Adaptive Sampling for Stochastic Risk-Averse Learning

Sebastian Curi [1] **Kfir Y. Levy** [2] **Stefanie Jegelka** [3] **Andreas Krause** [1]

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. 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 allows 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 significant losses on "difficult" examples 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. This criterion has been used in many applications, such as portfolio optimization (Krokhmal et al., 2002) or supply chain management (Carneiro et al., 2010). In short, the α-CVaR of a loss distribution is the average of the losses in the α-tail of such 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. Unfortunately, this approach fails when *stochastically* optimizing the CVaR – especially on non-convex problems, such as training deep models on Fashion-MNIST and CIFAR-10. A possible reason for this failure is that Monte Carlo estimates of gradients of the CVaR have high variance.

To address this issue, we propose a novel *adaptive sampling algorithm* (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 (Section 4.4). Finally, we demonstrate the performance of our algorithm in a suite of experiments (Section 5).

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 enlarge larger losses (Rabin, 2013); prospect theory that re-scales the probability of events (Kahneman & 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, particularly in portfolio optimization, as it does not rely on specific utility or weighing functions, which offers great flexibility. So we focus on it here.

CVaR in ML In machine learning, the CVaR criterion has been considered in several works. The ν-SVM algorithm by Schölkopf et al. (2000) can be interpreted as optimizing the CVaR of the loss, as shown by Gotoh & Takeda (2016). Also related, Shalev-Shwartz & Wexler (2016) propose an adaptive sampling algorithm 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. Furthermore, Fan et al. (2017) use an optimization algorithm proposed by Ogryczak & Tamir (2003) to optimize the maximum of

[1]Department of Computer Science, ETH, Zurich [2]Technion-Israel Institute of Technology [3]CSAIL, Massachusetts Institute of Technology, USA. Correspondence to: Sebastian Curi <sebastian.curi@inf.ethz.ch>.

the sum of k functions that 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 & 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 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. In this direction, Namkoong & Duchi (2016) generalize the work of Shalev-Shwartz & Wexler (2016) for a particular class of f-divergences, also using an adaptive sampling algorithm. Similarly, Ahmadi-Javid (2012) introduces the entropic value-at-risk by considering a different DRO set. Duchi et al. (2016); Namkoong & Duchi (2017); Esfahani & Kuhn (2018); Kirschner et al. (2020) address related DRO problems, but with different disturbance 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 related to Namkoong & Duchi (2016), but we use a different DRO set. Further, we provide *efficient* algorithms to apply the DRO problem to 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* $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to [0, 1]$. Overloading notation, 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 loss distribution 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):

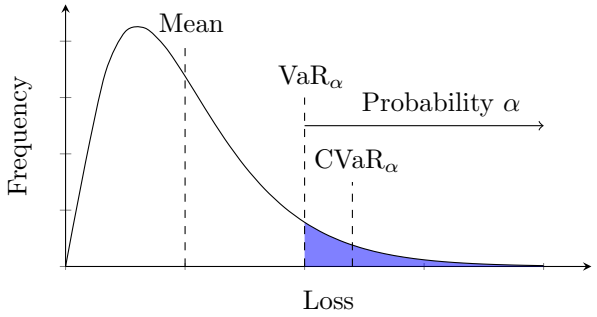

Figure 1. Illustration of the CVaR of a Loss

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

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 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 program 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

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

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 A.1. □

The result above is easily extended to classes $\mathcal{H}$ with finite VC (pseudo-)dimension.

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.

We realize that the root of the problem is 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 & Tuffin, 2009). To address this issue, we propose a novel sampling algorithm that *adaptively* learns to sample events from the distribution $Q^\star$. Furthermore, the algorithm adapts $Q^\star$ to the different parameters θ that are encountered during the optimization.

