Introduction to Online Convex Optimization and Online Learning

Sabyasachi Chatterjee

December 28, 2022

Contents

Abstract

This document contains lecture notes for a course titled 'Introduction to Online Learning" which I taught in Fall 2022. These notes borrow material from several sources. The references for these notes are Hazan et al. [14], Orabona [24] Shalev-Shwartz et al. [30], Bubeck [5], Bubeck et al. [6], Bubeck et al. [7], lecture notes in http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15850-f20/www/notes/cmu850-f20.pdf and lecture notes in https://haipeng-luo.net/courses/CSCI659/2022_fall/index.html. Needless to say, there are several typos throughout these notes and the literature references are not adequate. The focus was to explain some basic mathematical results in the field and give a complete and simplified exposition of the proofs.

1 Introduction: What is Online Learning?

Online Learning is a different paradigm of learning compared to Statistical Learning Theory.

1.1 Statistical Learning Theory

One of the standard and thoroughly studied models for learning is the framework of statistical learning theory. We start by briefly reviewing this model.

The basic protocol of statistical learning is the following:

1. Observe training data $Z_1, \ldots, Z_n$ which is assumed to be an i.i.d. sequence from an unknown probability distribution P.
2. Make decision (or choose action) $a(Z_1, \ldots, Z_n) \in \mathcal{A}$ where $\mathcal{A}$ is a given set of possible actions.
3. Suffer an (average) loss $\mathbb{E}_{Z\sim P} l\big(a(Z_1, \ldots, Z_n); Z\big)$ where $l : \mathcal{A} \times \mathcal{Z} \to \mathbb{R}_+$ is a given loss function and Z is test data.

Objective: Minimize (and control) the excess risk:

$$r_n = \mathbb{E}_{Z\sim P} l\big(a(Z_1; \ldots, Z_n); Z\big) - \inf_{a\in\mathcal{A}} \mathbb{E}_{Z\sim P} l\big(a; Z\big)$$

The excess risk represents how much extra (average) loss one suffers compared to the optimal decision.

Controlling the excess risk means finding an upper bound on r_n which holds either in expectation or better, with high probability (with respect to the sequence $Z_1, \ldots, Z_n$). Usually the upper bound is expressed in terms of some complexity measure of $\mathcal{A}$. Moreover if the upper bound depends on P one says that it is a distribution-dependent bound, while if it is independent from P it is distribution-free bound. This formulation is very general and encompasses many standard problems such as regression/classification, density estimation etc.

1.2 Online Learning

What if it is not appropriate to assume the data is i.i.d? Online Learning offers a different paradigm for learning in the sense that it makes no (or very little) assumptions on the data sequence. Instead we think of a learning/prediction problem as a repeated game. At each round t, we make a prediction and then nature or an adversary generates the data and then we measure how good our prediction is via a loss function.

At round t, the data Z_t arrives. We can then take action $a_t \in \mathcal{A}$. What could be a reasonable notion of learning here? There is no probabilistic assumption on the data sequence $Z_1, \ldots, Z_T$. We could compare the total sequential risk to the sequential risk we would have achieved if we had played the best action in hindsight, i.e, if the following term

$$\sum_{t=1}^{T} l(a_t, Z_t) - \inf_{a \in \mathcal{A}} \sum_{t=1}^{T} l(a, Z_t)$$

grows slower than $O(T)$ for all possible data sequences then we can say that we are learning.

Let us write down the online optimization (OO) protocol more precisely. Suppose the game goes on for T rounds. The learner has a set of actions to choose from a set $\mathcal{A}$ which we can call as the action space. We can think of $\mathcal{A} \subset \mathbb{R}^d$. For any round $t \in [T]$,

1. The learner plays $x_t \in \mathcal{A}$.
2. Nature or Adversary reveals a loss function $l_t : \mathcal{A} \to \mathbb{R}_+$.
3. Incur loss $l_t(x_t)$.

While this protocol may seem a bit abstract at a first glance, this OO framework can capture a wide variety of learning problems as we will see in a bit. Let us formalize the notion of an algorithm.

Definition 1.1. *An online learning algorithm $A = (A_1, \ldots, A_t)$ is a sequence of mappings $x_t = A_{t-1}(l_1, \ldots, l_{t-1})$ for $t = 1, \ldots, T$.*

Now, the question arises as to how do we measure the goodness of an online algorithm? For this, we can consider a notion of regret which is similar in flavor to the excess risk in statistical learning theory.

Definition 1.2. *For any fixed $u \in \mathcal{A}$, we can define*

$$Regret_{\mathcal{A}}(u) = \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(u).$$

Regret measures the excess loss incurred by $\mathcal{A}$ as compared to playing a point $u \in \mathcal{A}$ every round. An online learning algorithm is said to have the *no regret* property w.r.t $\mathcal{A}$ if

$$\frac{1}{T} \sup_{u \in \mathcal{A}} Regret_{\mathcal{A}}(u) = \frac{1}{T} \sum_{t=1}^{T} l_t(x_t) - \inf_{u \in \mathcal{A}} \sum_{t=1}^{T} l_t(u) \to 0$$

as $T \to \infty$.

1.3 Comparisons

As one can see, distributional assumptions are built in the definition of statistical learning. On the other hand, online learning does not necessarily assume that data is from some fixed distribution, which makes it much more suitable for dealing with time-varying environments. In fact, even if the data is entirely adversarial, which is indeed the case for applications such as spam detection, meaningful and strong guarantees can still be derived for online learning as we will see soon. Another key advantage is that online learning algorithms are usually more memory-efficient, in the sense that they usually do not need to store data from the past. That is, at each round, the new data is used to update the current states of the algorithm and then discarded. On the other hand, most statistical learning algorithms require storing the training set and touching each example multiple times. Moreover, it can in fact be shown that online learning is strictly harder than statistical learning in the sense that a full information online learning algorithm can be used to solve statistical learning. We might come back to this later in the course.

1.4 Convexity

Within the Online Optimization (OO) framework, if we restrict the action space to be convex and the loss functions to be convex, then we obtain the Online Convex Optimization framework. This framework was introduced by Zinkevich [34].

1. The learner plays $x_t \in \mathcal{A}$ where $K \subset \mathbb{R}^d$ is a convex set of all possible actions.
2. Nature or Adversary reveals a convex loss function $l_t : \mathcal{A} \to \mathbb{R}_+$.
3. Incur loss $l_t(x_t)$.

We can again consider regret of an online algorithm. **The surprising fact is that no regret algorithms exist within the OCO framework for essentially any bounded set** K.

1.5 Things to Come

In this course we (among other things)

1. will study fundamental Online Learning Algorithms such as Online Gradient Descent (OGD), Online Mirror Descent (OMD), Regularized Follow the Leader (RFTL) and analyze their regret.
2. will mostly restrict ourselves to the OCO framework althougn non convex loss functions are also of great interest.
3. see connections to other closely related fields such as Game Theory, Statistical Learning Theory.

4. Will study several variations of the OO framework including limited feedback versions as in Multi Armed Bandits and other adaptive notions of regret.

1.6 Example Problems

A variety of problem settings fall under the OCO framework. Some examples are as follows.

1. **Online Classification**. At each time step the learner plays a classifier $a_t \in \mathcal{A}$. Then nature or an adversary plays (x_t, y_t) and the learner incurs a loss $l_t(a_t, (x_t, y_t))$. For example, the class of classifier functions could be based on hyperplanes, that is of the form $\mathbb{I}(\beta^t x \geq 0)$. In this case, the action space $\mathcal{A}$ could be the Euclidean ball of radius R where R is a tuning parameter to be set by the learner. The natural loss function here is $l_t(\beta_t, (x_t, y_t)) = \mathbb{I}(sgn(\beta_t^t x_t) \neq y)$ which is non convex. Typically, a convex loss function such as the hinge loss is used for computational considerations.

2. **Online Linear Regression**. Similarly as above, if y is real valued, the natural loss function here is the square loss. That is, $l_t(\beta_t, (x_t, y_t)) = (\beta_t^t x_t - y_t)^2$. A slightly modified protocol is also sometimes appropriate here where the learner first observes x_t and then makes a prediction. It is then that the y_t is revealed. The standard online learning algorithm in this setting is what is called the Vovk Azoury Warmuth forecaster.

3. **Prediction From Experts Advice**

   Perhaps the most well known problem in prediction theory is the experts problem. Suppose we want to predict a binary outcome A or B; say whether it will rain tomorrow or not. There are n experts who provide their predictions. Based on these predictions, the learner has to make his/her own prediction. Then the true outcome is revealed and a loss of either 0 or 1 is incurred. This scenario is repeated iteratively, and at each iteration, the predictions of the various experts are arbitrary (and possibly even adversarial, trying to mislead the decision maker). The goal of the decision maker is to do as well as the best expert in hindsight.

   As we will see in the next lecture, the OCO setting captures this problem as a special case too. The action space is the set of all probability distributions over n elements (experts); that is, the n-dimensional probability simplex. The loss function turns out to be linear functions on the simplex. The fundamental importance of the experts problem in machine learning warrants special attention, and we shall return to it and analyze it in detail from the next lecture.

4. **Portfolio Selection** In this section we consider a portfolio selection model that does not make any statistical assumptions about the stock market (as opposed to the standard geometric Brownian motion model for stock prices), and is called the universal portfolio selection" model. At each iteration $t \in [T]$, the decision maker chooses a distribution of her wealth over n assets x_t. The adversary independently chooses market returns for the assets, i.e., a vector r_t with strictly positive entries such that each coordinate $r_t(i)$ is the price ratio for the

i'th asset between the iterations t and t + 1. The ratio between the wealth of the investor at iterations $t+1$ and t is $\frac{W_{t+1}}{W_t} = r_t^t x_t$, and hence the gain in this setting is defined to be the logarithm of this change ratio in wealth $log(r_t^t x_t)$. Notice that since x_t is the distribution of the investor's wealth, even if $x_{t+1} = x_t$, the investor may still need to trade to adjust for price changes. So in this case, the action space $\mathcal{A}$ is the probability simplex and the loss functions are $l_t(x) = -\log(r_t^t x)$ which are convex.

Here regret minimization corresponds to maximizing

$$\log W_{T+1} - \log W_1$$

as compared to the same quantity attained by the best benchmark from a pool of investing strategies. A universal portfolio selection algorithm is defined to be one that, in this setting, attains regret converging to zero. Historically, the first universal portfolio algorithm was given by Tom Cover which was computationally efficient. We will come back to this setting a little later in the course and see an efficient universal portfolio selection algorithm.

5. **Online Matrix Completion and Recommendation Systems**

   Here at every time step we predict a matrix at of size $m \times n$, the adversary reveals the i, j th entry of an unknown true matrix M. Natural action space is the set of low rank matrices, say rank bounded by some $k \geq 1$. The loss can be the square loss. A typical convex relaxation of this problem is to consider matrices with bounded trace norm rather than bounded rank.

6. **Product Recommendation (Multi-Armed Bandits)**

   At each time $t = 1, 2, \ldots,$

   - randomly recommend one of the K products a to a customer visiting the website;
   - observe the loss of this product $l_t(a)$ (e.g. 0 if clicked, 1 otherwise), but not the losses for the other products.

   Here, I do not see the losses for all the actions and hence this is an example that falls under the OCO framework but with limited feedback.

2 Prediction from Experts Advice

We consider a basic problem in online learning: how to choose dynamically among a set of experts in a way that competes with the best expert in hindsight. This abstract problem and the techniques behind the solution are important parts of the algorithm designer's toolkit.

2.1 Mistake Bound Model

Consider the following setup: N experts make predictions about a binary outcome such as $\{\text{rain}, \text{no rain}\}$ in T rounds:

1. At the beginning of round $t \in [T]$, each expert makes a prediction which we can collect as a vector $\mathcal{E}^t \in \{0,1\}^N$.

2. Learner makes prediction a_t and simultaneously, the actual outcome o_t is revealed.

The goal of the leaner is to have an algorithm that can compete with the best expert in hindsight.

Suppose there is an expert that never makes mistakes, then what is the natural algorithm in this case?

Fact: In this case there exists an algorithm which makes at most $\lceil \log_2 N \rceil$ mistakes. The algorithm is to predict by majority. The first time the learner is wrong, more than half of the experts are wrong, and they can be thrown out. Now the number of valid experts are reduced. The second time the learner is wrong, more than half of the currently valid experts are wrong, and they can be thrown out too. By repeating this procedure, the perfect expert can be found in at most $\lceil \log_2 N \rceil$ steps.

This reasoning can be extended.

Exercise 2.1. *In the case we do not have the perfect expert, show that there exists an algorithm based on the above reasoning which makes $O(m^* \ln N)$ mistakes, where m^* is the number of mistakes made by the best expert.*

We will now see more streamlined versions of majority algorithms.

2.2 Weighted Majority algorithm

The Weighted Majority (WM) algorithm of Littlestone and Warmuth [21] can be described as follows:

1. Assign a weight $w_i^{(t)}$, $i \in [N]$, to each expert at each round.

2. Come up with a_t by majority.

3. Upon observing o_t, update the weights:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} & \text{ith expert is correct,} \\ w_i^{(t)}/2 & \text{ith expert is wrong.} \end{cases}$$

Theorem 2.2. *The number of mistakes made by the WM algorithm is at most* $2.41(\min_{i\in[N]} M_i + \log_2 N)$ *where* M_i *is the number of mistakes made by expert* i*. Moreover, if the weights are updated by multiplying with* $1-\varepsilon$*,* $0 < \varepsilon < 1/2$*, instead of* $1/2$ *when an expert is wrong, then the bound on the number of mistakes becomes*

$$2(1+\varepsilon)M_i + O\left(\frac{\ln N}{\varepsilon}\right).$$

Remark 2.1. *No deterministic algorithm can avoid the factor* 2 *compared to the best expert. The reason is the following: consider the case where there are only two experts, one always predicts* 0 *and the other always predicts* 1*. Fix an algorithm, then there exists a "bad" sequence of outcomes such that the algorithm is always wrong, i.e. it makes* T *mistakes. One of the experts in this circumstance only make* $T/2$ *mistakes. This suggests the factor of* 2 *is inherent. Nonetheless, if we make prediction by flipping a coin, then the rate of mistakes should be close to* $T/2$*. So randomization should help getting rid of the factor of* 2*.*

2.3 Randomized Weighted Majority algorithm

Let us consider the Randomized Weighted Majority (RWM) algorithm:

1. Assign a weight $w_i^{(t)}$, $i \in [N]$, to each expert at each round. Pick expert i with probability $\propto w_i^{(t)}$.

2. Let i_t denote the expert chosen at round t. Predict $a_t = \mathcal{E}(i_t)$.

3. Upon observing o_t, update the weights:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} & \text{if ith expert is correct,} \\ w_i^{(t)}(1-\varepsilon) & \text{if ith expert is wrong.} \end{cases}$$

Theorem 2.3. *Let* M *denote the number of mistakes made by RWM with parameter* $\epsilon \in (0, 1/2)$*. Then we have the following expected mistake bound*

$$\mathbb{E}[M] \le (1+\varepsilon)M_i + \frac{\ln N}{\varepsilon}, \qquad \forall i \in [N].$$

Proof. (This is a potential based proof. The idea is to track the total sum of weights.) Define

$$S_1 = N \quad \text{and} \quad S_t = \sum_{i=1}^{N} w_i^{(t)}.$$

Let $m_i^{(t)}$ be the indicator of the i-th expert making a mistake in round t, then

$$S_{t+1} = \sum_{i=1}^{N} w_i^{(t+1)} = \sum_{i=1}^{N} w_i^{(t)}(1 - \varepsilon m_i^{(t)}).$$

This implies

$$S_{t+1} = S_t - \varepsilon S_t \sum_{i=1}^{N} \frac{w_i^{(t)} m_i^{(t)}}{S_t} = S_t \cdot [1 - \varepsilon P(\tilde{m}_t = 1)],$$

where $\tilde{m}_t$ is the indicator of the algorithm makes a mistake on iteration t. Hence,

$$S_{t+1} = S_1 \prod_{t=1}^{T} [1 - \varepsilon P(\tilde{m}_t = 1)] \leq N e^{-\varepsilon \sum_{t=1}^{T} P(\tilde{m}_t = 1)}.$$

On the other hand, denote by $M_t(i)$ the number of mistakes made by expert i until time t, we have

$$(1 - \varepsilon)^{\sum_{t=1}^{T} M_t(i)} = w_i^{(T+1)} \leq S_{T+1}.$$

Let us denote the total number of mistakes made by expert i as $M_i = \sum_{t=1}^{T} M_t(i)$.

As a result,

$$(1 - \varepsilon)^{M_i} \leq N e^{-\varepsilon \mathbb{E}[M]}.$$

Take logarithm on both sides, we see that

$$M_i \ln(1 - \varepsilon) \leq \ln N - \varepsilon \mathbb{E}[M],$$

which is the same as

$$\mathbb{E}[M] \leq \frac{\ln N - M_i \ln(1 - \varepsilon)}{\varepsilon}.$$

The desired bound is then obtained by applying the fact that

$$-\ln(1 - \varepsilon) \leq \varepsilon + \varepsilon^2$$

for $\varepsilon \in (0, 1/2)$. □

Remark 2.2. *If nothing is known about the M_i's, we can bound M_i by T and optimize*

$$M_i + \varepsilon T + \frac{\ln N}{\varepsilon}$$

with respect to ε, and obtain $\varepsilon^* = \sqrt{\ln N / T}$. Hence, the optimal bound is

$$\min_{i \in [N]} M_i + 2\sqrt{T \ln N},$$

where the second term is exactly the regret comparing to the best expert. In case, the best expert makes $o(\sqrt{T})$ mistakes, then one can set ϵ to be a constant so that we will also incur the same $o(\sqrt{T})$ mistakes plus a log N factor.

2.4 Hedge algorithm

Now we turn to a slightly broader setting, i.e. the so-called *dot-product game*:

1. Learner plays a vector of probabilities $p_t = (p_1^t, \ldots, p_N^t)$.
2. The adversary produces the loss vector $\ell^t = (\ell_1^t, \ldots, \ell_N^t)$.
3. Loss incurred is $\ell_t \cdot p_t$.

This is a generalization of the binary outcomes prediction game as the loss vector can now consist of real numbers instead of $0, 1$. Viewing this dot product game, we can see that fits perfectly into the OCO setting. Here, the action set $\mathcal{A}$ is the probability simplex in $\mathbb{R}^n$, and the loss $p \mapsto \ell_t \cdot p$ is a linear function on the action space.

The Hedge algorithm for this dot product game is

1. Start with weights $w_1(i) = 1 \ \forall i$.
2. In round $1 \leq t \leq T$, pick a distribution $p_t = \frac{w_t(i)}{\sum_i w_t(i)}$.
3. Observe the loss l_t.
4. After round t, update the weights $w_{t+1}(i) = w_t(i)e^{-\epsilon l_t(i)}$.

Theorem 2.4. *Suppose the loss vectors $\{l^t : 1 \leq t \leq n\}$ consist of entries such that $\epsilon l^t(i) \geq -1$ for all i, t where $\epsilon > 0$ is the parameter of the hedge algorithm. The regret of the resulting Hedge algorithm can be bounded as follows:*

$$\begin{aligned} Regret &= \sum_{t=1}^{T} l_t.p_t - \min_{p \in \Delta_n} \sum_{t=1}^{T} l_t.p \\ &\leq \frac{\log n}{\epsilon} + \epsilon \sum_{t=1}^{T} l_t^2 \cdot p_t. \end{aligned}$$

As a consequence, if the loss entries are bounded between -1 and 1; then we can write

$$Regret \leq \frac{\log n}{\epsilon} + \epsilon T \leq 2\sqrt{T \log n}$$

where in the last inequality we have optimized over ϵ.

Proof. We'll again use a potential based argument and keep track of the sum of weights as we did in the analysis for the randomized weighted majority algorithm. Define

$$S_t = \sum_i w_t(i).$$

$$
\begin{aligned}
S_{t+1} &= \sum_i w_{t+1}(i) \\
&= \sum_i w_t(i) e^{-\epsilon l_t(i)} \text{ (Used the weight update for Hedge algorithm)} \\
&\leq \sum_i w_t(i)[1 - \epsilon l_t(i) + \epsilon^2 l_t^2(i)] \text{ (Used } e^{-x} \leq 1 - x + x^2 \text{ for } x \geq -1) \\
&= S_t + \epsilon^2 S_t \sum_i p_t(i) l_t^2(i) - \epsilon S_t \sum_i p_t(i) l_t(i) \text{ (Multiplied and divided by } S_t) \\
&\leq S_t e^{\epsilon^2 l_t^2 \cdot p_t - \epsilon l_t \cdot p_t} \text{ (Used } 1 + x \leq e^x).
\end{aligned}
$$

Therefore, using this inequality recursively we can obtain

$$
\begin{aligned}
S_{T+1} &\leq S_1 e^{\epsilon^2 \sum_{t=1}^T l_t^2 \cdot p_t - \epsilon \sum_{t=1}^T l_t \cdot p_t} \\
&= n e^{\epsilon^2 \sum_{t=1}^T l_t^2 \cdot p_t - \epsilon \sum_{t=1}^T l_t \cdot p_t} \qquad (1)
\end{aligned}
$$

Now, we'll find a lower bound on S_{T+1}

$$
\begin{aligned}
S_{T+1} &\geq w_{T+1}(i) \ \ \forall i \\
&= e^{-\epsilon \sum_{t=1}^T l_t(i)} \text{ (Used the weight update for Hedge algorithm)} \qquad (2)
\end{aligned}
$$

Thus, combine Eq. 2 and Eq. 1 to get

$$
\begin{aligned}
e^{-\epsilon \sum_{t=1}^T l_t(i)} &\leq n e^{\epsilon^2 \sum_{t=1}^T l_t^2 \cdot p_t - \epsilon \sum_{t=1}^T l_t \cdot p_t} \\
\sum_{t=1}^T l_t \cdot p_t &\leq \sum_{t=1}^T l_t(i) + \frac{\log n}{\epsilon} + \epsilon \sum_{t=1}^T l_t^2 \cdot p_t \ \forall i \text{ (Took log on both sides)} \\
&\leq \min_i \sum_{t=1}^T l_t(i) + \frac{\log n}{\epsilon} + \epsilon \sum_{t=1}^T l_t^2 \cdot p_t.
\end{aligned}
$$

The proof is nearly done after observing that

$$
\min_i \sum_{t=1}^T l_t(i) = \min_{p \in \Delta_n} \sum_{t=1}^T l_t . p
$$

This is because of the fact that minimum of a linear function inside a convex set is attained at one of the extreme points. We can now optimize w.r.t ϵ and set it to $\sqrt{\frac{\log n}{T}}$ to get the final bound.

□

Remark 2.3. *We will see later in the course that the Hedge algorithm can be thought of as a generalized version of online gradient descent.*

Remark 2.4. *The boundedness of the loss function plays an important role in the above proof.*

3 Some Basic Results of Convex Optimization

Before venturing into online convex optimization, we will review some basic results in (offline) convex optimization. We will begin with a few remarks connecting convex optimization to online

convex optimization

1. Offline Convex optimization is a special case of online convex optimization where the action space $\mathcal{A} \subseteq \mathbb{R}^d$ is convex and the loss function $l_t : \mathcal{A} \to \mathbb{R}$ is a constant convex function f.

2. Low regret in this case implies "closeness" of the average point to optimal point as follows

$$\frac{1}{T}\sum_{t=1}^{T} f(a_t) - \min_{a\in\mathcal{A}} f(a) \leq \epsilon$$
$$\implies f\left(\frac{1}{T}\sum_{t=1}^{T} a_t\right) - \min_{a\in\mathcal{A}} f(a) \leq \epsilon \text{ (Used convexity of } f)$$

3.1 Gradient Descent

The gradient descent (GD) is a classical first order optimization algorithm with the update equation given by

$$x_{t+1} \leftarrow x_t - \eta \nabla f(x_t) \tag{3}$$

Given a convex function $f : \mathbb{R}^d \to \mathbb{R}$, let $x^* = \text{argmin}_{x\in\mathbb{R}^d} f(x)$ and let the suboptimality gap at x be defined as $\delta(x) = f(x) - f(x^*)$.

The suboptimality gap lower bounds correlation between negative gradient at a point and the direction to the optimal point as follows

$$\begin{aligned}\delta(x) &:= f(x) - f(x^*) \\ &\leq \nabla f(x) \cdot (x - x^*) \text{ (definition of convex function)} \\ &= -\nabla f(x) \cdot (x^* - x)\end{aligned} \tag{4}$$

This suggests that going in the negative gradient direction may not be a terrible idea.

3.1.1 Gradient Flow

We will first provide an approximate analysis of GD (Eq. 3) by taking infinitesimal step η resulting in gradient flow as

$$\frac{dx(t)}{dt} = -\nabla f(x(t)) \tag{5}$$

Lemma 3.1. *Gradient flow leads to low average suboptimality over the trajectory with*

$$\frac{1}{\tau}\int_0^\tau \delta(x(t))dt \leq \frac{D_0^2}{2\tau}$$

where $D_0^2 = \frac{1}{2}|x(0) - x^*|^2$.

Proof. We will track the time derivative of the squared distance to the optima.

$$\begin{aligned}
\int_0^\tau \frac{d|x(t)-x^*|^2}{dt}dt &= \int_0^\tau (x(t)-x^*)\cdot -\nabla f(x(t))dt \text{ (Used gradient flow equation, Eq. 5)} \\
&\leq \int_0^\tau -\delta(x(t))dt \text{ (Used Eq. 4)} \\
\frac{1}{2}|x(\tau)-x^*|^2 - \frac{1}{2}|x(0)-x^*|^2 &\leq \int_0^\tau -\delta(x(t))dt \\
\int_0^\tau \delta(x(t))dt &\leq \frac{1}{2}|x(0)-x^*|^2 \\
\frac{1}{\tau}\int_0^\tau \delta(x(t))dt &= \frac{D_0^2}{2\tau}
\end{aligned}$$

□

Remark 3.1. *The average point in the gradient descent trajectory has low suboptimality gap as* $\delta(\frac{1}{\tau}\int_0^\tau x(t)dt) \leq \frac{1}{\tau}\int_0^\tau \delta(x(t))dt \leq \frac{D_0^2}{2\tau}$. *Also, the above result gives us a* $O(1/\tau)$ *rate of convergence for gradient flow.*

3.2 Discrete Time Analysis of Gradient Descent

Previously, we analyzed gradient flow to gain intuition for the convergence of gradient descent. In this lecture, we do the discrete time analysis. We start with what we can call as *the basic lemma*.

Lemma 3.2 (Basic Lemma)**.** *Let* $g_1, \ldots, g_T \in \mathbb{R}^d$ *be arbitrary vectors and let us consider the sequence of states*

$$x_{t+1} = x_t - \eta g_t; t \in [T]$$

for some stepsize $\eta > 0$. *Then, for any* $x \in \mathbb{R}^d$,

$$\sum_{t=1}^T \langle x_t - x, g_t\rangle \leq \frac{|x_1-x|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T |g_t|^2.$$

Proof. Let $D_t = |x_t - x|$. Then,

$$\begin{aligned}
D_{t+1}^2 - D_t^2 &= |x_t - \eta g_t - x|^2 - |x_t - x|^2 \\
&= -2\eta\langle x_t - x, g_t\rangle + \eta^2|g_t|^2, \\
D_{T+1}^2 - D_1^2 &= \sum_{t=1}^T (D_{t+1}^2 - D_t^2) \\
&= -2\eta\sum_{t=1}^T \langle x_t - x, g_t\rangle + \eta^2\sum_{t=1}^T |g_t|^2.
\end{aligned}$$

Rearranging the terms gives

$$\sum_{t=1}^T \langle x_t - x, g_t\rangle = \frac{D_1^2}{2\eta} - \frac{D_{T+1}^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T |g_t|^2 \leq \frac{|x_1-x|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T |g_t|^2$$

□

We can now use the above lemma to get error bounds for Gradient Descent. Compared with gradient flow, there will be an additional error term in the discrete case. We formalize this idea in the following theorem.

