Adaptive Sampling for Stochastic Risk-Averse Learning

Sebastian Curi
ETH Zurich

Kfir Y. Levy
Technion

Stefanie Jegelka
MIT CSAIL

Andreas Krause
ETH Zurich

Abstract

We consider the problem of training machine learning models in a risk-averse manner. In particular, 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. Our approach solves the game using an efficient no-regret algorithm for each player. Critically, we can apply these algorithms to large-scale settings because the implementation relies on sampling from Determinantal Point Processes. 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. In short, the $\alpha$-CVaR of a loss distribution is the average of the losses in the $\alpha$-tail of the distribution.

Unfortunately, we see in experiments that common variants of stochastic gradient descent (SGD) fail to optimize the CVaR in real-world data sets such as 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 optimizes the CVaR (Section 4.2). Our approach naturally enables the use of standard stochastic optimizers (Section 4.3). We provide convergence guarantees of the algorithm (Section 4.4) and an efficient implementation (Section 4.5). 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). There are three different methods to model risk: utility functions that enlarge 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 case of the latter class. The CVaR is ubiquitous in applications, particularly in portfolio optimization, as it does not rely on the design of utility nor weighing functions, which makes its success as a risk-averse criterion less sensitive to designer priors.

CVaR in ML

In machine learning, the CVaR criterion has been considered in several works. The $\nu$-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 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 the definition of the CVaR for empirical measures. Furthermore, Fan et al. (2017) use an optimization algorithm proposed by Ogryczak and Tamir (2003) to optimize the maximum of the
sum of $k$ functions that is the same as the algorithm proposed by 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 of these works use the original formulation provided in 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 paper, we address this via adaptive sampling and develop a method that allows us to scale up previous work to larger datasets and more complex models.

Robust optimization The dual representation of the CVaR that we use in this paper has a distributionally robust optimization (DRO) interpretation (Shapiro et al., 2009, Section 6.3). In this direction, Namkoong and Duchi (2016) generalize the work of Shalev-Shwartz and 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 and Duchi (2017); Esfahani and Kuhn (2018); Staib and Jegelka (2019) address related DRO problems. In this work, we use the DRO formulation of the CVaR to phrase the optimization problem as a game. To solve the game, we propose an adaptive algorithm for the learning problem. The algorithm is related to Namkoong and Duchi (2016), but we use a different type of robust set. Furthermore, we provide efficient algorithms to apply the DRO problem to large-scale datasets.

Efficient Combinatorial Bandits 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. Cesa-Bianchi and Lugosi (2012) introduce this setting and introduce the CombBAND algorithm 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 adapt the algorithm proposed by Alatur et al. (2019) using Determinantal Point Processes (Kulesza et al., 2012). Our algorithm has $O(\log N)$ computational and $O(N)$ space complexity.

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., $D = \{(x_1, y_1), \ldots, (x_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y})^N \sim D^N$, and her goal is to learn a function $h_\theta : \mathcal{X} \to \mathcal{R}$ that is parametrized by $\theta \in \Theta \subset \mathcal{R}^d$. The performance of $h_\theta$ 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 $L_\theta(x, y) = l(\theta, x, y)$. The learner’s goal is to minimize the CVaR of the loss distribution w.r.t. the parameters $\theta$ and the (unknown) distribution $D$.

CVaR properties The CVaR of a random variable $L \sim P$ is defined as $C^\alpha[L] = \mathbb{E}_P[L | L \geq \ell^\alpha]$, where $\ell^\alpha$ 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.

![Figure 1: Illustration of the CVaR of a Loss](image-url)

The CVaR of a random variable is the expected value of the same random variable but w.r.t. a different law. This law arises from the following optimization problem (Shapiro et al., 2009, Section 6.3):

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

(1)

where $\mathcal{Q}^\alpha = \left\{Q \ll P, \frac{dQ}{dP} \leq \frac{1}{\alpha}\right\}$. The distribution $Q^\star$ that solves Problem (1) places all the mass in the tail, i.e., the blue shaded region of Figure 1. Rockafellar et al. (2000) prove strong duality for Problem (1). The dual program is:

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

(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 expectation $\hat{\mathbb{E}}$. The learning problem is:

$$\min_{\ell \in \mathbb{R}, \theta \in \Theta} \ell + \frac{1}{\alpha} \hat{\mathbb{E}} \left[\max \{0, L(\theta) - \ell\}\right].$$

(3)
Learning problem \((\text{3})\) has computable subgradients, and hence lends itself to subgradient-based optimization. Furthermore, when \(L(\theta)\) is a convex function, then the learning problem \((\text{3})\) is jointly convex in \((\ell, \theta)\).