4. Adaptive Sampling for CVaR Optimization

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

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

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. It is known that viewing both players as online learners that compete against each other, and equipping each of them with no-regret algorithms is a way to solve minimax games (Freund & Schapire, 1999). The partial (bandit) feedback model enables us to consider only a *small* subset of the data set at each iteration of the optimization algorithm. Instead, full-information feedback requires a full pass on the data set per iteration, invalidating all benefits of stochastic optimization.

Below, 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.

On a high level, we use an adaptive sampling scheme for the q-player and SGD for the θ-player. Initially, the q-player plays the uniform distribution and the θ-player selects any parameter in the set Θ. At each iteration t, the sampler player samples a data point (or a mini-batch) with respect to the distribution q_t. Then, the learner player 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** $t = 1, \dots, T$ **do**
4: **Sampler:** Sample element $i_t \sim q_t = \frac{1}{k}\mathbb{P}_{w_t}(i)$.
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

At every iteration t, the sampler player faces a vector of losses that the adversary sets through θ_t. We use the notation $L(\theta_t) = L_t$, and $L(\theta_t, x_i, y_i) = L_{t,i}$. The goal of the sampler player is to minimize its *regret*:

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

The regret measures how good the sequence of actions of the sampler are, compared to the best single action in hindsight (after seeing the sequence of iterates L_t).

The DRO set $\mathcal{Q}^\alpha$ is a polytope with $\binom{N}{k}$ vertexes, 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. Let us call the set of all size k subsets $\mathcal{I}_k = \left\{I \subseteq 2^{[N]} \mid |I| = k\right\}$ and the goal is to find the best such set. Hereby, the value of a set I at time t is simply the average of the losses $(1/k)\sum_{i\in I} L_i(\theta_t)$. For our online algorithm, we will, every round, sample a set $I \in \mathcal{I}_k$ and obtain (an unbiased estimate of) its value. This value changes every round, since θ_t changes. 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 (Lattimore & Szepesvári, 2018, Chapter 30). We follow Alatur et al. (2019) and develop an algorithm for the sampler.

To obtain no-regret, one must use randomized strategies. Let $\Delta_I := \left\{\tilde{W} \in \mathbb{R}^{\binom{N}{k}} | \sum_I \tilde{W}_I = 1, \tilde{W}_I \ge 0\right\}$ be the simplex

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

of distributions over the $\binom{N}{k}$ subsets. The simplex Δ_I is the convex hull of the set $\mathcal{I}_k$. Finding the best distribution $W_I^\star \in \Delta_I$ is equivalent to finding the best subset $I^\star \in \mathcal{I}_k$. In turn, this is equivalent to finding the best $q^\star \in \mathcal{Q}^\alpha$.

A well known algorithm to find the best distribution under bandit feedback $W_I^\star \in \Delta_I$ is the famous EXP.3 algorithm (Auer et al., 2002). In this algorithm, the learner 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 a multiplicative weight update.

In our setting, rather than sampling k-subset I_t, the q-player samples a single element i_t uniformly at random from the set $I_t \sim W_{I,t}$. Then we only observes the loss L_{i_t}, and use it to updates a weight vector $w_{t,i}$. The weight vectors of elements and of subsets are related as follows, $W_{t,I} = \sum_{i \in I} w_{t,i}$. It turns out that this bandit feedback is sufficient to find the best $W_I^\star \in \Delta_I$ by playing EXP.3.

Unfortunately, the EXP.3 algorithm is impractical because the dimension of $W_{I,t}$ is *exponential* in k. Furthermore, the regret of this algorithm depends on the dimension of $W_{I,t}$.

The first crucial insight is that, due to the combinatorial structure of the problem and additivity of the loss, a single element i_t sampled by the algorithm provides information bout the loss of all $\binom{N-1}{k-1}$ *sets* that contain i_t. Exploiting this enables to provide regret guarantees that depend sublinearly in N (rather than N^k).

Since we like to design practical algorithms, we cannot allow ourselves to explicitly maintain and update $W_{t,I}$. The second crucial insight is that we can further exploit the structure of the problem in order to efficiently do so by employing *k-Determinantal Point Processes*.

