor by going beyond simply optimizing the *average* performance, and considering risk in deep learning. The CVaR is also known to be an avenue towards enforcing *fairness constraints* in data sets (Williamson and Menon, 2019). Hence, our algorithm also contributes to optimizing fair deep models, by counteracting inherent biases in the data (e.g., undersampling of certain parts of the population).

Acknowledgments and Disclosure of Funding

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research, innovation programme grant agreement No 815943, a DARPA YFA award, and NSF CAREER award 1553284.

References

N. Agarwal, A. Gonen, and E. Hazan. Learning in non-convex games with an optimization oracle. *arXiv:1810.07362*, 2018.

A. Ahmadi-Javid. Entropic value-at-risk: A new coherent risk measure. *Journal of Optimization Theory and Applications*, 155(3):1105–1123, 2012.

P. Alatur, K. Y. Levy, and A. Krause. Multi-player bandits: The adversarial case. *Journal of Machine Learning Research*, 21(77):1–23, 2020.

N. Anari, S. O. Gharan, and A. Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pages 103–115, 2016.

P. Artzner et al. Coherent measures of risk. *Mathematical finance*, pages 203–228, 1999.

J.-Y. Audibert, S. Bubeck, and G. Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2013.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

S. Barthelmé, P.-O. Amblard, N. Tremblay, et al. Asymptotic equivalence of fixed-size and varying-size determinantal point processes. *Bernoulli*, pages 3555–3589, 2019.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Z. Borsos, S. Curi, K. Y. Levy, and A. Krause. Online variance reduction with mixtures. In *International Conference on Machine Learning*, pages 705–714, 2019.

D. B. Brown. Large deviations bounds for estimating conditional value-at-risk. *Operations Research Letters*, 35(6):722–730, 2007.

C. Brownlees, E. Joly, G. Lugosi, et al. Empirical risk minimization for heavy-tailed losses. *The Annals of Statistics*, 43(6):2507–2536, 2015.

S. Bubeck, N. Cesa-Bianchi, and S. M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *Conference on Learning Theory*, pages 41–1, 2012.

M. C. Carneiro et al. Risk management in the oil supply chain: a cvar approach. *Industrial & Engineering Chemistry Research*, pages 3286–3294, 2010.

N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

M. Dereziński, D. Calandriello, and M. Valko. Exact sampling of determinantal point processes with sublinear time preprocessing. *arXiv:1905.13476*, 2019.

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

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

J. Duchi, P. Glynn, and H. Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv:1610.03425*, 2016.

J. P. Eaton and C. A. Haas. *Titanic, triumph and tragedy*. WW Norton & Company, 1995.

P. M. Esfahani and D. Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171 (1-2):115–166, 2018.

Y. Fan, S. Lyu, Y. Ying, and B. Hu. Learning with average top-k loss. In *Advances in Neural Information Processing Systems*, pages 497–505, 2017.

Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

J.-y. Gotoh and A. Takeda. Cvar minimizations in support vector machines. *Financial Signal Processing and Machine Learning*, pages 233–265, 2016.

E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, pages 157–325, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

J. Kirschner, I. Bogunovic, S. Jegelka, and A. Krause. Distributionally robust bayesian optimization. In *The 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *COLT*, pages 93–105, 2010.

A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of risk*, 4:43–68, 2002.

A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

T. Lattimore and C. Szepesvári. Bandit algorithms. *preprint*, 2018.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

J. Lee, S. Park, and J. Shin. Learning bounds for risk-sensitive learning. *arXiv preprint arXiv:2006.08138*, 2020.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

Z. Mhammedi, B. Guedj, and R. C. Williamson. Pac-bayesian bound for the conditional value at risk. *arXiv preprint arXiv:2006.14763*, 2020.

K. G. Murty. *Linear programming*. Springer, 1983.

K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

H. Namkoong and J. C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems*, pages 2208–2216, 2016.

H. Namkoong and J. C. Duchi. Variance-based regularization with convex objectives. In *Advances in Neural Information Processing Systems*, pages 2971–2980, 2017.

A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

W. Ogryczak and A. Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.

A. Paszke et al. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*, 2017.

J. W. Pratt. Risk aversion in the small and in the large. In *Uncertainty in Economics*, pages 59–79. Elsevier, 1978.

M. Rabin. Risk aversion and expected-utility theory. In *Handbook of the Fundamentals of Financial Decision Making*, pages 241–252. World Scientific, 2013.

R. T. Rockafellar, S. Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2: 21–42, 2000.

G. Rubino and B. Tuffin. *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

A. Sani, A. Lazaric, and R. Munos. Risk-aversion in multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 3275–3283, 2012.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

S. Shalev-Shwartz and Y. Wexler. Minimizing the maximal loss: How and why. In *ICML*, pages 793–801, 2016.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

T. Soma and Y. Yoshida. Statistical learning with conditional value at risk. *arXiv preprint arXiv:2002.05826*, 2020.

T. Tarnopolskaya and Z. Zhu. Cvar-minimising hedging by a smoothing method. *ANZIAM*, 52: 237–256, 2010.

T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pages 375–389. Springer, 2010.

V. Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

R. Williamson and A. Menon. Fairness risk measures. In *International Conference on Machine Learning*, pages 6786–6797, 2019.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

D. W. Zimmerman. Teacher's corner: A note on interpretation of the paired-samples t test. *Journal of Educational and Behavioral Statistics*, 22(3):349–360, 1997.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.

A Approximate Sampling from k-DPP Marginals

The final piece of the ADA-CVaR algorithm is how to efficiently compute the marginal distribution $\mathbb{P}_w(i)$ of the k-DPP model, and how to sample from it.

Cost of sampling Compared to general k-DPPs, our setting has the crucial advantage that the kernel matrix is diagonal. Thus there is no need for performing an expensive eigendecomposition of the kernel matrix which is required in the general case. The marginals of diagonal k-DPPs are

$$\mathbb{P}_w(i) = w_i \frac{e_{-i}^{k-1}}{e_N^k}, \tag{9}$$

where $e_N^k = \sum_{|I|=k} \prod_{i \in I} w_i$ is the elementary symmetric polynomial of size k for the ground set $[N]$ and e_{-i}^{k-1} is the elementary symmetric polynomial of size $k-1$ for the ground set $[N] \backslash i$. Unfortunately, naively computing the elementary symmetric polynomials has a complexity of $O(N^2 k) = O(\alpha N^3)$ using (Kulesza et al., 2012, Algorithm 7). Even if this computation could be performed fast, exact computation of the elementary symmetric polynomials is numerically unstable.

In view of this, Barthelmé et al. (2019) propose an approximation to k-DPPs valid for large-scale ground sets which has better numerical properties. Their main idea is to relax the sample size constraint of the k-DPP with a soft constraint such that the *expected* sample size of the matched DPP is k. The total variation distance between the marginal probabilities of the k-DPP and DPP decays as $O(1/N)$. The marginal probabilities of this matched DPP are simply

$$\hat{\mathbb{P}}_w(i) = \frac{w_i e^{\nu}}{1 + w_i e^{\nu}}, \tag{10}$$

where ν softly enforces the sample size constraint $\sum_{i=1}^N \frac{w_i e^{\nu}}{1+w_i e^{\nu}} = k$. For a given ν, we can efficiently sample from this singleton-marginal distribution, which takes $O(\log(N))$ using the same sum-tree data-structure as Shalev-Shwartz and Wexler (2016).