Theorem 3.3. *Let f be a L lipschitz convex function, i.e, $|\nabla f(x)| \leq L$ for all x. Let $x_{t+1} = x_t - \eta \nabla f(x_t)$ where $\eta = \frac{D_1}{L\sqrt{T}}$. Then*

$$\frac{1}{T}\sum_{t=1}^{T}\{f(x_t) - f(x^*)\} \leq \underbrace{\frac{D_1^2}{2\eta T}}_{\text{gradient flow error}} + \underbrace{\frac{\eta}{2T}\sum_{t=1}^{T}|\nabla f(x_t)|^2}_{\text{discretization error}},$$

where x^* denotes the optimum and $D_1 = |x_1 - x^*|$. Then we have

$$\frac{1}{T}\sum_{t=1}^{T}\{f(x_t) - f(x^*)\} \leq \frac{LD_1}{\sqrt{T}}.$$

Proof. Pick $g_t = \nabla f(x_t)$ and $x = x^*$ in Lemma 3.2. Convexity of f and Lemma 3.2 gives

$$\frac{1}{T}\sum_{t=1}^{T}\{f(x_t) - f(x^*)\} \leq \frac{1}{T}\sum_{t=1}^{T}\nabla f(x_t)(x_t - x^*) \leq \frac{D_1^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}|\nabla f(x_t)|^2 \leq \frac{D_1^2}{2\eta T} + \frac{\eta}{2}L^2.$$

Now set $\eta = \frac{D_1}{L\sqrt{T}}$ to finish the proof. □

Remark 3.2. *The optimal step size depends on the initial distance D_1, the lipschitz constant L and the time horizon T. One can use a time varying step size decaying like $O(1/\sqrt{t})$ to get a similar upper bound. One can also use line search to choose the right step size.*

3.3 Constrained Case

Suppose we want to minimize a convex function inside a convex set $K \subset \mathbb{R}^d$. In this case, we can use the projected gradient descent algorithm

$$x_{t+1} = \text{Proj}_K(x_t - \eta g_t).$$

The same bound as in Theorem 3.3 holds for projected gradient descent as well. We record this fact as a corollary.

Corollary 3.4. *Let f be a L lipschitz convex function, i.e, $|\nabla f(x)| \leq L$ for all $x \in K$ where K is a convex subset of $\mathbb{R}^d$. Let*

$$x_{t+1} = \text{Proj}_K\big(x_t - \eta \nabla f(x_t)\big)$$

where $\eta = \frac{D_1}{L\sqrt{T}}$. Then

$$\frac{1}{T}\sum_{t=1}^{T}\{f(x_t) - f(x^*)\} \leq \underbrace{\frac{D_1^2}{2\eta T}}_{\text{gradient flow error}} + \underbrace{\frac{\eta}{2T}\sum_{t=1}^{T}|\nabla f(x_t)|^2}_{\text{discretization error}},$$

where x^* denotes the optimum and $D_1 = |x_1 - x^*|$. Then we have

$$\frac{1}{T}\sum_{t=1}^{T}\{f(x_t) - f(x^*)\} \leq \frac{LD_1}{\sqrt{T}}.$$

Proof. As a consequence of convexity, $|\mathrm{Proj}_K(x) - z|^2 \leq |x - z|^2$ for all $z \in K$. This is the Pythagorean theorem for convex projections. Theorem 3.3 applies similarly to the projected gradient descent because

$$|\mathrm{Proj}_K(x_t - \eta\nabla f(x_t)) - x^*|^2 \leq |x_t - \eta\nabla f(x_t) - x^*|^2$$

by the fact above. □

Remark 3.3. *1. If f is convex but not differentiable, we can use subgradients instead of gradients and Theorem 3.3 and Corollary 3.4 still continue to hold.*

2. The upper bound $LD_1/\sqrt{T}$ is often called a dimension-free bound as the dimension d of the problem does not enter explicitly into the bound. Moreover, in machine learning, often we truly are in very high dimensional settings but the lipschitz constant L is typically a constant.

3.4 Lower Bound for High Dimensions

To understand how tight the upper bound $LD_1/\sqrt{T}$ is, let us fix T. Let $d = T + 1$ and K be the unit ball in $\mathbb{R}^d$. The following is true.

Theorem 3.5. *Define $F_{L,d}$ to be the space of convex functions defined on K with lipschitz constant L. Then,*

$$\max_{d\geq 1}\left(\inf_{\hat{x}}\sup_{f\in F_L(K)}|f(\hat{x}) - f(x^*)|\right) \geq \frac{L}{\sqrt{T+1}}$$

where the infimum is over any output $\hat{x}$ of an algorithm after T gradient queries.

Proof. For any output $\hat{x}$ of an algorithm after T gradient queries,

$$\inf_{\hat{x}}\sup_{f\in F_{L,d}}|f(\hat{x}) - f(x^*)| \geq \inf_{\tilde{x}}\mathbb{E}_{f\in F_L(K)}|f(\hat{x}) - f(x^*)|$$

where $\mathbb{E}$ is expectation over a suitably chosen distribution on $F_L(K)$. We now define this distribution.

Choose a random orthonormal basis $\{v_1, \ldots, v_{T+1}\} \in \mathbb{R}^{T+1}$. Define the random function

$$f(x) = \max_{j=1,\ldots,T+1} v_j^\intercal x.$$

Note that this random function f defines a distribution on $F_L(K)$.

Now let us rename the orthonormal basis vectors so that after T queries, we do not know v_{T+1} but possibly we know $v_1, \ldots, v_T$. For any output of an algorithm $\hat{x} \in K$ after T queries , by symmetry we must have

$$\mathbb{E}(v_{T+1}^\intercal\hat{x} \mid \text{first } T \text{ queries}) = 0.$$

The logic is that conditionally on the first T queries, $\hat{x}$ is fixed, and the conditional distribution of v_{T+1} has to be symmetric about 0.

Therefore,

$$\mathbb{E}\{f(\hat{x}) \mid \text{first } T \text{ queries}\} \geq \mathbb{E}(v_{T+1}^{\intercal}\hat{x} \mid \text{first } T \text{ queries}) = 0.$$

On the other hand, taking the point $-\sum_{j=1}^{T+1} v_j/\sqrt{T+1}$ gives

$$\min_{x \in K} f(x) \leq -\frac{1}{\sqrt{T+1}}.$$

This implies that

$$\inf_{\tilde{x}} \mathbb{E}_{f \in F_1(K)} |f(\hat{x}) - f(x^*)| \geq \frac{1}{\sqrt{T+1}}.$$

□

Remark 3.4. *The above lower bound only works when d is taken to be larger than T.*

3.5 Center of Gravity Algorithm for Low Dimensions

We saw that the $O(1/\sqrt{T})$ rate is unimprovable in high dimensions. Suppose the dimension d is small. Then, an algorithm called the Center of Gravity algorithm can achieve better convergence rate. This is based on the following geometric inequality.

Lemma 3.6. *(Grunbaum's inequality)*

Let K be a convex body in $\mathbb{R}^d$. For any half-space H containing the center of gravity defined as

$$\mathrm{CG}(K) = \int_K x dx / \mathrm{Vol}(K)$$

(the center of gravity of K),

$$\mathrm{Vol}(K \cap H) \geq \frac{1}{e}\mathrm{Vol}(K).$$

The Center of Gravity algorithm is as follows.

Remark 3.5. *The idea of the algorithm is that for any $x \in H_t$, clearly $f(x) \geq f(x_t)$ and hence the H_t part in K is chopped off. Grunbaum's inequality ensures that in each step we are chopping off a constant fraction of K in volume. Hence, we will be left with an exponentially small sized set after a few iterations. This is the multivariate analogue of the Binary search algorithm that can be used in one dimension.*

Remark 3.6. *Note that this is more of a conceptual algorithm only as computing the Center of Gravity is difficult and costly when d is large. Nevertheless, under the first order oracle model (where you count only the number of first or zeroth order queries) it can theoretically improve the $O(1/\sqrt{T})$ rate.*

1 (Center of Gravity) Set $K_0 = K$. For $t = 0, \ldots, T$ do

1. Compute $x_t = \mathrm{CG}(K_t)$.
2. Define $H_t = \{x : \nabla f(x_t)^T (x - x_t) \geq 0\}$ and $H_t^c = \{x : \nabla f(x_t)^T (x - x_t) \leq 0\}$.
3. Define $K_{t+1} = K_t \cap H_t^c$.

After this, make $T + 1$ zeroth order queries and output

$$\hat{x} = \operatorname*{argmin}_{x \in x_0, \ldots, x_T} f(x).$$

Theorem 3.7. *Let $K \subset \mathbb{R}^d$ be a convex body. Let $f : K \to [-1, 1]$ be a bounded convex function. Fix $1 > \epsilon > 0$. After $T = O(d \log(1/\epsilon))$ steps of the Center of Gravity method, we have*

$$f(x_T) - f(x^*) \leq \epsilon \Leftrightarrow f(x_T) - f(x^*) \leq \exp(-\frac{cT}{d}),$$

where c is some absolute constant.

Proof. Lemma 3.6 gives

$$\mathrm{Vol}(K_T) < (1 - \frac{1}{e})^T \mathrm{Vol}(K) \leq \epsilon^d \mathrm{Vol}(K)$$

where the second inequality follows by definition of T.

Define

$$\Omega = \{(1 - \epsilon)x^* + \epsilon x, x \in K\}.$$

Note that $\mathrm{Vol}(\Omega) = \epsilon^d \mathrm{Vol}(K)$, so we have

$$\mathrm{Vol}(\Omega) > \mathrm{Vol}(K_T).$$

In particular, this implies that there must exists a time $r \in \{1, \ldots, T\}$ such that $K_{r+1} \notin \Omega$. Take any $x \in \Omega$. It is then true that $x \in H_r$ and hence

$$f(x_r) \leq f(x) \leq (1 - \epsilon) f(x^*) + \epsilon \leq f(x^*) + 2\epsilon.$$

Here, the first inequality is because $x \in H_r$, and we have used convexity and boundedness of f in the later inequalities.

Finally, note that by definition of $\hat{x}$,

$$f(\hat{x}) \leq f(x_r) \leq f(x^*) + 2\epsilon$$

which finishes the proof. □

3.6 Beyond $O(1/\sqrt{T})$ Rate

We have seen so far that gradient descent achieves $O(1/\sqrt{T})$ rate when tuned properly. We have also seen that this rate is unimprovable in high dimensions. We now explore more dimension free rates under stronger assumptions on the underlying convex function. It turns out that gradient descent achieves faster rates of convergence under the following assumptions:

1. Strong Convexity (curvature): $O(1/T)$.
2. Smoothness: $O(1/T)$.
3. Both Strong Convexity and Smoothness: $O(e^{-cT})$.

3.6.1 Strong Convexity

Definition 3.8. *A function $f : K \to \mathbb{R}$ is α-strongly convex if $f(x) - (\alpha/2)|x|^2$ is convex on K where K is a convex set.*

Lemma 3.9. *If $f(x)$ is α-strongly convex, then for all $x, y \in K$,*

$$f(y) \geq f(x) + \nabla f(x)(y - x) + \frac{\alpha}{2}|y - x|^2,$$

where K is some convex set.

Proof. This proof is left as an exercise. □

Let us investigate how strong convexity helps in attaining faster rates for gradient flow. Recall that for gradient flow is defined by

$$dx(t)/dt = -\nabla f(x(t)).$$

We will again track the squared distance

$$D^2(t) = |x(t) - x^*|^2/2.$$

By Lemma 3.9, we have

$$\begin{aligned}
\frac{d}{dt}D^2(t) &= -\{x(t) - x^*\}\nabla f(x(t)) \\
&\leq -\{f(x(t)) - f(x^*)\} - \alpha D^2(t) \\
&\leq -\alpha D^2(t), \\
D^2(t) &\leq D^2(0)\exp(-\alpha t).
\end{aligned}$$

The key point is that in the first inequality we have an extra $-\alpha D^2(t)$ term which arises due to strong convexity. So we see that atleast in terms of distance to optima, we have exponential convergence. *Can this argument be converted to show exponential convergence in function value as well?*

We now present the discrete time analysis.

Theorem 3.10. *Set the weight sequence*

$$w_t = \frac{2t}{T(T+1)}.$$

Then we have the following bound on the weighted suboptimality gap

$$\sum_{t=1}^{T} w_t\big(f(x_t) - f(x^*)\big) \leq \frac{2L^2}{\alpha(T+1)}.$$

Proof. For gradient descent with step size η_t, we have

$$\begin{aligned} \|x_{t+1} - x^*\|^2 &= \|x_t - x^*\|^2 - 2\eta_t\langle x_t - x^*, \nabla f(x_t)\rangle + \eta_t^2\|\nabla f(x_t)\|^2 \\ &\leq \|x_t - x^*\|^2 + 2\eta_t\left[f(x^*) - f(x_t) - \frac{\alpha}{2}\|x_t - x^*\|^2\right], \end{aligned} \tag{6}$$

where the second inequality is obtained by using the strong convexity property as in Lemma 3.9. Using the lipschitz property of f and rearranging we obtain

$$f(x_t) - f(x^*) \leq (\frac{1}{2\eta_t} - \frac{\alpha}{2})\|x_t - x^*\|^2 - \frac{1}{2\eta_t}\|x_{t+1} - x^*\|^2) + \frac{\eta_t}{2}L^2.$$

Now set $\eta_t = \frac{2}{\alpha(t+1)}$ and multiply the above display both sides by t to obtain

$$\begin{aligned} t\left(f(x_t) - f(x^*)\right) &\leq \frac{L^2}{\alpha} + \left[\frac{\alpha t(t+1)}{4} - \frac{\alpha t}{2}\right]\|x_t - x^*\|^2 - \frac{\alpha t(t+1)}{4}\|x_{t+1} - x^*\|^2 \\ &= \frac{L^2}{\alpha} + \frac{\alpha}{4}\left[t(t-1)\|x_t - x^*\|^2 - t(t+1)\|x_{t+1} - x^*\|^2\right]. \end{aligned} \tag{7}$$

Summing up across all $t = 1, \ldots, T$ and dividing by $\frac{T(T+1)}{2}$, the last term gives rise to a telescoping sum:

$$\begin{aligned} \frac{\sum_{t=1}^{T} t(f(x_t) - f(x*))}{T(T+1)/2} &\leq \frac{2L^2}{\alpha(T+1)} - \frac{\alpha}{2}\|x_{T+1} - x^*\|^2 \\ &\leq \frac{2L^2}{\alpha(T+1)}. \end{aligned} \tag{8}$$

□

Remark 3.7. *Thus, gradient descent for strongly convex and lipschitz functions converges at the rate of $O(1/\alpha T)$. Note that if the strong convexity parameter α is small (as might be the case in problems such as minimizing hinge loss with L2 regularization), the rate may not be very satisfactory.*

3.7 Smooth Functions

Definition 3.11. *A differentiable function f is L-smooth if*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{9}$$

for all x, y.

The following lemma would be useful in analysing gradient descent for smooth functions.

Lemma 3.12. *If f is L-smooth then*

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}\|y - x\|^2. \tag{10}$$

Proof.

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \langle \nabla f(x + t(y - x)), y - x \rangle dt \\ &= f(x) + \langle \nabla f(x), y - x \rangle + \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt \\ &\leq f(x) + \langle \nabla f(x), y - x \rangle + \int_0^1 L\|y - x\|^2 t dt \\ &= f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2, \end{aligned} \tag{11}$$

where the first equality arises from applying the Fundamental Theorem of Calculus on $g(t) = f(x + t(y - x))$, and the inequality arises from the Lipschitz property of the gradients. □

Gradient descent applied on a convex, smooth function yields a convergence rate of order $O(1/T)$. The following theorem formally states this result.

Theorem 3.13. *If f is convex and L-smooth, then*

$$f(x_T) - f(x^*) \leq \frac{4L\|x_1 - x^*\|^2}{T}, \tag{12}$$

when we set $\eta = \frac{1}{3L}$.

Remark 3.8. *The constants in the above theorem are not optimal. Interestingly, we get the convergence result in terms of the last iterate x_T. This is possible, since at each time gradient descent update, we have strict descent if the step size is chosen appropriately as shown below.*

Lemma 3.14. *If $1 - \frac{\eta L}{2} > 0$, then $f(x_{t+1}) \leq f(x_t)$ and*

$$\frac{f(x_t) - f(x^*)}{\eta(1 - \eta L/2)} \geq \|\nabla f(x_t)\|^2. \tag{13}$$

The proof of the above lemma is straightforward to see by setting $y = x_{t+1} = x_t - \eta_t \nabla f(x_t)$ and $x = x_t$ in Lemma 3.12.

We now prove Theorem 3.13.

Proof. We have that

$$\|x_{t+1} - x^*\|^2 = \|x_t - x^*\|^2 - 2\eta\langle x_t - x^*, \nabla f(x_t)\rangle + \eta^2\|\nabla f(x_t)\|^2. \tag{14}$$

Using Lemma 3.14 we can write

$$\|x_{t+1} - x^*\|^2 \leq \|x_t - x^*\|^2 - 2\eta\langle x_t - x^*, \nabla f(x_t)\rangle + \frac{\eta}{1 - \eta L/2}(f(x_t) - f(x^*))$$

Now using convexity we can further write

$$\left(2\eta - \frac{\eta}{1 - \eta L/2}\right)[f(x_t) - f(x^*)] \leq \|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2. \tag{15}$$

Using $\eta L = 1/3$ and the strict descent property as stated in Lemma 3.14, we can sum over t to obtain

$$f(x_T) - f(x^*) \leq \frac{5}{4\eta T}\|x_1 - x^*\|^2 \leq \frac{4L}{T}\|x_1 - x^*\|^2. \tag{16}$$

□

Remark 3.9. *For both strongly convex and smooth functions we get $O(1/T)$ rate. But as the proofs reveal, the reasons for this rate are quite different. In the case of strong convexity, we are able to leverage strong convexity to get an exponentially fast rate for gradient flow but we can only bound the discretization error term by $\eta^2 L^2$. On the other hand, we can bound the discretization error term proportional to $\|\nabla f(x_t)\|^2$ term much better because of Lemma 3.14. Basically Lemma 3.14 tells us that $\|\nabla f(x_t)\|^2$ gets smaller when arrive near the optima. So the overall updates get autmatically smaller even if our step size remains of constant order. It is useful to keep this intuition in mind.*

Remark 3.10. *We can have higher order versions of gradient descent. To motivate this, we can see first order gradient descent as the solution to a minimization of a first order approximation with quadratic penalty, i.e.,*

$$x_{t+1} = \arg\min_x f(x_t) + \langle \nabla f(x_t), x - x_t\rangle + \frac{1}{2\eta}\|x - x_t\|^2. \tag{17}$$

In order to extend to higher order derivatives, we can use a quadratic approximation with a cubic penalty as follows:

$$\arg\min_x f(x_t) + \langle \nabla f(x_t), x - x_t\rangle + \langle x - x_t, \nabla^2 f(x_t)(x - x_t)\rangle + \frac{1}{6\eta}\|x - x_t\|^3. \tag{18}$$

This is said to be the cubic regularized Newton method. Now, it is known that if the Hessians are Lipschitz (in a certain sense), we can prove convergence of the order $1/T^2$ for this cubic regularized Newton's method. So, if you add more smoothness to your convex function, it is possible to attain faster rates, albeit with higher order versions of gradient descent.

add linear regression example

3.8 Both Strongly Convex and Smooth Functions

If f is smooth and strongly convex, we can get exponentially fast rate of convergence of gradient descent. This is the content of the following theorem.

Theorem 3.15. *If f is α strongly convex and L smooth then gradient descent with step size $\eta = \frac{1}{2L}$ satisfy*

$$\|x_t - x^*\|^2 \leq (1 - \mu\eta)^t \|x_t - x^*\|^2.$$

Consequently, by the smoothness property,

$$f(x_t) - f(x^*) \leq L(1 - \mu\eta)^t \|x_t - x^*\|^2.$$

Proof. The proof is left as an exercise in Problem Set 1. □

3.9 Summary

We can now summarize the convergence rates we have seen for different function classes.

Gradient Descent	Optimal Rate	Function Class
$\frac{G\|x_1 - x^*\|}{\sqrt{T}}$	$\frac{G\|x_1 - x^*\|}{\sqrt{T}}$	Non Smooth G Lipschitz Convex Functions.
$\frac{G^2}{\alpha T}$	$\frac{G^2}{\alpha T}$	Non Smooth α Strongly Convex G Lipschitz Functions.
$\frac{L\|x_1 - x^*\|}{T}$	$\frac{L\|x_1 - x^*\|}{T^2}$	L Smooth Convex Functions.
$\exp(-T/\kappa)\|x_1 - x^*\|^2$	$\exp(-T/\sqrt{\kappa})\|x_1 - x^*\|^2$	L Smooth and α Strongly Convex Functions.

The first column gives the rates for Gradient Descent while the second column gives the optimal rate for the function classes. Note that for smooth functions, Gradient Descent does not attain the optimal rate. In the last row, $\kappa = \frac{L}{\alpha}$ is the condition number. In this case, it turns out that a different algorithm called the accelerated gradient descent attains the optimal rates. This is what we study in the next section.

3.10 Nesterov's Accelerated Gradient Descent

We now present Nesterov's Accelerated Gradient Descent [22]. This algorithm attains the optimal $O(1/T^2)$ rate for smooth functions. Gradient Descent has an inherent lack of memory and it turns out that it is possible to attain a better rate by using previous information when the underlying function is smooth. Nesterov's method reveals that the only thing that matters from the past is a momentum like term. This analysis is unfortunately not going to be very illuminating; see the blogpost https://blogs.princeton.edu/imabandit/2015/06/30/revisiting-nesterovs-acceleration/ for some nice references which attempt to explain this algorithm from different perspectives.

Define momentum at the t th iterate to be

$$m_t = \gamma_t(x_t - x_{t-1})$$

where $\gamma_t > 0$ is a tuning parameter sequence. Also let us use the notation x^+ to denote the one step gradient descent update with the ideal tuning parameter ($\eta = \frac{1}{L}$) for a L smooth function

$$x^+ = x - \frac{1}{L}\nabla f(x).$$

Definition 3.16. *Nesterov's AGD is defined by the following update rule*

$$x_{t+1} = (x_t + m_t)^+$$

Theorem 3.17. *Let f be a L smooth convex function. Define the sequence of weights satisfying $w_0 = 0, w_1 = 1$ and for $t \geq 1$,*

$$w_t^2 - w_{t-1}^2 = w_t.$$

Also define the sequence $\{\gamma_{t+1}\}_{t=0}^{T-1}$ such that

$$\gamma_{t+1} = \frac{w_t - 1}{w_{t+1}}.$$

Now by running Nesterov's AGD with the above choice of $\gamma_1, \ldots, \gamma_T$ the following holds:

$$f(x_{T+1}) - f(x^*) \leq \frac{cL}{2T^2}\|x_0 - x^*\|^2$$

for some absolute constant $c > 0$.

Proof. Denote

$$g_t = -\frac{1}{L}\nabla f(x_t + m_t).$$

Then we can write the AGD update as

$$x_{t+1} = x_t + m_t + g_t. \tag{19}$$

Denote $\delta_t = f(x_t) - f(x^*)$ to be the suboptimality gap. We will now obtain bounds on both $\delta_{t+1} - \delta_t$ and δ_t. Recall that for any x we have by setting $\eta = \frac{1}{L}$ in Lemma 3.14,

$$f(x^+) \leq f(x) - \frac{1}{2L}\|\nabla f(x)\|^2. \tag{20}$$

We have

$$\delta_{t+1} - \delta_t = f((x_t + m_t)^+) - f(x_t) \leq f(x_t + m_t) - f(x_t) - \frac{1}{2L}\|\nabla f(x_t + m_t)\|^2 \leq$$
$$\nabla f(x_t + m_t)^T m_t - \frac{1}{2L}\|\nabla f(x_t + m_t)\|^2 = -Lg_t^T m_t - \frac{L}{2}\|g_t\|^2$$

where in the first inequality we used (20) and in the second inequality we used convexity.

Similarly, we can obtain

$$\delta_{t+1} \leq f(x_t + m_t) - f(x^*) - \frac{1}{2L}\|\nabla f(x_t + m_t)\|^2 \leq -Lg_t^T(x_t + m_t - x^*) - \frac{L}{2}\|g_t\|^2.$$

Now, for a weight sequence w_t we can multiply the second last display above by $w_t - 1$ and add it to the last display to obtain

$$w_t\delta_{t+1} - (w_t - 1)\delta_t \leq -\frac{L}{2}w_t\|g_t\|^2 - Lg_t^T(w_t m_t + x_t - x^*) \tag{21}$$

$$= \frac{-L}{2w_t}\left(\|x_t - x^* + w_t m_t + w_t g_t\|^2 - \|x_t - x^* + w_t m_t\|^2\right) \tag{22}$$

where in the last equality we used $\|a\|^2 + 2a^T b = \|a+b\|^2 - \|b\|^2$ with $a = w_t g_t, b = w_t m_t + x_t - x^*$.

Now, we desire to have a telesoping sum on the R.H.S above. In order for this to happen we wish that

$$x_t - x^* + w_t m_t + w_t g_t = x_{t+1} - x^* + w_{t+1}m_{t+1} = x_t + m_t + g_t - x^* + w_{t+1}\gamma_{t+1}(x_{t+1} - x_t)$$

where in the last equality we used the identity (19).

Equivalently, we wish for

$$w_t(m_t + g_t) = (m_t + g_t)(1 + w_{t+1}\gamma_{t+1}).$$

Let us set $\{\gamma_{t+1}\}_{t=0}^{T-1}$ such that the above is true, that is,

$$\gamma_{t+1} = \frac{w_t - 1}{w_{t+1}}.$$

Then by denoting

$$u_t = \|x_t + w_t m_t - x^*\|^2$$

we can rewrite (21) as

$$w_t^2\delta_{t+1} - (w_t^2 - w_t)\delta_t \leq -\frac{L}{2}(u_{t+1} - u_t). \tag{23}$$

Now set w_t such that $w_0 = 0, w_1 = 1$ and

$$w_t^2 - w_{t-1}^2 = w_t.$$

With this choice, we can rewrite (23) as follows:

$$w_t^2\delta_{t+1} - w_{t-1}^2\delta_t \leq \frac{L}{2}(u_t - u_{t+1}).$$

The previous bound is to our liking because on both sides we have a telescoping sum like term. We can now sum over $t = 1, \ldots, T$ to obtain

$$w_T^2\delta_{T+1} \leq \frac{L}{2}u_1.$$

Now, it remains to check that $u_1 = \|x_0 - x^*\|^2$ and $w_T \geq cT$ for some absolute constant $c > 0$. The second fact is left as an exercise!

Note that

$$u_1 = \|x_1 + w_1 m_1 - x^*\|^2 = \|x_1 + w_1\gamma_1(x_1 - x_0) - x^*\|^2 = \|x_1 - (x_1 - x_0) - x^*\|^2 = \|x_0 - x^*\|^2$$

where in the first equality we used the definition of m_1 and in the second equality we used the fact that $\gamma_1 w_1 = w_0 - 1 = -1$ by definition.

