Fundamentals of Convex Optimization
Lecture 2, Date: October 24, 2014
Instructor: Nisheeth Vishnoi
Scribe: Slobodan Mitrović

Multiplicative Weights Update vs. Gradient Descent

1 Overview

In the previous lecture we discussed gradient descent-type methods from convex optimization and showed how they can be straightforwardly translated into regret-minimizing algorithms in the online convex optimization setting. The key observation was that several gradient-descent type methods are oblivious to the fact that the same function is used in every round. Today we turn this idea on its head: namely, we begin with an algorithm that solves an online problem, and end up with a (new) convex optimization method. As a result, we can get improved dependence on the number of iterations when we are in settings other than the Euclidean space or have guarantees on the function on the Euclidian norm. For instance, suppose we know that $\|\nabla f(x)\|_\infty = O(1)$ which, at best, implies a bound $\|\nabla f(x)\|_2 = O(\sqrt{n})$. We obtain a gradient-descent type algorithm that takes $O(\log n/\varepsilon^2)$ iterations as opposed to $O(\sqrt{n}/\varepsilon^2)$ iterations if we use the algorithm from the previous lecture. The journey we take in deducing this result is long and rich. The central player is the old, famous and versatile: the Multiplicative Weights Update (MWU) method. In the process we also explain how to bet on stocks, how to solve linear programs approximately and quickly, and how MWU can be thought of as a method in convex optimization. We also hint how one may solve semi-definite programs approximately using a matrix variant of the MWU.

2 The Multiplicative Weights Update Method

MWU has appeared in the literature at least as early as the 1950s and has been rediscovered many times in many fields since then. It has applications in optimization (solving LPs), game theory, finance (portfolio optimization), machine learning (Winnow, AdaBoost, Hedge), theoretical computer science (devising fast algorithms for LPs and SDPs), and many more. We refer the reader to the survey of Arora, Hazan, and Kale [1] for a survey on Multiplicative Weight Update method.

To describe MWU, consider the following game. We want to invest in the stock market, i.e., to buy/sell stocks and make money. To simplify, assume that each day we can either buy or sell one share of one particular kind of stock. Now, our goal is to trade stock each day in order to maximize our revenue.

Unfortunately, we do not know much about the stock market. However, we have access to *n experts* that know something (or a lot) about the stock market. At the beginning of each day, each expert i advices us on whether we should buy or sell. Once we hear all of the advice, we decide what to do. At the end of each day we observe whether our profit went up or down. If the advice of an expert was incorrect (i.e., they said "buy" when the best option was "sell"), we say that the expert made a *mistake*.

Now, of course, experts may make mistakes, or may even be adversarial and collude against us. Nevertheless, our goal is to, based on advice given by the experts, do as *well as possible*; i.e., consider the expert $\star$ who gave the most correct advice, our aim is to do almost as good as expert $\star$.

To formalize the setup, we define m_i^t and M^t, for each expert i and each day t, as follows

$$m_i^t \stackrel{\text{def}}{=} \text{number of mistakes made by expert } i \text{ until day } t,$$

and

$$M^t \stackrel{\text{def}}{=} \text{number of mistakes made by us until day } t.$$

Having these definitions in hand, our goal is to ensure that the *regret*

$$\text{Regret}_t \stackrel{\text{def}}{=} M^t - m_\star^t$$

is minimized, where $m_\star^t = \min_i m_i^t$. Note that M^t may even be smaller than $m_\star^t$.

Let us first consider some obvious approaches to regret-minimization given the advice of experts. One natural strategy is to take advice from an expert who was right on the previous day. However, one can very easily design examples where this strategy will perform very poorly.[1] Alternately, we can consider *Majority Vote* in which we simply see the advice for a given day, and take the majority option. In other words, we will buy if and only if the majority of the experts say to buy. However, in this case we can also design examples where this algorithm performs poorly.

[1]For example, if half the experts are right on even days and wrong on odd days, and the other half or the experts are wrong on even days and right on odd days, we will always make the wrong choice.

2.1 The Weighted Majority Algorithm

In the two aforementioned approaches we were either too radical (i.e., ignoring experts who just made a mistake) or too forgiving (i.e., treating all experts equally regardless of their history). Instead, we could consider a strategy which combines the best from both the worlds by considering *Weighted Majority*. Here, we take an expert's past advice into consideration using weights, and take the weighted majority in order to determine which advice to follow.