Proposition 2 (Excess Regret of Approximate Sampler)**.** *Let $\tilde{q}$ be the marginals of the approximate k-DPP* (10) *and q the exact marginals of the k-DPP* (9)*. Then, a sampler that plays* ADA-CVaR *using the approximate k-DPPs marginals suffers an extra regret of $\epsilon_{\text{approx}} T$, with $\epsilon_{\text{approx}} = O(1/N)$, for large enough N.*

Proof. Let $\widetilde{\text{SR}}_T$ be the regret of the player that selects the approximate k-DPPs marginals. Then,

$$\begin{aligned}
\widetilde{\text{SR}}_T &\coloneqq \max_{q \in \mathcal{Q}^\alpha} \sum_{t=1}^T q^\top L_t - \sum_{t=1}^T \tilde{q}_t^\top L_t \\
&= \text{SR}_T + \sum_{t=1}^T (\tilde{q}_t - q_t)^\top L_t \\
&\leq \text{SR}_T + \sum_{t=1}^T \|\tilde{q}_t - q_t\|_1 \|L_t\|_\infty \\
&\leq \text{SR}_T + \sum_{t=1}^T \|\tilde{q}_t - q_t\|_1 \\
&\leq \text{SR}_T + \epsilon_{\text{approx}} T
\end{aligned}$$

where the first inequality holds due to Hölder's inequality, the second inequality due to $L \in [0,1]$, and finally noticing that $\|\tilde{q}_t - q_t\|_1$ is proportional to the total variation distance. Using Theorem 2.1 from Barthelmé et al. (2019) the results follow. □

We note that this result implies an an extra $O(1/N)$ term in the excess CVaR bounds from Corollaries 1 and 2. This is fast compared to statistical $O(1/\sqrt{N})$ rates. For small N, the exact marginals can be computed efficiently so there is no need for the approximation.

B Omitted Proofs

B.1 Proof of Proposition 1

Proposition 1. Let $h : \mathcal{X} \to \mathcal{Y}$ be a function class with finite VC-dimension $|\mathcal{H}|$. Let $L(h) : \mathcal{H} \to [0, 1]$ be a random variable. Then, for any $0 < \alpha \leq 1$, with probability at least $1 - \delta$,

$$\mathbb{E}\left[\sup_{h\in\mathcal{H}}\left|\widehat{\mathbb{C}}^{\alpha}[L(h)] - \mathbb{C}^{\alpha}[L(h)]\right|\right] \leq \frac{1}{\alpha}\sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}.$$

Proof for Proposition 1. Brown (2007) proves that the following two inequalities hold jointly with probability $1 - \delta$, $\delta \in (0, 1]$ for a single $h \in \mathcal{H}$:

$$\mathbb{C}^{\alpha}(L(h)) \geq \widehat{\mathbb{C}}^{\alpha}(L(h)) - \frac{1}{\alpha}\sqrt{\frac{\log(2/\delta)}{N}}$$
$$\mathbb{C}^{\alpha}(L(h)) \leq \widehat{\mathbb{C}}^{\alpha}(L(h)) + \sqrt{\frac{5\log(6/\delta)}{\alpha N}}$$

Taking the union bound over all $h \in \mathcal{H}$:

$$\mathbb{C}^{\alpha}(L(h)) \geq \widehat{\mathbb{C}}^{\alpha}(L(h)) - \frac{1}{\alpha}\sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}$$
$$\mathbb{C}^{\alpha}(L(h)) \leq \widehat{\mathbb{C}}^{\alpha}(L(h)) + \sqrt{\frac{5\log(6|\mathcal{H}|/\delta)}{\alpha N}}$$

The theorem follows from taking the maximum between lower and upper bounds.

If $\mathcal{H}$ is a non-finite class we follow a standard argument with bounded norm of real valued functions, i.e., $\|\mathcal{H}\|_{\infty} \leq B$, we rely on the standard argument based on covering numbers. To define such covering, we fix $\epsilon > 0$ and consider a set $\mathcal{C}_{\mathcal{H},\epsilon}$ of minimum cardinality, such that for all $h \in \mathcal{H}$, there exists an $h' \in \mathcal{C}_{\mathcal{H},\epsilon}$, satisfying $|h - h'| \leq \epsilon$. Define the covering number as $\mathcal{N}_{\mathcal{H},\epsilon} = |\mathcal{C}_{\mathcal{H},\epsilon}|$, then taking the union bound for $\epsilon = 1\sqrt{N}$ we arrive to the result

$$\mathbb{E}\left[\sup_{h\in\mathcal{H}}\left|\widehat{\mathbb{C}}^{\alpha}[L(h)] - \mathbb{C}^{\alpha}[L(h)]\right|\right] \leq \frac{1}{\alpha}\sqrt{\frac{\log(2\mathcal{N}_{\mathcal{H},\epsilon}/\delta)}{N}}.$$

Finally, the logarithm of the covering number can be upper bounded using the pseudo-VC dimension of the function class, for a proof see Devroye et al. (2013, Chapter 29).

□

B.2 Proof of Lemma 1

Lemma 1. Let the sampler player play the K.EXP.3 Algorithm with $\eta = \sqrt{\frac{\log N}{NT}}$. Then, she suffers a sampler regret (5) of at most $O(\sqrt{TN\log N})$.

In order to prove this, we need to first show that K.EXP.3 is a valid algorithm for the sampler player. This we do next.

Proposition 3. *The marginals of any k-DPP with a diagonal matrix kernel $K = \mathrm{diag}(w)$ are in the set $\mathcal{Q}_k^{\alpha} = \left\{kq \in \mathbb{R}^N \mid q \in \mathcal{Q}^{\alpha}\right\}$.*

Proof. For any $w \in \mathbb{R}^N_{\geq 0}$ the marginals of the k-DPP with kernel $K = \mathrm{diag}(w)$ are:

$$\mathbb{P}_w(i) = \sum_{I\ni i}\mathbb{P}_w(I) = \frac{\sum_{I\ni i}\prod_{i'\in I} w_{i'}}{\sum_I \prod_{i'\in I} w_{i'}}. \tag{11}$$

From eq. (11), clearly $0 \leq \mathbb{P}_w(i) \leq 1$. Summing eq. (11) over i we get:

$$\sum_i \mathbb{P}_w(i) = \sum_i\sum_{I\ni i}\mathbb{P}_w(I) = \sum_I \mathbb{P}_w(I)\sum_{i\in I} 1 = k$$

This shows that $\mathbb{P}_w(i) \in \mathcal{Q}_k^{\alpha}$. □

Proposition 4. *Let* $\tilde{L}_I = \sum_{i \in I} L_i$. *Let* $\Delta = \left\{ \tilde{w} \in \mathbb{R}^{\binom{N}{k}} \mid, 0 \leq w_I \leq 1, \sum_I w_I = 1 \right\}$ *the set of distributions over the* $\binom{N}{k}$ *subsets of size* k *of the ground set* $[N]$.

$$\max_{q \in \mathcal{Q}^\alpha} \sum_{t=1}^{T} q^\top L_t = \max_{\tilde{w} \in \Delta} \frac{1}{k} \sum_{t=1}^{T} \tilde{w} \tilde{L}_I \tag{12}$$

Proof. Both left and right sides of (12) are linear programs over a convex polytope, hence the solution is in one of its vertices (Murty, 1983). The vertices of $\mathcal{Q}^\alpha$ are vectors $\frac{1}{k}\mathbf{1}_I = \frac{1}{k}[[i \in I]]$. These vectors have $\frac{1}{k}$ in coordinate i if the coordinate belongs to set I and 0 otherwise. The vertices of the simplex are just $[[I]]$, one for coordinate I.