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:

$$\mathbb{P}(I) = \frac{\det(K_I)}{\sum_{|J|=k} \det(K_J)},$$

where K is a positive definite kernel matrix and K_J is the submatrix of K indexed by the elements of J. ■

In particular, we consider k-DPPs with *diagonal* kernel matrices $K = \operatorname{diag} w$, with $w \in \mathbb{R}^N_{\geq 0}$. Such family of distributions is sufficient to describe, for example, the uniform distribution over the $\binom{N}{k}$ subsets and all the vertices of $\mathcal{Q}^\alpha$.

Let $\mathbb{P}_w$ indicate the vector of marginal probabilities under the diagonal k-DPP model with kernel $K = \operatorname{diag} w$, (i.e., a vector whose i-th entry $\mathbb{P}_w(i)$ is the marginal probability of sampling the i-th element under the k-DPP model). Then, for any $w \in \mathbb{R}^N_{\geq 0}$, the vector of marginals $\frac{1}{k}\mathbb{P}_w \in \mathcal{Q}^\alpha$.[2]

[2]This is true for *any* k-DPP, not just those with diagonal kernels

We can finally describe the sampler algorithm. We initialize the k-DPP kernel with the uniform distribution $w_1 = \mathbf{1}_N$. At each iteration t, the sampler player plays the distribution $q_t = \frac{1}{k}\mathbb{P}_{w_t} \in \mathcal{Q}^\alpha$ and samples an element $i_t \sim q_t$.[3] 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 $w_{t+1,i_t} = w_{t+1,i_t} e^{kL_{t,i_t}/q_{t,i_t}}$. The final algorithm is marked with "sampler" in Algorithm 1 and its adaptive nature motivates the final algorithm's name.

This algorithm addresses the disadvantages of the EXP.3 algorithm. Computationally, it only requires $O(N)$ memory. Statistically, after sampling every element i_t, the distribution induced in the $\binom{N-1}{k-1}$ sets that contain i_t are updated, yielding rates that depend sub-linearly on the data set size. Next, we prove the regret rates for this algorithm.

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 A.2. First, we prove in Proposition 2 that the iterates of the algorithm are effectively in $\mathcal{Q}^\alpha$. Next, we prove in Proposition 3 that the comparator in the regret of Alatur et al. (2019) 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. (2019, Lemma 1). □

It is instructive to consider the special cases $k = 1$ and $k = N$. For $k = N$, the marginal distribution q_t remains uniform, hence ADA-CVAR reverts back to SGD. For $k = 1$, q_t is in fact identical to the standard EXP.3 algorithm applied to singleton sets $I_t = \{i_t\}$.

4.2. Learner (θ-Player) Algorithm

Analogous to the sampler player, the learner player seeks to minimize its regret defined as:

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

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

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

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. This oracle model is standard in such games (Agarwal et al., 2018)

[3]Note that this is equivalent to first sample a set I_t from the k-DPP, and then i_t uniformly from I_t.

Lemma 2. *A learner player that plays the BTL algorithm suffers at most zero regret.*

Proof. See Appendix A.3. □

To implement BTL, at every round the learner player must solve a *weighted* empirical loss minimization in Problem (7). For convex problems, we know that it is not necessary to solve the BTL problem (7). In fact, algorithms such as online projected gradient descent (Zinkevich, 2003) achieve no-regret guarantees. We refer the reader to Appendix B for a discussion of the convex case. The non-convex case is more challenging as solving a non-convex optimization problem is in general NP-hard (Murty & Kabadi, 1987). Obtaining efficient and provably no-regret guarantees in the non-convex online 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 associated non-convex problems. Hence, we use the sequence of samples $\{i_\tau \sim q_\tau\}_{\tau=1}^{t}$ as an unbiased estimate of $\bar{q}_t$. Then, for each $i_t \sim q_t$, the learner chooses $\theta_t := \theta_{t-1} - \eta_l \nabla L_{i_t}(\theta_{t-1})$. [4] We show the algorithm with lines labeled "Learner" in Algorithm 1.