Let w_i^t be the weight of expert i at the beginning of day t. We can also think of w_i^t as our confidence in expert i – the larger w_i^t the more confidence we have. We are unbiased in the beginning, and let

$$w_i^1 = 1, \quad \forall i.$$

If expert i is right on day t, then we do not change w_i^t, but otherwise penalise that expert. Formally, define f_i^t to be

$$f_i^t \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if expert } i \text{ makes a mistake on day } t \\ 0 & \text{otherwise} \end{cases}$$

Then, we set

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t(1 - \varepsilon f_i^t)$$

for an $\varepsilon > 0$. This introduction of ε is *crucial* and is a reflection of our trust in the predictions of the experts. $\varepsilon = 1$ would correspond to the setting when we know that there is some expert who is always correct. On the other hand, if we set $\varepsilon = 0$, we are discarding the history. Morally, this parameter plays the same role as the η-parameter in gradient-descent type methods and could depend on t and, in principle, be different for each expert. For now, we just think of ε as a small enough number.

How do we make a decision given these weights? Towards this, let $\Phi^t \stackrel{\text{def}}{=} \sum_i w_i^t$ denote the sum of all weights of the experts. On day t we decide to buy if the sum of the weights of experts saying to buy is at least $\Phi^t/2$, and sell otherwise. This completes the description of the Weighted Majority Algorithm (WMA).

How well does WMA perform? Before we delve into the proofs, let us state two fact that we will use repeatedly.

Fact 1. *For any x the following holds*

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

Fact 2. *For any $|x| \leq 1/2$ we have*

$$-\ln(1-x) \leq x + x^2.$$

Now we are ready to prove the following theorem.

Theorem 1. *After t steps, let m_i^t denotes the number of mistakes made by expert i and M^t the number of mistakes WMA had made, and let $0 < \varepsilon \leq 1/2$. Then, for every i we have*

$$M^t - 2(1+\varepsilon)m_i^t \leq \frac{2\ln n}{\varepsilon}.$$

In other words, the performance of the WMA is within a factor of $2(1+\varepsilon)$ of the best expert up to an additive factor of about $\ln n/\varepsilon$. If we consider the average after time t, i.e., M^t/t, the above theorem implies that if $t \geq 2\ln n/\varepsilon^2$, then the difference between the average mistakes made by the WMA is no more than an additive ε of that of the best expert. This is formalized in the following corollary:

Corollary 2. *After T days, let $m_\star^T$ be the number of mistakes the best expert has made and let M^T be the number of mistakes WMA has made. If $T \geq \frac{2\ln n}{\varepsilon^2}$ where $0 < \varepsilon \leq 1/2$, then*

$$\frac{1}{T}\left(M^T - 2(1+\varepsilon)m_\star^T\right) \leq \varepsilon.$$

Note that the corollary says that in the long run, i.e., for large T, the algorithm essentially makes at most two times more mistakes than the best expert in hindsight![2]

The proof is easy and relies on observing that every time the WMA makes a mistake, the total weight on the experts reduces by a multiplicative factor of $(1-\varepsilon/2)$.

Proof. Assume that we make a mistake on day t. Then, at least half of the total weight of the experts will get decreased by $(1-\varepsilon)$, otherwise the weighted majority would have told us to take the other option. Hence,

$$\Phi^{t+1} \leq \Phi^t\left(\frac{1-\varepsilon}{2} + \frac{1}{2}\right) = \Phi^t\left(1 - \frac{\varepsilon}{2}\right).$$

[2]Interestingly, the factor of 2 in the bound is optimal for a deterministic strategy. We will later see that randomisation can help.

Therefore, every time we make a mistake the total weight of the experts decreases by the multiplicative factor of $\left(1 - \frac{\varepsilon}{2}\right)$. Hence, we can upper bound Φ^{t+1} as follows

$$\Phi^{t+1} \leq \Phi^1 \left(1 - \frac{\varepsilon}{2}\right)^{M^t} = n \left(1 - \frac{\varepsilon}{2}\right)^{M^t} \leq n e^{-\varepsilon M^t/2}, \tag{1}$$

where we recall that all weights were initialized to 1 so $\Phi^1 = n$ and use Fact 1.

On the other hand, since we have that Φ^{t+1} is the sum of all the weights of the experts on day $t+1$, we also have the following lower bound $\forall i$

$$\begin{aligned} \Phi^{t+1} &\geq w_i^{t+1} \\ &= w_i^1 \left(1 - \varepsilon\right)^{m^t} \\ &= \left(1 - \varepsilon\right)^{m^t}, \end{aligned} \tag{2}$$

where again we use the fact that $w_i^1 = 1$.