Next, we show that the learning problem \((\text{3})\) is a sensible learning rule 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 \(1 - \delta\),
\[
E\left[\sup_{h \in \mathcal{H}} \left| \mathbb{C}^\alpha[L(h)] - \mathbb{C}^\alpha[L(h)] \right| \right] \leq \frac{1}{\alpha} \sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}.
\]

**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 \((\text{3})\), 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 \(\alpha\) of points will contain gradient information. The rest 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 \((\text{3})\) extremely noisy, as we demonstrate empirically in Section 5.

We realize that the root of the problem is the mismatch between the sampling distribution \(P\) and the unknown distribution \(Q^*\), from which we would ideally want to sample. In fact, Problem \((\text{3})\) can be interpreted as a form of rejection sampling – samples with losses smaller than \(\ell\) are rejected. It is well known that Monte Carlo estimation of rare events suffers from high variance \((\text{Rubino and Tuffin, 2009})\).

To address this issue, we propose a novel sampling algorithm that \textit{adaptively} learns to sample events from the distribution \(Q^*\). Furthermore, the algorithm adapts \(Q^*\) to the different parameters \(\theta\) that are encountered during the optimization algorithm.

## 4 ADAPTIVE SAMPLING FOR EMPIRICAL CVAR LEARNING

### 4.1 Reformulation of CVaR Optimization

We propose to \textit{directly} address the DRO problem \((\text{1})\) on the empirical measure \(\hat{P}\) for learning. The DRO set is then \(Q^\alpha = \{q \in \mathbb{R}^N : 0 \leq q_i \leq \frac{1}{\alpha}, \sum q_i = 1\}\) with \(k = \lfloor \alpha N \rfloor\). The learning problem becomes:
\[
\min_{\theta} \max_{q \in Q^\alpha} \mathbb{E}_q[L_i(\theta)] = \min_{\theta} \max_{q \in Q^\alpha} q^\top L(\theta), \quad (\text{4})
\]
where \(L(\theta) \in \mathbb{R}^N\) has index \([L(\theta)]_i = L_i(\theta)\). The learning problem \((\text{1})\) can be interpreted as a game between a \(\theta\)-player (the learner), whose goal is to \textit{minimize} the objective function by selecting \(\theta \in \Theta\), against a \(q\)-player (the sampler), whose goal is to \textit{maximize} the objective function by selecting \(q \in Q^\alpha\).

Note that for each \(\theta\), the inner optimization in the game \((\text{4})\) is a linear program. Hence, its solution is a vertex of the set \(Q^\alpha\). Thus, the game becomes:
\[
\min_{\theta} \max_{q \in Q^\alpha} \mathbb{E}_q[L_i(\theta)] = \min_{\theta} \max_{q \in Q^\alpha} \frac{1}{k} \sum_{i \in I} L_i(\theta), \quad (\text{5})
\]
for \(I_k = \{I \subseteq [N] : |I| = k\}\). Then, for a fixed \(\theta\), the inner optimization is easily solved by simply sorting the losses, and selecting the largest \(k\). For large data, however, this is prohibitive, as it would require computing the losses for all data points, invalidating all benefits of stochastic optimization.

\textit{Fan et al., 2017} directly propose the combinatorial learning problem \((\text{5})\), without motivating it with the CVaR. Nevertheless, to solve it they use the high-variance algorithm for Problem \((\text{3})\).

Instead, we propose to solve the game \((\text{4})\) directly. A powerful approach for this is to use online no-regret algorithms for each player \((\text{Rakhlin and Sridharan, 2013})\). The learning protocol in Algorithm 1 proceeds as follows. In each round, the sampler-player samples a point from the data set using distribution \(q_i\). Based on it, the learner-player updates the model \(\theta\). The players then only observe the loss at the sampled point for the selected \(\theta\). If both players suffer sublinear regret, the game dynamics converge to the solution of Problem \((\text{1})\). We provide details below.

**Algorithm 1: Learning Protocol**

**Input:** Data set \(D = \{(x_1, y_1), \ldots, (x_N, y_N)\}\)

**Input:** Learning algorithm for \(q\)- and \(\theta\)-players.

**for** \(t = 1, \ldots, T\) **do**

\(q\)-player samples \(i_t \sim q_t \in Q^\alpha\).

\(\theta\)-player chooses \(\theta_t \in \Theta\).

Both players incur costs \(L_i(\theta_t)\).