□

Remark 3.11. *There is a result for AGD for both strongly convex and smooth functions which shows that it is sufficient to do $O(\sqrt{\kappa}\log\frac{1}{\epsilon})$ many iterations to attain ϵ suboptmality gap.*

Remark 3.12. *There are other types of accelerated GD methods; for instance one is by Nemirovski. It appears that Nesterov's is the most popular.*

3.11 Stochastic Gradient Descent

What if we do not have exact access to the gradient? The classic paper Robbins and Monro [28] considered the problem when we only observe a stochastic version, namely a vector g_t satisfying $\mathbb{E}[g_t \,|\, x_t] = \nabla f(x_t)$ for every t. Then, in the SGD algorithm, we let

$$x_{t+1} = x_t - \eta_t g_t.$$

The paper [28] showed that under appropriate choices of the step size η_t and appropriate assumptions on f we have $x_t \to x^*$. A nice survey of SGD is given in [23].

3.11.1 Useage in Machine Learning

Typically in Machine Learning, one is interested in optimizing functions of the form $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ where each f_i corresponds to a data point. Here, i_t is sampled uniformly from the finite set $\{1, \ldots, n\}$ and then the stochastic gradient is set to be

$$g_t(x_t) = \nabla f_{i_t}(x_t).$$

This is often called the *multi-pass SGD*.

Remark 3.13 (Single Pass SGD)**.** *In the i.i.d statistical learning, a single pass through the data actually gives a generalization errror bound. This is in contrast to the multipass version which minimizes the training error.*

3.11.2 Stochastic Convex Optimization

We now consider rates of convergence of SGD for convex functions.

Theorem 3.18 (Non Smooth Lipschitz Convex Funtions)**.** *Suppose f is a convex function, and the stochastic gradients g_t are all independent and satisfy*

$$\mathbb{E}[g_t \,|\, x_t] = \nabla f(x_t), \quad \textit{and} \quad E\big[\|g_t\|^2 \,|\, x_t\big] \leq \sigma^2. \tag{24}$$

If we choose the step size $\eta_t = \frac{\|x_1 - x^*\|\sigma}{\sqrt{t}}$, then

$$\frac{1}{\sum_{t=1}^T \eta_t} \sum_{t=1}^T \eta_t \big(\mathbb{E}[f(x_t)] - f(x^*)\big) \leq \frac{\|x_1 - x^*\|\sigma \log T}{\sqrt{T}}.$$

Proof. The proof is left as an exercise. □

Remark 3.14. *Under a random first order oracle model, where we are allowed to observe noisy function and gradient values, a lower bound scaling like $O(1/\sqrt{T})$ regret can be shown even when $d = 1$. Contrast this with the deterministic case, when the $O(1/\sqrt{T})$ holds only in high dimensions.*

As in the case of usual gradient descent, strong convexity leads to a faster rate of convergence for SGD as well.

Theorem 3.19 (Strongly Convex Functions)**.** *Suppose f is α-strongly convex, and the stochastic gradient g_t satisfies (24). If we choose step size $\eta_t = \frac{1}{\alpha t}$, then*

$$\frac{1}{T} \sum_{t=1}^T \big(\mathbb{E}[f(x_t)] - f(x^*)\big) \leq \frac{\sigma^2 \log t}{2\alpha t}.$$

Proof. The proof is left as an exercise. □

Remark 3.15. *Note that the above theorems show that SGD attains the same rate of convergence as GD upto log factors. In this sense, there is minimal loss in using stochastic gradients which appears at a first glance to be a surprising and interesting fact.*

3.11.3 Cannot Expect Fast Rates for General Smooth Convex Functions

Recall that when we use GD in the setting of convex optimization , the smoothness of function f will help us derive a better bound on the discretization error. This is because the gradient $\|\nabla f(x_t)\|$ gets smaller as x_t goes closer to the optima. However, things are different in SGD, because we cannot expect the stochastic gradient $\|g_t\|$ to get smaller near optima due to the variance of the noise. This prevents us from getting the $O(1/T)$ rate for general smooth convex functions and the $\exp(-T/\kappa)$ rate for both strongly convex and smooth convex functions using SGD. However, for functions of the form of a finite sum as is of interest in ML, one can modify SGD by reducing its variance to still attain fast rates. This is the content of the next section.

3.11.4 Variance Reduced SGD for Finite Sum Functions

Consider the finite sum setting, where we have

$$f = \frac{1}{n}\sum_{i=1}^{n} f_i$$

and assume that each f_i is L-smooth while the function f overall is still α-strongly convex. Such functions arise naturally in ML; for instance when we do regularized least squares or logistic regression.

For such functions, in order to achieve ε sub-optimality gap, one needs $O(\kappa \log \frac{1}{\varepsilon})$ iterations of GD (using Theorem 3.15) and one needs $O(\frac{1}{\alpha\varepsilon})$ iterations in SGD (using Theorem 3.19). However, to compute GD in each iteration we need to touch all the n data points. Therefore, we should really compare $O(n \log \frac{1}{\varepsilon})$ to $O(\frac{1}{\varepsilon})$. GD is preferable when we desire very high accuracy, that is, ε is very small. However, for reasonable ε and large n, SGD might also be preferable. It is thus a natural question to ask if there is a method which enjoys cheap computation per round like the SGD and yet gets $\log \frac{1}{\varepsilon}$ dependence on ε like GD. This is where the idea of variance reduction comes in. This was first proposed by [16] and has now been used in several other optimization problems.

We now introduce the Stochastic Variance Reduced Gradient (SVRG) algorithm.

Algorithm 1: SVRG algorithm

Data: access g of the gradient ∇f at any arbitrary point, step size η, length of each epoch T, number of iteration N, step size $\eta = cL$ with $0 < c \ll 1$

Result: estimation of the optimal point x^* of f

```
Initialize y_1;
for j ← 1 to N do
    Calculate ∇f(y_j);
    x_0 = y_j;
    for t ← 1 to T do
        i_t ∼ unif([n]);
        g_t = ∇f_{i_t}(x_t) − ∇f_{i_t}(y_j) + ∇f(y_j);
        x_{t+1} = x_t − ηg_t;
    end
    y_{j+1} = (1/T) Σ_{t=1}^{T} x_t;
end
Output y_{T+1} to approximate x*;
```

Remark 3.16. *In words, the SVRG algorithm computes the full gradient (using all data points) once in a while and uses this full gradient to construct a variance reduced stochastic gradient (using a single random data point) in the other rounds.*

For SVRG, we have the following result.

Theorem 3.20. *By taking $\eta = \frac{c}{L}$ and $T = \frac{CL}{\alpha}$ for constants c, C, we have*

$$\mathbb{E}\big[f(y_{j+1}) - f(x^*)\big] \leq \frac{2c + \frac{1}{cC}}{1 - 2c}\mathbb{E}\big[f(y_j) - f(x^*)\big].$$

Proof. Recall that we have

$$\begin{aligned} f(y_{j+1}) - f(x^*) &= f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \\ &\leq \frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^*) \\ &\leq \frac{1}{2\eta T}\|y_j - x^*\|^2 + \frac{\eta}{2T}\sum_{t=1}^{T}\|g_t\|^2 \\ &=: F_e + D_e. \end{aligned}$$

where the first inequality follows from convexity and the second inequality follows from the basic lemma 3.2. As usual, we can think of F_e as the flow error term and D_e to be the discretization error term.

By strong convexity, the flow error F_e can be bounded by

$$F_e \leq \frac{1}{\alpha\eta T}\big(f(y_j) - f(x^*)\big). \tag{25}$$

To handle the discretization error D_e, we will need the following lemma.

Lemma 3.21. *If $i \sim$ Unif $([n])$, then for any x we have*

$$\mathbb{E}\big\|\nabla f_i(x) - \nabla f_i(x^*)\big\| \leq 2L\big(f(x) - f(x^*)\big).$$

Let us see how we can finish the proof assuming the above lemma. We can write

$$\begin{aligned} \mathbb{E}\|g_t\|^2 &= \mathbb{E}\|\nabla f_{i_t}(x_t) - \nabla f_{i_t}(y_j) + \nabla f(y_j)\|^2 \\ &\leq 2\mathbb{E}\|\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x^*)\|^2 + 2\mathbb{E}\|\nabla f_{i_t}(y_j) - \nabla f(y_j) - \nabla f_{i_t}(x^*)\|^2. \end{aligned}$$

To bound the second term above in the R.H.S, we have

$$\begin{aligned} &\mathbb{E}\|\nabla f_{i_t}(y_j) - \nabla f(y_j) - \nabla f_{i_t}(x^*)\|^2 \\ &= \mathbb{E}\big\|\big(\nabla f_{i_t}(y_j) - \nabla f_{i_t}(x^*)\big) - \mathbb{E}\big(\nabla f_{i_t}(y_j) - \nabla f_{i_t}(x^*)\big)\big\|^2 \\ &\leq \mathbb{E}\big\|\nabla f_{i_t}(y_j) - \nabla f_{i_t}(x^*)\big\|^2 \end{aligned}$$

where we used the fact that $\mathbb{E}\big(\nabla f_{i_t}(y_j) - \nabla f_{i_t}(x^*)\big) = \nabla f(y_j)$ and the fact that variance is at most the second moment.

From the last two displays, we obtain

$$\begin{aligned} \mathbb{E}\|g_t\|^2 &\leq 2\mathbb{E}\|\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x^*)\|^2 + 2\mathbb{E}\big\|\nabla f_{i_t}(y_j) - \nabla f_{i_t}(x^*)\big\|^2 \\ &\overset{\text{(i)}}{\leq} 4L\big(f(x_t) - f(x^*)\big) + 4L\big(f(y_j) - f(x^*)\big). \end{aligned}$$

where in the second inequality we used Lemma 3.21.

This implies that the discretization error term is bounded as follows:

$$\begin{aligned} D_e &\le \frac{4L\eta}{2T}\sum_{t=1}^{T}\Big[\big(f(x_t)-f(x^*)\big)+\big(f(y_j)-f(x^*)\big)\Big] \\ &= 2L\eta\cdot\frac{1}{T}\sum_{t=1}^{T}\big[f(x_t)-f(x^*)\big]+2L\eta\big[f(y_j)-f(x^*)\big]. \end{aligned}$$

Combining the last display alongwith (25) we get

$$\begin{aligned} \mathbb{E}f(y_{j+1})-f(x^*) &\le \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[f(x_t)-f(x^*)\big] \le F_e+D_e \\ &\le \Big(2L\eta+\frac{1}{\alpha\eta T}\Big)\big(\mathbb{E}f(y_j)-f(x^*)\big)+2L\eta\cdot\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[f(x_t)-f(x^*)\big] \end{aligned}$$

which further implies

$$\mathbb{E}f(y_{j+1})-f(x^*) \le \frac{2L\eta+\frac{1}{\alpha\eta T}}{1-2L\eta}\big(\mathbb{E}f(y_j)-f(x^*)\big).$$

By choosing $L\eta = c \ll 1$ and $T = \frac{CL}{\alpha}$, the contraction number is $\frac{2c+\frac{1}{cC}}{1-2c} < 1$ by choosing c small enough and C large enough.

All that remains is to give a proof of Lemma 3.21.

Let us define

$$g_i(x) = f_i(x) - f_i(x^*) - \langle \nabla f_i(x^*), x - x^* \rangle$$

i.e. the Bregman divergence of f_i. Then $g_i(x^*) = 0$ and $g_i \ge 0$ due to the convexity of f_i. Notice that g_i is also L-smooth, since f_i is L-smooth. Therefore, by applying Lemma 3.14 with $\eta = 1/L$ we obtain Therefore, we have for any x,

$$\|\nabla g_i(x)\|^2 \le 2L\ \ g_i(x).$$

Since $\nabla g_i(x) = \nabla f_i(x) - \nabla f_i(x^*)$, taking expectation w.r.t. i yields

$$\mathbb{E}\|\nabla f_i(x) - \nabla f_i(x^*)\|^2 = \mathbb{E}\|\nabla g_i(x)\|^2 \le 2L\,\mathbb{E}g_i(x) = 2L\big(f(x)-f(x^*)\big).$$

This finishes the proof of Lemma 3.21 and hence finishes the proof of Theorem 3.20.

□

4 Online Convex Optimization

Recall the basic OCO framework introduced by Zinkevich [34].

1. The learner plays $x_t \in \mathcal{A}$ where $K \subset \mathbb{R}^d$ is a convex set of all possible actions.
2. Nature or Adversary reveals a convex loss function $f_t : \mathcal{A} \to \mathbb{R}_+$.
3. Incur loss $f_t(x_t)$.

For any algorithm that generates $x_1, x_2, \ldots, x_T$, its regret at a point $x \in K$ is

$$R_T = \sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x).$$

The goal of the learner is to minimize the regret. Ideally we would like to develop online algorithms whose regret would grow sublinearly $o(T)$ for all points $x \in K$.

4.1 Online (Projected) (Sub)Gradient Descent

The online gradient descent algorithm can be extended to the online setting. The update step is as follows:

$$x_{t+1} = P_K(x_t - \eta_t \nabla f_t(x_t)) \tag{26}$$

where P_K is the projection operator onto the convex set K. Essentially the same analysis of OGD we have seen so far extends to the online setting as well to give a $O(\sqrt{T})$ regret.

Theorem 4.1 (OGD for Non Smooth Lipschitz Losses)**.** *Let $x_1, \ldots, x_T$ denote the OGD updates with step size η. Suppose each of the loss functions $f_1, \ldots, f_T$ are G lipschitz and $Diam(K) \leq D$. Then we have the regret bound for any $x^* \in K$,*

$$\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \frac{D^2}{2\eta} + \frac{\eta}{2} T G^2.$$

Consequently, by setting $\eta = \frac{D}{G\sqrt{T}}$ we obtain the bound $DG\sqrt{T}$.

Proof. We will simply use the basic lemma 3.2 by setting $g_t = \nabla f_t(x_t)$,

$$\begin{aligned}\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \sum_{t=1}^{T} \nabla f_t^{\intercal}(x_t)(x_t - x^*) &\leq \frac{|x_1 - x^*|^2}{2\eta} + \frac{\eta}{2} \sum |\nabla f_t(x_t)|^2 \\ &\leq \frac{D^2}{2\eta} + \frac{\eta}{2} T G^2.\end{aligned}$$

□

Remark 4.1. *If instead, $\eta_t = \frac{D}{G\sqrt{t}}$ is taken to be time varying, then it is possible to show a regret bound $\frac{3}{2} DG\sqrt{T}$, inflated by $3/2$.*

Remark 4.2. *As before, the above proof works even if we consider a subgradient of f_t at x_t.*

4.1.1 Lower Bound

Theorem 4.2. *For any online algorithm there exists an online convex optimization problem such that this algorithm suffers a regret $\Omega(DG\sqrt{T})$.*

Proof. Let the action space be the n dimensional hypercube

$$K = \{x \in \mathbb{R}^n, |x|_\infty \leq 1\}. \tag{27}$$

The dimension n can be anything including 1.

For each vertex of the hypercube $v \in \{\pm 1\}$ define the linear loss function

$$f_v(x) = v^\intercal x, \quad \forall v \in \{\pm 1\}^n$$

Therefore, the diameter D of K and the lipschit constant of the loss functions are bounded by

$$D \leq 2\sqrt{n}, \text{ and } G \leq \sqrt{n} \tag{28}$$

Suppose at each round t, the loss function f_{v_t} is obtained by choosing v_t uniformly from the finite set K. Now note that under this assumption, for any online algorithm,

$$\mathbb{E}_{v_t}[f_t(x_t)] = \mathbb{E}_{v_t}[v_t^\intercal x_t] = 0$$

.

However,

$$\mathbb{E}_{v_t}\left[\min_x \sum_{t=1}^T f_t(x)\right] = \mathbb{E}_{v_t}\left[\min_x \sum_{t=1}^T \sum_{i=1}^n v_t(i)x_i\right] \tag{29}$$

$$= n\mathbb{E}\left[-\left|\sum_{t=1}^T v_t(1)\right|\right] \tag{30}$$

$$= -\Omega(n\sqrt{T}) \tag{31}$$

□

4.2 OGD Regret for Strongly Convex Losses

As in the offline case, we can improve the regret bound for OGD if all the loss functions $f_1, \ldots, f_T$ are strongly convex.

Theorem 4.3. *If all the loss functions f_t are α-strongly convex in addition to G lipschitz then by setting $\eta_t = \frac{1}{\alpha t}$, we can obtain the regret bound for any $x^* \in K$,*

$$\sum_{t=1}^T f_t(x_t) - f_t(x^*) \leq \frac{G}{2\alpha}(1 + \log T). \tag{32}$$

Proof. α-strongly convexity gives

$$2(f_t(x_t) - f_t(x^*)) \leq 2\nabla f_t^\intercal(x_t)(x_t - x^*) - \alpha \left|x_t - x^*\right|^2$$

From the Pythagorean theorem about convex projections we have

$$|x_{t+1} - x^*|^2 = |P_K(x_t - \eta_t \nabla f_t(x)t)) - x^*)|^2 \leq |x_t - \eta_t \nabla f_t(x)t - x^*|^2$$

and by expanding the square we get

$$|x_{t+1} - x|^2 \leq |x_t - x|^2 + \eta_t^2 \left|\nabla f_t(x_t)\right|^2 - 2\eta_t \nabla f_t^\intercal(x_t)(x_t - x^*)$$

which further can be rearranged to give us the upper bound

$$2\nabla f_t^\intercal(x_t)(x_t - x^*) \leq \frac{\left|x_t - x\right|^2 - \left|x_{t+1} - x\right|^2}{\eta_t} + \eta_t G^2.$$

Now use the lower bound on $2\nabla f_t^\intercal(x_t)(x_t - x^*)$ given by the first display alongwith the display above to obtain

$$\begin{aligned} 2\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) &\leq \sum_{t=1}^{T} |x_t - x^*|^2 \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \alpha\right) + G^2 \sum_{t=1}^{T} \eta_t \\ &\leq 0 + G^2 \sum_{t=1}^{T} \frac{1}{\alpha t} \leq \frac{G^2}{\alpha}(1 + \log T) \end{aligned}$$

where in the last inequality we used the fact that $\eta_t = \frac{1}{\alpha t}$ and $\eta_0 = 0$.

□

4.3 Exp Concave Functions

In this section, we consider a class of functions called exp concave functions. The motivation behind considering such functions is the following. Consider the following loss functions of interest:

- $f_t(\beta) = (x_t^T \beta - y_t)^2$ (Online Linear Regression)
- $f_t(\beta) = \log(1 + \exp -y_t x_t^t \beta)$ (Online Logistic Regression)
- $f_t(u) = -\log(\gamma_t^T u)$ (Online Portfolio Selection)

These loss functions are not strongly convex as it can be checked that the Hessian is a rank 1 matrix. Are we doomed to a $O(\sqrt{T})$ regret in these cases? The answer is no!

We will see that the loss functions above still posess curvature allowing for a faster regret bound. In particular, all the loss functions above are exp concave functions. The class of strongly convex lipschitz functions are exp concave but the reverse is not necessarily true. As we will see, this class of functions still posess a notion of curvature which allows $O(\log T)$ regret. The algorithm that achieves such log regret is the Online Newton Step algorithm which is also a fundamental online learning algorithm.

4.3.1 Universal portfolio selection

A practical example where an exponential concave loss function arises is the universal portfolio selection problem. Consider d stocks and S_d be the probability simplex in d dimensions. For time $t = 1, \cdots, T$,

- Choose a distribution $x_t \in \mathcal{S}_d$ where $x_t(i)$ corresponds to the proportion of current wealth invested in stock i.
- Adversary/market gives stock returns $\gamma_t \in \mathbb{R}_+^d$ where $\gamma_t(i) = \frac{p_{t+1}(i)}{p_t(i)}$.
- Let $w_t =$ Total wealth on day t. Then $w_{t+1} = w_t \sum_{i=1}^d x_t(i)\gamma_t(i)$ and hence $\frac{w_{t+1}}{w_t} = \gamma_t^T x_t$

From the expressions,

$$\log(w_{t+1}) - \log(w_t) = \log(\gamma_t^T x_t) \tag{33}$$

We can view this as an instance of OCO where the convex loss functions are $f_t(u) = -\log(\gamma_t^T u)$. The regret can hence be defined as

$$Regret = \sum_{t=1}^T \gamma_t x_t - \min_{x \in \mathcal{S}_d} \sum_{t=1}^T \log(\gamma_t^T x) \tag{34}$$

The above measures the excess loss as compared to the best constant rebalancing portfolio.

Observe that

$$\begin{aligned} \nabla(-\log(\gamma^T x)) &= \frac{-\gamma_t}{\gamma_t^T x} \\ \nabla^2(-\log(\gamma_t^T x)) &= \frac{\gamma_t \gamma_t^T}{(\gamma_t^T x)^2} \end{aligned} \tag{35}$$

Observe that the Hessian $\frac{\gamma_t \gamma_t^T}{(\gamma_t^T x)^2}$ is rank 1 and hence not stongly convex, however, it is possible to attain $O(\log(T))$ regret as the loss function that is to be minimized, $-\log(\gamma^T x)$ is exp concave.

4.3.2 Definition and Properties

Definition 4.4. *A function $f : K \to \mathbb{R}$ is α-exp-concave over K if $e^{-\alpha f}$ is concave.*

Lemma 4.5. *A twice differentiable function f is α exp concave iff $\nabla^2 f(x) \succeq \alpha \nabla f(x) \nabla f(x)^T$.*

We leave the proof of the above lemma as an exercise.

The next lemma shows that exp concave and lipschitz functions posess curvature in the direction of its gradient.

Lemma 4.6. *Let $f : K \to \mathbb{R}$ be an α-exp-concave function and D, G denote the radius of K and lipschitz constant of f. The following holds for all $\gamma \leq \frac{1}{2}\min\{\frac{1}{GD}, \alpha\}$ and all $x, y \in \mathcal{K}$:*

$$f(u) \geq f(w) + \langle u - w, \nabla f(x)\rangle + \frac{\alpha}{2}(\langle u - w, \nabla f(w)\rangle)^2$$

Proof. For all γ s.t. $2\gamma \leq \alpha$ the function $h(x) = e^{-2\gamma f(x)}$ is concave (can be inferred from composition of functions $x^{\frac{2\gamma}{\alpha}}$ which is concave and increasing and $\exp\{-\alpha f(x)\}$ which is concave). By the concavity of $h(x)$,

$$h(u) \leq h(w) + \nabla h(w)^T(u - w)$$

The gradient $\nabla h(w) = -2\gamma \exp\{-2\gamma f(w)\}\nabla f(w)$ gives

$$\exp -2\gamma f(u) \leq \exp\{-2\gamma f(w)\}\big(1 - 2\gamma \nabla f(w)^T(u - w)\big)$$

Simplifying the equation by taking $\log$ on both sides,

$$f(u) \geq f(w) - \frac{1}{2\gamma}\log(1 - 2\gamma \nabla f(w)^T(u - w))$$

By the Cauchy Schwarz inequality $|2\gamma \nabla f(w)^T(u-w)| \leq 2\gamma GD \leq 1$. Moreover, we will use the fact that for $|x| \leq 1$, $\log(1+x) \leq x - \frac{1}{4}x^2$. Using the above two facts for $x = 2\gamma \nabla f(w)^T(u - w)$, we get the inequality

$$f(u) \geq f(w) + \langle u - w, \nabla f(w)\rangle + \frac{\alpha}{2}(\langle u - w, \nabla f(w)\rangle)^2$$

□

4.4 Online Newton Step (ONS) algorithm

Next we provide the formal description of the Online Newton Step Algorithm.

Consider the action space $K \subseteq \mathbb{R}^d$ which we assume contains 0. Fix any arbitrary $\epsilon, \gamma > 0$. Then ONS(ϵ, γ) is as follows.

1. Initialize $x_1 = 0 \in K$ and $A_0 = \epsilon I_d$, where I_d is the $d \times d$ identity matrix.
2. For any time $t \in [T]$, let
   (a)
   $$\nabla_t := \nabla f_t(x_t) \quad \text{and} \quad A_t := A_{t-1} + \nabla_t \nabla_t^T.$$
   (b) Update x_{t+1} as
   $$x_{t+1} = P_{A_t}\left(x_t - \frac{1}{\gamma}A_t^{-1}\nabla_t\right),$$

where for any $u \in \mathbb{R}^d$,

$$P_{A_t}(u) := \operatorname*{argmin}_{x \in K} (x-u)^T A_t (x-u).$$

Remark 4.3. *Recall that in ordinary Newton-Raphson method we update x_{t+1} as*

$$x_{t+1} = x_t - H_t^{-1} \nabla_t,$$

where H_t is the corresponding Hessian matrix. In the ONS algorithm, we see that the update is of a similar form.

Remark 4.4. *A natural question at this point is how one can come up with the ONS algorithm. There is a Follow the Leader (FTL) interpretation of ONS which may be a more natural starting point to explain ONS; see the paper [15] where ONS was first introduced.*

Now we present the main result which establishes an upper bound on the regret of the ONS algorithm when the loss functions are exp-concave.

Theorem 4.7 (Regret Bound for Online Newton Step)**.** *Suppose $K \subseteq \mathbb{R}^d$ such that for every $v \in K$, $|v| \leq D$, and for every $t \in [T]$, let the loss function $f_t : K \to \mathbb{R}$ be an α-exp-concave function such that $|\nabla f| \leq G$. If the updates $\{x_t : t \in 0, \ldots, T\}$ are derived by the ONS(ϵ, γ) algorithm then the regret*

$$R_T := \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x) \leq \frac{\gamma}{2} \epsilon D^2 + \frac{d}{2\gamma} \log \left(1 + \frac{TG^2}{\epsilon d} \right).$$

Moreover, if we set

$$\epsilon = \frac{d}{\gamma^2 D^2} \quad \textit{and} \quad \gamma = \frac{1}{2} \min \left\{ \alpha, \frac{1}{2GD} \right\}$$

then

$$R_T \leq d \left\{ \log \left(1 + \frac{T}{16 d^2} \right) + 1 \right\}.$$

Remark 4.5. *Thus, the Online Newton Step algorithm, ideally tuned, does attain a $O(d \log T)$ regret.*

Proof. Let

$$x^* := \operatorname*{argmin}_{x \in K} \sum_{t=1}^{T} f_t(x).$$

For any $t \in [T]$, if we denote by $|\cdot|_{A_t}$ the norm induced by A_t, i.e.,

$$|u|_{A_t} := u^T A_t u, \quad u \in \mathbb{R}^d,$$

then

$$
\begin{aligned}
&|x_{t+1} - x^*|_{A_t}^2 \\
&\leq \left| x_t - \frac{1}{\gamma} A_t^{-1} \nabla_t - x^* \right|_{A_t}^2 \\
&= |x_t - x^*|_{A_t}^2 - \frac{2}{\gamma} \langle x_t - x^*, \nabla_t \rangle + \frac{1}{\gamma^2} \nabla_t A_t^{-1} \nabla_t \\
&= |x_t - x^*|_{A_{t-1}}^2 + \langle x_t - x^*, \nabla_t \rangle^2 - \frac{2}{\gamma} \langle x_t - x^*, \nabla_t \rangle + \frac{1}{\gamma^2} \nabla_t A_t^{-1} \nabla_t, \qquad (36)
\end{aligned}
$$

where the inequality follows from the Pythagoras Theorem and the last equality follows by using the fact that $A_t = A_{t-1} + \nabla_t \nabla_t^T$.

Therefor, for any $t \in [T]$ we have

$$
\begin{aligned}
f_t(x_t) - f_t(x^*) &\leq \langle x_t - x^*, \nabla_t \rangle - \frac{\gamma}{2} \langle x_t - x^*, \nabla_t \rangle^2 \\
&\leq \frac{\gamma}{2} \left(|x_t - x^*|_{A_{t-1}}^2 - |x_{t+1} - x^*|_{A_t}^2 \right) + \frac{1}{2\gamma} \nabla_t A_t^{-1} \nabla_t, \qquad (37)
\end{aligned}
$$

where the first inequality follows from Lemma 4.6 and the second one follows from (36).