Now, putting together the upper bound (1) and lower bound (2), we obtain

$$\left(1 - \varepsilon\right)^{m_i^t} \leq n e^{-\varepsilon M^t/2}.$$

Taking logarithms on both sides we get that

$$m_i^t \ln\left(1 - \varepsilon\right) \leq \ln n - \frac{\varepsilon}{2} M^t. \tag{3}$$

Using Fact 2, from (3) we conclude that

$$-\varepsilon(1+\varepsilon) m_i^t \leq \ln n - \frac{\varepsilon}{2} M^t,$$

which further implies that

$$M^t - 2(1+\varepsilon) m_i^t \leq \frac{2 \ln n}{\varepsilon}.$$

This gives a bound on the regret as desired. □

2.2 The Multiplicative Weights Update Algorithm

The algorithm we saw so far makes, up to a factor of two, optimal choices about trading a single item. Let us now consider a more general setting. As before, there are n experts, and following solely expert i's advice on day t incurs a cost f_i^t where $f^t : [n] \to [-1, 1]$. However, in many cases, as with stocks, we can take

fractional decisions. Hence, for a given vector $p^t \in \Delta_n$ which represents a convex combination of experts,[3] we incur a *loss* of $\langle p^t, f^t \rangle$.

Equivalently, we can think of p^t as a probability distribution from where we select a single expert x^t, and incur a loss of $f^t(x^t)$. In this case, the expected loss $\mathbb{E}_{x^t \sim p^t}[f^t(x^t)] = \langle p^t, f^t \rangle$. Furthermore, we compete against the best convex combination of the experts, and are interested in minimizing the *regret*

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle - \min_{p \in \Delta_n} \sum_{t=1}^{T} \langle p, f^t \rangle.$$

How do we update p^t? The strategy is similar to the WMA: maintain a weight w_i^t for the i-th expert where we start with $w_t^1 = 1$. Given the loss function f^t, the weights are updated as

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t (1 - \varepsilon f_i^t)$$

for a parameter $\varepsilon > 0$ which has the same intention as before. Since f_i^t could be negative or positive, the weight can increase or decrease. However, since $\|f_t\|_\infty \leq 1$ for all t, the weights always remain non-negative. Finally, since we need a probability distribution (or convex combination) $p^t \in \Delta_n$, we normalize the weight vector w^t by the total weight $\Phi^t \stackrel{\text{def}}{=} \sum_i w_i^t$ to obtain

$$p^t \stackrel{\text{def}}{=} \frac{w^t}{\Phi^t}.$$

We call this method the Multiplicative Weights Update (MWU) algorithm. We are now ready to prove the main theorem of this section. The only difference from 1 is that the factor of 2 goes away! The proof is similar to that of Theorem 1.

Theorem 3. *Assume that $\|f^t\|_\infty \leq 1$ for every t, and let $0 < \varepsilon \leq 1/2$. Then, the MWU algorithm produces a sequence of probability distributions $p^1, \ldots, p^T$ such that*

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle - \inf_{p \in \Delta_n} \sum_{t=1}^{T} \langle p, f^t \rangle \leq \frac{\ln n}{\varepsilon} + \varepsilon T. \tag{4}$$

Thus, the number of iterations after which the average regret becomes less than 2ε is at most $\frac{\ln n}{\varepsilon^2}$.

[3] A point p is in Δ_n if $\|p\|_1 = 1$ and $p_i \geq 0$ for all i.

Proof. We have

$$\Phi^{t+1} = \sum_i w_i^{t+1} = \sum_i w_i^t \left(1 - \varepsilon f_i^t\right).$$

Now, using the facts that $p_i^t = {}^{w_i^t}/_{\Phi^t}$, $\|p^t\|_1 = 1$ and $p_i^t \geq 0$, we rewrite the above equation as follows

$$\begin{aligned} \Phi^{t+1} &= \sum_i \left(p_i^t \Phi^t\right)\left(1 - \varepsilon f_i^t\right) \\ &= \Phi^t - \varepsilon \Phi^t \sum_i p_i^t f_i^t \\ &= \Phi^t \left(1 - \varepsilon \left\langle p^t, f^t \right\rangle\right). \end{aligned} \tag{5}$$

Following Fact 1, equality (5) can be written as

$$\Phi^{t+1} \leq \Phi^t e^{-\varepsilon \left\langle p^t, f^t \right\rangle}.$$

Therefore, after T rounds we have

$$\begin{aligned} \Phi^{T+1} &\leq \Phi^1 e^{-\varepsilon \sum_{t=1}^T \left\langle p^t, f^t \right\rangle} \\ &= n e^{-\varepsilon \sum_{t=1}^T \left\langle p^t, f^t \right\rangle}, \end{aligned} \tag{6}$$

since $\Phi^1 = n$.

