Learning piecewise Lipschitz functions in changing environments

Maria-Florina Balcan
Carnegie Mellon University

Travis Dick
Carnegie Mellon University

Dravyansh Sharma
Carnegie Mellon University

Abstract

Optimization in the presence of sharp (non-Lipschitz), unpredictable (w.r.t. time and amount) changes is a challenging and largely unexplored problem of great significance. We consider the class of piecewise Lipschitz functions, which is the most general online setting considered in the literature for the problem, and arises naturally in various combinatorial algorithm selection problems where utility functions can have sharp discontinuities. The usual performance metric of 'static' regret minimizes the gap between the payoff accumulated and that of the best fixed point for the entire duration, and thus fails to capture changing environments. Shifting regret is a useful alternative, which allows for up to s environment *shifts*. In this work we provide an $O(\sqrt{sdT \log T} + sT^{1-\beta})$ regret bound for β-dispersed functions, where β roughly quantifies the rate at which discontinuities appear in the utility functions in expectation (typically $\beta \geq 1/2$ in problems of practical interest [Balcan et al., 2019, Balcan et al., 2018a]). We also present a lower bound tight up to sub-logarithmic factors. We further obtain improved bounds when selecting from a small pool of experts. We empirically demonstrate a key application of our algorithms to online clustering problems on popular benchmarks.

1 Introduction

Online optimization is well-studied in the online learning community [Cesa-Bianchi and Lugosi, 2006, Hazan et al., 2016]. It consists of a repeated game with T iterations. At iteration t, the player chooses a point ρ_t from a compact decision set $\mathcal{C} \subset \mathbb{R}^d$; after the choice is committed, a bounded utility function $u_t : \mathcal{C} \to [0, H]$ is revealed. We treat u_t as a reward function to be maximized, although one may also consider minimizing a loss function. The goal of the player is to minimize the regret, defined as the difference between the online cumulative payoff (i.e. $\sum_{t=1}^{T} u_t(\rho_t)$) and the cumulative payoff using an optimal offline choice in hindsight. In many real world problems, like online routing [Awerbuch and Kleinberg, 2008, Talebi et al., 2018], detecting spam email/bots [Sculley and Wachman, 2007, Cormack et al., 2008] and ad/content ranking [Wauthier et al., 2013, Combes et al., 2015], it is often inadequate to assume a fixed point will yield good payoff at all times. It is more natural to compute regret against a stronger offline baseline, say one which is allowed to switch the point a few times (say s *shifts*), to accommodate 'events' which significantly change the function values for certain time periods. The switching points are neither known in advance nor explicitly stated during the course of the game. This stronger baseline is known as *shifting regret* [Herbster and Warmuth, 1998].

Shifting regret is a particularly relevant metric for online learning problems in the context of algorithm configuration. This is an important family of non-convex optimization problems where the goal is to decide in a data-driven way what algorithm to use from a large family of algorithms for a given problem domain. In the online setting, one has a configurable algorithm such as an algorithm for clustering data [Balcan et al., 2017], and must solve a series of related problems, such as clustering news articles each day for a news reader or clustering drugstore sales information to detect disease outbreaks. For problems of this nature, significant events in the world or changing habits of buyers might require changes in algorithm parameters, and we would like the online algorithms to adapt smoothly.

Related work: We present the first results for shifting regret for non-convex utility functions which potentially have sharp discontinuities. Restricting attention to specific kinds of decision sets $\mathcal{C}$

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

and utility function classes yields several important problems. If $\mathcal{C}$ is a convex set and utility functions are concave functions (i.e. corresponding loss functions are convex), we get the Online Convex Optimization (OCO) problem [Zinkevich, 2003], which is a generalization of online linear regression [Kivinen and Warmuth, 1997] and prediction with expert advice [Littlestone and Warmuth, 1994]. Algorithms with $O(\sqrt{sT \log NT})$ regret are known for the case of s *shifts* for prediction with N experts and OCO on the N-simplex [Herbster and Warmuth, 1998, Cesa-Bianchi et al., 2012] using weight-sharing or regularization. We show how to extend the result to arbitrary compact sets of experts, and more general utility functions where convexity can no longer be exploited. Our key insight is to view the regularization as simultaneously inducing multiplicative weights update with restarts matching all possible shifted expert sequences, which allows us to use the *dispersion* condition introduced in [Balcan et al., 2018a]. Related notions like adaptive regret [Hazan and Seshadhri, 2007], strongly adaptive regret [Daniely et al., 2015, Jun et al., 2017], dynamic regret [Zinkevich, 2003, Jadbabaie et al., 2015] and sparse experts setting [Bousquet and Warmuth, 2002] have also been studied for finite experts.

Intuitively, a sequence of piecewise L-Lipschitz functions is well-*dispersed* if not too many functions are non-Lipschitz in the same region in $\mathcal{C}$. An assumption like this is necessary, since, even for piecewise constant functions, linear regret is unavoidable in the worst case [Cohen-Addad and Kanade, 2017]. Our shifting regret bounds are $O(\sqrt{sdT \log T} + sT^{1-\beta})$ which imply low regret for sufficiently dispersed (large enough β) functions. In a large range of applications, one can show $\beta \geq \frac{1}{2}$ [Balcan et al., 2018a]. This allows us to obtain tight regret bounds modulo sublogarithmic terms, providing a near-optimal characterization of the problem. Our analysis also readily extends to the closely related notion of adaptive regret [Hazan and Seshadhri, 2007]. Note that our setting generalizes the Online Non-Convex Learning (ONCL) problem where all functions are L-Lipschitz throughout [Maillard and Munos, 2010, Yang et al., 2018] for which shifting regret bounds have not been studied.