Let $q^\star = \frac{1}{k}\mathbf{1}_{I^\star}$ be the solution of the l.h.s. of (12). Assume that $\hat{I} \neq I^\star$ is the solution of the right hand side. This implies that $\tilde{L}_{\hat{I}} \geq \tilde{L}_{I^\star}$. Therefore, $\sum_{i \in \hat{I}} L_i \geq \sum_{i \in I^\star} L_i$. This in turn implies that $\mathbf{1}_{\hat{I}} L_i \geq \mathbf{1}_{I^\star} L_i$, which contradicts the first predicate. In the case the equalities hold, then the values l.h.s and r.h.s. of equation (12) are also equal. □

Proof of Lemma 1.

$$\begin{aligned}
\mathrm{SR}_T &= \max_{q \in \mathcal{Q}^\alpha} \sum_{t=1}^{T} q^\top L_t - \sum_{t=1}^{T} q_t^\top L_t \\
&= \max_{\tilde{w} \in \Delta} \frac{1}{k} \sum_{t=1}^{T} \tilde{w} \tilde{L}_I - \sum_{t=1}^{T} q_t^\top L_t \\
&= \frac{1}{k} \left(\max_{\tilde{w} \in \Delta} \sum_{t=1}^{T} \tilde{w} \tilde{L}_I - \sum_{t=1}^{T} \sum_{i} \mathbb{P}_{w_t}(i) L_i \right) \\
&= \frac{1}{k} \left(\max_{\tilde{w} \in \Delta} \sum_{t=1}^{T} \tilde{w} \tilde{L}_I - \sum_{t=1}^{T} \sum_{I} \mathbb{P}_{w_t}(I) \tilde{L}_I \right) \\
&\leq O(\sqrt{NT \log(N)})
\end{aligned}$$ □

The first equality uses Proposition 4. The second equality uses Proposition 3 and the fact that the iterates q_t come from the K.EXP.3 algorithm. The third equality uses the definition of $\tilde{L}$. The final inequality is due to Alatur et al. (2020, Lemma 1).

B.3 Proof of Lemma 2

Lemma 2. A learner player that plays the BTL algorithm with access to an ERM oracle as in Assumption 1, achieves at most $\epsilon_{\mathrm{oracle}} T$ regret.

Proof. We proceed by induction. Clearly for $T = 1$, $\mathrm{LR}_1 \leq \epsilon_{\mathrm{oracle}}$. Assume true for $T - 1$, the inductive hypothesis is $\mathrm{LR}_{T-1} \leq \epsilon_{\mathrm{oracle}}(T-1)$. The regret at time T is:

$$\begin{aligned}
\mathrm{LR}_T &= \sum_{t=1}^{T} q_t^\top L(\theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} q_t^\top L(\theta), \\
&\leq \sum_{t=1}^{T} q_t^\top \left[L(\theta_t) - L(\theta_T) \right] + \epsilon_{\mathrm{oracle}} \\
&= \sum_{t=1}^{T-1} q_t^\top \left[L(\theta_t) - L(\theta_T) \right] + \epsilon_{\mathrm{oracle}} \\
&\leq \mathrm{LR}_{T-1} + \epsilon_{\mathrm{oracle}} \leq \epsilon_{\mathrm{oracle}} T
\end{aligned}$$ □

The first inequality is due to the definition of the oracle, the second inequality is by definition of the minimum in the learner regret, and the final inequality is due to the inductive hypothesis.

B.4 Proof of Theorem 1

Theorem 1. Let $L_i(\cdot) : \Theta \to [0, 1]$, $i = \{1, ..., N\}$ be a fixed set of loss functions. If the sampler player uses ADA-CVAR and the learner player uses the BTL algorithm, then the game has regret $O(\sqrt{TN \log N})$.

Proof. We decompose bound the game regret into the sum of player and sampler regret. To bound the regret, we bound it with the sum of the Learner and Sampler regret as follows:

$$\begin{aligned}
\mathrm{GameRegret}_T &= \sum\nolimits_{t=1}^{T} J(\theta_t, q^\star) - J(\theta^\star, q_t), \\
&\leq \max_{q \in \mathcal{Q}^\alpha} \sum\nolimits_{t=1}^{T} J(\theta_t, q) - J(\theta^\star, q_t), \\
\text{(Lemma 2)} \ &\leq \max_{q \in \mathcal{Q}^\alpha} \sum\nolimits_{t=1}^{T} J(\theta_t, q) - J(\theta_t, q_t), \\
\text{(Lemma 1)} \ &\leq O(\sqrt{TN \log N} + \epsilon_{\mathrm{oracle}} T).
\end{aligned}$$

□

B.5 Proof of Corollary 1

Corollary 1 (Online to Batch Conversion)**.** Let $L_i(\cdot) : \Theta \to [0, 1]$, $i = \{1, ..., N\}$ be a set of loss functions sampled from a distribution $\mathcal{D}$. Let $\theta^\star$ be the minimizer of the CVaR of the empirical distribution $\hat{\mathbb{C}}^\alpha$. Let $\bar{\theta}$ be the output of ADA-CVAR, selected uniformly at random from the sequence $\{\theta_t\}_{t=1}^T$. Its expected excess CVaR is bounded as:

$$\mathbb{E}\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] \leq \hat{\mathbb{C}}^\alpha[L(\theta^\star)] + + O(\sqrt{N \log N / T}) + \epsilon_{\mathrm{oracle}}$$

where the expectation is taken w.r.t. the randomization in the algorithm, both for the sampling steps and the final randomization in choosing $\bar{\theta}$.

Proof. We use the proof techinque from Agarwal et al. (2018)[Theorem 3]. The CVaR of $\bar{\theta}$ is $\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] = \max_{q \in \mathcal{Q}^\alpha} J(\bar{\theta}, q)$. Let $\bar{q}^\star \coloneqq \arg\max_{q \in \mathcal{Q}^\alpha} J(\bar{\theta}, q)$. For any $q \in \mathcal{Q}^\alpha$, we can use the sampler regret to bound:

$$\begin{aligned}
\mathbb{E}\left[J(\bar{\theta}, q)\right] &= \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} J(\theta_t, q)\right] \\
&\leq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} J(\theta_t, q_t)\right] + \frac{1}{T}\,\mathrm{SR}_T
\end{aligned}$$

Likewise, for any $\theta \in \Theta$,

$$\begin{aligned}
\mathbb{E}\left[J(\theta, \bar{q})\right] &= \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} J(\theta, q_t)\right] \\
&\geq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} J(\theta_t, q_t)\right] - \frac{1}{T}\,\mathrm{LR}_T
\end{aligned}$$

Using the minimax inequality, we bound the excess risk for any $q \in \mathcal{Q}$ and $\theta \in \Theta$ by the average game regret:

$$\begin{aligned}
\mathbb{E}\left[J(\bar{\theta}, q)\right] &\leq J(\theta^\star, q^\star) + \mathbb{E}\left[J(\bar{\theta}, q) - J(\theta, \bar{q})\right] \\
&\leq J(\theta^\star, q^\star) + \underbrace{\frac{1}{T}\left(\mathrm{LR}_T + \mathrm{SR}_T\right)}_{\epsilon}
\end{aligned}$$

Noting that, $\mathrm{GameRegret}_T = \mathrm{LR}_T + \mathrm{SR}_T$ and that the latter results holds in particular for $\bar{q}^\star$. Furthermore, the pair $(\bar{\theta}, \bar{q})$ is an ϵ-equilibrium point of the game in the sense: $J(\theta^\star, q^\star) - \epsilon \leq \mathbb{E}\left[J(\bar{\theta}, \bar{q})\right] \leq J(\theta^\star, q^\star) + \epsilon$. □

B.6 Proof of Corollary 2

Corollary 2 (Online to Batch Conversion)**.** Let $L(\cdot) : \Theta \to [0, 1]$ be a Random Variable induced by the data distribution $\mathcal{D}$. Let $\theta^\star$ be the minimizer of the CVaR at level α of such Random Variable. Let $\bar{\theta}$ be the output of ADA-CVAR, selected uniformly at random from the sequence $\{\theta_t\}_{t=1}^T$. Then, with probability at least δ the expected excess CVaR of $\bar{\theta}$ is bounded as:

$$\mathbb{E}\mathbb{C}^\alpha[L(\bar{\theta})] \leq \mathbb{C}^\alpha[L(\theta^\star)] + O(\sqrt{N \log N/T}) + \epsilon_{\text{oracle}} + \epsilon_{\text{stat}}$$

where $\epsilon_{\text{stat}} = \tilde{O}(\frac{1}{\alpha}\sqrt{\frac{1}{N}})$ comes from the statistical error and the expectation is taken w.r.t. the randomization in the algorithm, both for the sampling steps and the randomization in choosing $\bar{\theta}$.