Our next step is to provide a lower bound on Φ^{T+1}. As before, we observe that $\Phi^t \geq w_i^t$ for any t and any i, and obtain

$$\Phi^{T+1} \geq w_i^{T+1} = \prod_{t=1}^T (1 - \varepsilon f_i^t).$$

Using Fact 2 we can further write

$$\Phi^{T+1} \geq e^{-\varepsilon \sum_{t=1}^T f_i^t - \varepsilon^2 \sum_{t=1}^T \left(f_i^t\right)^2}. \tag{7}$$

Putting together the lower and the upped bound on Φ^{T+1}, i.e. equations (7) and (6), and taking logarithm we obtain

$$\ln n - \varepsilon \sum_{t=1}^T \left\langle p^t, f^t \right\rangle \geq -\varepsilon \sum_{t=1}^T f_i^t - \varepsilon^2 \sum_{t=1}^T \left(f_i^t\right)^2,$$

which after rearranging and dividing by ε gives

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle - \sum_{t=1}^{T} f_i^t \leq \frac{\ln n}{\varepsilon} + \varepsilon \sum_{t=1}^{T} \left(f_i^t\right)^2 \leq \frac{\ln n}{\varepsilon} + \varepsilon T.$$

The last inequality is true since $\|f_t\|_\infty \leq 1$. Since this is true for any i, by convexity we obtain that

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle - \sum_{t=1}^{T} \langle p, f^t \rangle \leq \frac{\ln n}{\varepsilon} + \varepsilon T$$

for all $p \in \Delta_n$. This completes the proof of the theorem. □

The width. What happens is instead of $\|f^t\|_\infty \leq 1$, we have $\|f^t\|_\infty \leq \rho$ (for all t) some $\rho > 1$? This parameter is often referred to as the *width* of the loss function. In this case, to maintain the non-negativity of the weights, the update rule must be modified to

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t \left(1 - \frac{\varepsilon f_i^t}{\rho}\right).$$

In this case, everything goes through as before except that the term $\frac{\ln n}{\varepsilon}$ in the regret bound in Theorem 1 becomes $\frac{\rho^2 \ln n}{\varepsilon}$. Thus, the number of iterations after which the average regret becomes less than 2ε is $\frac{\rho^2 \ln n}{\varepsilon^2}$.

2.3 Solving Linear Programs Approximately using MWU

In this section we illustrate one of many of the applications of the MWU algorithm: solving linear programs. Rather than solving the optimization version of linear programming, we chose a very simple setting to illustrate the powerful idea of using the MWU algorithm. The method has been used in numerous ways to speed up solutions to linear programs for fundamental problems. We consider the following feasibility problem: given an $m \times n$ matrix A, and $b \in \mathbb{R}^n$ does there exist an x such that $Ax \geq b$?

$$\exists ? x : Ax \geq b. \tag{8}$$

For this problem we give an algorithm that, given an error parameter $\varepsilon > 0$, either outputs a point x such that $Ax \geq b - \varepsilon \mathbb{1}$ or proves that there is no solution to this linear system of inequalities. We also assume the existence of an *oracle* that, given vector $p \in \Delta_n$, solves the following relaxed problem

$$\exists ? x : p^\top Ax \geq p^\top b. \tag{9}$$

Note that the above feasibility problem involves only *one* inequality. This, often, may be significantly easier algorithmically than the original feasibility problem. In any case, we will assume that. Clearly, if there is an x that satisfies (8), then x satisfies (9) as well for all $p \in \Delta_n$. On the other hand, if there is no solution to (9), then the system (8) is infeasible. Finally, we assume that when the oracle returns a feasible solution for a p, the solution x that it returns is not arbitrary but has bounded *width*:

$$\max_i |(Ax)_i - b_i| \leq 1.$$

In this setting, we can prove the following theorem:

Theorem 4. *If there is a solution to* (8)*, then there is an algorithm that outputs a x which satisfies the system* (8) *up to an additive error of* 2ε*. The algorithm makes at most $\frac{\ln m}{\varepsilon^2}$ calls to a width-bounded oracle for the problem 9. On the other hand, if there is no solution to* (8)*, then the algorithm says so.*

The algorithm in the proof of the theorem is the MWU: we just have to identify the *loss* function at time t. We maintain a probability distribution p^t at any time t and pass it to the oracle. As is usual, the starting probability distribution, p^1, is uniform. We pass this p^t to the oracle. If the oracle returns that the system 9 is infeasible, the algorithm returns that the system (8) is infeasible. As feasibility is preserved under taking convex combinations, we know that the algorithm is correct. On the other hand, if the oracle returns an x^t, we set the loss function to

$$f^t \stackrel{\text{def}}{=} Ax^t - b.$$

Since $|(Ax^t)_i - b_i| \leq 1$, $\|f^t\|_\infty \leq 1$. Finally, if the algorithm succeeds for $T \stackrel{\text{def}}{=} \frac{\ln m}{\varepsilon^2}$ iterations, then we let

$$\tilde{x} \stackrel{\text{def}}{=} \frac{1}{T} \sum_t x^t.$$

Thus, to prove the theorem, it is sufficient to prove that $\tilde{x}$ satisfies (8) up to an additive error of 2ε.