We demonstrate the effectiveness of our algorithm in solving the algorithm selection problem for a family of clustering algorithms parameterized by different ways to initialize k-means [Balcan et al., 2018b]. We consider the problem of online clustering, but unlike prior work which studies individual data points arriving in an online fashion [Liberty et al., 2016, Rakhlin et al., 2007], we look at complete clustering instances from some distribution(s) presented sequentially. Our experiments provide the first empirical evaluation of online algorithms for piecewise Lipschitz functions — prior work is limited to theoretical analysis [Balcan et al., 2018a] or experiments for the batch setting [Balcan et al., 2018b]. Our results also have applications in non-convex online problems like portfolio optimization [Merton, 1976] and online non-convex SVMs [Ertekin et al., 2010]. More broadly, for applications where one needs to tune hyperparameters that are not 'nice', our results imply it is necessary and sufficient to look at dispersion.

2 Problem setup

Consider the following repeated game. At each round $1 \leq t \leq T$ we are required to choose $\rho_t \in \mathcal{C} \subset \mathbb{R}^d$, are presented a piecewise L-Lipschitz function $u_t : \mathcal{C} \to [0, H]$ and experience reward $u_t(\rho_t)$.

In this work we will study s-shifted regret and (m-sparse, s-shifted) regret notions defined below.

Definition 1. *The s-shifted regret ('tracking regret' in [Herbster and Warmuth, 1998]) is given by*

$$\mathbb{E}\left[\max_{\substack{\rho_i^* \in \mathcal{C}, \\ t_0=1<t_1\cdots<t_s=T+1}} \sum_{i=1}^{s} \sum_{t=t_{i-1}}^{t_i-1} (u_t(\rho_i^*) - u_t(\rho_t))\right]$$

Note that for the i-th phase ($i \in [s]$) given by $[t_{i-1}, t_i - 1]$, the offline algorithm uses the same point ρ_i^*. The usual notion of regret compares the payoff of the online algorithm to the offline strategies that pick a fixed point $\rho^* \in \mathcal{C}$ for all $t \in [T]$ but here we compete against more powerful offline strategies that can use up to s distinct points ρ_i^* by switching the expert $s-1$ times. For $s = 1$, we retrieve the standard static regret.

Definition 2. *Extend Definition 1 with an additional constraint on the number of distinct experts used, $\left|\{\rho_i^* \mid 1 \leq i \leq s\}\right| \leq m$. We call this ($m$-sparse, s-shifted) regret [Bousquet and Warmuth, 2002].*

This restriction makes sense if we think of the adversary as likely to reuse the same experts again, or the changing environment to experience recurring events with similar payoff distributions.

Without further assumptions, no algorithm achieves sublinear regret, even when the payout functions are piecewise constant [Cohen-Addad and Kanade, 2017]. We will characterize our regret bounds in terms of the 'dispersion' [Balcan et al., 2018a, Balcan et al., 2019] of the utility functions, which roughly says that discontinuities are not too concentrated. Several other restrictions can be seen as a special case [Rakhlin et al., 2011, Cohen-Addad and Kanade, 2017].

Definition 3. *The sequence of utility functions $u_1, \dots, u_T$ is β-dispersed for the Lipschitz constant L if, for all T and for all $\epsilon \geq T^{-\beta}$, at most $\tilde{O}(\epsilon T)$ functions (the soft-O notation suppresses dependence on quantities beside ϵ, T and β) are not L-Lipschitz in any ball of size ϵ contained in $\mathcal{C}$. Further if the utility functions are obtained from some distribution, the random process generating them is said to be β-dispersed if the above holds in expectation, i.e. if for all T and for all $\epsilon \geq T^{-\beta}$,*

$$\mathbb{E}\left[\max_{\rho \in \mathcal{C}} \left|\{t \mid u_t \text{ not } L\text{-Lipschitz in } \mathcal{B}(\rho, \epsilon)\}\right|\right] \leq \tilde{O}(\epsilon T)$$

For 'static' regret, a continuous version of exponential weight updates gives a tight bound of $\tilde{O}(\sqrt{dT} + T^{1-\beta})$ [Balcan et al., 2018a]. They further show that in several cases of practical interest one can prove dispersion with $\beta = 1/2$ and the algorithm enjoys $\tilde{O}(\sqrt{dT})$ regret. This algorithm may, however, have $\Omega(T)$ s-shifted regret even with a single switch ($s = 2$), and hence is not suited to changing environments (Appendix B).

3 Algorithms with low shifting regret

In this section we describe online algorithms with good shifting regret, but defer the actual regret analysis to Section 4. First we present a discretization based algorithm that simply uses a finite expert algorithm given a discretization of $\mathcal{C}$. This algorithm will give us the reasonable target regret bounds we should shoot for, although the discretization results in exponentially many experts.

Algorithm 1 Discrete Fixed Share Forecaster

Input: β, the dispersion parameter

1. Obtain a $T^{-\beta}$-discretization $\mathcal{D}$ of $\mathcal{C}$ (i.e. any $c \in \mathcal{C}$ is within $T^{-\beta}$ of some $d \in \mathcal{D}$)

2. Apply an optimal algorithm for finite experts with points in $\mathcal{D}$ as the experts (e.g. fixed share [Herbster and Warmuth, 1998])

We introduce a continuous version of the fixed share algorithm (Algorithm 2). We maintain weights for all points similar to the Exponential Forecaster of [Balcan et al., 2018a] which updates these weights in proportion to their exponentiated scaled utility $e^{\lambda u_t(\cdot)}$ ($\lambda \in (0, 1/H]$ is a *step size parameter* which controls how aggressively the algorithm updates its weights). The main difference is to update the weights with a mixture of the exponential update and a constant additive boost at all points in some proportion α (the *exploration parameter*, optimal value derived in Section 4) which remains fixed for the duration of the game. This allows the algorithm to balance exploitation (exponential update assigns high weights to points with high past utility) with exploration, which turns out to be critical for success in changing environments. We will show this algorithm has good s-shifted regret in Section 4. It also enjoys good adaptive regret [Hazan and Seshadhri, 2007] (see Appendix D).