Now, note that

$$
\begin{aligned}
\nabla_t A_t^{-1} \nabla_t &= \mathrm{Tr} \left(\nabla_t \nabla_t A_t^{-1} \right) \\
&= \mathrm{Tr} \left((A_t - A_{t-1}) A_t^{-1} \right) \\
&\leq \log \det(A_t) - \log \det(A_{t-1}), \qquad (38)
\end{aligned}
$$

where the inequality follows from the following fact. Since the function $h(A) = \log \det(A)$ is concave with $\nabla h(A) = A^{-1}$ we have

$$
\log \det(A_{t-1}) - \log \det(A_t) \leq \left(\mathrm{vec}(A_{t-1}) - \mathrm{vec}(A_t) \right)^T A_t^{-1} = \mathrm{Tr} \left((A_{t-1} - A_t) A_t^{-1} \right),
$$

where $\mathrm{vec}(A)$ represents the vectorized form of any matrix A.

Thus, from (37) and (38) we have

$$
\begin{aligned}
&f_t(x_t) - f_t(x^*) \\
&\leq \frac{\gamma}{2} \left(|x_t - x^*|_{A_{t-1}}^2 - |x_{t+1} - x^*|_{A_t}^2 \right) + \frac{1}{2\gamma} \log \det(A_t) - \log \det(A_{t-1}), \qquad (39)
\end{aligned}
$$

which allows us to have a desired form of telescoping sum when summing over $t \in [T]$. Therefore, we finally have

$$
\begin{aligned}
&\sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \\
&\leq \frac{\gamma}{2} |x_1 - x^*|_{A_0}^2 + \frac{1}{2\gamma} \left(\log \det(A_T) - \log \det(A_0) \right) \\
&\leq \frac{\gamma \epsilon}{2} |x_1 - x^*|^2 + \frac{1}{2\gamma} \left(d \log \left(\epsilon + \frac{TG^2}{d} \right) - d \log \epsilon \right) \\
&\leq \gamma \epsilon D^2 + \frac{d}{2\gamma} \log \left(1 + \frac{TG^2}{\epsilon d} \right),
\end{aligned}
$$

where the first inequality follows from (39). The second inequality follows from the observations that: (i) since for every $t \in [T]$, $\mathrm{Tr}(\nabla_t \nabla_t^T) = |\nabla_t|^2 \leq G$ under the assumption, by the recursion we have $\mathrm{Tr}(A_T) \leq d\epsilon + TG^2$, and this further yields

$$\begin{aligned}\frac{1}{d}\log\det(A_T) = \frac{1}{d}\sum_{i=1}^{d}\log\lambda_i^{(T)} &\leq \log\left(\frac{1}{d}\sum_i \lambda_i^{(T)}\right) \\ &= \log\left(\frac{1}{d}\,\mathrm{Tr}(A_T)\right) \leq \log\left(\epsilon + TG^2\right),\end{aligned}$$

where $\{\lambda_i^{(T)} : i \in [d]\}$ represents the eigen values of A_T, and (ii) $\log\det(A_0) = d\log\epsilon$. The third inequality holds since by using the assumption we have

$$|x_1 - x^*|^2 \leq 2(|x_1^2| + |x^*|^2) \leq 2D^2.$$

The proof is complete. □

5 Follow the Regularized Leader Algorithm

5.1 Follow the Leader

Let us start with the follow the leader *(FTL) algorithm, whose update scheme is:*

$$x_t = \operatorname*{argmin}_{x \in K} \sum_{i=1}^{t-1} f_i(x).$$

The initial point x_1 can be an arbitrary point in K.

Lemma 5.1. *[FTL Regret Bound] Let $x_1, x_2, \ldots$ be the iterates of FTL. Then for all $u \in K$,*

$$\sum_{t=1}^{T}[f_t(x_t) - f_t(u)] \leq \sum_{t=1}^{T}[f_t(x_t) - f_t(x_{t+1})].$$

Remark 5.1. *We can think of the $f_t(x_t) - f_t(x_{t+1})$ term as measuring the* stability *of the updates. Note that x_{t+1} is the ideal point we should have played at round t. So, essentially the instantaneous regret bound measures how close x_t is to the ideal one step lookahead play x_{t+1} in terms of the loss function x_t.*

Proof. It suffices to show

$$\sum_{t=1}^{T} f_t(x_{t+1}) \leq \sum_{t=1}^{T} f_t(u).$$

We shall do this by induction: it is clear that $f_1(x_2) \leq f_1(u)$. Assume the inequality holds up to $T-1$, i.e.

$$\sum_{t=1}^{T-1} f_t(x_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u).$$

Add $f_T(x_{T+1})$ to both sides, we see that

$$\sum_{t=1}^{T-1}[f_t(x_{t+1}) + f_T(x_{T+1})] \leq \sum_{t=1}^{T-1} f_t(u) + f_T(x_{T+1}) \leq \sum_{t=1}^{T} f_t(x_{T+1}).$$

But by definition of x_{T+1},

$$\sum_{t=1}^{T} f_t(x_{T+1}) \leq \sum_{t=1}^{T} f_t(u)$$

for all u. Hence, we are done. □

Exercise 5.2 (Predict the next vector)**.** *Suppose*

$$f_t(x) = \frac{1}{2}\|x - z_t\|_2^2$$

for an arbitrary sequence of $z_t \in \mathbb{R}^d$. If we run FTL on this, then show that

$$\textit{Regret} \leq 4L^2(1 + \log T),$$

where $L = \max_{1 \leq t \leq T} \|x - z_t\|_2^2$.

FTL can fail on some instances of OCO. The following is an example.

Example (Failure of FTL)**.** *Let $K = [-1, 1]$, $f_t(x) = z_t x$, where*

$$z_t = \begin{cases} -1/2 & t = 1 \\ 1 & t \textit{ even} \\ -1 & t > 1, t \textit{ odd} \end{cases}$$

Then the FTL update reduces to

$$x_t = \begin{cases} 1 & t \textit{ even} \\ -1 & t \textit{ odd} \end{cases}$$

In this case, the total loss will be T, but the best possible loss is 0 (obtained by playing $x_t = 0$ every round). Hence, the regret of FTL is atleast T.

Remark 5.2. *The failure of FTL is due to x_t being unstable, i.e. shifting drastically from round to round. This can be fixed by regularizing the updates as we will now see.*

5.2 Follow the Regularized Leader

Now we consider the follow the regularized leader *(FTRL) algorithm:*

$$x_t = \operatorname*{argmin}_{x \in K} \left[\sum_{i=1}^{t-1} f_i(x) + R(x) \right],$$

where $R(x)$ is some regularization function. The initial point x_1 is a minimizer of R in K.

Since $\nabla f_i(x_i)^\top x$ can be used to estimate f_i at a point x_i, we can also define the linearized *FTRL algorithm:*

$$x_t = \operatorname*{argmin}_{x \in K} \left[\sum_{i=1}^{t-1} \nabla f_i(x_i)^\top x + R(x) \right].$$

The linearized version can be computationally simpler. We will see that we will obtain similar regret for both the versions.

Example. *Suppose $f_t(x) = g_t^\top x$, $R(x) = \|x\|_2^2/(2\eta)$. Then the FTRL updates are given by*

$$x_1 = 0 \quad \textit{and} \quad x_{t+1} = \operatorname*{argmin}_{x \in K} \left[\sum_{i=1}^{T} g_i^\top x + \frac{1}{2\eta}\|x\|_2^2 \right] = -\eta \sum_{i=1}^{t} g_i.$$

Note that this implies $x_{t+1} = x_t - \eta g_t$. That is, ***linearized FTRL with $R(x) = \|x\|_2^2/(2\eta)$ is equivalent to OGD****.*

We next present a regret bound for FTRL.

Lemma 5.3 (FTRL Regret Bound)**.** *Suppose $x_1, x_2, \ldots,$ is the sequence of iterates produced by FTRL. Then for all $u \in K$,*

$$\sum_{t=1}^{T} [f_t(x_t) - f_t(u)] \le R(u) - R(x_1) + \sum_{t=1}^{T} [f_t(x_t) - f_t(x_{t+1})].$$

Remark 5.3. *We can think of the first term $R(u) - R(x_1)$ as the* size *of K relative to R, and $f_t(x_t) - f_t(x_{t+1})$ as the stability term as before. The idea is that the regularization R ensures that x_t is close to the one step lookahead x_{t+1} to give stability to the iterates; we pay an additional cost (compared to FTL) which is the $R(u) - R(x_1)$ term.*

Proof. Run FTL on $f_0 = R$ and $f_1, \ldots, f_T$. Then we can use Lemma 5.1 and obtain

$$\text{Regret} + R(x_0) - R(u) \le S + R(x_0) - R(x_1),$$

where

$$S = \sum_{t=1}^{T} [f_t(x_t) - f_t(x_{t+1})].$$

Set $x_0 = u$ finishes the proof. □

Remark 5.4. *Since it is enough to consider linear loss functions and OGD is an instance of linearized FTRL, we can prove the OGD regret bound (for general convex lipschitz losses) using the previous lemma. To that end, take*

$$g_t(x) = \nabla f_t(x_t)^T x, \qquad R(u) = \frac{1}{2\eta}\|u\|_2^2$$

. Then note that $g_t(x_t) - g_t(x_{t+1}) = \eta\|\nabla f_t(x_t)\|^2$ and $R(x_1) = 0$. Thus the

$$\text{Regret} \le \frac{\|u\|_2^2}{2\eta} + \eta \sum_{t=1}^{T} \|g_t\|_2^2$$

which is exactly the bound for OGD.

5.2.1 Strongly Convex Regularizers

In the N experts problem, we have $K = S_N$ which is a subset of the unit ball and $f_t(x) = \ell_t^\top x$, where we assume $\|\ell_t\|_\infty \leq 1$. OGD in this case gives the regret bound as $O(GD\sqrt{T}) = O(\sqrt{NT})$, where $D = \mathrm{diam}(S_N) = 2$, and $G = \|\ell_t\|_2 \leq \sqrt{N}$. In constrast, the regret with hedge algorithm is $O(\sqrt{\log N}\sqrt{T})$. This suggest the OGD bound is suboptimal in N as compared to the Hedge algorithm. We have already seen that OGD is an instance of FTRL with ℓ_2^2 regularization. We will see in a bit that the Hedge algorithm can also be seen as an instance of FTRL with a different regularizer.

We will now explore some properties of the FTRL (Follow the Regularized Leader) algorithm in the setting where the regularization function is strongly convex. Let us use $|y|$ to represent the norm of a vector y in some vector space ($\mathbb{R}^n$ usually), and $|y|_*$ the dual norm of y, which is defined as

$$|y|_* := \sup_{x:|x|\leq 1} x^T y.$$

Notice that the norm $|\cdot|$ here needn't be the Euclidean norm.

Definition 5.4. *Suppose $f : K \to \mathbb{R}$. Then f is called $L - Lipschitz$ with respect to the norm $|\cdot|$ if for any x, $y \in K$ we have*

$$|f(x) - f(y)| \leq L|x - y|. \tag{40}$$

Lemma 5.5. *Suppose $f : K \to \mathbb{R}$. Then f is $L - Lipschitz$ with respect to the norm $|\cdot|$ if and only if for any $x \in K$ and $z \in \partial f(x)$, we have*

$$|z|_* \leq L, \tag{41}$$

where $\partial f(x)$ is the set of subgradients of f at point x.

The proof of the lemma is left as an exercise. Now we give the definition of a strongly convex function under a given norm $|\cdot|$.

Definition 5.6. *Assume K is a convex set. A convex function $f : K \to \mathbb{R}$ is $\alpha-$strongly convex with respect to norm $|\cdot|$ if for any x,$y \in K$ and $z \in \partial f(x)$, we have*

$$f(y) \geq f(x) + z^T(y - x) + \frac{\alpha}{2}|y - x|^2. \tag{42}$$

Remark 5.5. *If f is twice differentiable, a sufficient condition for f to be $\alpha-$strongly convex with respect to norm $|\cdot|$ is that*

$$(y - x)^t \nabla^2 f(z)(y - x) \geq \alpha$$

for all $x, y, z \in K$.

Example. *The function $f(x) = \frac{\|x\|_2^2}{2}$ is 1 strongly convex w.r.t to the Euclidean norm.*

Example. *Let us use the notation $H(p)$ to denote the entropy of a probability vector $p \in \mathbb{R}^n$, and define the function*

$$R(p) = -\sum_i p_i \log \frac{1}{p_i} = -H(p), \tag{43}$$

on the domain $\mathcal{S}_n$, which is the probability simplex in $\mathbb{R}^n$. Then,

Lemma 5.7. *R is 1 strongly convex on S_n w.r.t the ℓ_1 norm $|\cdot|_1$.*

Proof. It can be shown that this function is $1-$strongly convex with respect to the ℓ_1 norm $|\cdot|_1$. To see this, notice that the Hessian matrix at a point w is diagonal, with the ith diagonal element $1/w_i$, then according to Cauchy-Schwarz inequality and the fact that $\sum_i w_i = 1$ we have

$$p^T \nabla^2 R(w) p = (\sum_{i=1}^n \frac{p_i^2}{w_i}) \sum_i w_i \geq (\sum_{i=1}^n (\frac{p_i}{\sqrt{w_i}} \sqrt{w_i}))^2 = |p|_1^2, \tag{44}$$

□

Now we start to analyze the regret of a FTRL algorithm equipped with a strongly convex regularization function. By Lemma 5.3 we need to control the stability term. This is the content of the next lemma.

Lemma 5.8. *Let the regularization function R be $\alpha-$strongly convex over $\mathcal{K}$ with respect to $|\cdot|$, and assume f_t is L_t- Lipschitz w.r.t. the same norm $|\cdot|$. Then*

$$f_t(x_t) - f_t(x_{t+1}) \leq L_t |x_t - x_{t+1}| \leq \frac{L_t^2}{\alpha}. \tag{45}$$

Proof. Let $F_t(w) = \sum_{i=1}^{t-1} f_i(w) + R(w)$, so that $x_t = \arg\min_{w \in \mathcal{K}} F_t(w)$. Observe that $F_t(w)$ is $\alpha-$ strongly convex since R is. Then we get

$$F_t(x_{t+1}) \geq F_t(x_t) + \nabla F_t(x_t)(x_{t+1} - x_t) + \frac{\alpha}{2}|x_{t+1} - x_t|^2 \geq F_t(x_t) + \frac{\alpha}{2}|x_{t+1} - x_t|^2,$$

where the second inequality is due to the fact that x_t is a minimizer of F_t within K, $x_{t+1} \in K$ and the general first order optimality criterion for the optima. Similarly, at x_{t+1} we have

$$F_{t+1}(x_t) \geq F_{t+1}(x_{t+1}) + \frac{\alpha}{2}|x_t - x_{t+1}|^2.$$

Combine them and we get

$$\alpha |x_t - x_{t+1}|^2 \leq f_t(x_t) - f_t(x_{t+1}) \leq L_t |x_t - x_{t+1}|$$

which implies that

$$|x_t - x_{t+1}| \leq \frac{L_t}{\alpha}.$$

Combining the above with the fact that f_t is L_t lipschitz finishes the proof.

□

The above lemma directly yields the following theorem.

Theorem 5.9 (Regret for FTRL with Strongly Convex Regularizer)**.** *Let the regularization function R be $\alpha-$strongly convex over $\mathcal{K}$ with respect to $|\cdot|$, and assume f_t is L_t- Lipschitz w.r.t. the same norm $|\cdot|$. Assume $L > 0$ is such that $(\sum_{i=1}^{T} L_i^2)/T \leq L^2$.*

$$Regret(u) \leq R(u) - \min_{w \in \mathcal{K}} R(w) + \frac{TL^2}{\alpha}. \tag{46}$$

Proof. The proof directly follows from Lemma 5.3 and Lemma 5.8. □

Let us apply the above theorem to our two running examples.

1. *Firstly, if we choose $R(x) = |x|_2^2/2\eta$ then this is strongly convex w.r.t the Euclidean norm with $\alpha = 1/\eta$, hence we can recover the online gradient descent bound ($\mathcal{O}(\sqrt{T})$) for the FTRL algorithm. Note that this proves a regret bound for the actual FTRL algorithm and not the linearized version, although both the regret bounds are identical.*

2. *Let's consider the prediction with experts advice problem where the domain K is the probability simplex $\mathcal{S}_n$ and let us consider the FTRL algorithm with the regularizer $R(p) = -H(p)/\eta$ which is the negative entropy function. By Lemma 5.7, R is $1/\eta$ strongly convex w.r.t the ℓ_1 norm.*

   Now, the loss functions in this case are simply linear functions $f_t(x) = l_t^t x$ where we can assume that $|l_t|_\infty \leq 1$. We have by Holder's inequality,

   $$|f_t(x) - f_t(y)| = |l_t^T(x-y)| \leq |l_t|_\infty |x-y|_1 \leq |x-y|_1.$$

   This means that f_t is 1 Lipschitz w.r.t the ℓ_1 norm. Hence, we can apply Theorem 5.9.

   So we have the following regret bound:

   $$\begin{aligned} Regret(u) &\leq R(u) - \min_{p \in \mathcal{S}_n}(-\frac{1}{\eta}H(p)) + T\eta \\ &\leq \max_{p \in \mathcal{S}_n}(\frac{1}{\eta}H(p)) + T\eta \\ &= \frac{\log n}{\eta} + T\eta \\ &\leq 2\sqrt{T \log n}. \end{aligned} \tag{47}$$

   where in the second inequality we used $R(u) < 0$, in the equality step we used the fact that entropy is maximized for the uniform distribution equal to $\log n$ and in the last step we plugged in the optimal value of the tuning parameter η. We see that we recover the Hedge bound. This is not a coincidence as ***this FTRL algorithm is actually equivalent to the Hedge algorithm!***

Lemma 5.10. *For the experts advice problem, The FTRL algorithm with the regularizer $R(p) = -H(p)/\eta$ is equivalent to the Hedge algorithm.*

Proof. The proof is left as an exercise. □

Remark 5.6. *We see that both Online Gradient Descent and Hedge are instances of FTRL with different regularization functions.*

6 Online Mirror Descent (OMD)

One possible disadvantage of the FTRL method is that at each round we have to potentially solve a big optimization problem. A natural question is whether we can define updates which are local in nature in the sense that the updates only depend on the information at the current point. The FTRL updates, the way they are defined, consider the entire history to construct the next update. In this section, we consider another online convex optimization algorithm which is called online mirror descent (OGD). This method can be viewed as a generalization of online gradient descent. We will see that this method enjoys similar regret bounds as FTRL but the update step is local.

6.1 Gradient Descent Update as Regularized Optimization

To motivate OMD, let's recall the offline unconstrained gradient descent, with the update rule $x_{t+1} = x_t - \eta \nabla f(x_t)$. We can also view this rule as a solution of the optimization problem

$$\begin{aligned} x_{t+1} &= \arg\min_x f(x_t) + (\nabla f(x_t))^T (x - x_t) + \frac{1}{2\eta}|x - x_t|_2^2 \\ &= \arg\min_x (\nabla f(x_t))^T x + \frac{1}{2\eta}|x - x_t|_2^2. \end{aligned} \tag{48}$$

If the domain is some specified convex set K, then it is not hard to see (left as an exercise!) that the related optimization problem above will give the solution $x_{t+1} = P_{\mathcal{K}}(x_t - \eta \nabla f(x_t))$, where $P_{\mathcal{K}}(\cdot)$ is the Euclidean projection function to $\mathcal{K}$. Then the way to generalize gradient descent to mirror descent is to observation that instead of the penalty $|x - x_t|_2^2/2\eta$ we can use a more general penalty function. Before giving further specifics, we firstly introduce the notion of Bregman divergence.

Definition 6.1. *Assume h is a strictly convex function on convex set K. Then we define the Bregman divergence of h from y to x as*

$$D_h(y|x) := h(y) - h(x) - (\nabla h(x))^T (y - x). \tag{49}$$

Remark 6.1. *From the above definition, the Bregman divergence corresponding to a convex functions is just the difference between $h(y)$ and the value at y of the linear approximation of h at x. We can think of this as defining a notion of distance between y and x although this is not neccessarily symmetric.*

Example. *If $h(x) = |x|_2^2/2$, then $D_h(y|x) = |y-x|_2^2/2$. (Check!)*

Example. *Consider the negative entropy function $h(p) = \sum_i (p_i \log p_i)$ defined on the probability simplex S_n. Then $D_h(q|p) = KL(q,p)$, where $KL(q,p)$ is the Kullback Leibler divergence $KL(q,p) = \sum_i q_i \log \frac{q_i}{p_i}$. (Exercise!)*

From the first example, we see that the penalty function defining the Gradient Descent update $|x-x_t|_2^2/2\eta$ is a Bregman divergence $D_h(x|x_t)$, if we take $h(x) = |x|_2^2/2\eta$. This naturally suggests using a general Bregman divergence as a penalty term which then gives us the way to generalise gradient descent to mirror descent.

Definition 6.2. *Given a convex function $h : K \to \mathbb{R}$ for a convex set $K \subset \mathbb{R}^n$, let us define (projected) Online Mirror Descent with the following update rule:*

$$x_{t+1} = \arg\min_{x \in K} \{\eta(\nabla f(x_t))^T x + D_h(x|x_t)\},$$

To see the rule more explictly, we first consider the unconstrained case $K = \mathbb{R}^n$, which implies that we can take the gradient and set it to zero:

$$\eta \nabla f(x_t) + \nabla h(x) - \nabla h(x_t) = 0 \implies \nabla h(x_{t+1}) := \nabla h(x) = \nabla h(x_t) - \eta \nabla f(x_t).$$

Then to get back x_{t+1}, we just take the inverse map of ∇h (will exist under regularity conditions on h; e.g. strong convexity of h), and write

$$x_{t+1} = (\nabla h)^{-1}(\nabla h(x_t) - \eta \nabla f(x_t)). \tag{50}$$

Regarding our two running examples of convex functions h, it turns out that we recover both OGD and Hedge as the corresponding OMD algorithms. We leave this as an exercise.

Now we give a regret bound for the online mirror descent algorithm.

Theorem 6.3. *[OMD Regret Bound] Assume the function $h : K \to \mathbb{R}$ is $\alpha-$strongly convex on the convex set $\mathcal{K}$ w.r.t. a norm $|\cdot|$. Let $x_0, x_1, \ldots$ be the OMD iterates with step size η. Then we have the upper bound for the regret*

$$\sum (f_t(x_t) - f_t(x^*)) \leq \frac{D_h(x^*|x_1)}{\eta} + \eta \frac{\sum |\nabla f_t(x_t)|_*^2}{2\alpha}, \tag{51}$$

where $|\cdot|_*$ is the dual norm of $|\cdot|$.

Proof. For simplicity, we will only give the proof for the unconstrained case $K = \mathbb{R}^n$.

The proof is a potential based proof where our potential function is going to be $\phi_t = \frac{1}{\eta} D_h(x^*, x_t)$.

Then,

$$\begin{aligned}\phi_{t+1} - \phi_t &= \frac{1}{\eta}\{h(x^*) - h(x_{t+1}) - \nabla h(x_{t+1})(x^* - x_{t+1}) \\ &\quad - h(x^*) + h(x_t) + \nabla h(x_t)(x^* - x_t)\} \\ &= \frac{1}{\eta}(h(x_t) - h(x_{t+1}) + \nabla h(x_t)(x_{t+1} - x_t)) + \nabla f_t(x_t)(x^* - x_{t+1}) \\ &\leq -\frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 + \nabla f_t(x_t)(x^* - x_{t+1}),\end{aligned}$$

where we get the second equality using the update rule (50), and we get the inequality because of the $\alpha-$strongly convexity of h. Now,

$$\begin{aligned}&f_t(x_t) - f_t(x^*) + \phi_{t+1} - \phi_t \\ &\quad\leq f_t(x_t) - f_t(x^*) - \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 + \nabla f_t(x_t)(x^* - x_{t+1}) \\ &\quad= -(f_t(x^*) - f_t(x_t) - \nabla f_t(x_t)(x^* - x_t)) \\ &\qquad + \nabla f_t(x_t)(x_t - x_{t+1}) - \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 \\ &\quad\leq \nabla f_t(x_t)(x_t - x_{t+1}) - \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 \\ &\quad\leq |\nabla f_t(x_t)|_*|x_t - x_{t+1}| - \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 \\ &\quad\leq \frac{\eta}{2\alpha}|\nabla f_t(x_t)|_*^2 + \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 - \frac{\alpha}{2\eta}|x_{t+1} - x_t|^2 \\ &\quad= \frac{\eta}{2\alpha}|\nabla f_t(x_t)|_*^2,\end{aligned} \tag{52}$$

where the first inequality is due to the previous display, the second inequality is due to the convexity of f_t, the third follows by definition of dual norm, the fourth is simply the AM GM inequality.

So we have

$$f_t(x_t) - f_t(x^*) \leq \phi_t - \phi_{t+1} + \frac{\eta}{2\alpha}|\nabla f_t(x_t)|_*^2. \tag{53}$$

Take the sum, and discard the $-\phi_{T+1}$ term, then we get the result. □

Remark 6.2. *The proof in the general constrained case can be obtained similarly as above except that we have to use the general KKT condition characterizing the optima x_{t+1}. This is left as an exercise.*

7 Optimistic FTRL

The results we have seen in Online Learning hold for any sequence of convex loss functions. This makes the results very general. However, this generality comes at the cost that the results are often worst case. Hence, a major theme is to develop adaptive regret bounds that adapt to easier problems. One such easy and possible practically relevant case is when the loss function sequence changes gradually. Such a situation motivates the next algorithm we will study, a variant of FTRL called the Optimistic FTRL. This was proposed by Rakhlin and Sridharan [27].

7.1 Optimistic FTRL: Definition and Regret Bounds

As we know, to derive regret bounds it is enough to consider linear losses. Hence, for this section, we will assume that we are in an Online Linear Optimization (OLO) framework. So, let's say the loss functions are of the form $f_t(x) = l_t^T x$. FTRL updates take the following form:

$$x_t = \arg\min_{x \in K} \sum_{t=1}^{T-1} l_t^T x + \frac{1}{\eta} R(x),$$

where $R(x)$ is a convex function which acts as a regularizer.

Suppose, at the beginning of round t we knew l_t before playing x_t. Then we should play

$$x'_t = \arg\min_{x \in K} \sum_{t=1}^{t-1} l_t^T x + \frac{1}{\eta} R(x) + l_t^T x =: F'_T(x).$$

The above algorithm can be called the one step lookahead algorithm. Now, this one step lookahead algorithm would have regret bounded by $\max_{x \in K} R(x) - \min_{x \in K}$. This can be proved by similar arguments as in the proof of Lemma 5.1. We leave this as an exercise.

This suggests that if we could estimate l_t beforehand we could use the one step lookahead algorithm. Thus, if m_t is a good guess of l_t, we should turn to the optimization problem

$$x_t = \arg\min_{x \in \mathcal{K}} \sum_{t=1}^{t-1} l_t^T x + \frac{1}{\eta} R(x) + m_t^T x =: F_{t-1}(x). \tag{54}$$

This is called optimistic FTRL algorithm (with $m_0 = 0$). We know FTRL has regret $O(\sqrt{T})$. We desire that FTRL algorithm would have smaller regret when $m_t \approx l_t$ but not be too much worse than FTRL if m_t is not predictive of l_t. In fact, we have the following result.