Players see \(L_i(\theta_t)\) and \(\nabla L_i(\theta_t)\).

end

### 4.2 Sampler (\(q\)-Player) Algorithm

From an online optimization perspective, the sampler player is faced with a vector of losses that the adver-
sary sets through $\theta_t$. To avoid clutter, we use the notation $L(\theta_t) = L_t$, and $L(\theta_t, x_t, y_t) = L_{t,i}$. The goal of the sampler player is to control its regret:

$$SR_T := \max_{q \in \mathbb{Q}^N} \sum_{t=1}^T q^\top L_t - \sum_{t=1}^T q_{t,i}^\top L_{t,i}. \quad (6)$$

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 sampler player faces two challenges. First, it needs to select its distribution before the learner player. Second, it only observes the loss $L_t$ at a single point $i_t$.

We first note that this problem is an adversarial linear bandit problem (Lattimore and Szepesvári 2018 Chapter 27). Below, we exploit further structure, which allows to devise more efficient algorithms than available for general linear bandit problems. Utilizing Observation 3, we can restrict the player to select a subset of size $k$ elements from the ground set $[N]$ and compare against the best such subset. Thus, we face a combinatorial bandit problem (Lattimore and Szepesvári 2018 Chapter 30).

The algorithm with best regret bounds in this setting is Algorithm 1 from Alatur et al. (2019). Below, we introduce an algorithm k-EXP.3 based on their approach, which enables a highly efficient implementation in Section 4.5. The main idea is to maintain and update distributions over the $\binom{N}{k}$ subsets. Naively implemented, such an approach is not practical as it requires storing and updating a variable $\vec{w} \in \mathbb{R}^{\binom{N}{k}}$. Instead, we make use of special structure of 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 $k$ such that 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 $J$.  

We now introduce our algorithm, which we call k-EXP.3 (Algorithm 2). It is similar to the classical EXP.3 algorithm Auer et al. 2002, in that it only stores a variable $w \in \mathbb{R}^N$ and updates only one entry in each iteration. Instead of sampling $i_t$ proportionally to $w_{t,i}$, however, it samples from the marginal distribution of the k-DPP parameterized by $w_t$. For $k = 1$, the algorithms are identical, and for $k = N$, each $q_t$ is simply the uniform distribution. Crucially, the decision vector $q$ is proportional to the marginal distribution $\mathbb{P}_w(i)$ of the k-DPP with kernel $K = \text{diag} \, w$.

**Algorithm 2:** k-EXP.3

**Input:** Learning rate $\eta$

**Initialize weights** to $w_0 = 1_N$.

**for** $t = 1, \ldots, T$ **do**

- Sample element $i_t \sim q_t = \frac{1}{k} \mathbb{P}_w(i)$.
- Observe loss $L_{t,i_t}$.
- Build unbiased estimate $\hat{L}_t = \frac{L_{t,i_t}}{q_{t,i_t}}[i_t = i_t]$.
- Update weights $w_{t+1,i} = w_{t,i} e^{\eta \hat{L}_{t,i}}$.

**end**

**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 of at most $O(\sqrt{TN \log N})$.

**Proof.** For a detailed proof please refer to Appendix A.2. Here, we just sketch the proof. For the iterates $q_t$ of k-EXP.3, we need the following three facts. First, we prove in Proposition 2 that the iterates of the algorithm are effectively in $Q^N$. Second, we prove in Proposition 3 that the comparator in the regret of Alatur et al. (2019) and in the sampler regret have the same value (scaled by $k$). Finally, the result follows as a corollary from these propositions and Alatur et al. (2019 Lemma 1).

### 4.3 Learner (θ-Player) Algorithm

Analogous to the sampler player, the objective of the learner player is to control its regret defined as:

$$LR_T := \sum_{t=1}^T q_t^\top L(\theta) - \min_{\theta \in \Theta} \sum_{t=1}^T q_t^\top L(\theta). \quad (7)$$

The goal of this section is to discuss a sublinear-regret algorithm for the learner player. The key observation is that this player chooses $\theta_t$ after the sampler player selects $q_t$. For this reason, the learner player can play the Be-The-Leader (BTL) algorithm, namely:

$$\theta_t = \arg\min_{\theta \in \Theta} \sum_{\tau=1}^t q_{\tau}^\top L(\theta) = \arg\min_{\theta \in \Theta} \bar{q}_{\tau}^\top L(\theta), \quad (8)$$

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.

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

**Proof.** See Appendix A.3.

For each new $q_t$ that the sampler player selects, the learner player must solve a weighted empirical loss minimization in Problem (5). For convex problems,
we know that it is not necessary to solve exactly the BTL algorithm and algorithms such as online-SGD (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 and Kabadi, 1987). Obtaining provable 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 approximate \( \tilde{q}_i \) by the sequence of samples \( q_i \) that the sampler player provides. We perform stochastic optimization (SGD or its variants) with respect to these samples. Namely, for each \( i_t \sim q_t \), the learner chooses \( \theta_i := \theta_{i-1} + \eta \nabla L_i(\theta_{i-1}) \). Note that this is not online-SGD because in BTL the loss \( L \) is observed at index \( i_t \), whereas in SGD at index \( i_{t-1} \).