Towards that end, we begin by observing that, since x^t is feasible for the system (9) for p^t for every t, we have

$$\left\langle p^t, f^t \right\rangle = \left\langle Ax^t - b, p^t \right\rangle = \left(p^t\right)^\top Ax^t - \left(p^t\right)^\top b \geq 0.$$

Thus, the loss at every step is at least 0. Hence, from Theorem 3, for the choice of $T = \ln m/\varepsilon^2$ we obtain that for every i, $-\frac{1}{T}\sum_{t=1}^T f_i^t \leq 2\varepsilon$. This is the same as

$$-\frac{1}{T}\sum_{t=1}^{T}\left((Ax^t)_i - b_i\right) \leq 2\varepsilon.$$

Now, using the definition of $\tilde{x} = \frac{1}{T}\sum_t x^t$, we obtain that for all i,

$$(A\tilde{x})_i \geq b_i - 2\varepsilon.$$

Thus, $\tilde{x}$ is 2ε-feasible for (8). This concludes the proof of Theorem 4.

3 Multiplicative Weights Update as Gradient Descent

Now we begin our journey towards using the ideas developed in the previous sections to give a new algorithm for convex optimization. In this section we first see how the MWU algorithm is in fact a form of gradient descent. The connection is via the entropy function.

For a non-negative vector w, let $H(w)$ denote the function

$$H(w) \stackrel{\text{def}}{=} \sum_i w_i \ln w_i.$$

(This function is the negative entropy when $w \in \Delta_n$.) The gradient of $H(w)$ is the vector $x(w)$ where

$$x_i = (1 + \ln w_i)_i.$$

Thus, if w varies over the non-negative orthant, $x(w)$ varies over $\mathbb{R}^n$. Moreover, this map is invertible: given $x \in \mathbb{R}^n$, one can find a w such that $x = x(w)$:

$$w_i = e^{x_i - 1}.$$

Thus, at iteration t, for w^t we let $x^t \stackrel{\text{def}}{=} x(w^t)$. The loss function is f^t, and the loss is $F^t(w) \stackrel{\text{def}}{=} \langle f^t, w\rangle$. Thus, $\nabla_w F^t = f^t$. Thus, our assumption that $\|f^t\|_\infty \leq 1$ is the same as

$$\|\nabla_w F^t\|_\infty \leq 1.$$

Suppose we take the gradient step of size η in the x-space with respect to $\nabla_w F^t$. Then, we obtain a new point in the x-space:

$$x^{t+1} = x^t - \eta \nabla_w F^t = x^t - \eta f^t.$$

The corresponding w^{t+1} is obtained by equating, for each i,

$$1 + \ln w_i^{t+1} = 1 + \ln w_i^t - \eta f_i^t.$$

Exponentiating, the update step in the w-space is

$$w_i^{t+1} = w_i^t e^{-\eta f_i^t}.$$

While this is not quite the update in the MWU algorithm, it is the *Hedge* variant of the MWU. For these updates we can derive the same regret bound as Theorem 3. It is the same as the MWU algorithm when $\eta f_i^t \ll 1$ for all t, i since we can then use the approximation $e^{-x} \approx 1 - x$, giving us the familiar update step:

$$w_i^{t+1} = w_i^t(1 - \eta f_i^t).$$

Thus, if we allow going back and forth to a space via the gradient, there is a function, namely the negative entropy function, for which the MWU algorithm is nothing but a gradient-descent type method! How general is this paradigm?

3.1 A Gradient Descent Method Inspired from MWU

It turns out that there is a method for convex optimization that can be abstracted out from the previous section which works well beyond the MWU setting. As we mentioned in the beginning of this lecture, the method we will derive could give better convergence rate than doing the usual gradient descent in the ambient space. We are back in the setting where we are given a convex function f and a convex set $K \subseteq K'$, and the goal is to find a point $x \in K$ such that

$$f(x) - f(x^\star) \leq \varepsilon.$$

(The reader can keep in mind $K = \Delta_n$ and $K' = \mathbb{R}^n_{\geq 0}$ to draw the analogy with the previous section.) Assume that the gradient of f is bounded by 1 with respect to some norm $\|\cdot\|$. (This corresponds to the assumption $\|f^t\|_\infty \leq 1$ for all t in the MWU setting.) Just like we chose the negative entropy map H in the previous section, the method depends on a map M which is assumed to be continuously differentiable and strictly convex function M over K'. Further, similar to H, the map M should have additional properties (we do not spell them all out): the map $\nabla M : K' \mapsto S$ should be one-to-one and invertible. ($S = \mathbb{R}^n$ in the previous section.) We can now generalize the algorithm in the previous section.