Theorem 7.1. *The updates $\{x_t\}_{t=1}^T$ by optimistic FTRL satisfies for any $u \in K$,*

$$\sum_{t=1}^{T} (x_t - u)^T l_t \leq \underbrace{\frac{R(u)}{\eta} - \min_{x \in K} \frac{R(x)}{\eta}}_{RangeTerm} + \underbrace{\sum_{t=1}^{T} (x_t - x'_t)^T (l_t - m_t)}_{StabilityTerm} - \underbrace{\sum_{t=1}^{T} \frac{D_R(x_t, x'_{t-1}) + D_R(x'_t, x_t)}{\eta}}_{CurvatureGainTerm}$$

for any $u \in \mathcal{K}$. Here, D_R is the Bregman divergence associated with R.

Remark 7.1. *The above bound holds for any convex regularizer R. Later on, we will see a more streamlined result when R is strongly convex. The main thing to look at in the above bound is the presence of $l_t - m_t$ in the second term. This indicates a better bound when m_t is close to l_t. The third term involves Bregman divergences and is negative. So we could drop this term for the moment although this term will be crucial for us when we look at applying this result to zero sum games. This term can be thought of as the gain due to the curvature of R.*

Note that if we set $m_t = 0$ then x_t becomes the usual FTRL play. Then $x'_t = x_{t+1}$ and hence we can use the above theorem to derive a regret bound for usual FTRL which is a slightly improved version of Lemma 5.3.

Corollary 7.2. *The updates $\{x_t\}_{t=1}^T$ of usual (linearized) FTRL satisfies for any $u \in K$,*

$$\sum_{t=1}^{T}(x_t - u)^T l_t \leq \underbrace{\frac{R(u)}{\eta} - \min_{x\in K}\frac{R(x)}{\eta}}_{RangeTerm} + \underbrace{\sum_{t=1}^{T}(x_t - x_{t+1})^T l_t}_{StabilityTerm} - \underbrace{\sum_{t=1}^{T}\frac{D_R(x_{t+1}, x_t)}{\eta}}_{CurvatureGainTerm}$$

for any $u \in \mathcal{K}$.

To prove Theorem 7.1, we need to introduce several lemmas.

Lemma 7.3 (Stability)**.** *For $i = 1, 2$, let $F_i(x) = L_i^T x + R(x)$ with optimizer and optimal values (x_i^*, V_i^*) where R is a convex function defined on K. Then, we have*

$$V_2^* - V_1^* \leq (L_2 - L_1)^T x_1^* - D_R(x_1^*, x_2^*),$$

This is a stability of optimal value result. Moreover, if R is $1/\eta$ strongly convex w.r.t a norm $\|.\|$, then we have

$$\|x_1^* - x_2^*\|^2 \leq \eta(x_1^* - x_2^*)^T(L_2 - L_1)$$

and hence

$$\|x_1^* - x_2^*\| \leq \eta\|(L_2 - L_1)\|_*$$

which shows the stability of the optimizer under strong convexity. Here $\|.\|_*$ represents the dual norm of $\|.\|$.

Proof. By convexity of F and the first order optimality criterion, we have for any $x \in K$,

$$F_i(x) - F_i(x_i^*) - D_{F_i}(x, x_i^*) = \nabla F_i(x_i^*)^T(x - x_i^*) \geq 0.$$

This implies

$$\begin{aligned} V_2^* - V_1^* &= F_2(x_2^*) - F_1(x_1^*) \\ &\leq F_2(x_1^*) - D_{F_2}(x_1^*, x_2^*) - F_1(x_1^*) \\ &= (L_2 - L_1)^T x_1^* - D_R(x_1^*, x_2^*). \end{aligned}$$

This shows the stability of the optimal value. Now, in case R is $1/\eta$ strongly convex, we have $D_R(y, x) \geq \frac{1}{2\eta}\|y - x\|^2$. Plugging this in the above inequality we get,

$$V_2^* - V_1^* \leq (L_2 - L_1)^T x_1^* - \frac{1}{2\eta}\|x_1^* - x_2^*\|^2.$$

By symmetry, we also get the inequality

$$V_1^* - V_2^* \leq (L_1 - L_2)^T x_2^* - \frac{1}{2\eta}\|x_1^* - x_2^*\|^2.$$

Add the last two displays to obtain

$$\|x_1^* - x_2^*\|^2 \leq \eta(x_1^* - x_2^*)^T(L_2 - L_1).$$

□

Now, we are ready to prove our main regret bound.

Proof of Theorem 7.1. The main theme of this proof is to compare the one step lookahead updates with the optimistic FTRL updates. To this end, let us define the functions

$$F_{t-1}(x) = (\sum_{i=1}^{t-1} l_i + m_t)^t x + \frac{R(x)}{\eta}.$$

$$F_t^{'}(x) = (\sum_{i=1}^{t-1} l_i + l_t)^t x + \frac{R(x)}{\eta}.$$

Then, recalling the definition of x_t and x_t' we see that they are the optimizers of F_{t-1} and $F_t^{'}$. Let us also denote the optimal values of F_t and $F_t^{'}$ by V_t and $V_t^{'}$ respectively.

By using the stability of optimal value result in Lemma 7.3 we have

$$V_{t-1}' - V_{t-1} \leq -m_t^T x_t - \frac{1}{\eta} D_R(x_t, x_{t-1}')$$
$$V_{t-1} - V_t' \leq (m_t - l_t)^T x_t' - \frac{1}{\eta} D_R(x_t', x_t).$$

Summing these two inequalities yields

$$l_t^T x_t \leq V_t' - V_{t-1}' + (m_t - l_t)^T(x_t' - x_t) - \frac{D_R(x_t, x_{t-1}') + D_R(x_t', x_t)}{\eta}.$$

This implies

$$\sum_{t=1}^{T} l_t^T x_t \leq V_T^{'} - V_0^{'} + \sum_{t=1}^{T}(m_t - l_t)^T(x_t' - x_t) - \sum_{t=1}^{T} \frac{D_R(x_t, x_{t-1}') + D_R(x_t', x_t)}{\eta}$$

Now, by definition,

$$V_T^{'} - V_0^{'} = \sum_{t=1}^{T} l_i^T x_t^{'} + \frac{R(x_t^{'})}{\eta} - \min_{x \in K} \frac{R(x)}{\eta} \leq \sum_{t=1}^{T} l_i^T u + \frac{R(u)}{\eta} - \min_{x \in K} \frac{R(x)}{\eta}$$

for any $u \in K$.

Combining the last two displays finishes the proof. □

Now let's state a corollary of Theorem 7.1 in case R is 1 strongly convex.

Corollary 7.4. *The updates $\{x_t\}_{t=1}^T$ by optimistic FTRL with a 1 strongly convex (w.r.t a norm $\|.\|$) regularizer K satisfies for any $u \in K$,*

$$\sum_{t=1}^{T}(x_t - u)^T l_t \leq \sum_{t=1}^{T}(x_t - u)^T l_t \leq \underbrace{\frac{R(u)}{\eta} - \min_{x \in K}\frac{R(x)}{\eta}}_{RangeTerm} + \underbrace{\eta \sum_{t=1}^{T}\|l_t - m_t\|_*^2}_{StabilityTerm} - \underbrace{\frac{1}{4\eta}\sum_{t=2}^{T}\|x_t - x_{t-1}\|^2}_{CurvatureGainTerm}$$

Proof. To bound the stability term from Theorem 7.1 we just use the stability of optimizer result from Lemma 7.3. To bound the gain term due to curvature, note that

$$\sum_{t=1}^{T}\frac{D_R(x_t, x_t') + D_R(x_t, x_{t+1}')}{\eta} \geq \frac{1}{2\eta}\sum_{t=1}^{T}(\|x_t - x_t'\|^2 + \|x_t - x_{t+1}'\|^2) \geq$$
$$\frac{1}{2\eta}\sum_{t=2}^{T}\|x_t - x_t'\|^2 + \frac{1}{2\eta}\sum_{t=1}^{T-1}\|x_t - x_{t+1}'\|^2 = \frac{1}{2\eta}\sum_{t=2}^{T}(\|x_t - x_t'\|^2 + \|x_t' - x_{t-1}\|^2) \geq$$
$$\frac{1}{4\eta}\sum_{t=2}^{T}(\|x_t - x_t'\| + \|x_t' - x_{t-1}\|)^2 \geq \frac{1}{4\eta}\sum_{t=1}^{T}(\|x_t - x_{t-1}\|)^2$$

where in the first inequality we used the strong convexity of R, in the second inequality we dropped some terms and reindexed the summation, in the third inequality we used $a^2 + b^2 \geq 1/2(a+b)^2$ and in the fourth inequality we used the triangle inequality. □

Remark 7.2. *Ignoring the curvature gain term for now, we see that in Corollary 7.4, by setting η optimally, we get a regret bound of order $O(\sqrt{Range\sum_{t=1}^{T}\|l_t - m_t\|_*^2})$. If $\|l_t\|_*$ is bounded and hence $\|m_t\|_*$ is also bounded then we see that in the worst case we get a $O(\sqrt{T})$ regret bound with atmost a constant factor worse off. However, when m_t is predictive of l_t, that is when $\sum_{t=1}^{T}\|l_t - m_t\|_*^2 = o(T)$ we can get a much better bound. Similar to before, one can set a time varying tuning parameter $\eta_t = \sqrt{\frac{R}{\sum_{i=1}^{t-1}\|l_i - m_i\|_*^2}}$ and still obtain a similar guarantee as in Corollary 7.4.*

Remark 7.3. *What can we set m_t to be? We can in fact think of this problem as another online learning problem where we can set the loss function to be $\|l_t - m_t\|_*^2$. So one can use any online learning algorithm to set m_t. One natural choice is to set $m_t = l_{t-1}$. Then we obtain a regret bound of order $O(\sqrt{Range\sum_{t=1}^{T}\|l_t - l_{t-1}\|_*^2})$. The quantity $\|l_t - l_{t-1}\|_*^2$ is called the path length variation of the loss vectors l_t. We will use this choice of m_t and the full strength (including the curvature gain term) of Corollar 7.4 in Section 7.3.*

7.2 Application to Zero Sum Games

A zero sum two player game is a game defined by a payoff matrix $G \in [0,1]^{m \times n}$ where we have assumed the $[0,1]$ range for simplicity. There are m possible actions for the row player and n possible actions for the column player. For each pair of actions (i,j) the row player incurs a loss

(and the column player enjoys a gain) given by $G[i,j]$. For example, in the rock paper scissors game the payoff matrix G could be of the following form:

$$G_{3\times 3} = \begin{bmatrix} 1/2 & 1 & 0 \\ 0 & 1/2 & 1 \\ 1 & 0 & 1/2 \end{bmatrix}$$

where the actions (for both the players) are Rock, Paper and Scissors respectively.

Remark 7.4. *Not all two player games are zero sum games. In general, both players have a payoff matrix G_1, G_2 representing their losses respectively. Zero sum games are special cases where $G_2 = -G_1$. In this section, we will stick to discussing zero sum games only.*

The players might play a mixed strategy over the possible actions which is represented by a probability vector. The interpretation is that the row player samples from his/her possible actions following a distribution and similarly the column player does so independently of the row player.. For example, the row player can play a mixed strategy given by $p \in \Delta_m$ and the column player can play a mixed strategy given by $q \in \Delta_n$. Then the expected loss for the row player is $G(p,q) = p^T G q$.

Definition 7.5. *The pair of mixed strategies (p,q) is called the Nash equilibrium if $G(p,q') \leq G(p,q) \leq G(p',q), \ \ \forall p' \in \Delta_m, \forall q' \in \Delta_n$.*

For instance, in the rock paper scissor game discussed in the last lecture, one can verify that the unique Nash equilibrium is $p = q = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

Another natural concept in Games is called the minimax solution (or maximin solution).

Definition 7.6.

$$p^* = \operatorname*{argmin}_p \max_q G(p,q)$$
$$q^* = \operatorname*{argmax}_q \min_p G(p,q)$$

Here p^* and q^* are the least loss and largest reward respectively that the players can hope for if the other player plays second and plays optimally. (p^*, q^*) together is called a minimax solution of the game.

It is intuitive that playing second should be advantageous which can be shown by the following lemma.

Lemma 7.7.

$$\min_p \max_q G(p,q) \geq \max_q \min_p G(p,q)$$

Proof. The L.H.S is the least possible loss if the row player plays first and the R.H.S is the least possible loss if the row player plays second. Therefore, this lemma should hold. We must have for all p, q,

$$G(p, q) \geq \min_p G(p, q).$$

Therefore, taking max over q now gives for all p,

$$\max_q G(p, q) \geq \max_q \min_p G(p, q).$$

Finally, taking the minimum over p on the L.H.S above finishes the proof.

□

Perhaps surprisingly, the reverse inequality also holds for the zero sum games and the two values are exactly equal for a zero sum game. This is called the Von Neumann's minimax theorem.

Theorem 7.8 (Von Neumann's Minimax Theorem)**.** *For any two player zero sum game with $G \in [0, 1]^{N \times M}$, we have*

$$\min_p \max_q G(p, q) = \max_q \min_p G(p, q)$$

We will prove Von-Neumann's minimax theorem using online learning in a bit, but for now let us discuss the relationship between the Nash equilibrium and the minimax solution.

Theorem 7.9. *A pair of mixed strategies (p^*, q^*) is a Nash equilibrium if and only if it is also a minimax solution and $G(p^*, q^*)$ is called the value of the game.*

Proof. Let (p^*, q^*) be a Nash equilibrium, then by definition of Nash equilibrium, and optimality, we have

$$\min_p \max_q G(p, q) \leq \max_q G(p^*, q) = G(p^*, q^*) = \min_p G(p, q^*) \leq \max_q \min_p G(p, q)$$

Now, by the minimax theorem 7.8, the above inequalities are actually equalities. Thus, (p^*, q^*) is also the minimax solution.

For the reverse direction, suppose (p^*, q^*) is a minimax solution. Then,

$$\min_p \max_q G(p, q) = \max_q G(p^*, q) \geq G(p^*, q^*) \geq \min_p G(p, q^*) = \max_q \min_p G(p, q).$$

By the minimax theorem again, the above inequalities are actually equalities which mean that (p^*, q^*) is a Nash equilibrium. □

We are now going to provide a proof of the minimax theorem. The original proof is quite different and relies on fixed point theorems. The following proof uses online learning techniques.

Proof of Von Neumann's Minimax Theorem. It is enough to prove that $\min_p \max_q G(p,q) \leq \max_q \min_p G(p,q)$. From the row player's point of view, we can run the following repeated game:
For rounds $t = 1, 2, ..., T$,

1. Row player plays p_t.
2. Row player observes the loss vector $l_t = Gq_t \in \mathbb{R}^m$.
3. Row player incurs loss $l_t.p_t = G(p_t, q_t)$

The row player in any round has m possible actions to randomize and choose from. To do so the row player can use any online learning algorithm. To be concrete, let's say that the row player uses the Hedge algorithm. Suppose the online algorithm has regret $Row\ regret$, then it implies that

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} G(p_t, q_t) &= \min_p \frac{1}{T}\sum_{t=1}^{T} G(p, q_t) + \frac{Row\ regret}{T} \\ &= \min_p G(p, \bar{q}) + \frac{Row\ regret}{T} \quad \text{(Define } \bar{q} := \frac{1}{T}\sum_{t=1}^{T} G(p, q_t)\text{)} \\ &\leq \max_q \min_p G(p, q) + \frac{Row\ regret}{T} \end{aligned} \tag{55}$$

Thus, if the regret is sublinear, then for very large T, the average loss of the row player is close to the value of the game.

Similarly, from the column player's point of view, since $-G$ gives the loss we have

$$\frac{1}{T}\sum_{t=1}^{T} -G(p_t, q_t) = \min_q \frac{1}{T}\sum_{t=1}^{T} -G(p_t, q) + \frac{Col\ regret}{T}.$$

Therefore, we can write

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} G(p_t, q_t) &= \max_q \frac{1}{T}\sum_{t=1}^{T} G(p_t, q) - \frac{Col\ regret}{T} \\ &= \max_q G(\bar{p}, q) - \frac{Col\ regret}{T} \quad \text{(Define } \bar{p} := \frac{1}{T}\sum_{t=1}^{T} G(p_t, q)\text{)} \\ &\geq \min_p \max_q G(\bar{p}, q) - \frac{Col\ regret}{T}. \end{aligned} \tag{56}$$

Combining Eq. 55 and 56, and letting $T \to \infty$ we obtain

$$\min_p \max_q G(p,q) \leq \max_q \min_p G(p,q)$$

which when combined with Lemma 7.7 finishes the proof. □

Remark 7.5. *Running Hedge for both the row player and the column player actually provides us an algorithm to find an approximate Nash equilibrium. This can be seen by combining Eq. 55 and 56, and defining* $\epsilon_{row} := \frac{Row\ regret}{T}$, $\epsilon_{col} := \frac{Col\ regret}{T}$ *to get*

$$\begin{aligned}\min_p G(p, \bar{q}) + \epsilon_{row} &= \max_q G(\bar{p}, q) - \epsilon_{col} \\ \max_q G(\bar{p}, q) &= \min_p G(p, \bar{q}) + \epsilon \ \ (\epsilon := \epsilon_{row} + \epsilon_{col}) \\ &\leq G(\bar{p}, \bar{q}) + \epsilon\end{aligned}$$

Similarly,

$$\begin{aligned}\min_p G(p, \bar{q}) &= \max_q G(\bar{p}, q) - \epsilon \\ &\geq G(\bar{p}, \bar{q}) - \epsilon\end{aligned}$$

Thus, $\bar{p}, \bar{q}$ *is an* ϵ*-approximate Nash equilibrium pair. Since the Nash equilibrium and Minimax solutions are identical for a zero sum game, we get approximate mini-max solution too through online learning. Note that by the regret bound for Hedge (2.4), we have* $\epsilon = O(1/\sqrt{T})$. ***In the next section, we will see that we can actually make the convergence rate as fast as*** $O(1/T)$ ***if we use Optimistic Hedge instead of Hedge****.*

7.3 When Both Players run Optimistic Hedge Algorithm

Now let us consider the case when both the row player and the column player uses the following Optimistic Hedge algorithm to choose p_t, q_t *in every round,*

$$p_t(i) \propto \exp\left(-\eta\left(l_{t-1}(i) + \sum_{s<t} l_s(i)\right)\right)$$
$$q_t(i) \propto \exp\left(-\eta\left(l'_{t-1}(i) + \sum_{s<t} l'_s(i)\right)\right)$$

where $l_s(i) = G(i, q_s), l'_s(i) = G(p_s, i)$*. As we have seen before,* p_t *are the updates ofOptimistic FTRL,where* K *is the probability simplex in* $\mathbb{R}^m$*, the regularizer is* $1/\eta$ *times the negative entropy function which is* 1 *strongly convex w.r.t the* ℓ_1 *norm; see Lemma 5.7, and* $m_t = l_{t-1}$*. Similar statement holds for* q_t*.*

Theorem 7.10 (Fast Rate for Convergence to Nash Equlibrium)**.** *In a zero sum game with payoff matrix* $G \in [0, 1]^{m \times n}$*, if both plays optimistic hedge as defined above, then with* $\eta = \frac{1}{4}$*, the total regret (look at the notation in Remark 7.5):*

$$\epsilon = \epsilon_{row} + \epsilon_{col} = O\left(log(NM)\right)$$

and hence $(\bar{p}, \bar{q})$ *is an approximate Nash Equilibrium with error* $O\left(\frac{log(NM)}{T}\right)$.

Proof. By Corollary 7.4 and the fact that ℓ_∞ is the dual norm to ℓ_1 we have the following regret bound for the row player:

$$\epsilon_{row} \leq \frac{\log M}{\eta} + \eta \sum_{t=1}^{T} |l_t - l_{t-1}|_\infty^2 - \frac{1}{4\eta} \sum_{t=1}^{T} |p_t - p_{t-1}|_1^2.$$

Now, here the negative term above in the regret bound will actually come in handy. Note that we have

$$\begin{aligned} |l_t - l_{t-1}|_\infty &= \max_i |G(i, q_t) - G(i, q_{t-1})| \\ &= \max_i |G[i, \cdot]^T (q_t - q_{t-1})| \\ &\leq |q_t - q_{t-1}|_1. \end{aligned}$$

Therefore, by taking $l_0 = 0$,

$$T\epsilon_{row} \leq \frac{\log M}{\eta} + \eta + \eta \sum_{t=2}^{T} |q_t - q_{t-1}|_1^2 - \frac{1}{4\eta} \sum_{t=1}^{T} |p_t - p_{t-1}|_1^2. \tag{57}$$

Exactly the same argument applies to the column player, so we get

$$T\epsilon_{col} \leq \frac{\log N}{\eta} + \eta + \eta \sum_{t=2}^{T} |p_t - p_{t-1}|_1^2 - \frac{1}{4\eta} \sum_{t=1}^{T} |q_t - q_{t-1}|_1^2 \tag{58}$$

We can now sum (57) and (58), to get

$$\epsilon \leq \frac{\log MN}{\eta} + 2\eta + (\eta - \frac{1}{4\eta}) \sum_{t=2}^{T} \left(|p_t - p_{t-1}|_1^2 + |q_t - q_{t-1}|_1^2 \right)$$

Setting $\eta = \frac{1}{4}$, we have

$$\epsilon \leq 4 \log MN - \frac{1}{4} \sum_{t=2}^{T} \left(|p_t - p_{t-1}|_1^2 + |q_t - q_{t-1}|_1^2 \right) \tag{59}$$

and now we can drop the negative term to finish the proof. □

Remark 7.6. *Actually one can show that both $\epsilon_{row}, \epsilon_{col}$ are $O(\log(MN))$ which does not follow from their sum being $O(\log(MN))$ as one of them could be negative. To see this, first note that*

$$\epsilon = \max_q G(\bar{p}, q) + \min_q G(\bar{p}, q) \geq G(\bar{p}, \bar{q}) - G(\bar{p}, \bar{q}) \geq 0$$

where the first equality is by the second display in Remark 7.5. Therefore, combining this with (59), we get that

$$\sum_{t=2}^{T} |p_t - p_{t-1}|_1^2 + |q_t - q_{t-1}|_1^2 \leq O(\log MN).$$

This confirms our intuition that the plays of the row player or the column player enjoy an inherent stability property; the total movement of the plays is bounded by $O(1)$ in the above sense. This is precisely why we can hope to have faster rate of convergence than $O(1/\sqrt{T})$. The above display can now be combined with (57) to get $\epsilon_{row} = O(\log MN)/T$ and one can argue similarly for ϵ_{col}.

8 Online Learning in Non-Stationary Environments

So far, we have used the notion of regret

$$Regret = \sum_t f_t(x_t) - \min_{x \in K} \sum_t f_t(x).$$

Here, the regret is small means that our total loss is small compared to the best action in hindsight. However, in cases when there is no overall good single action this notion of regret is not meaningful. Consider the following example:

Example 8.1. *There are* 2 *experts.*

- *expert 1:*
  - *incurs 0 loss for first half* $\frac{T}{2}$
  - *incurs 1 loss for second half* $\frac{T}{2}$
- *expert 2:*
  - *incurs 1 loss for first half* $\frac{T}{2}$
  - *incurs 0 loss for second half* $\frac{T}{2}$

We see that the overall loss of both the experts is $T/2$*. If we run the Hedge algorithm, the regret guarantee in Theorem* (2.4) *then ensures that our total loss is at most* $T/2 + O(\sqrt{T})$ *which is rather disappointing. Infact, this is not just a case of our upper bound being loose. Note that the cumulative loss of the second expert is always more than that of the first expert. Therefore, the Hedge algorithm will put weight atleast* $1/2$ *to the first expert always.This means that in the last* $T/2$ *rounds, the expected loss per round is atleast* $1/2$ *which gives atleast* $T/4$ *loss in expectation.*

Ideally in a case like above, we would like an algorithm to have small regret on all intervals simultaneously. Given an algorithm $\mathcal{A}$*, for an interval* $I \subset [T]$*, define*

$$Regret_{\mathcal{A}}(I) = \sum_{t \in I} f_t(x_t) - \min_{x \in K} \sum_{t \in I} f_t(x)$$

We would like

$$Regret_{\mathcal{A}}(I) \leq O(\sqrt{|I|}) \tag{60}$$

for all intervals I *simultaneously.*

Question: Does an algorithm exist which satisfies* (60)*? The answer is yes!

If we could obtain such an algorithm, then in our example above we can ensure that in both the first half and the second half of the rounds, the regret is separately of $O(\sqrt{T})$*. This means the total loss of the algorithm is also of* $O(\sqrt{T})$*.*

Intuitively, the problem faced by the Hedge algorithm in our example is that it considers the entire history. We need an algorithm that quickly recognizes that the best expert has changed. If we

knew the exact time of change, we could restart the Hedge algorithm, forget the past history and this would serve our purpose.

A natural idea might be to consider the Hedge algorithm restarted from all possible time points $r \in [T]$. Let's call these algorithms $\{\mathcal{A}_r : r \in [T]\}$. Then, in round t, before playing we will have t algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_t$ providing their predictions. We can consider these as experts and try to aggregate their predictions. Thus, we are faced with another layer of an expert problem. The only difference is that the number of experts keeps growing in every round. It turns out we can handle such cases by a general framework called the Sleeping Experts framework. We will see that the general Sleeping Experts Aggregation Algorithm (SAA) will satisfy (60). This is the content of the next section.

8.1 Sleeping Experts Framework

The sleeping experts framework is as follows. This is a variant of the experts problem and was introduced by Freund et al. [13]. Here, the main difference with the usual experts setup is that each expert can abstain to provide any prediction.

Consider an Online Convex Optimization problem with action space K. Let there be a set of N experts. The protocol is as follows. For every round $t \in [T]$,

- *Each expert $S \in [N]$ either provides a prediction $x_{S,t} \in K$ or abstains. Let $A_t \subset [N]$ define the set of active experts in round t; those who provide their predictions.*
- *Then the learner plays x_t based on the recommendations of the active experts.*
- *The loss function $f_t : K \to \mathbb{R}$ is revealed and the learner incurs the loss $f_t(x_t)$.*

In this framework, we desire that we suffer as small a regret as possible against any expert on the rounds that it was active. Specifically, we define

Definition 8.2. *Let $R(S)$ denote the regret against expert S defined as follows:*

$$R(S) = \sum_{t:S \in A_t} f_t(x_t) - f_t(x_{S,t})$$

In this framework, we would like $R(S)$ uniformly small for all experts S.

8.2 Sleeping Experts Algorithm

The sleeping experts aggregation algorithm (SAA) is defined as follows; this version is based on [11]. The algorithm maintains a probability distribution $w(1)$ over the experts. Initialize with $w_{S,1} = 1/N$ for all experts S.

For every round $t = 1, \ldots, T$,

1. *Nature/Adversary reveals the set of active experts A_t.*

2. *The learner plays*
$$x_t = \sum_{s \in A_t} \frac{w_{S,t}}{\sum_{s \in A_t} w_{S,t}} x_{S,t}.$$

3. *Nature/Adversary reveals the loss function f_t.*

4. *Update the weights so that*

   - $$w_{S,t+1} = w_{S,t} \quad \forall S \notin A_t.$$

   - $$w_{S,t+1} = \frac{w_{S,t} \exp(-\alpha f_t(x_{S,t}))}{\sum_{S \in A_t} w_{S,t} \exp(-\alpha f_t(x_{S,t}))} \sum_{s \in A_t} w_{S,t} \quad \forall S \in A_t$$

To describe the algorithm in words, in every round the learner maintains a probability weight vector for the experts. In each round t, the learner plays the conditional mean of the experts predictions; conditioned on the active experts A_t. After observing the loss of the active experts in that round, then the weights of those active experts are updated with a multiplicative exponential factor and renormalizing so that the total weight of the active experts remain the same. The following is the main regret guarantee for the SAA algorithm.