Proof.

$$\begin{aligned}
\mathbb{E}\mathbb{C}^\alpha[L(\bar{\theta})] &\leq \mathbb{E}\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] + \epsilon_{\text{stat}} \\
&\leq \mathbb{E}\hat{\mathbb{C}}^\alpha[L(\theta^\star)] + O(\sqrt{N \log N/T}) + \epsilon_{\text{oracle}} + \epsilon_{\text{stat}} \\
&\leq \mathbb{E}\mathbb{C}^\alpha[L(\theta^\star)] + O(\sqrt{N \log N/T}) + \epsilon_{\text{oracle}} + 2\epsilon_{\text{stat}},
\end{aligned}$$

Where the first and last inequality hold by the uniform convergence results from Proposition 1 and the second inequality by Corollary 1. □

C Learner Player Algorithm for Convex Losses

In the convex setting, there are online learning algorithms that have no-regret guarantees and there is no need to play the BTL algorithm (7) exactly. Instead, stochastic gradient descent (SGD) Zinkevich (2003) or online mirror descent (OMD) (Beck and Teboulle, 2003) both have no-regret guarantees. We focus now on SGD but, for certain geometries of Θ and appropriate mirror maps, OMD has exponentially better regret guarantees (in terms of the dimension of the problem).

Algorithm 2: Ada-CVaR-CVX

input Learning rates η_s, η_l.
1: **Sampler:** Initialize k-DPP $w_1 = \mathbf{1}_N$.
2: **Learner:** Initialize parameters $\theta_1 \in \Theta$.
3: **for** $t = 1, \ldots, T$ **do**
4: **Sampler:** Sample element $i_t \sim q_t = \frac{1}{k}\mathbb{P}_{w_t}(i)$.
5: **Sampler**: Build estimate $\hat{L}_t = \frac{L_{i_t}(\theta_t)}{q_{t,i_t}}[[i == i_t]]$.
6: **Sampler**: Update k-DPP $w_{t+1,i} = w_{t,i}e^{\eta_s \hat{L}_{t,i}}$.
7: **Learner:** $\theta_{t+1} = \theta_t - \eta_l \nabla L_{i_t}(\theta_t)$.
8: **end for**
output $\bar{\theta} = \frac{1}{T}\sum_{t=1}^T \theta_t$, $\bar{q} = \frac{1}{T}\sum_{t=1}^T q_t$

Lemma 3. *Let assume that $L_i(\cdot) : \Theta \to [0, 1]$ be any sequence of convex losses, with $\|\nabla L_i\|_2 \leq G$ and $\|\Theta\|_2 \leq D$, then a learner player that plays SGD algorithm suffers at most regret $O(GD\sqrt{T})$.*

Proof. Hazan (2016, Chapter 3). □

Note that even if there are algorithms for the strongly convex case or exp-concave case that have $\log(T)$ regret, it does not bring any advantage in our case as the $\sqrt{T}$ term in the sampler regret dominates and is unavoidable (Audibert et al., 2013).

Corollary 3. *Let $L_i(\cdot) : \Theta \to [0, 1]$ be any sequence of convex losses. Let the learner and sampler player play Algorithm 2, then the game has regret $O(\sqrt{TN \log N} + \beta\sqrt{T})$, where β is a problem-dependent constant.*

This result changes the ϵ_{oracle} term in Corollaries 1 and 2 by a $O(1/\sqrt{T})$ term.

D Experimental Setup

Implementation and Resources: We implemented all our experiments using PyTorch (Paszke et al., 2017). We ran our experiments convex experiments on an Intel(R) Xeon(R) CPU E5-2697 v4 2.30GHz machine. Our deep lerning experiments ran on an NVIDIA GeForce GTX 1080 Ti GPU.

Datasets: For classification we use the Adult, Australian Credit Approval, German Credit Data, Monks-Problems-1, Spambase, and Splice-junction Gene Sequences datasets from the UCI repository (Dua and Graff, 2017) and the Titanic Disaster dataset from (Eaton and Haas, 1995). For regression we use the Boston Housing, and Abalone, and CPU small from the UCI repository, the sinc dataset is synthetic recreated from (Fan et al., 2017), and normal and pareto datasets are synthetic datasets recreated from (Brownlees et al., 2015) with Gaussian and Pareto noise, respectively. For vision datasets, we use MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky et al., 2014).

Models: For convex regression and classification models we use linear models. For MNIST we use LeNet-5 neural network (LeCun et al., 1995). For Fashion MNIST we use LeNet-5 using dropout (Hinton et al., 2012). For CIFAR-10 we use ResNet18 (He et al., 2016) neural network and VGG-16 (Simonyan and Zisserman, 2014) with batch normalization (Ioffe and Szegedy, 2015).

Dataset Preparation: We split UCI datasets into train, validation, and test set using a 50/30/20 split. For vision tasks we use as validation set the same images in the train set, without applying data-augmentations. For discrete categorical data, we use a one-hot-encoding. We standarize continuous data.

Hyper-Parameter Search: We ran a grid search over the hyperparameters for all the algorithms with a five different random seeds. In regression tasks, we selected the set of hyper-parameters with the lowest CVaR in the validation set. In classification tasks, we selected the set of hyper-parameters with the highest accuracy in the validation set. The hyperparameters are:

1. Optimizer SGD with momentum or ADAM (Kingma and Ba, 2014).
2. Initial learning rates:
   $\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00001\}$,
3. Momentum: 0.9
4. Learning rate decay: 0.1 at epochs 20 and 40.
5. Batch Size $\{64, 128\}$,
6. Adaptive algorithm learning rate: $\{1.0, 0.5, 0.1, *\}$, where (*) is the optimal learning rate,
7. Mixing with uniform distribution: $\{0, 0.01, 0.1\}$,
8. Adaptive algorithm learning rate decay scheduling: $\{\text{constant}, O(1/\sqrt{t}), \text{Adagrad}\}$,
9. SOFT-CVAR algorithm temperature: $\{0.1, 1, 10\}$.
10. Random seeds: $\{0, 1, 2, 3, 4\}$.
11. Early stopping: $\{\text{True}, \text{False}\}$.

Experimental Significance: In UCI and synthetic datasets, the test results of each algorithm is paired because the same data split is used across the algorithms. Likewise, for vision datasets, the neural network initialization is paired across experiments. Therefore, we use a paired t-test to determine statistical significance for $p \leq 0.05$ (Zimmerman, 1997). In TRUNC-CVAR and SOFT-CVAR experiments, we usually encountered numerical overflows, we discarded such experiments to determine significance.