The initial point x^1 is chosen to be

$$x^1 \stackrel{\text{def}}{=} \arg\inf_{x \in K} M(x).$$

(For negative entropy, this results in the choice of the vector where all the coordinates are the same, i.e., the uniform distribution over the experts or the vector $\frac{1}{n}\mathbb{1}$). Let $x^t \in K$. Once we map x^t to $\nabla M(x^t)$ and do the gradient step, we may move out of K but should remain in K'. Let y^{t+1} be the point in K' which corresponds to

the resulting point in S. In other words, given x^t and ∇f, for $t \geq 1$ we let y^{t+1} be the point such that

$$\nabla M\left(y^{t+1}\right) = \nabla M\left(x^t\right) - \eta \nabla f\left(x^t\right). \tag{10}$$

As y^{t+1} might lie outside of K, we project it back to K and let x^{t+1} be its projection. We obtain the projection by minimizing the Bregman divergence as follows

$$x^{t+1} \stackrel{\text{def}}{=} \arg\inf_{x \in K} D_M\left(x, y^{t+1}\right).$$

Recall that the *Bregman divergence* $D_M(x, y)$ associated with M for points $x, y \in K'$ is defined to be

$$D_M(x, y) \stackrel{\text{def}}{=} M(x) - M(y) - \langle \nabla M(y), x - y \rangle.$$

(In the MWU setting, the Bregman divergence of the negative entropy function corresponds to the Kullback-Liebler divergence and projecting according to it is nothing but normalizing a non-negative vector to make its ℓ_1 norm 1.) In this setting, we can prove the following theorem which is a generalization the MWU method (and from which we can recover the MWU method):

Theorem 5. *Let the gradient of f be bounded in a norm $\|\cdot\|$ by G. Let M be a map satisfying the properties listed above and, in addition, is l-strongly convex with respect to norm $\|\cdot\|_\star$. Let $D \stackrel{\text{def}}{=} \sup_{x \in K} M(x) - M(x^1)$. Then, for $\eta \stackrel{\text{def}}{=} \frac{D}{\sqrt{T}}$, the algorithm above produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \leq \varepsilon$$

where

$$T = \frac{1}{l} \cdot \left(\frac{DG}{\varepsilon}\right)^2.$$

To complete the analogy to the MWU method in the previous section, we note that if we let $x^1 \stackrel{\text{def}}{=} \frac{1}{n}\mathbb{1}$, then $D = \sqrt{\ln n}$. Further, Pinsker's inequality implies that the negative entropy function is 1-strongly convex with respect to the $\|\cdot\|_1$ norm, the dual norm to $\|\cdot\|_\infty$. The proof of this theorem is left as an exercise. No new idea, other than those discussed in this lecture and the last are required to complete the proof.

Thus, we have achieved the goal promised in the beginning of the lecture. In conclusion, we started with the MWU algorithm in the online convex optimization setting, interpreted it as a gradient descent method and then abstracted the essence to obtain a new method in convex optimization. This method is often referred to as *Mirror Descent* or *Dual Averaging* algorithm. In the appendix we present a matrix version of the MWU method which has applications to approximately solving SDPs quickly.

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

A A Multiplicative Weights Update for Matrices

In this section[4] we discuss the generalization of our MWU framework to the matrix world. As before we have n experts. So far, the decision for round t was essentially the index i of the expert we follow in this round. After this decision is made, the adversary reveals the cost vector $f^t \in [-1,1]^n$, and we pay f_i^t. This is equivalent to choosing on every round t a vector $v^t \in \{e_1, e_2, ..., e_n\}$ and having loss $\langle v^t, f^t \rangle$. Recall that the strategy we devised for the game was randomized and our v^t was a random vector chosen according to the distribution $P[v^t = e_i] = p_i^t$, where p^t was the probability distribution at round t. Then the expected loss was simply $\sum_i p_i f_i^t = \langle p^t, f^t \rangle$.