Algorithm 2 Fixed Share Exponential Forecaster (Fixed Share EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$

1. $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$

2. For each $t = 1, 2, \dots, T$:

   i. $W_t := \int_{\mathcal{C}} w_t(\rho) d\rho$

   ii. Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$

   iii. Observe $u_t(\cdot)$

   iv. Let $e_t(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$. For each $\rho \in \mathcal{C}$, set

$$w_{t+1}(\rho) = (1-\alpha) e_t(\rho) + \frac{\alpha}{\textsc{Vol}(\mathcal{C})} \int_{\mathcal{C}} e_t(\rho) d\rho \quad (1)$$

Notice that it is not clear how to implement the Algorithm 2 from its description. We cannot store all the weights or sample easily since we have uncountably many points $\rho \in \mathcal{C}$. We will show how to efficiently sample according to p_t without necessarily computing it exactly or storing the exact weights in Section 5.

Algorithm 3 Generalized Share Exponential Forecaster (Generalized Share EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$, discount rate $\gamma \in [0, 1]$

1. $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$

2. For each $t = 1, 2, \dots, T$:

   i. $W_t := \int_{\mathcal{C}} w_t(\rho) d\rho$

   ii. Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$

   iii. Let $e_t(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$ and $\beta_{i,t} = \frac{e^{-\gamma(t-i)}}{\sum_{j=1}^{t} e^{-\gamma(t-j)}}$. For each $\rho \in \mathcal{C}$, set $w_{t+1}(\rho)$ to $(1-\alpha) e_t(\rho) + \alpha \left(\int_{\mathcal{C}} e_t(\rho) d\rho\right) \sum_{i=1}^{t} \beta_{i,t} p_i(\rho)$

As it turns out adding equal weights to all points for exploration does not allow us to exploit recurring environments of the (m-sparse, s-shifted) setting very well. To overcome this, we replace the uniform update with a prior consisting of a weighted mixture of all the previous probability distributions used for sampling

(Algorithm 3). Notice that this includes uniformly random exploration as the first probability distribution $p_1(\cdot)$ is uniformly random, but the weight on this distribution decreases exponentially with time according to *discount rate* γ (more precisely, it decays by a factor $e^{-\gamma}$ with each time step). While exploration in Algorithm 2 is limited to starting afresh, here it includes partial resets to explore again from all past states, with an exponentially discounted rate (cf. Theorems 6, 7).

4 Analysis of algorithms

We will now analyse the algorithms in Section 3. At a high level, the algorithms have been designed to ensure that the optimal solution, and its neighborhood, in hindsight have a large total density. We achieve this by carefully setting the parameters, in particular the *exploration parameter* which controls the rate at which we allow our confidence on 'good' experts to change. Lipschitzness and dispersion are then used to ensure that solutions sufficiently close to the optimum are also good on average.

4.1 Regret bounds

In the remainder of this section we will have the following setting. We assume the utility functions $u_t : \mathcal{C} \to [0, H], t \in [T]$ are piecewise L-Lipschitz and β-dispersed (definition 3), where $\mathcal{C} \subset \mathbb{R}^d$ is contained in a ball of radius R.

Theorem 4. *Let $R^{finite}(T, s, N)$ denote the s-shifted regret for the finite experts problem on N experts, for the algorithm used in step 2 of Algorithm 1. Then Algorithm 1 enjoys s-shifted regret $R^{\mathcal{C}}(T, s)$ which satisfies*

$$R^{\mathcal{C}}(T,s) \le R^{finite}\left(T, s, \left(3RT^{\beta}\right)^{d}\right) + (sH+L)O(T^{1-\beta}).$$

The proof of Theorem 4 is straightforward using the definition of dispersion and is deferred to Appendix A. This gives us the following target bound for our more efficient algorithms.

Corollary 5. *The s-shifted regret of Algorithm 1 is $O(H\sqrt{sT(d\log(RT^{\beta}) + \log(T/s))} + (sH + L)T^{1-\beta})$.*

Proof. There are known algorithms e.g. Fixed-Share ([Herbster and Warmuth, 1998]) which obtain $R^{\text{finite}}(T, s, N) \le O(\sqrt{sT\log(NT/s)})$. Applying Theorem 4 gives the desired upper bound. □

Under the same conditions, we will show the following bounds for our algorithms. In the following statements, we give approximate values for the parameters α, γ and λ under the assumptions $m \ll s, s \ll T$. See proofs in Appendix C for more precise values.

Theorem 6. *The s-shifted regret of Algorithm 2 with $\alpha = s/T$ and $\lambda = \sqrt{s(d\log(RT^{\beta}) + \log(T/s))/T/H}$ is $O(H\sqrt{sT(d\log(RT^{\beta}) + \log(T/s))} + (sH + L)T^{1-\beta})$.*

Remark. *The algorithms assume knowledge of s/T, the average number of shifts per time. For unknown s, the strongly adaptive algorithms of [Daniely et al., 2015, Jun et al., 2017] can be used with the same meta-algorithms and substituting continuous exponential forecasters as black-box algorithms.*

Similarly for Algorithm 3 we can show low (m-sparse, s-shifted) regret as well. (In particular this implies s-shifted regret almost as good as Algorithm 2.)

Theorem 7. *The (m-sparse, s-shifted) regret of Algorithm 3 is $O(H\sqrt{T(md\log(RT^{\beta}) + s\log(mT/s))} + (mH + L)T^{1-\beta})$ for $\alpha = s/T$, $\gamma = s/mT$ and $\lambda = \sqrt{(md\log(RT^{\beta}) + s\log(T/s))/T/H}$.*

4.2 Proof sketch and insights

We start with some observations about the weights W_t in Algorithm 2.

Lemma 8 (Algorithm 2)**.** $W_{t+1} = \int_{\mathcal{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho$.

The update rule (1) had the uniform exploration term scaled just appropriately so this relation is satisfied. We will now relate W_t with weights resulting from pure exponential updates, i.e. $\alpha = 0$ in Algorithm 2 (also the Exponential Forecaster algorithm of [Balcan et al., 2018a]). The following definition corresponds to weights for running Exponential Forecaster starting at some time τ.