Evaluation Metric Normalization We evaluate the CVaR and the Average Loss for regression and we add Accuracy, and the CVaR and the Average Loss of the surrogate loss for classification. We normalize the mean score s of algorithm a on a given task by $(s_a - \min_a s_a)/(\max_a s_a - \min_a s_a)$ and we divide the standard deviation by $\max_a s_a$.

E Extended Experimental Results

In Table 1 we show the results of Section 5.1, in Table 2 the results of Section 5.2, and in Table 3 the results of Section 5.3.

Table 1: Test mean $\pm$ s.d. over five independent data splits. In shaded bold we indicate the best algorithms. In regression, we show the CVaR/(mean) loss. ADA-CVAR is competitive to benchmarks optimizing the CVaR. In classification, we show the accuracy / precision in "Accuracy" rows and CVaR/loss in "Surrogate" rows. ADA-CVAR has similar accuracy to MEAN/SOFT-CVAR, but with a lower CVaR.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Regression Loss	Abalone	**8.92 ± 3.3**	0.61 ± 0.1	**7.94 ± 2.7**	0.79 ± 0.1	14.25 ± 0.3	0.63 ± 0.0	11.07 ± 3.7	**0.51 ± 0.1**
	Boston	**3.09 ± 0.8**	0.28 ± 0.1	**3.02 ± 1.0**	0.36 ± 0.0	3.28 ± 1.4	**0.24 ± 0.1**	4.51 ± 1.9	**0.27 ± 0.1**
	Cpu	**2.32 ± 0.2**	0.58 ± 0.0	**2.12 ± 0.2**	0.57 ± 0.1	2.85 ± 0.0	0.40 ± 0.0	9.95 ± 1.4	**0.30 ± 0.0**
	Normal	**0.22 ± 0.1**	0.03 ± 0.0	0.42 ± 0.3	0.06 ± 0.0	**0.18 ± 0.0**	**0.01 ± 0.0**	0.55 ± 0.2	0.07 ± 0.0
	Pareto	**0.39 ± 0.3**	0.02 ± 0.0	0.43 ± 0.1	0.05 ± 0.0	**0.30 ± 0.3**	**0.01 ± 0.0**	0.69 ± 0.1	0.06 ± 0.0
	Sinc	**7.70 ± 3.1**	**0.80 ± 0.2**	**7.82 ± 3.5**	**0.74 ± 0.2**	**7.82 ± 3.5**	**0.72 ± 0.2**	8.40 ± 3.9	**0.71 ± 0.2**
Classification Accuracy	Adult	**0.85 ± 0.0**	0.72 ± 0.0	0.71 ± 0.1	0.44 ± 0.2	**0.85 ± 0.0**	**0.74 ± 0.0**	**0.85 ± 0.0**	**0.74 ± 0.0**
	Australian	**0.82 ± 0.0**	**0.79 ± 0.0**	0.66 ± 0.1	0.62 ± 0.1	**0.82 ± 0.0**	**0.81 ± 0.0**	0.81 ± 0.0	0.78 ± 0.0
	German	0.74 ± 0.0	0.65 ± 0.0	0.64 ± 0.0	0.43 ± 0.0	**0.78 ± 0.0**	**0.71 ± 0.0**	**0.77 ± 0.0**	**0.67 ± 0.1**
	Monks	**0.62 ± 0.1**	**0.59 ± 0.1**	0.53 ± 0.1	0.50 ± 0.1	**0.63 ± 0.0**	0.68 ± 0.0	0.61 ± 0.1	0.66 ± 0.1
	Phoneme	**0.75 ± 0.0**	**0.59 ± 0.0**	0.47 ± 0.1	0.26 ± 0.0	**0.75 ± 0.0**	**0.60 ± 0.0**	**0.76 ± 0.0**	**0.60 ± 0.0**
	Spambase	0.90 ± 0.0	0.88 ± 0.0	0.81 ± 0.0	0.71 ± 0.0	**0.93 ± 0.0**	**0.92 ± 0.0**	**0.92 ± 0.0**	0.91 ± 0.0
	Splice	**0.94 ± 0.0**	**0.93 ± 0.0**	0.90 ± 0.0	0.89 ± 0.0	0.92 ± 0.0	0.91 ± 0.0	0.93 ± 0.0	0.92 ± 0.0
	Titanic	**0.78 ± 0.0**	**0.74 ± 0.0**	0.55 ± 0.2	0.40 ± 0.3	**0.78 ± 0.0**	**0.75 ± 0.0**	**0.78 ± 0.0**	**0.75 ± 0.0**
Classification Surrogate	Adult	1.95 ± 0.1	0.37 ± 0.0	**0.70 ± 0.0**	0.69 ± 0.0	3.08 ± 0.1	**0.32 ± 0.0**	3.13 ± 0.1	0.32 ± 0.0
	Australian	1.33 ± 0.1	0.49 ± 0.0	**0.83 ± 0.0**	0.66 ± 0.0	1.78 ± 0.1	0.47 ± 0.0	1.93 ± 0.1	**0.45 ± 0.0**
	German	1.41 ± 0.2	0.56 ± 0.0	**0.86 ± 0.0**	0.67 ± 0.0	2.04 ± 0.2	**0.50 ± 0.0**	1.90 ± 0.2	**0.51 ± 0.0**
	Monks	**0.89 ± 0.0**	**0.66 ± 0.0**	**0.88 ± 0.0**	0.68 ± 0.0	1.36 ± 0.1	0.69 ± 0.0	1.07 ± 0.0	**0.66 ± 0.0**
	Phoneme	0.83 ± 0.0	0.64 ± 0.0	**0.69 ± 0.0**	0.69 ± 0.0	2.56 ± 0.0	**0.47 ± 0.0**	2.63 ± 0.0	**0.47 ± 0.0**
	Spambase	**1.04 ± 0.1**	0.49 ± 0.0	1.15 ± 0.3	0.49 ± 0.0	6.30 ± 1.7	**0.23 ± 0.0**	5.23 ± 1.4	**0.23 ± 0.0**
	Splice	1.57 ± 0.2	0.27 ± 0.0	**1.06 ± 0.1**	0.49 ± 0.0	5.45 ± 0.9	**0.20 ± 0.0**	1.75 ± 0.2	**0.22 ± 0.0**
	Titanic	0.78 ± 0.0	0.66 ± 0.0	**0.71 ± 0.0**	0.70 ± 0.0	1.70 ± 0.0	**0.52 ± 0.0**	1.68 ± 0.0	**0.52 ± 0.0**