### 4.4 Game Dynamics

We now show that if both players play the no-regret algorithms discussed above, they (approximately) solve the game. The minimax equilibrium of the game is the point \((\theta^*, q^*)\) that satisfies \( q^T L(\theta^*) \leq q^T L(\theta^*) \leq q^T L(\theta) \). We assume that this point exists (e.g., when the sets \( Q^\alpha \) and \( \Theta \) are compact). The game regret is:

\[
\text{GameRegret}_T := \sum_{t=1}^{T} J(\theta_t, q^*) - J(\theta^*, q_t).
\]

**Theorem 1** (Game Sublinear-Regret). Let \( L_i(\cdot) : \Theta \rightarrow [0,1], i = \{1,\ldots,N\} \) be a fixed set of loss functions. If the sampler player uses \( k \cdot \text{EXP.3} \) (Algorithm 2) and the learner player uses the BTL algorithm, then the game has regret \( O(\sqrt{T N \log N}) \).

**Proof.** To bound the regret, we bound it with the sum of the Learner and Sampler regret as follows:

\[
\text{GameRegret}_T = \sum_{t=1}^{T} J(\theta_t, q^*) - J(\theta^*, q_t),
\]

\[= \max_{q \in Q^\alpha} \sum_{t=1}^{T} J(\theta_t, q) - J(\theta^*, q_t),
\]

\[\leq \max_{q \in Q^\alpha} \sum_{t=1}^{T} J(\theta_t, q) - J(\theta_t, q_t),
\]

\[\text{(Lemma 1)} \leq O(\sqrt{T N \log N}). \]

**Theorem 2** (Implications for learning with the CVaR). Let \( L_i(\cdot) : \Theta \rightarrow [0,1], i = \{1,\ldots,N\} \) be a set of loss functions sampled from a distribution \( D \). Let \( \theta^* \) be the minimizer of the CVaR of the empirical distribution \( \hat{C}^\alpha \). Let \( \theta_t \) be the output of the sequence of iterates of the two-player algorithm. The average excess CVaR of the algorithm is bounded as:

\[
\frac{1}{T} \sum_{t=1}^{T} \hat{C}^\alpha[L(\theta_t)] - \hat{C}^\alpha[L(\theta^*)] \leq O(\text{GameRegret}_T / T).
\]

**Proof.** The average excess CVaR is bounded by the average 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 goes to zero.

### 4.5 Efficient Sampling from k-DPP Marginals

In Section 4.2 we propose an algorithm that maintains and updates the diagonal elements of a k-DPP \( w_t \), but we did not address how to sample nor how to compute the marginal distribution \( q_t \).

The challenges in our setting are the following. First, we aim for a sampling algorithm with low computational complexity (at most \( O(\log(N)) \)), to maintain the computational advantages of stochastic optimization. Second, we aim for an algorithm that is numerically stable for large k-DPPs, to scale to large-scale data sets. Finally, the k-DPP changes between iterations, hence we need a method that efficiently adapts to changing distributions.

**Sample Complexity** 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 \( O(Nk) \) (Li et al., 2016; Anari et al., 2016). Compared to general k-DPPs, our setting has the advantage that the k-DPP is diagonal and there is no need for performing an eigendecomposition of the kernel matrix. Instead, we directly sample from the singleton-marginal distribution, which takes \( O(\log(N)) \) using the same sum-tree data-structure as Shalev-Shwartz and Wexler (2016). The marginals of diagonal k-DPPs are:

\[
q_i = \frac{w_i e_{i-1}^k}{e_N^k},
\]

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 \( e_{N-1}^k \) is the elementary symmetric polynomial of size \( k - 1 \) for the ground set \([N] \setminus i\).
Large-Scale Approximation Naively computing the elementary symmetric polynomials has a complexity of $O(N^2k) = O(\alpha N^3)$ using (Kulesza et al., 2012 Algorithm 7). Using a specialized binary tree algorithm takes $O(N \log(N) k^2) = O(\alpha^2 N^3)$. Even if this computation could be performed fast, exact computation of the elementary symmetric polynomials is numerically unstable.