Definition 9. *For any $\rho \in \mathcal{C}$ and $\tau \le \tau' \in [T]$ define $\tilde{w}(\rho; \tau, \tau')$ to be the weight of expert ρ, and $\tilde{W}(\tau, \tau')$ to be the normalizing constant, if we ran the Exponential Forecaster of [Balcan et al., 2018a] starting from time τ up till time τ', i.e. $\tilde{w}(\rho; \tau, \tau') := e^{\lambda \sum_{t=\tau}^{\tau'-1} u_t(\rho)}$ and $\tilde{W}(\tau, \tau') := \int_{\mathcal{C}} \tilde{w}(\rho; \tau, \tau') d\rho$.*

We consider Algorithm 4 obtained by a slight modification in the update rule (1) of Fixed Share EF (Algorithm 2) which makes it easier to analyze. Essentially we replace the deterministic α-mixture by a randomized one, so at each turn we either explicitly 'restart' with probability α by putting the same weight on each point, or else apply the exponential update. We note that Algorithm 4 is introduced to simplify the proof of Theorem 6, and in particular does *not* result in low regret itself. The issue is that even though the weights are correct in expectation (Lemma 10), their ratio (probability $p_t(\rho)$) is not. In particular, the optimal parameter value of α for Fixed Share EF allows the possibility of pure exponential updates over a long period of time with a constant probability in Algorithm

4, which implies linear regret (see Appendix B, Theorem 21). This also makes the implementation of Fixed Share EF somewhat trickier (Section 5).

Algorithm 4 Random Restarts Exponential Forecaster (Random Restarts EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$

1. $\hat{w}_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For each $t = 1, 2, \ldots, T$:
   i. $\hat{W}_t := \int_{\mathcal{C}} \hat{w}_t(\rho) d\rho$
   ii. Sample ρ with probability proportional to $\hat{w}_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{\hat{w}_t(\rho)}{\hat{W}_t}$
   iii. Sample z_t uniformly in $[0, 1]$ and set

$$\hat{w}_{t+1}(\rho) = \begin{cases} e^{\lambda u_t(\rho)} \hat{w}_t(\rho) & \text{if } z_t < 1 - \alpha \\ \frac{\int_{\mathcal{C}} e^{\lambda u_t(\rho)} \hat{w}_t(\rho) d\rho}{\textsc{Vol}(\mathcal{C})} & \text{otherwise} \end{cases}$$

The expected weights of Algorithm 4 (over the coin flips used in weight setting) are the same as the actual weights of Algorithm 2 (proof in Appendix C).

Lemma 10. ***(Algorithm 2)*** *For each $t \in [T]$, $w_t(\rho) = \mathbb{E}[\hat{w}_t(\rho)]$ and $W_t = \mathbb{E}[\hat{W}_t]$, where the expectations are over random restarts $\mathbf{z}_t = \{z_1, \ldots, z_{t-1}\}$.*

The next lemma provides intuition for looking at our algorithm as a weighted superposition of several exponential update subsequences with restarts. This novel insight establishes a tight connection between the algorithms and is crucial for our analysis.

Lemma 11. ***(Algorithm 2)*** *W_{T+1} equals the sum*

$$\sum_{s \in [T]} \sum_{t_0 = 1 < t_1 \cdots < t_s = T+1} \frac{\alpha^{s-1}(1-\alpha)^{T-s}}{\textsc{Vol}(\mathcal{C})^{s-1}} \prod_{i=1}^{s} \tilde{W}(t_{i-1}, t_i)$$

Proof Sketch. Each term corresponds to the weight when we pick a number $s \in [T]$ for the number of times we start afresh with a uniformly random point ρ at times $\mathbf{t}_s = \{t_1, \ldots, t_{s-1}\}$ and do the regular exponential weighted forecaster in the intermediate periods. We have a weighted sum over all these terms with a factor $\alpha/\textsc{Vol}(\mathcal{C})$ for each time we restart and $(1-\alpha)$ for each time we continue with the Exponential Forecaster. □

We will now prove Theorem 6. The main idea is to show that the normalized exploration helps the total weights to provide a lower bound for the algorithm payoff. Also the total weights are competitive against the optimal payoff as they contain the exponential updates with the optimal set of switching points in Lemma 11 with a sufficiently large ('probability') coefficient.

Proof sketch of Theorem 6. We provide an upper and lower bound to $\frac{W_{T+1}}{W_1}$. The upper bound uses Lemma 8 and helps us lower bound the performance of the algorithm (see Appendix C) as

$$\frac{W_{T+1}}{W_1} \leq \exp\left(\frac{P(\mathcal{A})(e^{H\lambda} - 1)}{H}\right) \tag{2}$$

where $P(\mathcal{A})$ is the expected total payoff for Algorithm 2. We now upper bound the optimal payoff OPT by providing a lower bound for $\frac{W_{T+1}}{W_1}$. By Lemma 11 we have

$$W_{T+1} \geq \frac{\alpha^{s-1}(1-\alpha)^{T-s}}{\textsc{Vol}(\mathcal{C})^{s-1}} \prod_{i=1}^{s} \tilde{W}(t^*_{i-1}, t^*_i)$$

by dropping all terms save those that 'restart' exactly at the OPT expert switches $t^*_{0:s}$. Now using β-dispersion we can show (full proof in Appendix C)

$$\frac{W_{T+1}}{W_1} \geq \frac{\alpha^{s-1}(1-\alpha)^{T-s}}{(RT^\beta)^{sd}} e^{\lambda\left(OPT - (sH+L)O(T^{1-\beta})\right)}$$

Putting together with the upper bound (2), rearranging and optimizing the difference for α and λ concludes the proof. (See Appendix C for a full proof.) □

We now analyze Algorithm 3 for the sparse experts setting. We can adapt proofs of Lemmas 8 and 11 to easily establish Lemmas 12 and 13.

Lemma 12 (Algorithm 3)**.** $W_{t+1} = \int_{\mathcal{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho$.

Lemma 13. *Let $\pi_t(\rho) = \sum_{i=1}^{t-1} \beta_{i,t} p_i(\rho)$. For Algorithm 3, W_{T+1} can be shown to be equal to the sum*

$$\sum_{s \in [T]} \sum_{t_0 = 1 < \ldots t_s = T+1} \alpha^{s-1}(1-\alpha)^{T-s} \prod_{i=1}^{s} \tilde{W}(\pi_{t_{i-1}}; t_{i-1}, t_i)$$

where $\tilde{W}(p; \tau, \tau') := \int_{\mathcal{C}} p(\rho) \tilde{w}(\rho; \tau, \tau') d\rho$.