Table 2: Test accuracy/loss (mean $\pm$ s.d.) over five independent data splits with *train/test distribution shift*. In shaded bold we indicate the best algorithms. ADA-CVAR has superior test accuracy than benchmarks. It has comparable test loss to TRUNC-CVAR.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Distribution Shift 10%	Adult	**0.65 ± 0.0**	**0.63 ± 0.0**	0.61 ± 0.1	0.69 ± 0.0	0.63 ± 0.1	0.84 ± 0.3	0.60 ± 0.1	0.90 ± 0.4
	Australian	**0.68 ± 0.0**	**0.58 ± 0.0**	0.46 ± 0.1	0.70 ± 0.0	0.48 ± 0.3	0.91 ± 0.4	0.46 ± 0.3	0.87 ± 0.3
	German	**0.48 ± 0.1**	**0.73 ± 0.0**	0.44 ± 0.1	0.75 ± 0.0	0.45 ± 0.1	1.00 ± 0.2	**0.48 ± 0.1**	0.80 ± 0.1
	Monks	**0.49 ± 0.0**	0.72 ± 0.0	0.39 ± 0.1	0.73 ± 0.0	0.43 ± 0.3	1.27 ± 0.6	0.40 ± 0.2	0.91 ± 0.3
	Phoneme	**0.58 ± 0.1**	**0.68 ± 0.0**	0.52 ± 0.2	0.69 ± 0.0	0.51 ± 0.2	1.08 ± 0.4	0.55 ± 0.2	0.92 ± 0.3
	Spambase	**0.77 ± 0.0**	**0.46 ± 0.0**	0.74 ± 0.1	0.60 ± 0.0	0.71 ± 0.2	0.61 ± 0.2	0.69 ± 0.2	0.58 ± 0.2
	Splice	**0.84 ± 0.0**	**0.40 ± 0.0**	0.77 ± 0.1	0.57 ± 0.0	0.73 ± 0.1	0.52 ± 0.2	0.50 ± 0.3	0.81 ± 0.4
	Titanic	0.46 ± 0.2	**0.70 ± 0.0**	**0.50 ± 0.3**	**0.69 ± 0.0**	0.43 ± 0.3	1.17 ± 0.5	0.41 ± 0.3	0.87 ± 0.3

Table 3: "Accuracy" rows: Avg. test accuracy/(min class) precision. "Surrogate" rows: Avg. test CVaR/loss. Average over 5 different random seeds. ADA-CVAR matches the accuracy and average loss performance of MEAN, but has a lower CVaR. TRUNC-CVAR and SOFT-CVAR struggle to learn in non-convex tasks.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Accuracy	MNIST	**0.99 ± 0.0**	**0.98 ± 0.0**	**0.99 ± 0.0**	**0.98 ± 0.0**	**0.99 ± 0.0**	**0.98 ± 0.0**	**0.99 ± 0.0**	**0.98 ± 0.0**
	Fashion-MNIST	**0.99 ± 0.0**	**0.99 ± 0.0**	**0.99 ± 0.0**	0.98 ± 0.0	**0.99 ± 0.0**	0.98 ± 0.0	**0.99 ± 0.0**	0.97 ± 0.0
	Cifar-10	**0.94 ± 0.0**	**0.87 ± 0.0**	0.59 ± 0.0	0.50 ± 0.0	0.86 ± 0.1	0.64 ± 0.2	**0.94 ± 0.0**	**0.87 ± 0.0**
Surrogate	MNIST	**0.31 ± 0.0**	**0.03 ± 0.0**	0.35 ± 0.1	0.03 ± 0.0	0.41 ± 0.1	0.04 ± 0.0	0.33 ± 0.0	**0.03 ± 0.0**
	Fashion-MNIST	**0.31 ± 0.0**	**0.03 ± 0.0**	0.67 ± 0.0	0.14 ± 0.0	0.35 ± 0.0	0.04 ± 0.0	0.38 ± 0.0	0.04 ± 0.0
	Cifar-10	2.36 ± 0.1	**0.25 ± 0.1**	**2.02 ± 0.0**	1.32 ± 0.0	2.79 ± 0.1	0.32 ± 0.0	2.49 ± 0.1	**0.24 ± 0.0**

E.1 Comparison to other Techniques that Address Distribution Shift

In section 5.2, how ADA-CVAR and the other CVaR optimization algorithms guarded against distribution shift. Now, we compare how do this algorithms compare with algorithms that address distribution shift. We compare against up-sampling and down-sampling techniques for distribution shift for *all* the benchmarks considered in the experiment section. Namely, we perform up-sampling coupled with ADA-CVAR, TRUNC-CVAR, SOFT-CVAR, and MEAN. We expect up-sampling techniques to perform better than raw CVaR optimization when the distribution shift is random *and* the test distribution is *balanced*.

The CVaR guards against *worst-case* distribution shifts for all distributions in the *DRO* set. Certainly, balanced test-sets are contained in the *DRO*, but the *worst-case* guarantees that CVaR brings might be too pessimistic for this particular case. We expect specialized techniques to perform better. Nonetheless, the privileged information that the test-set is balanced might not always be available. In such cases, it could be preferable to optimize for the CVaR of the up-sampled data set, possibly for a large value of α.

Train-Set Shift In this subsection, we repeat the distribution shift experiment. Here, the train set is shifted (1-to-10 ratio) and the test-set is kept constant. In sake of presentation clarity, we show again the results in table 4. We repeat the experiments but upsampling the training set to balance the dataset. We show the results in table 5. When we compare MEAN with upsampling (last column in table 5) and ADA-CVAR without upsampling (first column in table 6), we see that both methods are comparable. This is because ADA-CVAR addresses this distribution shift algorithmically whereas MEAN does so by re-balancing the dataset. Nevertheless, ADA-CVAR can also re-sample the data set to balance it. This is ADA-CVAR with upsampling (first column in table 5, that outperforms all other algorithms with (and without) upsampling.

Table 4: Test accuracy/loss (mean ± s.d. for 5 random seeds) for train-set distribution shift without train set re-balancing. We highlight the best algorithms.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Train Set Shift	Adult	**0.65 ± 0.0**	**0.63 ± 0.0**	0.61 ± 0.1	0.69 ± 0.0	0.63 ± 0.1	0.84 ± 0.3	0.60 ± 0.1	0.90 ± 0.4
	Australian	**0.68 ± 0.0**	**0.58 ± 0.0**	0.46 ± 0.1	0.70 ± 0.0	0.48 ± 0.3	0.91 ± 0.4	0.46 ± 0.3	0.87 ± 0.3
	German	**0.48 ± 0.1**	**0.73 ± 0.0**	0.44 ± 0.1	0.75 ± 0.0	0.45 ± 0.1	1.00 ± 0.2	**0.48 ± 0.1**	0.80 ± 0.1
	Monks	**0.49 ± 0.0**	**0.72 ± 0.0**	0.39 ± 0.1	0.73 ± 0.0	0.43 ± 0.3	1.27 ± 0.6	0.40 ± 0.2	0.91 ± 0.3
	Phoneme	**0.58 ± 0.1**	**0.68 ± 0.0**	0.52 ± 0.2	0.69 ± 0.0	0.51 ± 0.2	1.08 ± 0.4	0.55 ± 0.2	0.92 ± 0.3
	Spambase	**0.77 ± 0.0**	**0.46 ± 0.0**	0.74 ± 0.1	0.60 ± 0.0	0.71 ± 0.2	0.61 ± 0.2	0.69 ± 0.2	0.58 ± 0.2
	Splice	**0.84 ± 0.0**	**0.40 ± 0.0**	0.77 ± 0.1	0.57 ± 0.0	0.73 ± 0.1	0.52 ± 0.2	0.50 ± 0.3	0.81 ± 0.4
	Titanic	0.46 ± 0.2	**0.70 ± 0.0**	**0.50 ± 0.3**	**0.69 ± 0.0**	0.43 ± 0.3	1.17 ± 0.5	0.41 ± 0.3	0.87 ± 0.3

Table 5: Test accuracy/loss (mean ± s.d. for 5 random seeds) for train-set distribution shift with train set re-balancing via upsampling We highlight the best algorithms.