In the matrix case we allow the decision v^t to be any vector of ℓ_2-norm 1 (i.e. vector from unit sphere S^{n-1}). The loss at round t is given by

$$\langle v^t, F^t v^t \rangle = (v^t)^\top F v^t,$$

where F^t is the loss matrix chosen by the adversary right after our decision at round t. We assume that this matrix is symmetric. Similarly to the basic case, where we had the requirement $|f_i^t| \leq 1$, we impose a bound on the cost matrix: $\|F^t\| \leq 1$ ($\|\cdot\|$ is the spectral norm, i.e. we require that the eigenvalues of F^t lie all in the interval $[-1,1]$). Observe that in our previous setting all the loss matrices were diagonal and we could pick our decision only from a finite set $\{e_1, e_2, ..., e_n\} \subseteq \mathrm{S}^{n-1}$. Like before we will give a randomized strategy for this online problem. Our goal is, of course, to minimize the expected total loss comparing

[4] Scribed by Damian Straszak.

to the best expert. However, now the set of experts is much bigger, it consists of all the unit length vectors.

Suppose our algorithm chooses the vector $v \stackrel{\text{def}}{=} v^t$ according to distribution $\mu \stackrel{\text{def}}{=} \mu^t$ and the cost matrix is $F \stackrel{\text{def}}{=} F^t$. We calculate the loss at round t:

$$\mathbb{E}_{v\sim\mu}[\langle v, Fv\rangle] = \mathbb{E}_{v\sim\mu}[(vv^T) \bullet F] = \mathbb{E}_{v\sim\mu}[vv^T] \bullet F$$

where $A \bullet B \stackrel{\text{def}}{=} \text{Tr}(A^T B)$ is the usual matrix inner product. So if we denote $P^t = \mathbb{E}_{v\sim\mu}[vv^T]$, then the loss at round t is simply $P^t \bullet F^t$. Observe that P^t is PSD and $\text{Tr}(P^t) = 1$. Our algorithm, instead of working directly with a distribution over $\mathbb{S}^{n-1}$, will keep a matrix P^t. Our objective is to ensure that $\sum_{t=1}^T P^t \bullet F_t$ as small as possible, compared to $\min_{\|w\|_2=1} \sum_{t=1}^T w^T F^t w$. The latter is nothing but the smallest eigenvalue of $\sum_{t=1}^T F^t$. We proceed with the description of our algorithm based on MWU.

Matrix Multiplicative Weights Update (MMWU):

1. Initialize $Q^1 = I$.
2. In round t, use the matrix
$$P^t = \frac{Q^t}{\Phi^t},$$
where $\Phi^t \stackrel{\text{def}}{=} \text{Tr}(Q^t)$.
3. Observe the loss matrix F^t, and update Q^t as follows
$$Q^{t+1} = e^{-\varepsilon \sum_{k=1}^t F^k}.$$
where: $e^A \stackrel{\text{def}}{=} \sum_{k=0}^\infty \frac{A^k}{k!}$

We will now prove an analogue of Theorem 3 in the matrix world.

Theorem 6. *Assume that $-I \preceq F^t \preceq I$ for every t, and let $0 < \varepsilon \leq 1$. Then, the algorithm MMWU produces a sequence of matrices $P^1, \ldots, P^T$ with $\text{Tr}(P^t) = 1$ such that:*

$$\sum_{t=1}^T P^t \bullet F^t - \inf_{v\in\mathbb{S}^{n-1}} \sum_{t=1}^T v^T F^t v \leq \frac{\ln n}{\varepsilon} + \varepsilon T. \tag{11}$$

Thus, after $T = \frac{\ln n}{\varepsilon^2}$,

$$\frac{1}{T}\sum_{t=1}^{T} P^t \bullet F^t \leq \lambda_{\min}\left(\frac{1}{T}\sum_{t=1}^{T} F^t\right) + 2\varepsilon.$$

Thus, if we know that $P^t \bullet F^t \geq 0$ for all t, then

$$-2\varepsilon I \preceq \frac{1}{T}\sum_{t=1}^{T} F^t.$$

This is a powerful implication and has far reaching consequences when one carefully constructs the setting, just as in Section 2.3. For instance, this theorem is sufficient to construct fast and approximate SDP solvers and show the existence of near-linear spectral sparsifiers. We omit the details.

The proof has the same structure as the proof of Theorem 3. However, in the matrix world where we are plagued with non-commutativty: $AB \neq BA$ in general for matrices. For instance it would have been great if $e^{A+B} = e^A e^B$. But this is false in general. What is true instead is the following fact which is known as the Golden-Thompson inequality:

Fact 3. *Let A, B be symmetric matrices, then:*

$$\mathrm{Tr}\left(e^{A+B}\right) \leq \mathrm{Tr}\left(e^A e^B\right).$$

We note that this is not true for 3 matrices A, B, C! We also need some simple inequalities necessary in the proof which essentially are a consequence of the corresponding scalar inequalities.