Corollary 14. *$W_T \geq \alpha(1-\alpha)^{T-t} \tilde{W}(\pi_t; t, T) W_t$, for all $t < T$.*

Proof. Consider the probability of last 'reset' (setting $w_t(\rho) = W_t \pi_t(\rho)$) at time t when computing W_{T+1} as the expected weight of a random restart version which matches Algorithm 3 till time t. □

Now to prove Theorem 7, we show that the total weight is competitive with running exponential updates on all partitions (in particular the optimal partition) of $[T]$ into m subsets with s switches, intuitively the property of restarting exploration from all past points crucially allows us to "jump" across intervals where a given expert was inactive (or bad).

Proof sketch of Theorem 7. We provide an upper and lower bound to $\frac{W_{T+1}}{W_1}$ similar to Theorem 6. Using Lemma 12 we can show that inequality 2 holds here as well. By Corollary 14 and Lemma 23 (which relates $\pi_t(.)$ to past weights, proved in Appendix C), and β-dispersion we can show a better lower bound Putting together the lower and upper bounds, rearranging and optimizing for γ, α, λ concludes the proof. □

5 Efficient implementation

In this section we show that the Fixed Share Exponential Forecaster algorithm (Algorithm 2) can be implemented efficiently when u_t's are piecewise concave (dimishing returns). In particular we overcome the need to explicitly compute and update $w_t(\rho)$ (there are uncountably infinite ρ in $\mathcal{C}$) by showing that we can sample the points according to $p_t(\rho)$ directly.
The high-level strategy is to show (Lemma 16) that $p_t(\rho)$ is a mixture of t distributions which are Exponential Forecaster distributions from [Balcan et al., 2018a] i.e. $\tilde{p}_i(\rho) := \frac{\tilde{w}(\rho;i,t)}{\tilde{W}(i,t)}$ for each $1 \le i \le t$, with proportions $C_{i,t}$. As shown in [Balcan et al., 2018a] these distributions can be approximately sampled from (exactly in the one-dimensional case, $\mathcal{C} \subset \mathbb{R}$), summarized below as Algorithm BDV-18. We need to sample from one of these t distributions with probability $C_{t,i}$ to get the distribution p_t, and we can approximate these coefficients efficently (or compute exactly in one-dimensional case). The rest of the section discusses how to do these approximations efficiently, and with small extra expected regret. Asymptotically we get the same bound as the exact algorithm. (Formal proofs in Appendix E).

Algorithm BDV-18: Simply *integrate* pieces of the exponentiated utility function, pick a piece with probability proportional to its integral, and *sample from* that piece. [Lovász and Vempala, 2006] show how to efficient *sample from* and *integrate* logconcave distributions. See [Balcan et al., 2018a] for more details.

The coefficients have a simple form in terms of normalizing constants W_t's of the rounds so far, so we first express W_{t+1} in terms of W_t's from previous rounds and some $\tilde{W}(i,j)$'s.

Lemma 15. *In Algorithm 2, for $t \ge 1$,*

$$W_{t+1} = (1-\alpha)^{t-1}\tilde{W}(1,t+1) + \frac{\alpha}{\text{VOL}(\mathcal{C})}\sum_{i=2}^{t}\left[(1-\alpha)^{t-i}W_i\tilde{W}(i,t+1)\right]$$

As indicated above, $p_t(\rho)$ is a mixture of t distributions.

Lemma 16. *In Algorithm 2, for $t \ge 1$, $p_t(\rho) = \sum_{i=1}^{t} C_{t,i}\frac{\tilde{w}(\rho;i,t)}{\tilde{W}(i,t)}$. The coefficients $C_{t,i}$ are given by*

$$C_{t,i} = \begin{cases} 1 & i = t = 1 \\ \alpha & i = t > 1 \\ (1-\alpha)\frac{W_{t-1}}{W_t}\frac{\tilde{W}(i,t)}{\tilde{W}(i,t-1)}C_{t-1,i} & i < t \end{cases}$$

and $(C_{t,1}, \ldots, C_{t,t})$ lies on the probability simplex Δ^{t-1}.

The observations above allow us to write the algorithms for efficiently implementing Fixed Share EF, for which we obtain formal guarantees in Theorem 17. We present an approximate algorithm (Algorithm 5) with the same expected regret as in Theorem 6 (and also present an exact algorithm, Algorithm 6 in Appendix E, for $d = 1$). We say Algorithm 5 gives a (η, ζ) estimate of Algorithm 2, i.e. with probability at least $1-\zeta$, its expected payoff is within a factor of e^{η} of that of Algorithm 2.

Algorithm 5 Fixed Share Exponential Forecaster - efficient approximate implementation

Input: approximation parameter $\eta \in (0,1)$, confidence parameter $\zeta \in (0,1)$

1. $W_1 = \text{VOL}(\mathcal{C})$
2. For each $t = 1, 2, \ldots, T$:
   i. Estimate $C_{t,j}$ using Lemma 16 for each $1 \le j \le t$.
   ii. Sample i with probability $C_{t,i}$.
   iii. Sample ρ with probability approximately proportional to $\tilde{w}(\rho;i,t)$ by running Algorithm BDV-18 with approximation-confidence parameters $(\eta/3, \zeta/2)$.
   iv. Estimate W_{t+1} using Lemma 15. Algorithm BDV-18 to get $(\eta/6T, \eta/2T^2)$ estimates for all $\tilde{W}(\tau,\tau')$ and memoize values of $W_i, i \le t$.