Theorem 8.3 (SAA Algorithm Competitive against Any Expert: General Convex Losses)**.** *Consider the sleeping experts setting. Suppose the loss functions f_t are convex and take values in $[0, M]$. Then for the SAA algorithm with tuning parameter α, we have the regret bound against any fixed expert S,*
$$\sum_{t: S \in A_t} f_t(x_t) \leq \frac{\alpha M}{1 - \exp(-\alpha M)} \sum_{t: S \in A_t} f_t(x_{S,t}) + \frac{M \log N}{1 - \exp(-\alpha M)}.$$

Proof. Fix an expert S and a a round $1 \leq t \leq T$ where expert $S \in A_t$ is active. Then we can write:

$$\begin{aligned}
&log(w_{S,t+1}) - log(w_{S,t}) = -log(w_{S,t}) + log(\frac{w_{S,t} e^{-\alpha f_t(x_{S,t})}}{\sum_{S \in A_t} e^{-\alpha f_t(x_{S,t})} w_{S,t}}) + log(\sum_{S \in A_t} w_{S,t}) \\
&\geq -\alpha f_t(x_{S,t}) - log(\sum_{S \in A_t} w_{S,t}[1 - (1 - e^{-\alpha M})\frac{f_t(x_{S,t})}{M}]) + log(\sum_{S \in A_t} w_{S,t}) \\
&= -\alpha f_t(x_{S,t}) - log(1 - C_M \frac{\sum_{S \in A_t} f_t(x_{S,t}) w_{S,t}}{\sum_{S \in A_t} w_{S,t}}) \\
&\geq -\alpha f_t(x_{S,t}) - log(1 - C_M f_t(x_t)) \\
&\geq -\alpha f_t(x_{S,t}) + C_M f_t(x_t)
\end{aligned} \tag{61}$$

where $C_M = \frac{1 - e^{-\alpha M}}{M}$.

In the above, the first inequality follows from using the following inequality which holds for all $x \in [0, M]$,

$$e^{-\alpha x} \leq 1 - (1 - e^{-\alpha M})\frac{x}{M}.$$

The best way to see it is perhaps that by rearranging to get

$$\frac{x}{M} \leq \frac{1 - e^{-\alpha x}}{1 - (1 - e^{-\alpha M})}$$

and then noting that both the L.H.S and the R.H.S are 0 at 0 and 1 at M; the L.H.S is a linear function and the R.H.S is a concave function.

In (61), the second inequality follows due to convexity of f_t and Jensen's inequality; the third inequality follows by using the elementary inequality $-log(1 - x) \geq x \quad \text{for} \quad 0 < x < 1$.

Now, we can sum (61) over all the rounds $\{t : S \in A_t\}$ to obtain,

$$\sum_{t:S\in A_t} f_t(x_t) \leq \frac{1}{C_M}\sum_{t=1}^{T}(log(w_{S,t+1}) - log(w_{S,t})) + \frac{\alpha}{C_M}\sum_{t:S\in A_t} f_t(x_{S,t}) \leq \frac{M \log N}{1 - \exp(-\alpha M)} + \frac{\alpha M}{1 - \exp(-\alpha M)}\sum_{t:S\in A_t} f_t(x_{S,t}).$$

In the first inequality above, we used the fact that $w_{S,t+1} = w_{S,t}$ for t such that $S \notin A_t$ and in the second inequality we telescoped the sum and used the fact that w_1 is the uniform distribution over the set of all experts. □

Remark 8.1. *Note that the constant $\frac{\alpha M}{1-\exp(-\alpha M)}$ is always bigger than 1. We can make this constant as close to 1 as possible by taking α close to 0 with the caveat that the coefficient of $\log N$ blows up as $1/\alpha$. We can make this constant actually equal to 1 if we additionally assume the loss functions f_t are exp-concave. This is the content of the next theorem.*

Theorem 8.4 (SAA Algorithm Competitive against Any Expert: Exp-Concave Losses)**.** *Consider the sleeping experts setting. Suppose the loss functions f_t are convex and take values in $[0, M]$. Then for the SAA algorithm with tuning parameter α, we have the regret bound against any fixed expert S,*

$$\sum_{t:S\in A_t} f_t(x_t) \leq \sum_{t:S\in A_t} f_t(x_{S,t}) + \frac{\log N}{\alpha}.$$

Proof. Let us fix any expert S and any time t such that $S \in A_t$. By exp-concavity of f_t we can start with

$$e^{-\alpha f_t(x_t)} \geq \sum_{S\in A_t} \hat{w}_{S,t} e^{-\alpha f_t(x_{S,t})}$$

where $\hat{w}_{S,t} = w_{S,t}/\sum_{S\in A_t} w_{S,t}$. Taking log,

$$\alpha f_t(x_t) \leq -log(\sum_{S\in A_t} \hat{w}_{S,t} e^{-\alpha f_t(x_{S,t})})$$

and equivalently,

$$
\begin{aligned}
f_t(x_t) - f_t(x_{S,t}) &\leq \frac{1}{\alpha}[-log(\sum_{S \in A_t} \hat{w}_{S,t} e^{-\alpha f_t(x_{S,t})}) + log(e^{-\alpha f_t(x_{S,t})})] \\
&= \frac{1}{\alpha} \log \frac{e^{-\alpha f_t(x_{S,t})}}{\sum_{S \in A_t} \hat{w}_{S,t} e^{-\alpha f_t(x_{S,t})}} \\
&= \frac{1}{\alpha} log(\frac{w_{S,t+1}}{w_{S,t}})
\end{aligned} \tag{62}
$$

Summing over t such that $S \in A_t$ and noting that $w_{S,t} = w_{S,t+1}$ for t such that $S \notin A_t$ we get,

$$
\sum_{t:S \in A_t} [f_t(x_t) \ - f_t(x_{S,t})] \leq \sum_{t=1}^{T} \frac{1}{\alpha} log(\frac{w_{S,t+1}}{w_{S,t}}) = \frac{1}{\alpha}(log(w_{S,T+1}) - log(w_{S,1})) \leq \frac{1}{\alpha} \log N.
$$

□

8.3 Sleeping Experts as a Meta Algorithm

The sleeping experts (SAA) algorithm can be thought of as a meta algorithm which can be used on top of any OCO algorithm $\mathcal{A}$. Suppose we have an OCO problem where $\mathcal{A}$ is a good algorithm like the Hedge algorithm for the vanilla experts problem. We can now use the SAA algorithm to guarantee (60). For any interval $I = [e, s] \subset [T]$, let us denote $\mathcal{A}_{[e,s]}$ to denote the algorithm $\mathcal{A}$ started from time point r, i.e, it starts predicting from time e onwards and only uses data from time e onwards as well and is only run till time s. This can be framed in the sleeping experts framework where $N = T^2$ and experts correspond to algorithms $\{A_I : \text{intervals } I \subset [T]\}$. At round $1 \leq t \leq T$, the active set consist of experts corresponding to intervals $A_t = \{I : t \in I\}$. Then, the regret guarantees in Theorems 8.3, 8.4 establish that for any interval $I \subset [T]$,

$$
\sum_{t=e}^{s} f_t(x_t) \leq Const. \sum_{t=r}^{s} f_t(x_{\mathcal{A}_{[e,s]},t}) + O(\log T).
$$

In the case when the loss functions f_t are exp-concave the $Const$ factor is 1 and then we can write

$$
\sum_{t=e}^{s} f_t(x_t) - \min_{x \in K} \sum_{t \in I} f_t(x) \leq \sum_{t=r}^{s} f_t(x_{\mathcal{A}_{[e,s]},t}) - \min_{x \in K} \sum_{t \in I} f_t(x) + O(\log T) = Regret_{\mathcal{A}}(I) + O(\log T).
$$

The above regret bound holds simultaneously for all intervals I the desire of which motivated this entire section.

8.4 Application to Online Signal Denoising

We will now give an application of the SAA algorithm to online signal denoising. For the sake of simplicity, our signal is going to be either a vector in $\mathbb{R}^n$ or a matrix in $\mathbb{R}^{n \times n}$. Let $L_{d,n} = [n]^d$ where $d = 1$ or 2.

Consider the following protocol for rounds $t = 1, \ldots, T$:

1. *Adversary reveals $\rho(t) \in L_{d,n}$.*
2. *Learner predicts $\hat{y}_{\rho(t)}$.*
3. *Adversary reveals $y_{\rho(t)}$ or equivalently, a loss $f_t(x) = (x - y_{\rho(t)})^2$.*

This problem falls under the OCO framework with $K = \mathbb{R}$ or the range of y and $T = L_{d,n}$. The sequence of indices revealed $\rho(t)$ could be adversarially chosen and is a permutation of the indices in $L_{d,n}$. For example, when $d = 1$ and $\rho(t) = t$; this is the forecasting problem.

One of the most basic algorithms in this case is perhaps the running mean algorithm which predicts

$$\hat{y}_{\rho(t)} = \frac{1}{t-1} \sum_{i=1}^{t-1} y_{\rho(i)}.$$

The usual regret for this problem would be

$$\sum_{t=1}^{T} (\hat{y}_t - y_t)^2 - \sum_{t=1}^{T} (y_t - \bar{y}_t)^2 \leq O(log(n))$$

where the above inequality follows from the following Lemma 5.2.

Lemma 8.5. *For any sequence of numbers $x \in \mathbb{R}^n$ we have*

$$\sum_{i=1}^{n} (\bar{x}_{1:i-1} - x_i)^2 \leq \sum_{i=1}^{n} (\bar{x} - x_i)^2 + 4(1 + \log(n))$$

where $\bar{x}_{1:0} = 0$.

*This notion of regret competes against all constant sequences. **What if we want to compete against the best** $k \geq 1$ **piecewise constant sequence**? To do this, we can use the running mean algorithm as a base algorithm and sleeping experts on top of it.*

For the subsequent discussion, for the sake of exposition assume n is a power of 2 (although this is not necessary). We can define dyadic intervals (subset of $[n]$) to be of the form $[(a-1)2^s, a2^s] \subset [n]$ for non negative integers a, s. In the 2D case, we can define a dyadic rectangle as a product of two dyadic intervals.

We further define for any round $t \in [T]$, the set of active experts as corresponding to those dyadic intervals I containing t; i.e $A_t = \{\text{Dyadic } I : I \supset \{\rho(t)\}\}$.

The following lemma says that square losses are exp concave in a bounded region.

Lemma 8.6. *The function $e^{-\alpha(x-z)^2}$ for $\alpha > 0$ is concave in x, $\forall x, z \in [\frac{-1}{\sqrt{8\alpha}}, \frac{1}{\sqrt{8\alpha}}]$.*

For example, the above lemma means that if we assume $max|y_i| \leq 1$, then the square loss function $f(x) = (x - y_i)^2$ is $\frac{1}{8}$-exp-concave. We can now write the following result.

Lemma 8.7. *Let y be any seq/matrix with entries bounded by 1. Suppose that $\hat{y}$ are the predictions of the SAA algorithm (with experts corresponding to dyadic intervals/rectangles) applied on top of the running mean algorithm. Then for every dyadic interval S, we have the regret bound*

$$\sum_{t\in s}(\hat{y}_t - y_t)^2 \leq \sum_{t\in s}(y_t - y_{\textbf{\textit{running mean}},t})^2 + O(log(n)).$$

Proof. The proof is a direct implication of Theorem 8.4. □

The following basic fact would be useful.

Lemma 8.8. *Take any interval $I \subset [n]$, then I can be decomposed into $log(n)$ many disjoint dyadic intervals.*

The above lemma yields the following result which we write as a theorem.

Theorem 8.9. *[Adaptive Regret Bounds against Piecewise Constant Sequences]*

Let y be any seq/matrix ($d = 1, 2$ respectively) with entries bounded by 1. Suppose that $\hat{y}$ are the predictions of the SAA algorithm (with experts corresponding to dyadic intervals/rectangles) applied on top of the running mean algorithm. Then for every interval/rectangle $S \subset [n]$, we have the regret bound

$$\sum_{t\in s}(\hat{y}_t - y_t)^2 \leq \sum_{t\in s}(y_t - \overline{y}_S)^2 + O(log(n)^{d+1}).$$

Proof. Just use Lemma 8.7 alongwith Lemmas 8.8, 8.5. □

We now make some remarks regarding Theorem 8.9.

Remark 8.2. *Let us denote the set of all rectangular partitions of $L_{d,n}$ by $\mathcal{P}_{d,n,rect}$.*

Note that we can use Theorem 8.9 to conclude

$$\sum_{t=1}^{N}(\hat{y}_t - y_t)^2 \leq \inf_{P\in\mathcal{P}_{d,n,rect}} Approx.Error(y, P) + |P|log(n)^{d+1}$$

where $Approx.Error(P) = \|y - y_P\|^2$ with y_P taking the value on an entry in a given rectangle of P equal to mean of the entries of y within that rectangle, $|P|$ is the number of rectangles of the partition P. This bound shows that the regret against any piecewise constant sequence is bounded by the number of pieces of that sequence/matrix up to a log factor.

Remark 8.3. *We could have used the set of all intervals/rectangles as experts instead of dyadic ones. The only reason for using dyadic rectangles is the computational efficacy. Note the total number of dyadic rectangles is $O(N)$ whereas the number of all rectangles is $O(N^2)$. Moreover, for every $t \in L_{d,n}$ the number of dyadic rectangles containing it is $(\log n)^d$ whereas it is $O(N^2)$ for general rectangles. So the computational complexity of the SAA algorithm with all intervals as experts is $O(N^3)$ whereas for dyadic intervals as experts is $O(N(\log n)^d)$.*

Remark 8.4. *We can define a stochastic version of the signal denoising problem (both when $d = 1, 2$) where there is a true sequence/matrix θ^* and at every round $t \in [T]$, we wish to predict $\theta^*_{\rho(t)}$ by $\hat{\theta}_{\rho(t)}$ and then we observe a noisy version $y_{\rho(t)} = \theta^*_{\rho(t)} + \epsilon_{\rho(t)}$. We wish to bound the MSE $\frac{1}{T}\|\hat{\theta} - \theta^*\|^2$. Under the assumption that the noise ϵ consists of bounded mean 0 i.i.d entries with $\|y\|_\infty \leq 1$; the pointwise guarantee in Theorem 8.9 actually implies the MSE bound*

$$\sum_{t=1}^{N}(\hat{\theta}_t - \theta^*_t)^2 \leq \inf_{P \in \mathcal{P}_{d,n,rect}} Approx.Error(\theta^*, P) + |P| log(n)^{d+1}$$

where $\hat{\theta}$ are the predictions of the SAA algorithm with dyadic intervals as experts. Note that in the proof we use the boundedness of the data crucially. A natural question in this stochastic setting is what if the noise is unbounded. The case of gaussian noise can be handled by incurring an extra multiplicative log factor essentially because the max of N gaussians is $O(\sqrt{\log N})$; see Chatterjee and Goswami [9]. I believe that the case when the noise is heavy tailed is an open problem.

Remark 8.5. *What if we want to compete with all piecewise linear sequences instead of piecewise constant sequences? We can use the Vovk Azoury Warmuth forecaster (see the next section) as a base algorithm and apply the SAA algorithm with experts as dyadic rectangles on top of it; this has been done in [9] in the stochastic setting with i.i.d gaussian noise.*

8.5 Vovk Azoury Warmuth Forecaster

Consider the online linear regression problem where in every round the learner observes the covariate vector $x_t \in \mathbb{R}^d$; then the learner estimates a slope vector $\beta_t \in \mathbb{R}^d$ and uses it to predict y_t by $\beta_t^T x_t$ and then observes y_t and incurs the loss $l_t(\beta_t) = (y_t - \beta_t^T x_t)^2$. A classical online algorithm for this problem is the VAW forecaster Vovk [33]. Here,

$$\hat{\beta}_t = \operatorname*{argmin}_{\beta \in \mathbb{R}^d} \sum_{s=1}^{t-1}(y_s - \beta x_s)^2 + (\beta^T x_t)^2 + \|\beta\|^2.$$

We can see that $\hat{\beta}_t$ is a modified version of ridge regression estimator where y_t is set to be 0.

Remark 8.6. *[33] argues that the above predictor is better than the usual ridge regression in certain adversarial cases. The paper [25] argues the efficacy of the VAW forecaster as compared to ridge regression in a stochastic setting.*

Let us give a regret bound for the VAW forecaster (see, e.g., Rakhlin and Sridharan [26, pp. 38–40] for a proof).

Proposition 8.10 (Regret bound for Vovk-Azoury-Warmuth forecaster)**.** *The VAW forecaster defined above satisfies*

$$\sum_{t \in [T]} (\hat{y}_t - y_t)^2 - \inf_{\beta \in \mathbb{R}^d} \Big\{ \sum_{t \in [T]} (y_t - \beta \cdot x_t)^2 + \|\beta\|^2 \Big\} \leq d\|y_\infty\|^2 \log(1 + T \max_{t \in [T]} \|x_t\|^2/d).$$

9 Adversarial Multi-Arm Bandits

The MAB problem is a canonical problem in sequential decision making and has a long history going back to Thompson [32], Lai et al. [19]. Nice monographs on this area are Bubeck et al. [7], Lattimore and Szepesvári [20], Slivkins et al. [31]. Standard application domains include clinical trials, ad placement etc. There are two variants of the problem where one considers deterministic/adversarial and stochastic losses. We will first consider the adversarial setting.

9.1 Adversarial Setting

The adversarial setting was first introduced by Auer et al. [3]. This is a limited feedback version of the experts problem where there are K arms/actions/experts available. Let T be the total number of rounds. At round $t = 1, \dots, T$:

1. *Learner picks one arm $a_t \in [K]$ and simultaneously, nature/adversary decides the loss vector $\ell_t \in [0,1]^K$.*
2. *Learner observes and incurs the loss $\ell_t(a_t)$.*

The (expected) regret for this problem is defined to be

$$\text{Regret}_{\mathcal{A}} = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \inf_{a \in [K]} \sum_{t=1}^{T} \ell_t(a)\right].$$

Remark 9.1. *We shall focus on the case of **oblivious adversary**. That is, the loss vector ℓ_t does not depend on $a_1, \dots, a_t$, and can be considered as being laid down before the game starts. There are some subleties that arise when we allow for adaptive adversaries which we would not discuss here. Note that we cannot use the Hedge algorithm directly to solve the problem, since we do not have full information at the end of each round. Moreover, it can be shown that for any deterministic strategy, there exists a sequence of ℓ_t such that the regret is $O(T)$. Thus, we will need to consider randomized algorithms as in the full information setting.*

Remark 9.2. *MAB is a foundational model which exhibits the exploration exploitation tradeoff. On the one hand, one would like to select arms that have suffered small losses in the past (exploitation), on the other hand one would also like to select other actions to find out whether they can lead to even smaller losses. These two desires are in conflict since one can observe the loss of only one action per round. A good algorithm has to balance these two desires.*

9.2 Exp3 Algorithm and its Regret Bound

The simplest algorithm for adversarial bandits is called Exp3, which stands for "exponential-weight algorithm for exploration and exploitation". The basic idea is to run the Hedge algorithm except

that we do not observe the entire loss vector. So, we plug in an estimate $\hat{l}_t$ of the loss vector l_t and feed it to Hedge.

Specifically, the algorithm goes as follows:

At round $t \in [T]$,

1. *Plug in $\hat{\ell}_t$ and run the Hedge algorithm. To be more precise, sample $a_t \sim p_t$ at round t, where*

$$p_t \propto \exp\left(-\eta \sum_{s<t} \hat{\ell}_s(a)\right)$$

for some $\eta > 0$.

2. *Estimate ℓ_t by*

$$\hat{\ell}_t(a) = \frac{\ell_t(a)}{p_t(a)} \mathbf{1}(a = a_t),$$

which is an unbiased estimator for ℓ_t.

Remark 9.3. *Anytime an action is chosen, probability of choosing that action decreases in the next run (because $\hat{\ell}_t(a) = 0$ for $a \neq a_t$). This encourages exploration. Of course, exploitation is also built in the algorithm.*

Theorem 9.1 (Exp3 Regret Bound)**.** *For the Exp3 algorithm, we have the following regret bound:*

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \inf_{a \in [K]} \sum_{t=1}^{T} \ell_t(a)\right] \leq 2\sqrt{TK \ln K}.$$

Proof. Applying the regret bound for Hedge in Theorem 2.4 to the Exp3 algorithm, we get

$$\sum_{t=1}^{T} p_t \cdot \hat{\ell}_t - \sum_{t=1}^{T} \hat{\ell}_t(a^*) \leq \frac{\ln K}{\eta} + \eta \sum_{t=1}^{T} \sum_{a=1}^{K} p_t(a) \cdot \hat{\ell}_t^2(a),$$

where a^* denote the best action in hindsight. Note that here we used the nonnegativity of l_t crucially as the above bound only holds when $\eta \hat{l}_t(a) \geq -1$ which need not be true in this bandit setting if l_t is allowed to be negative.

Let us denote $\mathscr{F}_t = \sigma(a_1, \ldots, a_t)$ to be the sigma field containing all the history up to round t. Taking expectations we see that the left-hand side becomes (recall also that $\hat{\ell}_t$ is unbiased):

$$\mathbb{E} \sum_{t=1}^{T} \mathbb{E}\big(p_t \cdot \hat{\ell}_t - \hat{\ell}_t(a^*)\big) | \mathscr{F}_{t-1} = \mathbb{E} \sum_{t=1}^{T} p_t \cdot \ell_t - \ell_t(a^*) = \sum_{t=1}^{T} \mathbb{E}[\ell_t(a_t) - \ell_t(a^*)],$$

and similarly by using

$$\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}\left[\sum_{a=1}^{K} p_t(a) \cdot \hat{\ell}_t^2(a) \;\middle|\; \mathcal{F}_{t-1}\right]\right] = \sum_{t=1}^{T} \mathbb{E}\left[\sum_{a=1}^{K} p_t(a) \cdot \frac{\ell_t^2(a)}{p_t(a)}\right] \leq \sum_{t=1}^{T} \sum_{a=1}^{K} \ell_t^2(a)$$

the right-hand side becomes at most

$$\frac{\ln K}{\eta} + \eta T K.$$

By optimizing the right-hand side with respect to η, we finally obtain a bound on the expected regret:

$$\sum_{t=1}^{T} \mathbb{E}[\ell_t(a_t) - \ell_t(a^*)] \leq 2\sqrt{TK \ln K}.$$

□

Remark 9.4. *In the Hedge analysis, the losses can take any value in $[-1, 1]$ but here the non negativity of the losses is crucially used.*

Remark 9.5. *Theorem 9.1 only gives a bound for the expected regret. High probability bound for the regret will take more work, infact the Exp3 algorithm needs to be modified (in order to reduce its variance) to what is called the Exp3.P algorithm; see Theorem 3.3 in [7].*

9.3 Lower Bound

In this section, we will work with rewards $Y_{i,t} \in [0, 1]$ (ith arm, tth round). One can always go back and translate everything to losses by setting $1 - Y_{i,t} = l_{i,t}$. The main result of this section is the following.

Theorem 9.2 (Multi Arm Bandits Lower Bound)**.** *Fix any (possibly randomized) algorithm $\mathcal{A}$. There exists a universal constant C (not unreasonably large) such that the following lower bound is true*

$$\sup_{\mathbb{P}} \max_{i \in [K]} \mathbb{E} \sum_{t=1}^{T} (Y_{i,t} - Y_{a_t,t}) \geq C\sqrt{KT}. \tag{63}$$

where $\mathbb{P}$ refers to the joint distribution of the rewards where for each arm i, the sequence $Y_{i,t}$ are i.i.d draws from a bernoulli distribution $\mathbb{P}_i$ independently of other arms and the expectation above is with respect to both the randomness of the rewards and the internal randomness of the algorithm.

Remark 9.6. *Note that Theorem 9.2 implies that for any (possibly randomized) algorithm $\mathcal{A}$, there exists a bad sequence of rewards on which the regret is lower bounded by $O(\sqrt{KT})$.*

Remark 9.7. *Note that the term we are lower bounding is not the same as regret where the max should be inside the expectation. We are bounding a smaller term which is usually called the pseudo regret but we will abuse notation and call this regret whenever necessary.*

Actually, we will show the following lemma which will immediately yield Theorem 9.2.

Lemma 9.3. *For any arm $i \in [K]$, let $\mathbb{E}_i$ denote the expectation w.r.t the joint distribution of rewards (we call it random environment $\mathcal{E}_i$)) when $P_i = Bernoulli(\frac{1+\epsilon}{2})$ and all other $P_j = Bernoulli(\frac{1-\epsilon}{2})$. Then for any (possibly randomized) algorithm $\mathcal{A}$, we have*

$$\max_{i \in K} \mathbb{E}_i \sum_{t=1}^{T} (Y_{i,t} - Y_{a_t,t}) \geq C\sqrt{KT}.$$

Proof of Theorem 9.2. The display in Lemma 9.3 suffices to prove Theorem 9.2. We can interchange sup and max in the display (63). □

All that remains is to prove Lemma 9.3.

Proof of Lemma 9.3. It is enough to lower bound (max is bigger than average)

$$\underbrace{\frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_i\sum_{t=1}^{T}(Y_{i,t}-Y_{a_t,t})}_{T^*}.$$

Let us denote

$$R_i=\sum_{t=1}^{T}(Y_{i,t}-Y_{a_t,t}).$$

Then, with our notation

$$T^*=\frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_iR_i.$$

Note that this E_i refers to taking expectation over both the randomness of the rewards under the environment $\mathcal{E}_i$ and the internal randomness of the algorithm.

We will now proceed in steps.

1. **Step 1: Reduction to Deterministic Algorithm**

   What is a randomized algorithm? At every round, it constructs a probability on the set of actions and samples an action according to it. One way of thinking about this is that given a sequence of i.i.d Uniform $(0,1)$ random variables $U_1,\ldots,U_T$; the algorithm uses U_1 to generate a_1, then uses (a_1,Y_{1,a_1},U_2) to generate a_2, then uses $(a_1,Y_{1,a_1},a_2,Y_{2,a_2})$ to generate a_3 and so on. Importantly, the $U_1,\ldots,U_T$ are i.i.d $U(0,1)$ independent of the rewards $\{Y_{i,t}\}_{i\in[K],t\in[T]}$. With this representation of a randomized algorithm, one can see that for every fixed realization of $U_1,\ldots,U_T$, the algorithm now becomes a deterministic algorithm. We can now write

   $$\mathbb{E}_iR_i=\mathbb{E}_{U_1,\ldots,U_T}\mathbb{E}_{\{Y_{i,t}\}_{i\in[K],t\in[T]}\sim\mathcal{E}_i}R_i.$$

   Therefore, we can further write

   $$T^*=\mathbb{E}_{U_1,\ldots,U_T}\frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{\{Y_{i,t}\}_{i\in[K],t\in[T]}\sim\mathcal{E}_i}R_i.$$

   For every fixed realization of $U_1,\ldots,U_T$, lower bounding the inner average term suffices for our purpose. Therefore, it suffices to lower bound T^* for every deterministic algorithm. In the remainder of the proof we will assume that the algorithm $\mathcal{A}$ is deterministic.

2. **Step 2: Simplifying T^***

   Let us denote

   $$n(i) = \sum_{t=1}^{T} \mathbb{I}(a_t = i).$$

   Then, $n(i)$ simply counts the number of times arm i is pulled.

   We can now write

   $$\mathbb{E}_i(Y_{i,t} - Y_{a_t,t}) = \mathbb{E}_i \epsilon \mathbb{I}(a_t \neq i).$$

   Summing the above over all the rounds give us the following expression:

   $$T^* = T\epsilon\big(1 - \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_i \frac{n(i)}{T}\big). \tag{64}$$

3. **Step 3: Pinsker's Inequality**

   Define the sequence of random variables

   $$\tilde{Y}_t = Y_{a_t,t}.$$

   Let us denote Q_i to be the joint distribution of $(\tilde{Y}_1, \ldots, \tilde{Y}_T)$ under the environment $\mathcal{E}_i$. Let us also define the the environment $\mathcal{E}_0$ to be such that all the rewards of all the arms are drawn i.i.d from Bernoulli($\frac{1-\epsilon}{2}$). Note that since the losses are drawn from Bernoulli distributions therefore Q_i is a probability measure on the hypercube $\{0,1\}^T$.