Barthelmé et al. (2019) observe this issue and propose an approximation to k-DPPs valid for large-scale ground sets which has better numerical properties. They show empirically that for $N = 200$, computing the marginals $[10]$ leads to numerical overflow. The main idea Barthelmé et al. (2019) propose 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 has rate $O(1/N)$ when $N \to \infty$ and $k/N \to \alpha$. The marginal probabilities of this matched DPP are:

$$
\hat{q}_i = \frac{w_i e^\nu}{1 + w_i e^\nu},
$$

where $\nu$ softly enforces the sample size constraint $\sum_{i=1}^N \frac{w_i e^\nu}{1 + w_i e^\nu} = k$. Direct sampling from these marginals still takes $O(\log(N))$ but the numerical properties are superior to those in Eq. (10).

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 $\nu$. We found three possible solutions. First, we can take $O(N)$ operations to solve for $\nu$. In practice, solving this equation is extremely fast when using the previous solution as a warm start. Second, we can just keep $\nu$ 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 $\nu$ when $w_i$ changes in the coupling constraint. The gradient is $\frac{\partial \nu}{\partial w_i} = -\frac{1}{w_i}$, therefore the update rule is $\nu_{t+1} = \nu_t - (e^{\eta L_{w_i} (\theta_i)} - 1)$. Note that this also introduces an approximation error, thus, every $O(N)$ steps one must solve again for $O(N)$. In experiments, we observe no performance difference between the first and third strategy.

5 EXPERIMENTS

5.1 Experimental Setup

We optimize the CVaR of a loss at level $\alpha = 0.1$, hence $k = \lfloor 0.1N \rfloor$, where $N$ is the size of the data set. In classification tasks, we consider the cross-entropy loss as a surrogate of the 0/1 loss, and in regression tasks, we consider the squared loss.

We test our algorithm on eight different UCI classification data sets, three UCI regression data sets and three synthetic regression data sets (Dua and Graff, 2017). We use linear classifiers/regressors to ensure a convex optimization setting.

We also test our algorithm in large-scale data sets and classes of nonlinear function approximators, yielding non-convex problems. We use LeNet-5 neural network (LeCun et al., 1995) for MNIST (LeCun et al., 1995) and we add dropout (Hinton et al., 2012) for Fashion-MNIST (Xiao et al., 2017); for CIFAR-10 (Krizhevsky et al., 2014) we use (Simonyan and Zisserman, 2014) with batch-norm (Ioffe and Szegedy, 2015).

Our algorithm applies our adaptive sampling algorithm to select data points for the learner player, who uses a variant of SGD to optimize $\theta$. We compare it to three baselines: first, an i.i.d. sampling scheme that optimizes Problem (3) (CVaR); second, an i.i.d. sampling scheme that uses the mean of the losses as a learning algorithm (MEAN); third, an i.i.d. sampling scheme that uses a mini-batch relaxation of the $\sum_i [x_i]_+$ non-linearity with $T \log(\sum_i e^{x_i/T})$ proposed by Nemirovski and Shapiro (2006) (SOFT). For optimizing $\theta$ we employ ADAM (Kingma and Ba, 2014). For more details, please refer to Appendix C.

5.2 Convex Learning Results

We show results for classification tasks in Figure 2 and for regression tasks in Figure 3. Here, we only use linear function approximators, so we evaluate the different algorithms in a controlled convex setting, where the CVaR algorithm has convergence guarantees.

Regression Tasks In regression tasks, our adaptive algorithm outperforms all other algorithms in terms of the CVaR of the loss. Furthermore, the average loss of adaptive also outperforms the average loss of CVaR. In fact, it is also competitive with MEAN, and in normal and pareto data sets it outperforms it. This shows that our adaptive algorithm benefits from the initial iterates of the sampling algorithm, when it learns about the average loss. As the optimization algorithm advances, the sampling algorithm focuses on harder examples, reducing also the CVaR.

Classification Tasks In classification tasks, the original CVaR algorithm performs best in terms of the CVaR of the surrogate loss. However, this comes at considerable cost for the accuracy, which for CVaR is worse than the other algorithms in almost all data.
Our adaptive algorithm achieves the best of both worlds, also in classification tasks. In the first few epochs, when samples come from the uniform distribution, it optimizes the accuracy and, once the accuracy is good, it learns about the extreme events. adaptive has accuracy comparable to mean in almost all data sets and outperforms it in terms of the CVaR. The soft algorithm also yields good accuracy but usually a much higher CVaR than the adaptive algorithm.

The Value-at-Risk is also a (non-coherent) risk measure that is commonly used in practice: it is the $1 - \alpha$ quantile of the distribution. Quantiles are non-differentiable functions of the samples and there are no easy algorithms for minimizing the quantile of a loss. Instead, the CVaR is a tight convex upper bound for the quantile (Nemirovski and Shapiro, 2006). Thus, the VaR evaluation criterion is a common use case of our algorithm. We see that, in convex settings, the cvar and our adaptive algorithm perform similarly.