Theorem 17. *If utility functions are piecewise concave and L-Lipschitz, we can approximately sample a point ρ with probability $p_{t+1}(\rho)$ in time $\tilde{O}(Kd^4T^4)$ for approximation parameters $\eta = \zeta = 1/\sqrt{T}$ and $\lambda = \sqrt{s(d\ln(RT^{\beta}) + \ln(T/s))/T}/H$ and enjoy the same regret bound as the exact algorithm. (K is number of discontinuities in u_t's).*

Note that in this section we concerned ourselves with developing a *poly*(d,T) algorithm. For special cases of practical interest, like one-dimensional piecewise constant functions, we can implement much faster $O(K \log KT)$ algorithms as noted in Section 7.

6 Lower bounds

We prove our lower bound for $\mathcal{C} = [0,1]$ and $H = 1$. Also we will consider functions which are β-dispersed

and 0-Lipschitz (piecewise constant). For such utility functions $u_1, \ldots, u_T$ we have shown in Section 4 that the s-shifted regret is $O(\sqrt{sT \log T} + sT^{1-\beta})$. Here we will establish a lower bound of $\Omega(\sqrt{sT} + sT^{1-\beta})$.

We show a range of values of s, β where the stated lower bound is achieved. For $s = 1$, this improves over the lower bound construction of [Balcan et al., 2018a] where the lower bound is shown only for $\beta = 1/2$. In particular our results establish an almost tight characterization of static and dynamic regret under dispersion.

Theorem 18. *For each $\beta > \frac{\log 3s}{\log T}$, there exist utility functions $u_1, \ldots, u_T : [0,1] \to [0,1]$ which are β-dispersed, and the s-shifted regret of any online algorithm is $\Omega(\sqrt{sT} + sT^{1-\beta})$.*

Proof. We perform the construction in $\Theta(s)$ phases, each phase accumulating $\Omega(\sqrt{T/s}+T^{1-\beta})$ regret, yielding the desired lower bound.

Let $I_1 = [0,1]$. In the first phase, for the first $\frac{T-3sT^{1-\beta}}{s}$ functions we have a single discontinuity in the interval $\left(\frac{1}{2}\left(1-\frac{1}{3s}\right), \frac{1}{2}\left(1+\frac{1}{3s}\right)\right) \subseteq \left(\frac{1}{3}, \frac{2}{3}\right)$. The functions have payoff 1 before or after (with probability 1/2 each) their discontinuity point, and zero elsewhere. We introduce $3T^{1-\beta}$ functions each for the same discontinuity point, and set the discontinuity points $T^{-\beta}$ apart for β-dispersion. This gives us $\frac{1/3s}{T^{-\beta}} - 1$ potential points inside $[\frac{1}{3}, \frac{2}{3}]$, so we can support $3T^{1-\beta}\left(\frac{1/3s}{T^{-\beta}} - 1\right) = \frac{T}{s} - 3T^{1-\beta}$ such functions ($\frac{T}{s} - 3T^{1-\beta} > 0$ since $\beta > \frac{\log 3s}{\log T}$). By Lemma 31 (Appendix F) we accumulate $\Omega(\sqrt{\frac{T-3sT^{1-\beta}}{s}}) = \Omega(\sqrt{T/s})$ regret for this part of the phase in expectation. Let I_1' be the interval from among $[0, \frac{1}{2}\left(1-\frac{1}{3s}\right)]$ and $[\frac{1}{2}\left(1+\frac{1}{3s}\right), 1]$ with more payoff in the phase so far. The next function has payoff 1 only at first or second half of I_1' (with probability 1/2) and zero everywhere else. Any algorithm accumulates expected regret 1/2 on this round. We repeat this in successively halved intervals. β-dispersion is satisfied since we use only $\Theta(T^{1-\beta})$ functions in the interval I' of size greater than 1/3, and we accumulate an additional $\Omega(T^{1-\beta})$ regret. Notice there is a fixed point used by the optimal adversary for this phase.

Finally we repeat the construction inside the largest interval with no discontinuities at the end of the last phase for the next phase. Note that at the i-th phase the interval size will be $\Theta(\frac{1}{i})$. Indeed at the end of the first round we have 'unused' intervals of size $\frac{1}{2}\left(1-\frac{1}{3s}\right), \frac{1}{4}\left(1-\frac{1}{3s}\right), \frac{1}{8}\left(1-\frac{1}{3s}\right), \ldots$ At the $i = 2^j$-th phase, we'll be repeating inside an interval of size $\frac{1}{2^{j+1}}\left(1-\frac{1}{3s}\right) = \Theta(\frac{1}{i})$. This allows us to run $\Theta(s)$ phases and get the desired lower bound (intervals must be of size at least $\frac{1}{s}$ to support the construction). □

7 Experiments

The simplest demonstration of significance of our algorithm in a changing environment is to consider the 2-shifted regret when a single expert shift occurs. We consider an artifical online optimization problem first, and will then look at applications to online clustering. Let $\mathcal{C} = [0,1]$. Define utility functions

$$u^{(0)}(\rho) = \begin{cases} 1 & \text{if } \rho < \frac{1}{2} \\ 0 & \text{if } \rho \geq \frac{1}{2} \end{cases} \quad \text{and} \quad u^{(1)}(\rho) = \begin{cases} 0 & \text{if } \rho < \frac{1}{2} \\ 1 & \text{if } \rho \geq \frac{1}{2} \end{cases}$$

Now consider the instance where $u^{(0)}(\rho)$ is presented for the first $T/2$ rounds and $u^{(1)}(\rho)$ is presented for the remaining rounds. We observe constant average regret for the Exponential Forecaster algorithm, while Fixed Share regret decays as $O(1/\sqrt{T})$ (Figure 2). While the example is simple and artificial, it qualitatively captures why Fixed Share dominates Exponential Forescaster here — because the best expert changes and the old expert is no longer competitive. (cf. Appendix B)