   Note that since $\mathcal{A}$ is deterministic, the random sequence $a_1, \ldots, a_t$ is completely determined by the random sequence $(\tilde{Y}_1, \ldots, \tilde{Y}_T)$. Now, we can write

   $$\mathbb{E}_i n(i) - \mathbb{E}_0 n(i) = \sum_{v \in \{0,1\}^T} n(i)(v)\,(Q_i(v) - Q_0(v)) \leq nTV(Q_0, Q_1) \leq n\sqrt{\frac{1}{2}KL(Q_0, Q_1)}$$

   where $n(i)(v)$ just refers to $n(i)$ for the action sequence $a_1, \ldots, a_t$ when the sequence $(\tilde{Y}_1, \ldots, \tilde{Y}_T)$ equals v and in the last step we used Pinsker's inequality.

4. **Step 4: Computing KL Divergence**

   The following lemma allows us to compute the KL Divergences.

   Lemma 9.4 (KL Divergence Decomposition Lemma)**.** *Let $\mathcal{E}_0, \mathcal{E}_1$ be two random environments where the rewards corresponding to arm $a \in [K]$ are i.i.d samples from $Q_{0,a}, Q_{1,a}$ respectively, independently of other arms. Suppose the MAB algorithm is deterministic. Let the joint distribution of $(\tilde{Y}_1, \ldots, \tilde{Y}_T) = (Y_{1,a_1}, \ldots, Y_{T,a_T})$ under the two random environments be denoted by Q_0, Q_1 respectively. Then, we have*

   $$KL(Q_0, Q_1) = \sum_{i=1}^{K} \mathbb{E}_0 n(i)\ \ KL\big(Q_{0,i}, Q_{1,i}\big).$$

Proof of Lemma 9.4. The proof is due to the chain rule that KL divergence obeys. Specifically,

$$KL(P_0, P_1) = KL(P_0^{(1)}, P_1^{(1)}) + \sum_{t=2}^{T} \sum_{(\tilde{y}_1,\ldots,\tilde{y}_T)} P_0(\tilde{Y}_{1:t-1} = (\tilde{y}_1, \ldots, \tilde{y}_T)) KL(P_0^{(t)}, P_1^{(t)} | \tilde{Y}_{1:t-1}).$$

In the above, $P_0^{(t)}, P_1^{(t)}$ refers to the distribution of $\tilde{Y}_t$ under the random environments $\mathcal{E}_0, \mathcal{E}_1$. Note that because $\mathcal{A}$ is a deterministic algorithm, a_t is completely determined by $\tilde{Y}_{1:t-1}$. Now observe that for any $t \in [T]$, we have the t th term on the R.H.S above equals

$$\sum_{a=1}^{K} P_0(a_t = a) KL(Q_{0,a}, Q_{1,a}).$$

□

We now apply Lemma 9.4 in our setting and use the fact that there exists a constant c such that

$$KL\big(Bernoulli(\frac{1-\epsilon}{2}), Bernoulli(\frac{1+\epsilon}{2})\big) \leq c\epsilon^2$$

for all $\epsilon \in (0, 1/2)$.

The last display in Step 3 along with the above lets us conclude that

$$\mathbb{E}_i n(i) \leq \mathbb{E}_0 n(i) + n\sqrt{\frac{1}{2}\mathbb{E}_0 n(i) c\epsilon^2}. \tag{65}$$

5. **Step 5: Combining Everything**

We can now write due to (65) and (64),

$$T^* \geq T\epsilon - \epsilon \frac{1}{K} \underbrace{\sum_{i=1}^{K} \mathbb{E}_0 n(i)}_{=T} - \frac{T\epsilon}{K} \sum_{i=1}^{K} \sqrt{\frac{1}{2}\mathbb{E}_0 n(i) c\epsilon^2}.$$

Now, by Jensen's inequality,

$$\frac{1}{K}\sum_{i=1}^{K} \sqrt{\mathbb{E}_0 n(i)} \leq \sqrt{\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_0 n(i)} \leq \sqrt{\frac{T}{K}}.$$

Therefore, we can write

$$T^* \geq T\epsilon\big(1 - \frac{1}{K} - c\epsilon\sqrt{T/K}\big) \geq \big(1/2 - c\epsilon\sqrt{T/K}\big).$$

Set $\epsilon = \frac{\sqrt{k}}{4c\sqrt{T}}$ to conclude

$$T^* \geq T\epsilon/4 = \frac{\sqrt{TK}}{16c}.$$

□

Remark 9.8. *What did we learn from the above proof? One of the hardest cases for a MAB algorithm is when one of the means is slightly $\delta = O(\sqrt{KT})$ higher than the rest. Let's try to intuitively summarize the above proof. Fix an algorithm and consider the completely random environment $\mathcal{E}_0$ where all the arms draw from roughly fair coin flips. Under this environment, atleast one of the arms must be pulled less than T/K times. Change the environment slightly where now we raise the mean of this arm by δ. This change δ is small enough so that the behaviour of the algorithm does not change much in the sense that this arm will be still not be picked atleast $\Omega(T)$ times. In these rounds, we will incur regret $O(T\delta = \sqrt{KT})$.*

9.4 Regularization by Tsallis Entropy

As we can see, we have shown an upper bound to the regret of Exp3 scaling like $O(\sqrt{K \log KT})$ and we have also shown a lower bound scaling like $O(\sqrt{KT})$. It turns out that this $\log K$ factor can be removed by using a different algorithm. We now present an intuitive argument motivating this algorithm.

We have seen that the term $\sum_{a=1}^K p_t(a)\hat{l}_t(a)^2$ term in the regret bound of Hedge was critical for us, in the sense that without the $p_t(a)$ weighting, the expectation of this term could be very big. Thus, it would be even nicer if our bound was of the form $\sum_{a=1}^K p_t(a)^{3/2}\hat{l}_t(a)^2$ say as then by taking expectation we could obtain a bound $\sum_{a=1}^K p_t(a)^{1/2} \leq \sqrt{K}$. This would then give us an overall regret bound $\frac{1}{\eta}\log K + \eta T\sqrt{K}$ which is better. This is not of course true for Exp3. However it turns out that it is possible to indeed obtain the $\eta T\sqrt{K}$ term but at the cost of increasing the first $\frac{1}{\eta}\log K$ term to $\frac{1}{\eta}\sqrt{K}$. Nevertheless, by choosing η optimally this indeed then gives me the $O(\sqrt{KT})$ bound without the log factor.

Recall that Hedge can be seen as FTRL with neg entropy regularizer. Then, it can be checked that with $\psi(p) = \sum_{i=1}^K p(i)\log p(i)$, the term $\sum_{a=1}^K p_t(a)\hat{l}_t(a)^2 = \|l_t\|^2_{\nabla^{-2}\psi(p_t)}$. Recall the notation $\|v\|_M^2 = v^T M v$ for a positive definite matrix M. We can think of $\|l_t\|^2_{\nabla^{-2}\psi(p_t)}$ as a local norm term. Can we get such a local norm term if we use other regularizers? By reverse engineering, we can check that if we define $\psi(p) = -\sum_{i=1}^K \sqrt{p(i)}$ then $\|l_t\|^2_{\nabla^{-2}\psi(p_t)} = \sum_{a=1}^K p_t(a)^{3/2}\hat{l}_t(a)^2$ which is precisely what we desire. The question is can we prove such a local norm bound for this type of regularizer? The answer turns out to be yes.

9.4.1 Tsallis Entropy

Definition 9.5. *The (negative) Tsallis Entropy is a function ψ on the probability simplex S_k for some parameter $0 < \beta < 1$ defined as follows:*

$$\psi(p) = \frac{1 - \sum_{a=1}^{K} p(a)^\beta}{1 - \beta}.$$

We now list some properties of Tsallis Entropy.

1. *Note that ψ is a convex function on the simplex just like negative Shannon entropy. So we could indeed think of ψ as some sort of entropy.*
2. *Letting $\beta \to 1$, by using L' Hospital's rule one can check that ψ converges to Shannon entropy. So we can think of Tsallis entropy as a family of entropies parametrized by $\beta \in (0, 1]$ with $\beta = 1$ corresponding to Shannon entropy.*
3. *The function ψ is maximized at the corners with maximum value 0 and minimized at the uniform distribution with value $\frac{1-K^{1-\beta}}{1-\beta}$.*
4. *Hessian $\nabla^2\psi(p)$ is a diagonal matrix with the ath diagonal entry $\beta p(a)^{\beta-2}$.*

9.4.2 Local Norm Bound for FTRL with Tsallis Entropy Regularizer

Lemma 9.6. *Consider the following FTRL algorithm*

$$p_t = \operatorname*{argmin}_{p \in S_k} \langle p, \sum_{s<t} \hat{l}_s \rangle + \frac{1}{\eta}\psi(p) \tag{66}$$

where $\eta > 0$ is a tuning parameter, ψ is the Tsallis entropy with parameter $\beta \in (0, 1)$, and $\hat{l}_1, \ldots, \hat{l}_T$ are arbitrary loss vectors in $\mathbb{R}^K_+$. Then the following holds for any $a^* \in [K]$,

$$\sum_{t=1}^{T} \langle p_t, \hat{l}_t \rangle - \sum_{t=1}^{T} \hat{l}_t(a^*) \leq \frac{1}{\eta} Range(\psi) + \frac{\eta}{2} \|\hat{l}_t\|^2_{\nabla^{-2}\psi(p_t)} = \frac{K^{1-\beta} - 1}{\eta(1-\beta)} + \frac{\eta}{2\beta} \sum_{t=1}^{T} \sum_{a=1}^{K} p_t(a)^{2-\beta} \hat{l}_t(a)^2. \tag{67}$$

Proof. The equality in (67) follows from the maximum and minimum value of ψ and the expression for the Hessian of ψ. To derive this bound we will use the general bound for FTRL given in Corollary 7.2. In light of this corollary, it suffices for us to show the following for every $t \in [T]$,

$$(p_t - p_{t+1})^T \hat{l}_t - \frac{1}{\eta} D_\psi(p_t, p_{t+1}) \leq \frac{\eta}{2} \|\hat{l}_t\|^2_{\nabla^{-2}\psi(p_t)}. \tag{68}$$

Let's extend the definition of ψ to the entire non-negative orthant for the subsequent discussion. We can bound the L.H.S in (68) by

$$\sup_{q \in \mathbb{R}^k_+} \langle p_t - q, \hat{l}_t \rangle - \frac{1}{\eta} D_\psi(q, p_t).$$

Let q_t be a maximizer of the above (show that it exists). Then, we can write

$$\langle p_t - q_t, \hat{l}_t \rangle - \frac{1}{\eta} D_\psi(q_t, p_t) = \langle p_t - q_t, \hat{l}_t \rangle - \frac{1}{2\eta} \|q_t - p_t\|_{\nabla^2 \psi(\xi)} \leq \frac{\eta}{2} \|\hat{l}_t\|^2_{\nabla^{-2} \psi(\xi)}.$$

for some ξ lying between p_t and q_t where in the inequality we have used $a^t b = a^t M^{1/2} M^{-1/2} b \leq \frac{1}{2\eta} \|a\|^2_M + \frac{\eta}{2} \|b\|^2_{M^{-1}}$ for any positive definite matrix M and any vectors a, b and any $\eta > 0$.

It remains now to show that

$$\|\hat{l}_t\|^2_{\nabla^{-2} \psi(\xi)} \leq \|\hat{l}_t\|^2_{\nabla^{-2} \psi(p_t)}$$

The above will follow if we can show that for all arms a and any ψ on the line segment between p_t and q_t we have $p_t(a) \geq p_t(\psi)$ which will in turn follow if we can show that

$$p_t(a) \geq q_t(a) \quad \forall a.$$

Fix an arm $a \in [K]$. If $q_t(a) = 0$ we are done. So let's assume $q_t(a) > 0$. Because q_t is the maximizer it should now satisfy a normal equation where we set the gradient to be 0. The normal equation is

$$\nabla \psi(q_t) = \nabla \psi(p_t) - \eta \hat{l}_t.$$

Looking at the a th coordinate of the above normal equation gives

$$\frac{1}{q_t(a)^{1-\beta}} = \frac{1}{p_t(a)^{1-\beta}} + \frac{\eta(1-\beta)}{\beta} \hat{l}_t(a).$$

Use $\hat{l}_t(a) \geq 0$ to conclude that $p_t(a) \geq q_t(a)$.

□

9.4.3 Regret Bound

Theorem 9.7 (Regret Bound for Tsallis Entropy Regularized FTRL for MAB)**.** *Consider the following MAB algorithm $\mathcal{A}$, at time t, sample $a_t \sim p_t$ with p_t defined in (66) with $\hat{l}_t(a) = \frac{l_t(a)}{p_t(a)} \mathbb{I}(a_t = a)$ being the inverse importance weighted estimator. The losses are assumed to be in $[0, 1]$. Then we have*

$$Regret_{\mathcal{A}} \leq \frac{K^{1-\beta} - 1}{\eta(1-\beta)} + \frac{\eta K^\beta T}{2\beta}.$$

By setting $\beta = 1/2$ and $\eta = 1/\sqrt{T}$ we obtain the minimax rate optimal regret $O(\sqrt{KT})$.

Proof. We can just take expectation of both sides in (67). The L.H.S just becomes the regret and the R.H.S becomes

$$\frac{K^{1-\beta}-1}{\eta(1-\beta)}+\frac{\eta K^{\beta}}{\beta}\sum_{t=1}^{T}\sum_{a=1}^{K} l_t(a)^2 p_t(a)^{1-\beta}.$$

Now just use the fact that

$$\sum_{a=1}^{K} p_t(a)^{1-\beta} \leq K^{\beta}$$

and $0 \leq l_t(a) \leq 1$. □

Remark 9.9. *Note that we could have optimized η for any fixed $\beta \in (0,1)$; by setting $\eta = K^{1/2-\beta}T^{-1/2}$ we would recover the $O(\sqrt{KT})$ bound.*

Remark 9.10. *One needs to solve an optimization problem (can be done efficiently) every round to compute p_t, there is no closed form solution.*

Remark 9.11. *As a finishing remark for this section, there are analogous small variation results and competing against best k piecewise constant strategies in the bandit setting as well. See the bibliographic remarks at the end of Chapter 3 in [7].*

10 Stochastic Multi Arm Bandits

There is a huge literature on stochastic version of the MAB problem where each arm generates an independent sample from a certain distribution. This is clearly an easier problem than the adversarial version, the goal here is usually to derive instance optimal bounds which could be better than the worst case $O(\sqrt{KT})$ bound. Let us denote P_a to be the probability distribution corresponding to arm a with mean $\mu(a)$. We will assume for simplicity of exposition that P_A is supported on $[0,1]$ although this can be relaxed. It is more standard in the literature to control the pseudo regret in this problem, mainly because it is natural in the stochastic setting and also is easier to analyze. Recall that the pseudo regret (we will call it regret anyway in this section) is given by

$$\text{Regret}_{\mathcal{A}} = \max_{a\in[K]} \mathbb{E}\left[\sum_{t=1}^{T}\ell_t(a_t) - \sum_{t=1}^{T}\ell_t(a)\right]$$

Let us denote $a^* = \text{argmin}_{a\in[K]}\mu(a)$ and $\Delta(a) = \mu(a) - \mu(a^*)$ to be the average suboptimality gap for arm a. Note that we can simplify the regret and write it in a couple of different ways.

$$\text{Regret}_{\mathcal{A}} = \max_{a\in[K]} \mathbb{E}\left[\sum_{t=1}^{T}\mu(a_t) - \sum_{t=1}^{T}\mu(a)\right] = \mathbb{E}\sum_{t=1}^{T}\Delta(a_t) = \sum_{t=1}^{T}\sum_{a\in[K]}\Delta(a)P(a=a_t) = \sum_{a\in[K]}\Delta(a)\mathbb{E}n(a)$$

where $n(a)$ is the random variable which denotes the number of times arm a is pulled.

Remark 10.1. *As before in the adversarial case, a good algorithm needs to balance exploration and exploitation. On the one hand, the learner wants to follow the sample means and pull the minimum one, on the other hand the sample means have to be reliable which means the learner wants to pull each arm enough number of times.*

Clearly, it is natural to base an algorithm by just looking at the running sample means of the arms. Let us use the notation $n_t(a)$ to denote the number of times arm a is pulled in the first t rounds and let us use $\hat{\mu}_t(a) = \sum_{l=1}^{t} l_t(a)\mathbb{I}(a = a_t)$ to denote the sample mean of the observations corresponding to arm a that we have seen till round t. To that end, let us state the following lemma.

Lemma 10.1. *For any stochastic MAB algorithm, the following good event $\mathcal{G}$ holds with probability atleast $1 - \frac{2K}{T}$,*

$$\mathcal{G} = \{|\hat{\mu}_t(a) - \mu(a)| \leq 2\sqrt{\frac{\log T}{n_t(a)}} \;\; \forall t \in [T], \forall a \in [K]\}.$$

Proof. The proof basically relies on Hoeffding's inequality and union bounds. There is a slight subtle issue that one needs to take care of which is that the sample sizes $n_t(a)$ are also random. To do this, let's consider a random table of losses $\{l_t(a)\}_{a\in[K],t\in[T]}$ such that for each a, $l_1(a), \ldots, l_T(a)$ are i.i.d draws from the distribution P_a independently of other arms. We can treat the set of all possible such tables as the sample space for the MAB experiment in the sense that at any round t, if a MAB algorithm pulls arm a, let it observe $l_{n_{t-1}(a)+1}(a)$. Then, given a deterministic algorithm such as UCB (defined below), the random variables $a_1, \ldots, a_T$ are functions of the random table $\{l_t(a)\}_{a\in[K],t\in[T]}$.

Now we can observe that

$$\max_{a\in[K],t\in[T]} \big(|\hat{\mu}_t(a) - \mu(a)| - 2\sqrt{\frac{\log T}{n_t(a)}}\big) \leq \max_{a\in[K],m\in[T]} \big(|\overline{l_t(a)}_{1:m} - \mu(a)| - 2\sqrt{\frac{\log T}{m}}\big) \tag{69}$$

For any fixed $m \in [T]$ and any $a \in [K]$, by Hoeffding's inequality for all $u > 0$,

$$P(|\overline{l_t(a)}_{1:m} - \mu(a)| > u) \leq 2\exp(-mu^2/2).$$

As a consequence, we can write

$$P(|\overline{l_t(a)}_{1:m} - \mu(a)| > 2\sqrt{\frac{\log T}{m}}) \leq \frac{2}{T^2}.$$

Take union bound over all $t \in [T]$ and all $a \in [K]$ on the above display to obtain that

$$P(\max_{a\in[K],m\in[T]} \big(|\overline{l_t(a)}_{1:m} - \mu(a)| - 2\sqrt{\frac{\log T}{m}}\big) < 0) \geq 1 - \frac{2K}{T}.$$

Now the above display alongwith (69) finishes the proof. □

Remark 10.2. *The first algorithm one might think of is to explore then exploit. This algorithm proceeds by first sampling every arm some number of times and then sticking with the empirically best action. A quick analysis shows that this would be suboptimal. If we want to estimate the true mean up to error ϵ we would need $\log T/\epsilon^2$ many samples. Thus, the regret for the first $K \log T/\epsilon^2$ would be $O(1)$ per round in the worst case. For the remaining $T - K \log T/\epsilon^2$ rounds we will incur at most $O(\epsilon)$ regret. So, the overall regret incurred would be at most $\tilde{O}(K/\epsilon^2)+T\epsilon = \tilde{O}(T^{2/3}K^{1/3})$ which is worse than $\sqrt{KT}$.*

10.1 UCB Algorithm

The Upper Confidence Bound (UCB) algorithm is one of the canonical bandit algorithms, first proposed by [19]. Since we consider losses instead of rewards, our algorithm will actually be LCB but we will still call it UCB. UCB applies the principle of optimism under uncertainty.

Definition 10.2. *Define*

$$LCB_t(a) = \hat{\mu}_{t-1}(a) - 2\sqrt{\frac{\log T}{n_{t-1}(a)}}.$$

Now, UCB plays the action

$$a_t = \operatorname*{argmin}_{a \in [K]} LCB_t(a).$$

We now make a couple of remarks.

- *UCB is a deterministic algorithm in contrast to Exp3.*
- *When $n_{t-1}(a) = 0$, we have $LCB_t(a) = -\infty$ and hence arm a is picked. This means that in the first K rounds, every arm is picked once.*
- *Minimizing the first term in $LCB_t(a)$ encourages exploitation and the second term encourages exploration. Interestingly, if we had minimized the upper confidence bar, then exploration would not have been encouraged. So the fact that we are using the lower confidence bar is actually critical.*

We will now start the analysis of UCB which is pretty straightforward. We will bound the regret on the good event $\mathcal{G}$.

Lemma 10.3. *Under the event $\mathcal{G}$, UCB ensures that*

$$\Delta(a_t) \leq 4\sqrt{\frac{\log T}{n_t(a_t)}} \ \forall t \in [T]. \tag{70}$$

Proof. We can write

$$\Delta(a_t) = \mu(a_t) - \mu(a^*) \leq \mu(a_t) - LCB_t(a^*) \leq \mu(a_t) - LCB_t(a_t) =$$
$$\mu(a_t) - \hat{\mu}_{t-1}(a_t) + 2\sqrt{\frac{\log T}{n_t(a_t)}} \leq 4\sqrt{\frac{\log T}{n_t(a_t)}}$$

where the first inequality follows due to Lemma 10.1, the second inequality follows by definition of the UCB algorithm and the last inequality again follows due to Lemma 10.1 applied to round $t-1$. □

We now give a worst case regret bound for UCB.

Theorem 10.4.

$$\text{Regret}_{UCB} \leq 8\sqrt{KT}\log T + 3K.$$

Proof. Recall that the regret is

$$\mathbb{E}\sum_{t=1}^{T}\Delta(a_t).$$

We can bound this as follows. On the event $\mathcal{G}$ using Lemma 10.3,

$$\sum_{t=1}^{T}\Delta(a_t) = \sum_{t=1}^{K}\Delta(a_t) + \sum_{t=K+1}^{T}\Delta(a_t) \leq K + 4\sqrt{\log T}\sum_{t=K+1}^{T}\frac{1}{\sqrt{n_{t-1}(a_t)}}.$$

Moreover, we have

$$\sum_{t=K+1}^{T}\frac{1}{\sqrt{n_{t-1}(a_t)}} = \sum_{t=K+1}^{T}\sum_{a=1}^{K}\frac{1}{\sqrt{n_{t-1}(a_t)}}\mathbb{I}(a_t = a) = \sum_{a=1}^{K}\left(1 + 1/\sqrt{2} + \cdots + 1/\sqrt{n_{T-1}(a)}\right) =$$
$$2\sum_{a=1}^{K}\sqrt{n_T(a)} \leq 2\sqrt{KT}$$

where in the last equality we used the fact that $\sum_{s=1}^{m} 1/\sqrt{s} \leq 2\sqrt{m}$ and in the last inequality we used Jensen's inequality.

Combining the last two displays we can write

$$\mathbb{E}\sum_{t=1}^{T}\Delta(a_t)(\mathbb{I}(\mathcal{G}) + \mathbb{I}(\mathcal{G}^c)) \leq 8\sqrt{KT}\log T + K + 2K.$$

□

Remark 10.3. *Note that $\sqrt{KT}$ is the minimax optimal rate even in the adversarial setting.*

We will now show an instance optimal bound. For that, we state the next lemma which bounds the number of times an arm a is pulled by the UCB algorithm.

Lemma 10.5. *Under event $\mathcal{G}$, we have*

$$n_T(a) \leq \frac{16 \log T}{\Delta_a^2} + 1 \quad \forall a \in [K].$$

Proof. For each arm a, set t to be the last time that arm a is pulled. Then, by Lemma 10.3 we have

$$\Delta(a) \leq 4\sqrt{\frac{\log T}{n_T(a) - 1}}.$$

Rearranging the above display finishes the proof. □

We are now ready to state the following instance optimal regret bound for UCB.

Theorem 10.6.

$$\text{Regret}_{UCB} \leq 3K + 16 \log T \Big(\sum_{a \in [K]: \Delta(a) > 0} \frac{1}{\Delta(a)} \Big)$$

Proof. We can write

$$\mathbb{E} \sum_{t=1}^{T} \Delta(a) \mathbb{E}\big[n_T(a) (\mathbb{I}(\mathcal{G}) + \mathbb{I}(\mathcal{G}^c)) \big] \leq K + 16 \log T \sum_{a \in [K]: \Delta(a) > 0} \frac{1}{\Delta(a)} + 2K$$

where the inequality is a consequence of Lemma 10.5. □

Remark 10.4. *In particular, if all the $\Delta(a)$ are $O(1)$ then the regret of UCB is $O(K \log T)$ which is much faster than the worst case rate $O(\sqrt{KT})$. On the other hand, if all the $\Delta(a)$ are $O(\sqrt{K/T})$ then we get back the worst case rate $O(\sqrt{KT})$. This instance bound is known to be the best instance optimal regret bound that you can get in the stochastic MAB problem.*

10.2 Some Other Remarks

- *There is another algorithm called successive elimination which is similar in spirit to UCB. In the case of two arms, this algorithm maintains confidence intervals for the two means. If at round t, the intervals do not overlap then it plays the lower of the two arms, otherwise it flips a fair coin to decide the arm to play. This enjoys similar regret guarantees as UCB and is perhaps more intuitive than UCB.*

- *High Probability Bounds: Note that regret can be negative so simply giving an expected regret bound may not be satisfactory. Indeed, it is important to give high probability bounds in these settings. High probability bounds are available in the literature for the UCB algorithm; see Audibert et al. [2].*

- *Heavy Tailed Bandits: The analysis we presented relied on Hoeffding's inequality and thus holds also for sub gaussian distributions. However, robust versions of UCB exist which work under only a finite variance assumption; see Bubeck et al. [8]. There should also be some quantile formulation of this problem if we want to include heavy tailed distributions such as the Cauchy.*

- *Thomson Sampling: In [32], a simple strategy was proposed for Bernoulli arms. Assume a uniform prior for $\mu(a) \in [0,1]$. Let $\pi_{a,t}$ be the posterior distribution (a beta distribution) for $\mu(a)$ at the tth round and let us sample $\theta_{a,t} \sim \pi_{a,t}$ independently from the past. Then define*

$$a_t = \operatorname*{argmin}_{a \in [K]} \theta_{a,t}.$$

Surprisingly, a full theoretical analysis of this algorithm came only in 2010's; see Agrawal and Goyal [1], Russo and Van Roy [29].

- *What if the arms are correlated? Does this setting make sense? For example, we could think of the arms forming a Gaussian Process with a certain dependence structure.*

11 Contextual Bandits

Contextual Bandits is an important extension of the MAB model with the following protocol:

1. *The environment first decides a context $x_t \in \mathcal{X}$ for some context space $\mathcal{X}$ and a loss vector $l_t \in [0,1]^K$ specifying the loss of K actions.*
2. *The learner observes the context x_t and then selects an action $a_t \in [K]$.*
3. *The learner observes and incurs the loss $l_t(a_t)$.*

Compared to the standard MAB problem, the only extra element here are the contexts which are supposed to give some information about the arms/actions. Think of clinical trials where x_t corresponds to patient covariates, arms/actions are the treatments and losses measure the response of the patient to treatment. Similarly, another application could be personalized ad recommendation where x_t correspond to user's covariates, the arms are the ads and losses are whether the user clicks on the ad or not.