These experiments suggest that in the convex setting, our adaptive algorithm is competitive to the cvar algorithm and outperforms the soft variant. Furthermore, our algorithm benefits from the initial states where the sampling distribution is close to uniform to get competitive performance w.r.t. the mean algorithm, when comparing the expected loss.

5.3 Large-Scale Non-Convex Learning

Figure 4 shows results for Fashion-MNIST, MNIST, and CIFAR-10, and Figure 5 illustrates the learning dynamics of Fashion-MNIST. To obtain confidence intervals in the plots, we repeat the experiments with five different random seeds. The multi-class F1-score is the harmonic average of the minimum one vs. all precision and recall. For the CIFAR-10 data set, the soft algorithm yielded numerical overflow during training.

CVaR-Accuracy Tradeoff In the left figures, we see that the cvar algorithm has the lowest CVaR but has terrible accuracy, particularly in Fashion-MNIST and CIFAR-10 data sets. This agrees with the results from the UCI data sets. Our adaptive algorithm outperforms all others in the Accuracy-to-CVaR ratio in the bottom right figure in Figure 4.

In Figure 5, we see that our adaptive algorithm starts optimizing the average loss (as the mean algorithm), but then follows the optimization of the CVaR (as the cvar algorithm). However, it reaches simultaneously...
Figure 4: Results for Vision Data sets. In the top row we plot the CVaR, average and VaR of the loss, normalized to one to compare between data sets. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. Full bars indicate the mean and error bars one standard deviation. For CIFAR-10, SOFT yields numerical overflow, thus we omit the results.

Figure 5: Learning Dynamics on the Fashion-MNIST data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

6 CONCLUSIONS

We consider the CVaR of the loss distribution as a risk-averse learning criterion. We notice that the typical way of solving the optimization algorithm is not useful for modern machine learning tasks due to high variance of the gradient estimates. To address this issue, we propose an adaptive sampling algorithm that is based on a distributionally robust formulation of the CVaR. We provably solve the game by applying SGD on the sequence of examples that the adaptive sampling algorithm provides. Furthermore, we provide an efficient implementation for the adaptive sampling algorithm based on DPPs. Finally, we demonstrate in a range of experiments that our adaptive algorithm is superior to the CVAR algorithm in Problem 3 in regression and classification tasks, both in the convex and non-convex learning settings.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme grant agreement No. 815943. It also received funding from a Sloan Research Fellowship and the Defense Advanced Research Projects Agency (grant numberYFA17 N66001-17-1-4039). The views, opinions, and/or findings contained in this article are those of
the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

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. *arXiv preprint arXiv:1902.08036*, 2019.

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, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical finance*, 9(3):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*, 25(4B):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.

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.

M. C. Carneiro, G. P. Ribas, and S. Hamacher. Risk management in the oil supply chain: a cvar approach. *Industrial & Engineering Chemistry Research*, 49(7):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.

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

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

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

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

E. Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3–4):157–325, 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 preprint arXiv:1207.0580*, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint 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 preprint arXiv:1412.6980*, 2014.

A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. online: [http://www.cs.toronto.edu/kriz/cifar.html](http://www.cs.toronto.edu/kriz/cifar.html), 55, 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.

C. Li, S. Sra, and S. Jegelka. Fast mixing markov chains for strongly rayleigh measures, dpps, and constrained sampling. In Advances in Neural Information Processing Systems, pages 4188–4196, 2016.

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, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. 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: A calibration theorem. In Handbook of the Fundamentals of Financial Decision Making: Part I, pages 241–252. World Scientific, 2013.

S. Rakhlin and K. Sridharan. Optimization, learning, and games with predictable sequences. In Advances in Neural Information Processing Systems, pages 3066–3074, 2013.

R. T. Rockafellar, S. Uryasev, et al. Optimization of conditional value-at-risk. Journal of risk, 2:21–42, 2000.
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 \( 1 - \delta \),

\[
E \left[ \sup_{h \in \mathcal{H}} |\hat{C}^\alpha[L(h)] - C^\alpha[L(h)]| \right] \leq \frac{1}{\alpha} \log \left( \frac{|\mathcal{H}|}{\delta} \right).
\]

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 \( h \in \mathcal{H} \):

\[
C^\alpha(L(h)) \geq \hat{C}^\alpha(L(h)) - \frac{1}{\alpha} \sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}
\]

\[
C^\alpha(L(h)) \leq \hat{C}^\alpha(L(h)) + \frac{\sqrt{5\log(6|\mathcal{H}|/\delta)}}{\alpha N}
\]

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

\[
C^\alpha(L(h)) \geq \hat{C}^\alpha(L(h)) - \frac{1}{\alpha} \sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{N}}
\]

\[
C^\alpha(L(h)) \leq \hat{C}^\alpha(L(h)) + \frac{\sqrt{5\log(6|\mathcal{H}|/\delta)}}{\alpha N}
\]

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

A.2 Proof of Lemma 1

Lemma 1. Let the sampler play the k.EXP.3 Algorithm with \( \eta = \sqrt{\frac{\log N}{NT}} \). Then, she suffers a sampler regret 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 \( K = \text{diag}(w) \) are in the set \( Q_k^\alpha = \{ kw \in \mathbb{R}^N \mid q \in Q^\alpha \} \).