Fact 4. *Let A, B, C be symmetric matrices, let A be PSD and $B \preceq C$, then:*

$$\mathrm{Tr}(AB) \leq \mathrm{Tr}(AC).$$

Fact 5. *Let A be a symmetric matrix and v be a vector of unit length, then:*

$$e^{v^\top A v} \leq \mathrm{Tr}\left(e^A\right).$$

Proof. Suppose the eigenvalues of A are $\lambda_1 \leq \cdots \leq \lambda_n$. The eigenvalues of e^A are precisely $e^{\lambda_1}, \ldots, e^{\lambda_n}$. We know that $v^\top A v \leq \lambda_n$. Therefore $e^{v^\top A v} \leq e^{\lambda_n} \leq \mathrm{Tr}\left(e^A\right)$. □

Fact 6. *Let A be a symmetric matrix satisfying $\|A\| \leq 1$, then:*

$$e^{-A} \preceq I - A + A^2.$$

This is easy to see because the eigenspaces of all the matrices $A, A^2, A^3, ..., e^A$ are the same, so it is enough to prove the above inequality for A being a number in the interval $[-1, 1]$. Now we are ready to prove the theorem.

Proof. As in the proof of Theorem 3 we try to obtain upper and lower bounds on the potential function Φ^{t+1}, which in our case is defined to be $\Phi^{t+1} \stackrel{\text{def}}{=} \text{Tr}(Q^{t+1})$. Let us start with the upper bound:

$$\begin{aligned}
\Phi^{t+1} &= \text{Tr}(Q^{t+1}) = \text{Tr}\left(e^{-\varepsilon \sum_{k=1}^{t} F^k}\right) \\
&\stackrel{G-T}{\leq} \text{Tr}\left(e^{-\varepsilon \sum_{k=1}^{t-1} F^k} e^{-\varepsilon F^t}\right) \\
&\leq \text{Tr}\left(e^{-\varepsilon \sum_{k=1}^{t-1} F^k}\left(I - \varepsilon F^t + \varepsilon^2 (F^t)^2\right)\right) \\
&= \text{Tr}\left(Q^t\right) - \varepsilon \text{Tr}\left(Q^t F^t\right) + \varepsilon^2 \text{Tr}\left(Q^t (F^t)^2\right).
\end{aligned}$$

Now we use the fact that $Q^t = P^t \text{Tr}(Q^t)$:

$$\text{Tr}\left(Q^t F^t\right) = \text{Tr}(Q^t)\text{Tr}\left(P^t F^t\right) = \Phi^t (P^t \bullet F^t)$$

Similarly:

$$\text{Tr}\left(Q^t (F^t)^2\right) = \text{Tr}(Q^t)\text{Tr}\left(P^t F^t\right) = \Phi^t (P^t \bullet (F^t)^2).$$

Therefore we can conclude that:

$$\Phi^{t+1} \leq \Phi^t (1 - \varepsilon (P^t \bullet F^t) + \varepsilon^2 (P^t \bullet (F^t)^2)) \leq \Phi^t e^{-\varepsilon (P^t \bullet F^t) + \varepsilon^2 (P^t \bullet (F^t)^2)}.$$

Expanding further,

$$\Phi^{t+1} \leq n e^{-\varepsilon \sum_{k=1}^{t} (P^k \bullet F^k) + \varepsilon^2 \sum_{k=1}^{t} (P^k \bullet (F^k)^2)}. \tag{12}$$

It remains to obtain a suitable lower bound for Φ^{t+1}. To this end let us fix any vector v with $\|v\|_2 = 1$. We use Fact 5 with $A = -\varepsilon \sum_{k=1}^{t} F^k$:

$$e^{-\varepsilon \sum_{k=1}^{t} v^\top F^k v} \leq \text{Tr}\left(e^{-\varepsilon \sum_{k=1}^{t} F^k}\right) = \Phi^{t+1}. \tag{13}$$

Putting inequalities 12 and 13 together and taking logarithms of both sides yields

$$-\varepsilon \sum_{k=1}^{t} v^{\top} F^k v \leq -\varepsilon \sum_{k=1}^{t} (P^k \bullet F^k) + \varepsilon^2 \sum_{k=1}^{t} (P^k \bullet (F^k)^2) + \ln n.$$

After dividing by ε and rearranging, we get for every $v \in \mathbb{S}^{n-1}$:

$$\sum_{k=1}^{t} (P^k \bullet F^k) \leq \sum_{k=1}^{t} v^T F^k v + \varepsilon \sum_{k=1}^{t} (P^k \bullet (F^k)^2) + \frac{\ln n}{\varepsilon}.$$

□