Train Set Shift **Upsample**	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
	Adult	0.51 ± 0.0	$\mathbf{0.69 \pm 0.0}$	0.62 ± 0.1	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.67 \pm 0.1}$	0.73 ± 0.3	$\mathbf{0.65 \pm 0.2}$	0.78 ± 0.4
	Australian	$\mathbf{0.77 \pm 0.0}$	$\mathbf{0.56 \pm 0.0}$	0.50 ± 0.1	0.69 ± 0.0	0.56 ± 0.3	$\mathbf{0.79 \pm 0.4}$	0.54 ± 0.3	$\mathbf{0.77 \pm 0.3}$
	German	$\mathbf{0.58 \pm 0.0}$	$\mathbf{0.68 \pm 0.0}$	0.46 ± 0.1	0.74 ± 0.0	0.50 ± 0.2	0.92 ± 0.2	0.51 ± 0.1	0.76 ± 0.1
	Monks	$\mathbf{0.63 \pm 0.0}$	$\mathbf{0.66 \pm 0.0}$	0.42 ± 0.1	0.72 ± 0.0	0.49 ± 0.3	1.10 ± 0.3	0.45 ± 0.2	$\mathbf{0.85 \pm 0.3}$
	Phoneme	$\mathbf{0.65 \pm 0.1}$	$\mathbf{0.68 \pm 0.0}$	0.57 ± 0.2	$\mathbf{0.69 \pm 0.0}$	0.57 ± 0.2	0.94 ± 0.4	0.60 ± 0.2	$\mathbf{0.82 \pm 0.3}$
	Spambase	$\mathbf{0.88 \pm 0.0}$	$\mathbf{0.43 \pm 0.0}$	0.78 ± 0.1	0.60 ± 0.0	0.76 ± 0.2	$\mathbf{0.54 \pm 0.2}$	0.74 ± 0.2	$\mathbf{0.51 \pm 0.2}$
	Splice	$\mathbf{0.89 \pm 0.0}$	$\mathbf{0.30 \pm 0.0}$	0.80 ± 0.1	0.54 ± 0.1	0.77 ± 0.1	0.45 ± 0.2	0.61 ± 0.3	0.67 ± 0.4
	Titanic	$\mathbf{0.51 \pm 0.2}$	$\mathbf{0.70 \pm 0.0}$	$\mathbf{0.54 \pm 0.3}$	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.52 \pm 0.3}$	1.02 ± 0.5	$\mathbf{0.49 \pm 0.3}$	0.80 ± 0.3

Test Shift Next, we consider another distribution shift. Namely, the train set is kept constant and instead the the test-set is shifted (1-to-10 ratio). In this case, up-sampling should not change the test results. We show the results without re-sampling in table 6 and with upsampling in table 7. In this setting, upsampling does not help because the training sets are balanced. Comparing each case, we see that ADA-CVAR has superior performance to the benchmarks.

Table 6: Test accuracy/loss (mean ± s.d. for 5 random seeds) for test-set distribution shift without train set re-balancing. We highlight the best algorithms.

Test Set Shift	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
	Adult	0.50 ± 0.0	$\mathbf{0.69 \pm 0.0}$	0.64 ± 0.1	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.66 \pm 0.1}$	$\mathbf{0.72 \pm 0.1}$	0.64 ± 0.1	0.77 ± 0.1
	Australian	$\mathbf{0.76 \pm 0.0}$	$\mathbf{0.61 \pm 0.0}$	0.53 ± 0.1	0.69 ± 0.0	0.61 ± 0.3	0.71 ± 0.2	0.58 ± 0.3	0.72 ± 0.1
	German	$\mathbf{0.52 \pm 0.1}$	$\mathbf{0.69 \pm 0.0}$	0.46 ± 0.1	0.73 ± 0.0	0.51 ± 0.1	0.90 ± 0.2	$\mathbf{0.52 \pm 0.1}$	0.75 ± 0.1
	Monks	$\mathbf{0.63 \pm 0.1}$	$\mathbf{0.66 \pm 0.0}$	0.44 ± 0.2	0.71 ± 0.0	0.50 ± 0.2	0.99 ± 0.2	0.48 ± 0.2	0.81 ± 0.2
	Phoneme	0.50 ± 0.2	$\mathbf{0.69 \pm 0.0}$	0.54 ± 0.2	$\mathbf{0.69 \pm 0.0}$	0.56 ± 0.2	0.92 ± 0.4	$\mathbf{0.58 \pm 0.1}$	0.82 ± 0.3
	Spambase	$\mathbf{0.91 \pm 0.0}$	$\mathbf{0.52 \pm 0.0}$	0.81 ± 0.1	0.62 ± 0.0	0.79 ± 0.1	$\mathbf{0.48 \pm 0.2}$	0.77 ± 0.2	$\mathbf{0.46 \pm 0.2}$
	Splice	$\mathbf{0.91 \pm 0.0}$	$\mathbf{0.29 \pm 0.0}$	0.82 ± 0.1	0.54 ± 0.1	0.81 ± 0.1	$\mathbf{0.39 \pm 0.2}$	0.67 ± 0.3	0.58 ± 0.2
	Titanic	$\mathbf{0.47 \pm 0.2}$	$\mathbf{0.70 \pm 0.0}$	$\mathbf{0.52 \pm 0.3}$	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.52 \pm 0.3}$	0.99 ± 0.2	$\mathbf{0.50 \pm 0.3}$	0.81 ± 0.2

Table 7: Test accuracy/loss (mean ± s.d. for 5 random seeds) for test-set distribution shift with train set re-balancing via upsampling. We highlight the best algorithms.

Test Set Shift **Upsample**	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
	Adult	0.46 ± 0.0	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.65 \pm 0.1}$	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.66 \pm 0.1}$	$\mathbf{0.72 \pm 0.1}$	0.64 ± 0.1	0.76 ± 0.1
	Australian	$\mathbf{0.76 \pm 0.0}$	$\mathbf{0.61 \pm 0.0}$	0.55 ± 0.1	0.68 ± 0.0	0.64 ± 0.3	0.67 ± 0.4	0.61 ± 0.3	0.69 ± 0.3
	German	$\mathbf{0.52 \pm 0.1}$	$\mathbf{0.69 \pm 0.0}$	0.46 ± 0.1	0.73 ± 0.0	$\mathbf{0.52 \pm 0.1}$	0.89 ± 0.2	$\mathbf{0.53 \pm 0.1}$	0.75 ± 0.1
	Monks	$\mathbf{0.63 \pm 0.1}$	$\mathbf{0.66 \pm 0.0}$	0.46 ± 0.2	0.71 ± 0.0	0.51 ± 0.2	0.92 ± 0.5	0.49 ± 0.2	0.79 ± 0.2
	Phoneme	0.50 ± 0.2	$\mathbf{0.69 \pm 0.0}$	0.52 ± 0.2	$\mathbf{0.69 \pm 0.0}$	0.55 ± 0.2	0.90 ± 0.4	$\mathbf{0.57 \pm 0.1}$	0.82 ± 0.3
	Spambase	$\mathbf{0.91} \pm 0.0$	$\mathbf{0.54 \pm 0.0}$	0.82 ± 0.1	0.63 ± 0.0	0.80 ± 0.1	$\mathbf{0.44 \pm 0.2}$	0.79 ± 0.2	$\mathbf{0.43 \pm 0.2}$
	Splice	$\mathbf{0.92} \pm 0.0$	$\mathbf{0.28 \pm 0.0}$	0.84 ± 0.1	0.53 ± 0.1	0.83 ± 0.1	$\mathbf{0.36 \pm 0.2}$	0.71 ± 0.3	0.53 ± 0.4
	Titanic	$\mathbf{0.47 \pm 0.2}$	$\mathbf{0.70 \pm 0.0}$	$\mathbf{0.52 \pm 0.3}$	$\mathbf{0.69 \pm 0.0}$	$\mathbf{0.52 \pm 0.3}$	0.99 ± 0.2	$\mathbf{0.50 \pm 0.3}$	0.81 ± 0.2

Double Shift Finally, we consider a case where the train set is imbalanced (1-to-10 ratio), and the test set is even more imbalanced (1-to-100 ratio). We show the results without re-sampling in table 8 and with upsampling in table 9. In this case, we upsampling is detrimental. ADA-CVAR clearly outperforms all the other algorithms with upsampling and without upsampling.