Proof. For any \( w \in \mathbb{R}^N \) the marginals of the k-DPP with kernel \( K = \text{diag}(w) \) are:

\[
P_w(i) = \sum_{I \ni i} \prod_{i' \notin I} w_{i'} = \frac{\prod_{i' \in I} w_{i'}}{\sum_{I \ni i} \prod_{i' \in I} w_{i'}}.
\]

From eq. (12), clearly \( 0 \leq P_w(i) \leq 1 \). Summing eq. (12) over \( i \) we get:

\[
\sum_i P_w(i) = \sum_{I \ni i} \prod_{i' \notin I} w_{i'} = \sum_{I \ni i} \prod_{i' \in I} w_{i'} \leq k.
\]

This shows that \( P_w(i) \in Q_k^\alpha \).

Proposition 3. Let \( \tilde{L}_t = \sum_{i \leq t} L_i \). Let \( \Delta = \{ \hat{w} \in \mathbb{R}^N \mid 0 \leq w_i \leq 1, \sum_i w_i = 1 \} \) the set of distributions over the \( \binom{N}{k} \) subsets of size \( k \) of the ground set \([N]\).

\[
\max_{q \in Q^\alpha} \sum_{t=1}^T q^\top L_t = \max \frac{1}{k} \sum_{t=1}^T \hat{w} \tilde{L}_t
\]

Proof. Both left and right sides of (13) are linear programs over a convex polytope, hence the solution is in one of its vertices (Murty, 1983). The vertices of \( Q^\alpha \) are vectors \( \frac{1}{N} \mathbf{1}_I \). 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 \( q^* = \frac{1}{N} \mathbf{1}_I \) be the solution of the l.h.s. of (13). Assume that \( I \neq I^* \) is the solution of the right hand side. This implies that \( \tilde{L}_I \geq \tilde{L}_{I^*} \). Therefore, \( \sum_{i \in I} L_i \geq \sum_{i \in I^*} L_i \). This in turn implies that \( 1_i L_i \geq 1_{I^*} L_{I^*} \), which contradicts the first predicate. In the case the equalities hold, then the values l.h.s and r.h.s. of equation (13) are also equal.

Proof of Lemma 2.

\[
SR_T = \max_{q \in Q^\alpha} \sum_{t=1}^T q^\top L_t - \sum_{t=1}^T q_t^\top L_t
\]

\[
= \max_{\hat{w} \in \Delta} \frac{1}{k} \sum_{t=1}^T \hat{w} \tilde{L}_t - \sum_{t=1}^T q_t^\top L_t
\]

\[
= \frac{1}{k} \left( \max_{a \in Q^\alpha} \sum_{t=1}^T \hat{w} \tilde{L}_t - \sum_{t=1}^T \sum_i P_w(i)L_i \right)
\]

\[
\leq O(\sqrt{NT \log N})
\]

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 \( T = 1 \), \( LR_1 = 0 \). Assume true for \( T - 1 \), the inductive hy-
pothesis is \( \text{LR}_{T-1} \leq 0 \). The regret at time \( T \) is:

\[
\text{LR}_T = \sum_{t=1}^{T} q_t^\top L(\theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} q_t^\top L(\theta),
\]

\[
= \sum_{t=1}^{T} q_t^\top [L(\theta_t) - L(\theta_T)] = \sum_{t=1}^{T-1} q_t^\top [L(\theta_t) - L(\theta_T)],
\]

\[
= \text{LR}_{T-1} + \min_{\theta \in \Theta} \sum_{t=1}^{T-1} q_t^\top L(\theta) - \sum_{t=1}^{T-1} q_t^\top L(\theta_T) \leq 0. 
\]

\[ \square \]

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 \( \mathcal{B} \) exactly. Instead, stochastic gradient descent (SGD) \cite{Zinkevich:2003} or online mirror descent (OMD) \cite{BeckTeboulle:2003} both have no-regret guarantees. We focus now on SGD but, for certain geometries of \( \Theta \) and appropriate mirror maps, OMD has exponentially better regret guarantees (in terms of the dimension of the problem).