11.1 Agnostic/Adversarial Setting

How should we define the regret here and what should be the role of the contexts? One way to formulate the problem is as follows. We can define a policy $\pi : \mathcal{X} \to [K]$ as a function from the context space to actions. A policy is implementable in the sense that whenever I observe context x_t it tells me to play $\pi(x_t)$. Given some class of policies Π we can now desire to compete against this class of policies. So for a given algorithm $\mathcal{A}$ we can define its regret to be

$$\text{Regret}_{\mathcal{A}} = \max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \sum_{t=1}^{T} \ell_t(\pi(x_t))\right]$$

where the expectation $\mathbb{E}$ is with respect to the potential randomness inherent in the algorithm $\mathcal{A}$.

Remark 11.1. *Note that if we consider the set of all constant functions as my policy class Π; each policy then corresponds to always pulling an arm a and we are back to the usual MAB problem. In this sense, CB is a generalization of MAB. An example of a more complicated policy class could be the following. Consider a prefixed partition of the context space $\mathcal{X}$ into m subsets. We could consider piecewise constant policies, constant on each of these m subsets. In this case, the cardinality of the policy class becomes K^m which becomes large quickly if m is large.*

11.1.1 Two Natural Approaches

There are two natural approaches to solving this problem.

- *One can view Contextual Bandits (CB) as a combination of several MAB problems, one for each context $x_t \in \mathcal{X}$. In other words, one maintains a MAB algorithm for each context and in round t, depending on the context x_t one uses the MAB algorithm corresponding to this context to select the next arm. This would not be a good strategy if $|\mathcal{X}|$ is very large, as in that case the contexts wont get repeated too many times.*
- *Another approach is to treat each policy $\pi \in \Pi$ as an expert. This then reduces the problem to an $N = |\Pi|$ armed MAB problem. Each policy π tells me to play $a_t = \pi(x_t)$ and then we observe the loss corresponding to this policy by observing $l_t(\pi(x_t))$. Then we will get $O(\sqrt{NT})$ regret, which may be unacceptable if N is very large.*

11.2 Exp 4 Algorithm for Adversarial Contextual Bandits

Suppose I am given a finite (possibly very large) class of policies Π with $N = |\Pi|$. The basic idea is to consider the second approach outlined above and treat this as an MAB problem. This means I can use the Exp3 algorithm which will give me $O(\sqrt{NT})$ regret. It turns out we can tweak this Exp3 algorithm slightly to obtain $\tilde{O}(\sqrt{KT})$ regret which is much better. This algorithm is the Exp4 algorithm proposed in Auer et al. [4].

The idea is just like in the Exp3 algorithm, we will estimate the entire loss vector $l_t \in [0,1]^N$ by $\hat{l}_t \in [0,1]^N$ and then compute a probability distribution on the set of policies $P_t \in S_N$ defined as

$$P_t(\pi) \propto \exp(-\eta \sum_{s<t} \hat{l}_s(\pi))$$

for a tuning parameter $\eta > 0$.

We sample a policy $\pi_t \sim P_t$ and then play action $a_t = \pi(x_t)$. This is equivalent to saying that we sample an action $a_t \sim p_t \in S_k$ where

$$p_t(a) = \sum_{\pi \in \Pi : \pi(x_t) = a} P_t(\pi).$$

In words, p_t is a probability distribution on the set of arms induced by the probability distribution P_t on the set of policies.

So far, this is simply running the Exp3 algorithm. The main difference is how we compute $\hat{l}_t$. Note that the feedback $l_t(a_t)$ gives the loss for not just the policy π_t but also all other policies which whould have chosen a_t. The usual Exp3 algorithm would set

$$\hat{l}_t(\pi) = \frac{l_t(\pi(x_t))}{P_t(\pi)}\mathbb{I}(\pi = \pi_t).$$

This would mean that the $\hat{l}_t$ random vector would only have one non zero entry. However, it is more natural to estimate the loss of an action instead of a policy. So, the Exp4 algorithm sets

$$\hat{l}_t(\pi) = \frac{l_t(\pi(x_t))}{p_t(\pi(x_t))}\mathbb{I}(a_t = \pi(x_t)).$$

Note the p_t in the denominator instead of P_t. This $\hat{l}_t$ has potentially many more non zero entries. Note that this $\hat{l}_t$ is also unbiased for l_t.

We can now check that the second moment is

$$\mathbb{E}\hat{l}_t^2(\pi) = \frac{l_t^2(\pi(x_t))}{p_t(\pi(x_t))}.$$

Note that if we had used the vanilla estimates $\hat{l}_t$ then we would have P_t in the denominator instead of p_t which would have potentially increased the second moment by a lot. We now summarize the Exp4 algorithm below.

At round $t \in [T]$,

1. *Compute $P_t \in S_N$ such that $P_t(\pi) \propto \exp(-\eta \sum_{s<t} \hat{l}_s(\pi(x_s)))$.*
2. *Sample $a_t \sim p_t \in S_K$ where $p_t(a) = \sum_{\pi \in \Pi : \pi(x_t) = a} P_t(\pi)$.*
3. *Observe $l_t(a_t)$ and construct $\hat{l}_t \in \mathbb{R}_+^N$ such that*

$$\hat{l}_t(\pi) = \frac{l_t(\pi(x_t))}{p_t(\pi(x_t))}\mathbb{I}(a_t = \pi(x_t)).$$

We are now ready to state the main regrt bound for Exp4.

Theorem 11.1 (Regret Bound for Exp4)**.** *For any finite policy class Π with $|\Pi| = N$, the Exp4 algorithm attains the expected regret bound*

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \inf_{\pi \in \Pi} \sum_{t=1}^{T} \ell_t(\pi(x_t))\right] \leq 2\sqrt{TK \ln N}.$$

Proof. Similar to the Exp3 analysis, the proof starts from the local norm bound for Hedge, which holds due to the nonnegativity of the estimated loss vectors, for any $\pi \in \Pi$,

$$\sum_{t=1}^{T} \langle P_t, \hat{l}_t \rangle - \sum_{t=1}^{T} \hat{l}_t(\pi) \leq \frac{\log N}{\eta} + \sum_{t=1}^{T} \sum_{\pi \in \Pi} P_t(\pi) \hat{l}_t^2(\pi).$$

As in the Exp3 proof, by unbiasedness of $\hat{l}_t$, the L.H.S above just becomes the regret against the policy π after taking expectation. Taking expectation for the local norm term on the R.H.S gives

$$\mathbb{E} \sum_{\pi \in \Pi} P_t(\pi) \hat{l}_t^2(\pi) \leq \sum_{\pi \in \Pi} \frac{P_t(\pi)}{p_t(\pi(x_t))} = \sum_{a \in [K]} \sum_{\pi \in \Pi : \pi(x_t) = a} \frac{P_t(\pi)}{p_t(a)} = K.$$

Therefore, we get a bound of $\frac{\log N}{\eta} + \eta T K$ which give us the $O(\sqrt{KT \log N})$ bound after optimizing over η. □

Remark 11.2. *Note that the computational complexity of Exp4 is $O(N)$. In cases when N is too large and we cannot afford to search over all policies, we would need to make some assumptions about the loss generating process such as the losses are stochastic with means that are nice functions of the context. This is what we describe next.*

11.3 Realizable Setting

We will now consider the stochastic setting where arm a draws losses (independently of the past) i.i.d from a probability distribution $P_a(x)$ when the context is x. Let $\mathscr{F}$ be a function class consisting of functions $\mathcal{X} \times [K] \to [0, 1]$. We will assume that there exists a function $f^* \in \mathscr{F}$ such that the mean of $P_a(x) = f^*(x, a)$, in other words, $\mathbb{E} l_t(x, a) = f^*(x, a)$. This is called the realizable setting in the literature. For example, $\mathcal{F}$ could represent K functions from a given function class such as Linear functions, Holder Smooth functions, Shape Constrained function classes such as monotonicity and convexity, large function classes such as neural networks with a lot of parameters.

In this setting, the natural class of policies is indexed by functions in $\mathscr{F}$. That is,

$$\Pi = \{\pi_f : f \in \mathscr{F}\}$$

where the policy π_f is defined as

$$\pi_f(x) = \operatorname*{argmin}_{a \in [K]} f(x, a).$$

In this realizable setting we can write the regret of an algorithm as

$$\text{Regret}_{\mathcal{A}} = \max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(a_t) - \sum_{t=1}^{T} \ell_t(\pi(x_t))\right] = \mathbb{E}\sum_{t=1}^{T}\left(f^*(x_t, a_t) - f^*(x_t, \pi_{f^*}(x_t))\right). \tag{71}$$

Remark 11.3. *One possible approach in the realizable setting is to use a functional version of UCB or successive elimination. This will require us to build confidence bands for the underlying mean functions which are valid with high probability. Constructing confidence bands for nonparametric function classes is a hard problem. There exists methods for constructing confidence bands for Holder Smooth Functions and this can likely be used to construct CB algorithms for this function class. We also took this approach for the class of univariate isotonic function class; see Chatterjee and Sen [10]. However, this approach seems hard to generalize to other nonparametric function classes. Although, it must be said that confidence bands which are valid for most parts of the context space $\mathcal{X}$ should be sufficient for the CB problem and this could be doable for other nonparametric function classes.*

11.4 SquareCB Algorithm

This algorithm was proposed in Foster and Rakhlin [12]. The goal of this section is to explain the essential idea behind this algorithm. The main message here is to say that one can reduce the contextual bandits problem to online regression.

11.4.1 Online Regression Oracle

Given a function class $\mathscr{F}$ consisting of functions $\mathcal{X} \times [K] \to [0, 1]$, the learner has access to an oracle that solves the following problem:

1. *The environment decides $z_t \in \mathcal{X} \times [K]$.*
2. *The oracle O_{sq} predicts $\hat{y}_t \in [0, 1]$.*
3. *The environment reveals $y_t \in [0, 1]$.*

The oracle O_{sq} can attain (for any possibly adaptively chosen) sequence $z_{1:T}$ and $y_{1:T}$, the following regret bound

$$\sum_{t=1}^{T}(\hat{y}_t - y_t)^2 - \min_{f \in \mathcal{F}}(f(z_t) - y_t)^2 \leq R(T) \tag{72}$$

where $R(T)$ is a $o(T)$ sequence.

This problem is open to the best of my knowledge for many nonparametric function classes. Indeed, this requirement on the oracle may be too strong for some function classes. For

example, in the case of univariate isotonic functions, online regression with sublinear regret would not be possible when the contexts are arbitrary; see [17], [18]. However, assuming a mild requirement on the context sequence such as they cover the entire context space could again make this problem viable for general nonparametric function classes.

In some simple cases we can give example of such an oracle.

- *When $\mathscr{F}$ is finite, this reduces to an expert problem. In this case, due to exp concavity of the square loss, one can show that Hedge can attain $R(T) = O(\log |\mathscr{F}|)$ which is a T independent rate.*
- *When $\mathscr{F}$ consists of K linear functions with the slopes $\beta_1, \ldots, \beta_k$ all with at most unit euclidean norm and $\mathcal{X}$ is the unit euclidean ball in $\mathbb{R}^d$, then one can check that by running K separate OGD's one can obtain a dimension independent regret bound $R(T) = \sqrt{TK}$ or we can run K separate VAW forecasters to obtain $R(T) = O(kd \log T)$.*

11.4.2 Motivating the Algorithm

Consider the following algorithmic framework. At time t, given context x_t, ask the oracle O_{sq} to predict the loss for each of the K arms; denote them by $\hat{y}_t(1), \ldots, \hat{y}_t(K)$. Then, based on these predictions, come up with a probability distribution over the arms $p_t \in S_K$, sample $a_t \sim p_t$, observe $l_t(a_t)$ and then feed the instance $z_t = (x_t, a_t)$ and $y_t = l_t(a_t)$ to the oracle. The main question that arises is how to choose p_t?

Note that this algorithm is a randomized algorithm and the next point $z_t = (x_t, y_t)$ is chosen based on the past history and hence we truly require (72) to hold for any adaptively chosen sequence $z_{1:T}$ and $y_{1:T}$.

What does (72) give us in this setting? We can simplify the expected (pseudo) regret of the oracle as follows:

$$\begin{aligned}
\max_{f \in \mathscr{F}} \mathbb{E} \sum_{t=1}^{T} (\hat{y}_t(a_t) - l_t(a_t))^2 - \sum_{t=1}^{T} (l_t(a_t) - f(x_t, a_t))^2 = \\
\mathbb{E} \sum_{t=1}^{T} (\hat{y}_t(a_t) - l_t(a_t))^2 - \sum_{t=1}^{T} (l_t(a_t) - f^*(x_t, a_t))^2 = \\
\mathbb{E} \sum_{t=1}^{T} \big(\hat{y}_t(a_t) - f^*(x_t, a_t)\big)\big(\hat{y}_t(a_t) + f^*(x_t, a_t) - 2l_t(a_t)\big) = \\
\mathbb{E} \sum_{t=1}^{T} \big(\hat{y}_t(a_t) - f^*(x_t, a_t)\big)^2 = \\
\mathbb{E} \sum_{t=1}^{T} \sum_{a=1}^{K} p_t(a) \big(\hat{y}_t(a) - f^*(x_t, a)\big)^2.
\end{aligned}$$

Therefore, the assumed pointwise regret bound of the oracle ensures that with our algorithmic framework (with whatever choice of p_t), the expected pseudo regret of the oracle

$$\underbrace{\mathbb{E}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\big(\hat{y}_t(a) - f^*(x_t,a)\big)^2}_{T_2} \leq R(T). \tag{73}$$

Coming back to the CB problem in the realizable setting recall that (71) we can write the expected regret and simplify it as follows:

$$\mathbb{E}\sum_{t=1}^{T}\big(f^*(x_t,a_t) - f^*(x_t,\pi_{f^*}(x_t))\big) = \underbrace{\mathbb{E}\sum_{t=1}^{T}\sum_{a=1}^{K} p_t(a)\big(f^*(x_t,a) - f^*(x_t,\pi_{f^*}(x_t))\big)}_{T_1}. \tag{74}$$

So far, we have still not shed light on how to choose p_t. Now we start indicating how p_t can be chosen. Note that we want to bound T_1 but what we have is a bound on T_2. So it is natural to want to bound T_1 by T_2. Perhaps what we want is to show is an inequality of the form

$$T_1 \leq \frac{\gamma}{4}T_2 + Rem \tag{75}$$

where $\gamma/4$ stands for some constant and Rem stands for remainder which we would like to make as small as possible.

To establish (75), it suffices to establish the analogous inequality for any fixed $t \in [T]$. So let us fix $t \in [T]$ and let us denote $f^*(x_t,a)$ by $\mu(a)$. Then the tth term in T_2 can be written as $p_t^T(\hat{y}_t - \mu)^2$ and the tth term in T_1 can be written as $p_t^T(\mu - \mu(a^*)\mathbf{1})$.

So it would suffice for us to show an inequality of the form

$$p_t^T(\mu - \mu(a^*)\mathbf{1}) \leq \frac{\gamma}{4}p_t^T(\hat{y}_t - \mu)^2 + Rem \tag{76}$$

and find an expression for the Rem term.

To this end, let us define the optimization problem

$$OPT(\hat{y}_t) = \inf_{p\in S_K}\max_{a^*\in[K]}\sup_{\mu\in\mathbb{R}^K} p^T(\mu - \mu(a^*)\mathbf{1}) - \frac{\gamma}{4}p^T(\hat{y}_t - \mu)^2$$

Observe that if we define p_t to be the argmin of the above optimization problem then (76) holds with $Rem = OPT(\hat{y}_t)$. Amazingly enough, the value of $OPT(\hat{y}_t)$ turns out to be small enough for our purposes and does not depend on $\hat{y}_t$. This is the content of the next lemma.

Lemma 11.2. *For any $\hat{y} \in \mathbb{R}^K$, we have $OPT(\hat{y}) = \frac{K-1}{\gamma}$ and the corresponding minimizer p^* can be expressed as*

$$p^*(a) = \frac{1}{\gamma\big(\hat{y}(a) + \lambda\big)}$$

where λ is the positive number that makes p^* a valid probability vector. This λ can be efficiently computed by binary search.

Proof. We can rewrite

$$OPT(\hat{y}) = \inf_{p \in S_K} \max_{a^* \in [K]} \sup_{\mu \in \mathbb{R}^K} (p - e_{a^*})^T \mu - \frac{\gamma}{4} p^T (\hat{y} - \mu)^2.$$

Let us first fix p and a^* and look at the innermost maximization over μ. This is a quadratic problem in μ and hence we can easily find the maximizer by setting the gradient to zero. A little bit of calculation will then give us

$$OPT(\hat{y}) = \inf_{p \in S_K} \max_{a^* \in [K]} \frac{1}{\gamma}\big(\frac{1}{p(a^*)} - 1\big) + (p - e_{a^*})^T \hat{y}.$$

To solve this minmax problem, first let us observe that for any fixed p:

$$\max_{a^* \in [K]} \frac{1}{\gamma}\big(\frac{1}{p(a^*)} - 1\big) + (p - e_{a^*})^T \hat{y} \geq \frac{1}{\gamma} \sum_{a=1}^{K} p(a)\big(\frac{1}{p(a)} - 1\big) = \frac{K-1}{\gamma}$$

where we obtained the inequality by taking expectation when $a^* \sim p$. The above therefore gives us a lower bound for $OPT(\hat{y})$. On the other hand, setting p such that $\frac{1}{\gamma}\big(\frac{1}{p(a)} - 1\big) + (p - e_a)^T \hat{y}$ is the same for all a will give us an equality in the previous display thereby asserting that indeed $OPT(\hat{y}) = \frac{K-1}{\gamma}$.

Therefore, an optimal p should satisfy that (by only considering the terms that depend on a)

$$\frac{1}{\gamma}\big(\frac{1}{p(a)}\big) - \hat{y}(a)$$

is the same for all $a \in [K]$ which then yields the expression for p^*. Note that because $\hat{y}(a) \in [0, 1]$ for all $a \in [K]$, a $\lambda \geq 0$ will exist which makes p^* a valid probability vector. Moreover, we can start with a really large candidate value of λ which will clearly make the sum of the entries of p^* smaller than 1, and then find the correct λ by binary search. So computing the value of λ is an easy problem. □

Actually the paper [12] proposes a slightly different choice of p_t which can be thought of as an approximate minimizer of the optimization problem. This is made precise in the next lemma.

Lemma 11.3. *Let*

$$b = \operatorname*{argmin}_{a \in [K]} \hat{y}_t(a).$$

Then define for all $a \neq b$*,*

$$p_t(a) = \frac{1}{\gamma\big(\hat{y}(a) - \hat{y}(b) + K\big)}.$$

Of course, $p_t(b) = 1 - \sum_{a \neq b} p_t(a)$*. Then, we have*

$$\max_{a^* \in [K]} \sup_{\mu \in \mathbb{R}^K} p_t^T(\mu - \mu(a^*)\mathbf{1}) - \frac{\gamma}{4} p_t^T(\hat{y}_t - \mu)^2 \leq 2\frac{K-1}{\gamma}.$$

Proof. As in the proof of the previous lemma, it suffices to consider the following term for any $a^* \in [K]$,

$$\frac{1}{\gamma}\big(\frac{1}{p(a^*)} - 1\big) + (p - e_{a^*})^T \hat{y}$$

where $p = p_t$.

Let's look at the second term first. We can write this term as

$$\begin{aligned}
&\sum_{a \in [K]} p(a)\hat{y}(a) - \quad \hat{y}(a^*) = \sum_{a \neq a^*} p(a)\big(\hat{y}(a) - \hat{y}(a^*)\big) = \\
&\sum_{a \neq a^*} p(a)\big(\hat{y}(a) - \hat{y}(b)\big) + \sum_{a \neq a^*} p(a)\big(\hat{y}(b) - \hat{y}(a^*)\big) = \\
&\sum_{a \neq a^*} p(a)\big(\hat{y}(a) - \hat{y}(b)\big) + (1 - p(a^*))\big(\hat{y}(b) - \hat{y}(a^*)\big) = \\
&\sum_{a \in [K]} p(a)\big(\hat{y}(a) - \hat{y}(b)\big) + \big(\hat{y}(b) - \hat{y}(a^*)\big) \leq \frac{K-1}{\gamma} + \big(\hat{y}(b) - \hat{y}(a^*)\big)
\end{aligned}$$

where in the last expression we used the definition of p_t.

Therefore, we can write

$$\frac{1}{\gamma}\big(\frac{1}{p(a^*)} - 1\big) + (p - e_{a^*})^T \hat{y} \leq \frac{1}{\gamma}\big(\frac{1}{p(a^*)} - 1\big) + \frac{K-1}{\gamma} + \big(\hat{y}(b) - \hat{y}(a^*)\big).$$

When $a^* \neq b$, by plugging in the expression for $p_t(a^*)$ we get $2\frac{K-1}{\gamma}$. When $a^* = b$, use $p_t(b) \geq \frac{1}{K}$ to again obtain that the above expression is at most $2\frac{K-1}{\gamma}$. □

11.4.3 SquareCB Algorithm and its Regret Bound

We now summarize the SquareCB algorithm.

Input*: Online Regression Oracle* O_{sq} *and parameter* $\gamma > 0$*. For* $t = 1, \ldots, T$ *do*

1. *Receive context* x_t*.*

2. *Ask Oracle O_{sq} to predict the loss for each of the K arms; denote them by $\hat{y}_t(1), \ldots, \hat{y}_t(K)$.*
3. *Compute p_t such that*
   - ***Option** 1:*

$$p_t(a) = \frac{1}{\gamma\big(\hat{y}(a) + \lambda\big)}$$

     where λ is found via binary search.
   - ***Option** 2: Let*

$$b = \operatorname*{argmin}_{a \in [K]} \hat{y}_t(a).$$

     Then define for all $a \neq b$,

$$p_t(a) = \frac{1}{\gamma\big(\hat{y}(a) - \hat{y}(b) + K\big)}$$

     and $p_t(b) = 1 - \sum_{a \neq b} p_t(a)$.
4. *Sample $a_t \sim p_t$, observe $l_t(a_t)$ and feed $(z_t, y_t) = ((x_t, a_t), l_t(a_t))$ to the Oracle O_{sq}.*

Theorem 11.4. *Under the realizable setting, the SquareCB algorithm defined above satisfies a regret bound = $O(\gamma R(T) + \frac{TK}{\gamma}) = O(\sqrt{TR(T)K})$ after optimizing over γ.*

Proof. The proof is immediate from our previous discussions. □

Let us now make some remarks regarding the SquareCB algorithm.

- *Note that since the CB problem in the realizable setting is a generalization of the stochastic MAB problem, the SquareCB algorithm gives an alternative algorithm for the usual stochastic MAB problem as well. What is interesting about this algorithm is that this bypasses the need to do inference or maintain confidence intervals. So in the usual MAB problem, the oracle would just predict based on the point estimates which are the running means of the arms and not construct confidence intervals. For general nonparametric function classes, constructing confidence bands may be harder than actually coming up with an online regression oracle or atleast that is the hope.*
- *Can a version of this algorithm be developed which works for heavy tailed noise? This seems an open problem.*
- *Do we need the full power of the assumption (72)? In the paper [12], a simpler assumption (tailored towards realizability) is given which suffices for the analysis of SquareCB to go through. This is the slightly weaker requirement that for every (possibly adaptively chosen) sequence $\{(x_t, a_t)_{t=1}^T\}$, we have*

$$\sum_{t=1}^{T} (\hat{y}_t - f^*(x_t, a_t))^2 \leq R(T).$$

References

[1] Agrawal, S. and N. Goyal (2012). Analysis of thompson sampling for the multi-armed bandit problem. In Conference on learning theory*, pp. 39–1. JMLR Workshop and Conference Proceedings.*

[2] Audibert, J.-Y., R. Munos, and C. Szepesvári (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. Theoretical Computer Science 410*(19), 1876–1902.*

[3] Auer, P., N. Cesa-Bianchi, and P. Fischer (2002). Finite-time analysis of the multiarmed bandit problem. Machine learning 47*(2), 235–256.*

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

[5] Bubeck, S. (2011). Introduction to online optimization. Lecture Notes 2.

[6] Bubeck, S. et al. (2015). Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning 8*(3-4), 231–357.*

[7] Bubeck, S., N. Cesa-Bianchi, et al. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Foundations and Trends® in Machine Learning 5*(1), 1–122.*

[8] Bubeck, S., N. Cesa-Bianchi, and G. Lugosi (2013). Bandits with heavy tail. IEEE Transactions on Information Theory 59*(11), 7711–7717.*

[9] Chatterjee, S. and S. Goswami (2022). Spatially adaptive online prediction of piecewise regular functions. arXiv preprint arXiv:2203.16587.

[10] Chatterjee, S. and S. Sen (2021). Regret minimization in isotonic, heavy-tailed contextual bandits via adaptive confidence bands. arXiv preprint arXiv:2110.10245.

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

[12] Foster, D. and A. Rakhlin (2020). Beyond ucb: Optimal and efficient contextual bandits with regression oracles. In International Conference on Machine Learning*, pp. 3199–3210. PMLR.*

[13] Freund, Y., R. E. Schapire, Y. Singer, and M. K. Warmuth (1997). Using and combining predictors that specialize. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 334–343.*

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

[15] Hazan, E., A. Agarwal, and S. Kale (2007). Logarithmic regret algorithms for online convex optimization. Machine Learning 69*(2), 169–192.*

[16] Johnson, R. and T. Zhang (2013). Accelerating stochastic gradient descent using predictive variance reduction. Advances in neural information processing systems 26.

[17] Kotłowski, W., W. M. Koolen, and A. Malek (2016). Online isotonic regression. In Conference on Learning Theory*, pp. 1165–1189. PMLR.*

[18] Kotlowski, W., W. M. Koolen, and A. Malek (2017). Random permutation online isotonic regression. Advances in Neural Information Processing Systems 30.

[19] Lai, T. L., H. Robbins, et al. (1985). Asymptotically efficient adaptive allocation rules. Advances in applied mathematics 6*(1), 4–22.*

[20] Lattimore, T. and C. Szepesvári (2020). Bandit algorithms*. Cambridge University Press.*

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

[22] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In Doklady an ussr*, Volume 269, pp. 543–547.*

[23] Netrapalli, P. (2019). Stochastic gradient descent and its variants in machine learning. Journal of the Indian Institute of Science 99*(2), 201–213.*

[24] Orabona, F. (2019). A modern introduction to online learning. arXiv preprint arXiv:1912.13213.

[25] Ouhamma, R., O.-A. Maillard, and V. Perchet (2021). Stochastic online linear regression: the forward algorithm to replace ridge. Advances in Neural Information Processing Systems 34*, 24430–24441.*

[26] Rakhlin, A. and K. Sridharan (2012). Statistical learning theory and sequential prediction. Lecture Notes in University of Pennsyvania 44.

[27] Rakhlin, A. and K. Sridharan (2013). Online learning with predictable sequences. In Conference on Learning Theory*, pp. 993–1019. PMLR.*

[28] Robbins, H. and S. Monro (1951). A stochastic approximation method. The annals of mathematical statistics*, 400–407.*

[29] Russo, D. and B. Van Roy (2016). An information-theoretic analysis of thompson sampling. The Journal of Machine Learning Research 17*(1), 2442–2471.*

[30] Shalev-Shwartz, S. et al. (2012). Online learning and online convex optimization. Foundations and Trends® in Machine Learning 4*(2), 107–194.*

[31] Slivkins, A. et al. (2019). Introduction to multi-armed bandits. Foundations and Trends® in Machine Learning 12*(1-2), 1–286.*

[32] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25*(3-4), 285–294.*

[33] Vovk, V. (1998). Competitive on-line linear regression. Advances in Neural Information Processing Systems*, 364–370.*

[34] Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936.*