4.3. Guarantees for CVaR Optimization

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

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

Theorem 1 (Game Sublinear-Regret)**.** *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. To bound the regret, we bound it with the sum of the Learner and Sampler regret as follows:

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

□

[4] Note that the learner player chooses θ_t after observing the gradient of $L_{i_t}(\cdot)$, hence she effectively plays approximate BTL.

The result above immediately implies a performance guarantee of any random iterate θ_t w.r.t. Problem (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:*

$$\mathbb{E}\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] \leq \hat{\mathbb{C}}^\alpha[L(\theta^\star)] + \epsilon,$$

where $\epsilon = O(\text{GameRegret}_T / T)$ and 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 sketch. For a detailed proof please refer to Appendix A.4. The excess CVaR is bounded by the duality gap, which in turn is upper-bounded by the average game regret. In Theorem 1 we proved that the Game Regret is sublinear, hence the average excess CVaR tends to zero. □

In the convex case, no randomization is necessary and we return the average iterate. In practice for both the convex and non-convex setting, our experiments use the last iterate.

4.4. Efficient 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 generic case. The marginals of diagonal k-DPPs are $\mathbb{P}_w(i) = w_i \frac{e_{-i}^{k-1}}{e_N^k}$, 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 $[N] \setminus i$. 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

asymptotically as $O(1/N)$. The marginal probabilities of this matched DPP are simply

$$\hat{\mathbb{P}}_w(i) = \frac{w_i e^{\nu}}{1 + w_i e^{\nu}}, \tag{9}$$

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 & Wexler (2016).

DPP Update The remaining challenge is how to update the approximate DPP between two different iterations of the sampling algorithm. Solving the coupling constraint $\sum_{i=1}^{N} \frac{w_i e^{\nu}}{1+w_i e^{\nu}} = k$ takes $O(N)$ operations and there is no closed-form solution for ν. We found three possible solutions. First, we can take $O(N)$ operations to solve for ν. In practice, solving this equation is extremely fast when using the previous solution as a warm start. Second, we can just keep ν constant every epoch and update it only every $O(N)$ steps. This deteriorates the approximation slightly, particularly in small scale applications. The third option is to use the implicit gradient theorem to calculate a first order approximation of ν when w_i changes in the coupling constraint. The gradient is $\frac{d\nu}{dw_i} = -\frac{1}{w_i}$, therefore the update rule is $\nu_{t+1} = \nu_t - (e^{\eta L_{i_t}(\theta_t)} - 1)$. Note that this also introduces an approximation error, thus, every $O(N)$ steps one must solve again for ν. In experiments, we observe no major performance difference between these strategies.

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 guarding against distribution shifts.

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 & Zhu (2010) compare different smoothing techniques for the $\sum_i [x_i]_+$ non-linearity. From these, we use the relaxation $T \log(\sum_i e^{x_i/T})$ proposed by Nemirovski & Shapiro (2006). At each iteration, we heuristically approximate the population sum with a mini-batch. Finally, we compare also 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 & Graff, 2017). Table 1 presents the CVaR ($\alpha = 0.01$) and (average) loss for linear regression and classification (logistic regression) on the test set. We also report test accuracy and precision for the classification (logistic regression) tasks. Overall, ADA-CVAR performs comparably or better to benchmarks in linear regression. 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 its learnt predictive distribution is close to uniform. This can be seen because the test log-likelihood (test loss) value is close to $\log 2$ and the test CVaR, is also close to $\log 2$. ADA-CVAR has a comparable accuracy to ERM (MEAN algorithm) but a much better (lower) CVaR.