**Lemma 3.** Let assume that \( L_i(\cdot) : \Theta \rightarrow [0,1] \) 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.** \cite{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 \cite{Audibert:2013}.

**Corollary 1.** Let \( L_i(\cdot) : \Theta \rightarrow [0,1] \) be any sequence of convex losses. Let the learner player play SGD (or OMD with an appropriate mirror map) and the sampler player play Algorithm 3, then the game has regret \( O(\sqrt{T}N\log N + \beta\sqrt{T}) \), where \( \beta \) is a problem-dependent constant.

C Experimental Setup

**Implementation:** We implement all our experiments using PyTorch \cite{Paszke:2017}.

**Datasets:** For classification we use the Adult, Australian Credit Approval, German Credit Data, Monk’s Problems-1, Spambase, and Splice-junction Gene Sequences datasets from the UCI repository \cite{Dua:2017} and the Titanic Disaster dataset from \cite{Eaton:1995}. For regression we use the Boston Housing, Abalone, and Energy Efficiency datasets from the UCI repository, the sinc dataset is synthetic recreated from \cite{Fan:2017}, and normal and pareto datasets are synthetic datasets recreated from \cite{Brownlees:2015} with Gaussian and Pareto noise, respectively. For vision datasets, we use MNIST \cite{LeCun:1998}, Fashion-MNIST \cite{Xiao:2017}, and CIFAR-10 \cite{Krizhevsky:2014}.

**Dataset Preparation:** We split UCI datasets into train and test set using a 80/20 split. We use a 10-fold cross-validation scheme on the training set. For image classification we use as validation set the same images in the train set, however without applying data-augmentations. For discrete categorical data, we use a one-hot-encoding. We normalize continuous data.

**Hyper-Parameter Search:** For UCI datasets, we ran a grid search over the hyperparameters for all the algorithms with a single random seed. Then we select the set of hyper-parameters with the highest cross-validation score. The hyperparameters are:

1. ADAM initial learning rate \{0.001, 0.004, 0.01\},
2. Batch Size \{64, 128\},
3. Adaptive algorithm learning rate \{1.0, 0.5, 0.1, *\}, where (* ) is the optimal learning rate,
4. Mixing with uniform distribution \{0, 0.01, 0.1\},
5. Adaptive algorithm learning rate decay scheduling \{constant, O(1/\sqrt{T}), Adagrad\},
6. Sampling \{i.i.d. sampling, shuffle and cycle\},
7. SOFT algorithm Temperature \{0.1, 1.0, 10.0, 100\}.

For the image classification data sets, we perform the same hyperparameter search and select the highest validation score but for a single data-fold. For the test results, we repeat the experiments with five different random seeds, but left the hyper-parameters unchanged.

**Evaluation Metrics:** We report results on the test set. In regression tasks, we evaluate primarily the CVaR of the loss, and secondarily the mean and the VaR (the quantile) of the loss. In classification tasks, the CVaR of the 0/1 loss is not a useful metric. If the mis-classification rate is larger than \( \alpha \), then the CVaR of the 0/1 loss is zero, and if it is smaller than \( \alpha \) it is just the miss-classification rate scaled up by \( 1/\alpha \). Therefore, we consider both the average accuracy and the CVaR w.r.t. the cross-entropy loss.

**Further Experimental Results:** In Figures 6 to 13 we plot the learning dynamics of the classification data sets. In Figures 14 to 20 we plot the learning dynamics of the regression data sets.
Figure 6: Learning Dynamics on the Adult data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 7: Learning Dynamics on the Australian Credit Approval data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 8: Learning Dynamics on the German Credit Data data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.
Figure 9: Learning Dynamics on the Monks-Problems data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 10: Learning Dynamics on the Phoneme data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 11: Learning Dynamics on the Spambase data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.
Figure 12: Learning Dynamics on the Splice-junction Gene Sequences data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 13: Learning Dynamics on the Titanic data. In the top row we plot the CVaR, mean and VaR of the loss. In the bottom row we plot the accuracy, F1-score and accuracy-to-cvar ratio. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 14: Learning Dynamics on the Abalone data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 15: Learning Dynamics on the Boston Housing data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.
Figure 16: Learning Dynamics on the CPU-SMALL data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 17: Learning Dynamics on the Normal data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 18: Learning Dynamics on the Pareto data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 19: Learning Dynamics on the Sinc data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.

Figure 20: Learning Dynamics on the Energy Efficiency data. We plot the CVaR, mean and VaR of the loss. All these are from the validation set. In solid line we plot the mean and in shaded we include one standard deviation.