Table 8: Test accuracy/loss (mean $\pm$ s.d. for 5 random seeds) for imbalanced train and test-sets without train set re-balancing. We highlight the best algorithms.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Double Set Shift	Adult	**0.94 ± 0.0**	**0.50 ± 0.0**	0.70 ± 0.2	0.69 ± 0.0	0.70 ± 0.2	0.63 ± 0.3	0.69 ± 0.2	0.66 ± 0.4
	Australian	**0.90 ± 0.0**	**0.39 ± 0.0**	0.59 ± 0.2	0.67 ± 0.1	0.69 ± 0.3	0.59 ± 0.4	0.66 ± 0.3	0.61 ± 0.3
	German	**0.95 ± 0.0**	**0.43 ± 0.0**	0.52 ± 0.2	0.70 ± 0.1	0.58 ± 0.2	0.79 ± 0.3	0.59 ± 0.2	0.69 ± 0.2
	Monks	**0.95 ± 0.0**	**0.48 ± 0.1**	0.50 ± 0.2	0.70 ± 0.0	0.58 ± 0.3	0.81 ± 0.4	0.56 ± 0.3	0.72 ± 0.3
	Phoneme	**0.58 ± 0.3**	**0.67 ± 0.0**	**0.57 ± 0.2**	0.69 ± 0.0	**0.61 ± 0.2**	0.79 ± 0.4	**0.63 ± 0.2**	0.73 ± 0.3
	Spambase	**0.98 ± 0.0**	**0.19 ± 0.0**	0.85 ± 0.1	0.60 ± 0.1	0.83 ± 0.1	0.39 ± 0.2	0.82 ± 0.2	0.39 ± 0.2
	Splice	**0.98 ± 0.0**	**0.18 ± 0.0**	0.86 ± 0.1	0.52 ± 0.1	0.85 ± 0.1	0.31 ± 0.2	0.75 ± 0.3	0.46 ± 0.4
	Titanic	**0.85 ± 0.2**	**0.49 ± 0.0**	0.52 ± 0.3	0.69 ± 0.0	0.59 ± 0.3	0.85 ± 0.3	0.57 ± 0.3	0.74 ± 0.3

Table 9: Test accuracy/loss (mean $\pm$ s.d. for 5 random seeds) for imbalanced train and test-sets with train set re-balancing via up-sampling. We highlight the best algorithms.

	Data Set	ADA-CVAR		TRUNC-CVAR		SOFT-CVAR		MEAN	
Double Set Shift **Upsample**	Adult	0.40 ± 0.1	**0.69 ± 0.0**	**0.68 ± 0.2**	**0.69 ± 0.0**	**0.72 ± 0.2**	**0.60 ± 0.3**	**0.71 ± 0.2**	**0.62 ± 0.4**
	Australian	**0.86 ± 0.0**	**0.46 ± 0.0**	0.61 ± 0.2	0.66 ± 0.1	0.71 ± 0.3	**0.56 ± 0.2**	0.67 ± 0.3	0.59 ± 0.2
	German	**0.78 ± 0.1**	**0.58 ± 0.0**	0.55 ± 0.2	0.69 ± 0.1	0.60 ± 0.2	0.75 ± 0.2	0.60 ± 0.2	0.68 ± 0.1
	Monks	**0.68 ± 0.1**	**0.64 ± 0.0**	0.51 ± 0.2	0.69 ± 0.0	0.59 ± 0.2	**0.78 ± 0.3**	0.57 ± 0.2	**0.72 ± 0.2**
	Phoneme	**0.49 ± 0.3**	**0.69 ± 0.0**	**0.59 ± 0.2**	**0.69 ± 0.0**	**0.63 ± 0.2**	**0.75 ± 0.2**	**0.65 ± 0.2**	**0.70 ± 0.2**
	Spambase	**0.95 ± 0.0**	**0.26 ± 0.1**	0.86 ± 0.1	0.60 ± 0.1	0.84 ± 0.1	**0.36 ± 0.2**	0.83 ± 0.2	**0.37 ± 0.2**
	Splice	**0.96 ± 0.0**	**0.18 ± 0.0**	0.87 ± 0.1	0.50 ± 0.1	0.86 ± 0.1	**0.29 ± 0.2**	0.77 ± 0.3	0.43 ± 0.2
	Titanic	**0.62 ± 0.3**	**0.68 ± 0.0**	**0.53 ± 0.3**	**0.69 ± 0.0**	**0.58 ± 0.3**	0.82 ± 0.4	**0.57 ± 0.3**	**0.72 ± 0.3**

Experiment Conclusion Techniques that address class imbalance, such as up-sampling, are useful when there is a-priori knowledge that the test-set is balanced (cf. Train Shift experiment). When this is not the case, such techniques may be detrimental (cf. Double Shift experiment). ADA-CVAR, and the CVaR DRO optimization problem, is orthogonal to techniques for imbalanced dataset and can be used together, as shown by the previous experiments. Overall, ADA-CVAR also has lower standard errors than MEAN and SOFT-CVAR. This indicates that it is more robust to the random sampling procedures.

F More Related Work

F.1 Combinatorial Bandit Algorithms

A central contribution of our work is an *efficient* sampling algorithm based on an instance of combinatorial bandits, the k-set problem. In this setting, the learner must choose a subset of k out of N experts with maximum rewards, and there are $\binom{N}{k}$ such sets. Koolen et al. (2010) introduce this problem in the full information setting (k-sets) and Cesa-Bianchi and Lugosi (2012) extend it to the bandit setting. Cesa-Bianchi and Lugosi (2012) propose an algorithm, called CombBand, that attains a regret of $O(k^{3/2}\sqrt{NT\log(N/k)})$ when the learner receives bandit feedback. Audibert et al. (2013) prove a lower bound of $O(k\sqrt{NT})$, which is attained by Alatur et al. (2020) up to a $\sqrt{\log(N)}$ factor. However, the computational and space complexity of the CombBand algorithm is $O(kN^3)$ and $O(N^3)$, respectively. Uchiya et al. (2010) propose an efficient sampling algorithm that has $O(N\log(k))$ computational and $O(N)$ space complexity. Instead, we efficiently represent the algorithm proposed by Alatur et al. (2020) using Determinantal Point Processes (Kulesza et al., 2012). This yields an algorithm with $O(\log(N))$ computational and $O(N)$ space complexity.

F.2 Sampling from kDPPs

We propose to directly sample from the marginals of the (relaxed) k-DPP. Our setting has the advantage that the k-DPP is diagonal and there is no need for performing an eigendecomposition of the kernel matrix, which takes $O(N^3)$ operations. However, there are efficient methods that avoid the eigendecomposition and return a sample (of size k) of the k-DPP. Sampling uniformly at random from such sample can be done in constant time. State-of-the-art exact sampling methods for k-DPPs take at least $O(N\,\mathrm{poly}(k))$ operations using rejection sampling (Dereziński et al., 2019), whereas approximate methods that use MCMC have mixing times of $O(Nk)$ (Anari et al., 2016). Hence, such algorithms are ineficient compared to our sampling algorithm that takes only $O(\log(N))$. This is because our method is specialized on diagonal k-DPPs and latter algorithms remain valid for general k-DPPs.