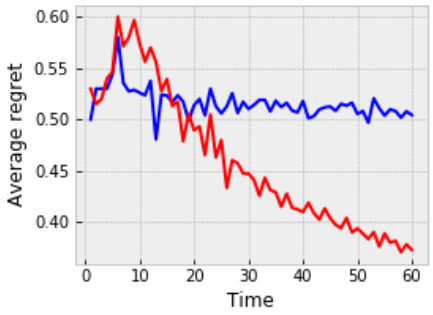
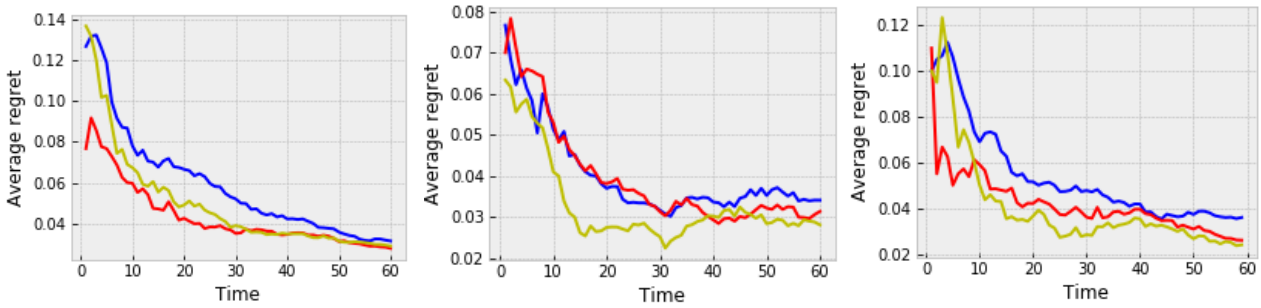


Figure 2: Average 2-shifted regret vs game duration T for a game with single expert shift. Color scheme: **Exponential Forecaster**, **Fixed Share EF**

[Arthur and Vassilvitskii, 2007] proposed k-means++, a celebrated algorithm which shows the importance of initial seed centers in clustering using the k-means algorithm (also called Llyod's method). [Balcan et al., 2018b] generalize it to $(\bar{\alpha}, 2)$-Lloyds++-clustering, which interpolates between random initial seeds (vanilla k-means, $\bar{\alpha} = 0$), k-means++ ($\bar{\alpha} = 2$) and farthest-first traveral ($\bar{\alpha} = \infty$) [Gonzalez, 1985, Dasgupta and Long, 2005] using a single parameter $\bar{\alpha}$. The clustering objective (we use the Hamming distance to the optimal clustering, i.e. the fraction of points assigned to different clusters by the algorithm and the target clustering) is a piecewise constant function of $\bar{\alpha}$, and the best clustering may be obtained for a value of $\bar{\alpha}$ specific to a given problem domain. In an online problem, where clustering instances arrive in a sequential fashion, determining good values of $\bar{\alpha}$ becomes an online optimization problem on piecewise Lipshitz functions.



(a) MNIST (b) Omniglot_small_1 (c) Omniglot (full)

Figure 1: Average 2-shifted regret vs game duration T for online clustering against 2-shifted distributions. Color scheme: **Exponential Forecaster**, **Fixed Share EF**, **Generalized Share EF**

We perform our evaluation on four benchmark datasets to cover a range of examples-set sizes, N and number of clusters, k: *MNIST*, 28×28 binary images of handwritten digits with 60,000 training examples for 10 classes [Deng, 2012]; *Omniglot*, 105×105 binary images of handwritten characters across 30 alphabets with 19,280 examples [Lake et al., 2015]; *Omniglot_small_1*, a "minimal" Omniglot split with only 5 alphabets and 2720 examples.

We consider a sequence of clustering instances drawn from the four datasets and compare our algorithms Fixed Share EF (Algorithm 2) and Generalized Share EF (Algorithm 3) with the Exponential Forecaster algorithm of [Balcan et al., 2018a]. At each time $t \leq T \leq 60$ we sample a subset of the dataset of size 100. For each T, we take uniformly random points from half the classes (even class labels) at times $t = 1, \ldots, T/2$ and from the remaining classes (odd class labels) at $T/2 < t \leq T$. We determine the hamming cost of $(\bar{\alpha}, 2)$-Lloyds++-clustering for $\alpha \in \mathcal{C} = [0, 10]$ which is used as the piecewise constant loss function (or payoff is the fraction of points assigned correctly) for the online optimization game. Notice the Lipschitz constant $L = 0$ since we have piecewise constant utility, and utility function values lie in $[0, 1]$. We set exploration parameter $\alpha = 1/T$ and decay parameter $\gamma = 1/T$ in our algorithms. We plot average 2-shifted regret until time T (i.e. R_T/T) and take average over 20 runs to get smooth curves. (Figure 1). Unlike Figure 2, the optimal clustering parameters before the shift might be relatively competitive to new optimal parameters. So the Exponential Forecaster performance is not terrible, although our algorithms still outperform it noticeably.

We observe that our algorithms have significantly lower regrets (about 15-40% relative for the datasets considered, for $T \geq 40$) compared to the Exponential Forecaster algorithm across all datasets. We also note that the exact advantage of adding exploration to exponential updates varies with datasets and problem instances. In Appendix G we have compiled further experiments that reaffirm the strengths of our approach against different *changing environments* and also compare against the static setting.

Remark. *For the applications considered above, the utility functions are piecewise constant with $d = 1$. For these it is possible to simply maintain the weight on each piece of $\Sigma_t u_t(\rho)$ in $O(K \log Kt)$ time for round t where each $u_t(\cdot)$ has $O(K)$ pieces by using a simple interval tree data structure [Cohen-Addad and Kanade, 2017]. The tree lazily maintains weight for each of $O(Kt)$ pieces, takes $O(\log Kt)$ time each for lazy insertion of $O(K)$ new pieces and allows drawing with probability proportional to weight in $O(\log Kt)$ time. Similarly $O(K \log Kt)$ updates are possible for Algorithm 3. Section 5 addresses the harder problem of polynomial time implementation of Algorithm 2 for arbitrary d.*

8 Discussion and open problems

We presented approaches which trade off exploitation with exploration for the online optimization problem to obtain low shifting regret for the case of general non-convex functions with sharp but dispersed discontinuities. Optimizing for the stronger baseline of shifting regret leads to empirically better payout, as we have shown via experiments bearing applications to algorithm configuration. Our focus here is on the full-information setting which corresponds to the entire utility function being revealed at each iteration, and we present almost tight theoretical results for it. Other relevant settings include bandit and semi-bandit feedback where the function value is revealed for only the selected point or a subset of the space containing the point. It would be interesting to obtain low shifting regret in these settings [Auer et al., 2002].