5.2. Convex CVaR Distributional Robustness

We use the same classification data sets and classifiers as in Experiment 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 systematic shifts in class frequencies are quite common in practice. Indeed, the CVaR can also be seen as a way of enforcing *fairness constraints* in imbalanced data sets (Williamson & Menon, 2019). Here, we consider $\alpha = 0.1$-CVaR which is compatible with the data imbalance. Table 2 shows test accuracy and (average) test negative log-likelihood. Overall, ADA-CVAR has higher test accuracy than the benchmarks. TRUNC-CVAR, however, has higher log-likelihood. It turns out this is due to the fact that its predictive distribution is close to uniform. This behavior guards against worst-case distribution shifts, where uniform predictions minimize large losses (negative log-likelihood) on small sets of examples that are difficult to predict. Such a *worst case* distribution might be too pessimistic to be encountered in practice, however. Instead, ADA-CVAR benefits from the varying distributions during training and protects better against non-adversarial distribution shifts.

5.3. Non-Convex (Deep Learning) CVaR Optimization

We test our algorithm in common non-convex optimization tasks that arise in deep learning. We consider MNIST (LeCun et al., 1995) with LeNet-5 neural network (LeCun et al., 1995), Fashion-MNIST (Xiao et al., 2017) with LeNet-5 using dropout (Hinton et al., 2012), and CIFAR-10 (Krizhevsky et al., 2014) with the ResNet18 (He et al., 2016) neural network. To achieve state-of-the-art accuracy

Table 1. Test mean ± 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**

we perform data-augmentation on the training set. Thus, the effective data set size is infinite. To address this, we consider a mixture of distributions in a similar spirit to Borsos et al. (2019). Each data point corresponds to the "center" 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. We show the result in Table 3. In MNIST and Fashion-MNIST, all algorithms achieve state-of-the art accuracy. In Fashion-MNIST, ADA-CVAR attains best test CVaR performance, indicating that the ADA-CVAR model has better calibration than the MEAN. For CIFAR-10, only ADA-CVAR and MEAN attain state-of-the art accuracy and lowest losses. The lowest CVaR is attained by TRUNC-CVAR at the cost of lower accuracy. Again, this is because the predictive model that the CVaR produces is close to uniform. Once again, our algorithm benefits from the adaptive sampling distributions to achieve high-accuracy yet low CVaR.

Gradient Magnitude and Training Time In SGD, 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 happens in SOFT-CVAR. This makes training of these algorithms considerably harder due to exploding gradients and noisier gradient estimates. With the same learning rates, these algorithms usually produce numerical overflows and, in order to stabilize learning, we used considerably smaller learning rates. In turn, this made the number of iterations required until convergence longer. 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 require 7 epochs, MEAN 9 epochs, SOFT-CVAR 21 epochs, 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. We keep the test set unchanged. We consider VGG16 (Simonyan & Zisserman, 2014) with

Table 2. Test accuracy/loss (mean ± 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	**0.60 ± 0.0**	0.86 ± 0.0	**0.61 ± 0.0**	**0.69 ± 0.0**	0.59 ± 0.0	0.97 ± 0.0	0.55 ± 0.0	1.05 ± 0.0
	Australian	**0.57 ± 0.1**	**0.62 ± 0.0**	0.54 ± 0.0	0.67 ± 0.0	0.53 ± 0.0	0.72 ± 0.0	0.47 ± 0.0	0.83 ± 0.0
	German	0.48 ± 0.0	**0.74 ± 0.0**	**0.49 ± 0.1**	**0.73 ± 0.0**	0.46 ± 0.0	0.99 ± 0.1	**0.51 ± 0.1**	0.78 ± 0.1
	Monks	**0.49 ± 0.0**	0.77 ± 0.0	**0.48 ± 0.1**	**0.70 ± 0.0**	**0.49 ± 0.0**	0.95 ± 0.0	**0.49 ± 0.0**	0.83 ± 0.1
	Phoneme	**0.55 ± 0.0**	0.94 ± 0.0	0.41 ± 0.0	**0.69 ± 0.0**	0.48 ± 0.0	1.20 ± 0.0	0.53 ± 0.0	0.99 ± 0.0
	Spambase	**0.78 ± 0.0**	**0.46 ± 0.0**	0.75 ± 0.0	0.56 ± 0.0	0.69 ± 0.0	0.59 ± 0.0	0.65 ± 0.0	0.58 ± 0.0
	Splice	**0.85 ± 0.0**	**0.38 ± 0.0**	0.83 ± 0.0	0.55 ± 0.0	0.80 ± 0.0	0.46 ± 0.0	0.52 ± 0.0	0.76 ± 0.0
	Titanic	**0.52 ± 0.0**	0.78 ± 0.0	**0.51 ± 0.2**	**0.69 ± 0.0**	0.32 ± 0.0	1.05 ± 0.0	0.32 ± 0.0	0.90 ± 0.1

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	MNIST		F-MNIST		CIFAR-10	
Accuracy	ADA	**0.99**	**0.98**	**0.99**	**0.99**	**0.94**	**0.87**
	TRUNC	**0.99**	**0.98**	**0.99**	**0.98**	0.59	0.50
	SOFT	**0.99**	**0.98**	**0.99**	**0.98**	0.86	0.64
	MEAN	**0.99**	**0.98**	**0.99**	**0.98**	**0.94**	**0.87**
Surrogate	ADA	**0.31**	**0.03**	**0.31**	**0.03**	2.19	**0.25**
	TRUNC	0.35	0.03	0.37	0.04	**2.02**	1.32
	SOFT	0.41	0.04	0.35	0.04	2.79	0.32
	MEAN	**0.33**	**0.03**	0.38	0.04	2.49	**0.24**

batch normalization (Ioffe & Szegedy, 2015) and Resnet-18 networks. In Figure 2, we show the test accuracy for different values of β. Differently to Experiment 5.2, the algorithms do not know a-priori the amount of distribution shift (α) to protect against. We consider a fixed $\alpha = 0.1$ for ADA-CVAR, TRUNC-CVAR, and SOFT-CVAR. Uniformly over all distribution shifts, ADA-CVAR is clearly superior to the benchmarks.

When a high-capacity network learns a perfectly accurate model (maybe over-fitting), then the average loss and the CVaR have both zero value. Resnet-18 achieves higher performance than VGG16 without distribution shifts due to its higher capacity. This explains the similar performance between ADA-CVAR and MEAN with Resnet-18 for most levels β. Instead, there is a stark discrepancy between ADA-CVAR and MEAN in VGG16, showing the advantage of training in a risk averse manner.

In this setting, TRUNC-CVAR returns almost uniform predictive models. The log-likelihood of such models is higher than other models, but the predictive accuracy is low. This is because TRUNC-CVAR focuses on the *worst* case distribution that is not encountered in practice. Instead, ADA-CVAR benefits from the adaptive distributions encountered during training to achieve high accuracy and high likelihood.

Finally, SOFT-CVAR under performs, particularly at lower values of β. The smoothed loss amplifies larger losses, effectively reducing the CVaR of the training set, but it is not robust to distribution shifts, particularly with the high-capacity models.

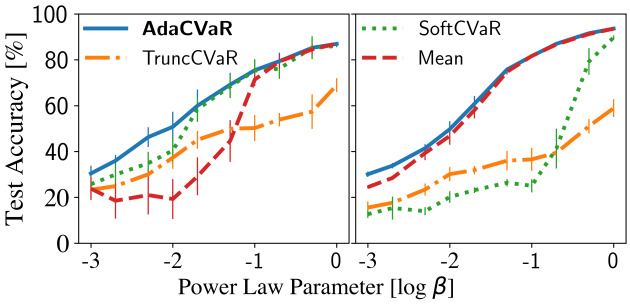

Figure 2. Distributional shift robustness for different class size distributions on the test set (amount of shift decreases from left to right). We plot the mean ± 1 s.d. over five different random seeds. Left: CIFAR-10 with VGG16 with Batch Normalization. Right: CIFAR-10 with Resnet-18. For all tested scaling parameters, ADA-CVAR has better or equal test accuracy.