9 Acknowledgements

We thank Ellen Vitercik for helpful feedback. This work was supported in part by NSF grants CCF-1535967, IIS-1618714, IIS-1901403, CCF-1910321, SES-1919453,an Amazon Research Award, a Microsoft Research Faculty Fellowship, a Bloomberg Data Science research grant, and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program. Views expressed in this work do not necessarily reflect those of any funding agency.

References

[Adamskiy et al., 2012] Adamskiy, D., Koolen, W. M., Chernov, A., and Vovk, V. (2012). A closer look at adaptive regret. In *International Conference on Algorithmic Learning Theory*, pages 290–304. Springer.

[Arthur and Vassilvitskii, 2007] Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

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

[Awerbuch and Kleinberg, 2008] Awerbuch, B. and Kleinberg, R. (2008). Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114.

[Balcan et al., 2019] Balcan, M.-F., Dick, T., and Pegden, W. (2019). Semi-bandit Optimization in the Dispersed Setting. *arXiv e-prints*, page arXiv:1904.09014.

[Balcan et al., 2018a] Balcan, M.-F., Dick, T., and Vitercik, E. (2018a). Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614. IEEE.

[Balcan et al., 2017] Balcan, M.-F., Nagarajan, V., Vitercik, E., and White, C. (2017). Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory*, pages 213–274.

[Balcan et al., 2018b] Balcan, M.-F. F., Dick, T., and White, C. (2018b). Data-driven clustering via parameterized lloyd's families. In *Advances in Neural Information Processing Systems*, pages 10641–10651.

[Bassily et al., 2014] Bassily, R., Smith, A., and Thakurta, A. (2014). Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE.

[Ben-David et al., 2009] Ben-David, S., Pál, D., and Shalev-Shwartz, S. (2009). Agnostic online learning. In *COLT*.

[Bousquet and Warmuth, 2002] Bousquet, O. and Warmuth, M. K. (2002). Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396.

[Cesa-Bianchi et al., 2012] Cesa-Bianchi, N., Gaillard, P., Lugosi, G., and Stoltz, G. (2012). Mirror descent meets fixed share (and feels no regret). In *Advances in Neural Information Processing Systems*, pages 980–988.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

[Cohen-Addad and Kanade, 2017] Cohen-Addad, V. and Kanade, V. (2017). Online optimization of smoothed piecewise constant functions. In *Artificial Intelligence and Statistics*, pages 412–420.

[Combes et al., 2015] Combes, R., Magureanu, S., Proutiere, A., and Laroche, C. (2015). Learning to rank: Regret lower bounds and efficient algorithms. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):231–244.

[Cormack et al., 2008] Cormack, G. V. et al. (2008). Email spam filtering: A systematic review. *Foundations and Trends® in Information Retrieval*, 1(4):335–455.

[Daniely et al., 2015] Daniely, A., Gonen, A., and Shalev-Shwartz, S. (2015). Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411.

[Dasgupta and Long, 2005] Dasgupta, S. and Long, P. M. (2005). Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569.

[Deng, 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[Ertekin et al., 2010] Ertekin, S., Bottou, L., and Giles, C. L. (2010). Nonconvex online support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):368–381.

[Gonzalez, 1985] Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

[Hazan et al., 2016] Hazan, E. et al. (2016). Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325.

[Hazan and Seshadhri, 2007] Hazan, E. and Seshadhri, C. (2007). Adaptive algorithms for online decision problems. In *Electronic colloquium on computational complexity (ECCC)*, volume 14.

[Herbster and Warmuth, 1998] Herbster, M. and Warmuth, M. K. (1998). Tracking the best expert. *Machine learning*, 32(2):151–178.

[Jadbabaie et al., 2015] Jadbabaie, A., Rakhlin, A., Shahrampour, S., and Sridharan, K. (2015). Online optimization: Competing with dynamic comparators. In *Artificial Intelligence and Statistics*, pages 398–406.

[Jun et al., 2017] Jun, K.-S., Orabona, F., Wright, S., and Willett, R. (2017). Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951.

[Kivinen and Warmuth, 1997] Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *information and computation*, 132(1):1–63.

[Lake et al., 2015] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

[Liberty et al., 2016] Liberty, E., Sriharsha, R., and Sviridenko, M. (2016). An algorithm for online k-means clustering. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 81–89. SIAM.

[Littlestone and Warmuth, 1994] Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and computation*, 108(2):212–261.

[Lovász and Vempala, 2006] Lovász, L. and Vempala, S. (2006). Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 57–68. IEEE.

[Maillard and Munos, 2010] Maillard, O.-A. and Munos, R. (2010). Online learning in adversarial lipschitz environments. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 305–320. Springer.

[Merton, 1976] Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144.

[Rakhlin et al., 2007] Rakhlin, A., Abernethy, J., and Bartlett, P. L. (2007). Online discovery of similarity mappings. In *Proceedings of the 24th international conference on Machine learning*, pages 767–774. ACM.

[Rakhlin et al., 2011] Rakhlin, A., Sridharan, K., and Tewari, A. (2011). Online learning: Stochastic, constrained, and smoothed adversaries. In *Advances in neural information processing systems*, pages 1764–1772.

[Sculley and Wachman, 2007] Sculley, D. and Wachman, G. M. (2007). Relaxed online svms for spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422. ACM.

[Talebi et al., 2018] Talebi, M. S., Zou, Z., Combes, R., Proutiere, A., and Johansson, M. (2018). Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 63(4):915–930.

[Wauthier et al., 2013] Wauthier, F., Jordan, M., and Jojic, N. (2013). Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning*, pages 109–117.

[Yang et al., 2018] Yang, L., Deng, L., Hajiesmaili, M. H., Tan, C., and Wong, W. S. (2018). An optimal algorithm for online non-convex learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):25.

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