6. Conclusions

The CVaR is a natural criterion for training ML models in a risk aware fashion, guarding against certain kinds of distribution shifts. As we have seen, 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 formulation of the CVaR, and tackles it using 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 has higher robustness to (non-adversarial) distribution shifts.

References

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

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

Alatur, P., Levy, K. Y., and Krause, A. Multi-player bandits: The adversarial case. *arXiv:1902.08036*, 2019.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Zimmerman, D. W. Teachers corner: A note on interpretation of the paired-samples t test. *Journal of Educational and Behavioral Statistics*, 22(3):349–360, 1997.

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

A. Proofs

A.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. □

A.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 2. *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:

$$\mathbb{P}_w(i) = \sum_{I\ni i}\mathbb{P}_w(I) = \frac{\sum_{I\ni i}\prod_{i'\in I} w_{i'}}{\sum_I \prod_{i'\in I} w_{i'}}. \tag{10}$$

From eq. (10), clearly $0 \leq \mathbb{P}_w(i) \leq 1$. Summing eq. (10) 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 3. *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 $[N]$.*

$$\max_{q\in\mathcal{Q}^{\alpha}}\sum_{t=1}^{T} q^{\top}L_t = \max_{\tilde{w}\in\Delta}\frac{1}{k}\sum_{t=1}^{T}\tilde{w}\tilde{L}_I \tag{11}$$

Proof. Both left and right sides of (11) 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 (11). 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 (11) 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 3. The second equality uses Proposition 2 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. (2019, Lemma 1).

A.3. Proof of Lemma 2

Lemma 2. A learner player that plays the BTL algorithm suffers at most zero regret.

Proof. We proceed by induction. Clearly for $T = 1$, $\mathrm{LR}_1 = 0$. Assume true for $T-1$, the inductive hypothesis

is $\mathrm{LR}_{T-1} \leq 0$. The regret at time T is:

$$\begin{aligned}\mathrm{LR}_T &= \sum_{t=1}^{T} q_t^\top L(\theta_t) - \min_{\theta\in\Theta} \sum_{t=1}^{T} q_t^\top L(\theta), \\ &= \sum_{t=1}^{T} q_t^\top \left[L(\theta_t) - L(\theta_T)\right] = \sum_{t=1}^{T-1} q_t^\top \left[L(\theta_t) - L(\theta_T)\right], \\ &= \mathrm{LR}_{T-1} + \min_{\theta\in\Theta} \sum_{t=1}^{T-1} q_t^\top L(\theta) - \sum_{t=1}^{T-1} q_t^\top L(\theta_T) \leq 0.\end{aligned}$$

□

A.4. 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:

$$\mathbb{E}\hat{\mathbb{C}}^\alpha[L(\bar{\theta})] \leq \hat{\mathbb{C}}^\alpha[L(\theta^\star)] + \epsilon,$$

where $\epsilon = \mathrm{GameRegret}_T / T$ and 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 $\bar{q}^\star := \arg\max_{q\in\mathcal{Q}^\alpha} J(\bar{\theta}, q)$. 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. 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 & 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 2. *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.*

C. 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 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 & Graff, 2017) and the Titanic Disaster dataset from (Eaton & 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).

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 normalize 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 & 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.

D. More Related Work

D.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 & Lugosi (2012) extend it to the bandit setting. Cesa-Bianchi & 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. (2019) 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. (2019) using Determinantal Point Processes (Kulesza et al., 2012). This yields an algorithm with $O(\log(N))$ computational and $O(N)$ space complexity.

D.2. Sampling from k-DPPs

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 $O(N\,\text{poly}(k))$ 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 inefficient 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.