CS 435, 2018
Lecture 4, Date: 15 March 2018
Instructor: Nisheeth Vishnoi

Mirror Descent and the Multiplicative Weight Update Method

In this lecture, we derive a new optimization algorithm – called Mirror Descent – via a different local optimization principle. First, the Mirror Descent algorithm is developed for optimizing convex functions over the probability simplex. Subsequently, we show how to generalize it and, importantly, derive the Multiplicative Weights Update (MWU) algorithm from it. This latter algorithm is then used to present a fast approximate algorithm to solve the bipartite matching problem on graphs.

Contents

1 A Local Optimization Principle and Regularizers

Consider a convex program

$$\min_{x \in K} f(x), \tag{1}$$

where $f : K \to \mathbb{R}^n$ is a convex function over a convex set. In the last lecture we introduced the (projected) gradient descent algorithm and proved that when f has Lipschitz gradient then it can solve the problem (1) up to precision ε in time proportional to $\frac{1}{\varepsilon}$.

This setting includes several important classes of functions f, such as quadratic functions $f(x) = x^\top Ax + x^\top b$ or squared distances to convex sets $f(x) = \text{dist}^2(x, K')$ (for some convex $K' \subseteq \mathbb{R}^n$). However, it leaves out convex functions which are non-differentiable, such as the ℓ_1 norm $f(x) = \sum_{i=1} |x_i|$ or the ℓ_∞ norm $f(x) = \max(|x_1|, \ldots, |x_n|)$. Such functions are still somewhat well-behaved (though in a different sense) and appear in applications very frequently.

To construct an algorithm for optimizing such functions, we introduce a general idea. Our algorithm will be iterative, given points[1] $x^1, x^2, \ldots, x^t$ it finds a new point x^{t+1} based on its history. How shall we choose the next point x^{t+1} to converge to the minimizer $x^\star$ quickly? An obvious choice would be to take

$$x^{t+1} = \operatorname*{argmin}_{x \in K} f(x),$$

it certainly converges to $x^\star$ quickly (in one step), yet clearly it is not very helpful because then x^{t+1} **is hard to compute**. To counter this problem one might try to construct a function f_t – a "simple model" of f – that **approximates** f in a certain sense and is **easy to minimize**. Then the update rule of our algorithm becomes

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

If the approximation f_t of f becomes more and more accurate with increasing t, then intuitively, the sequence of iterates should converge to the minimizer $x^\star$.

One can view the gradient descent algorithm for Lipschitz gradient we introduced last time to be of this type, where

$$f_t(x) = f(x^t) + \left\langle \nabla f(x^t), x - x^t \right\rangle + L \left\| x - x^t \right\|^2.$$

If f has L-Lipschitz gradient then $f(x) \leq f_t(x)$ for all $x \in K$ and moreover $f^t(x)$ is a good approximation of $f(x)$ for x in a small neighborhood around x^t. In this setting (as proved in the previous lecture), such an update rule guarantees convergence of x^t to the global minimizer.

[1] In this lecture, we index points produced by an algorithm using upper-indices: $x^1, x^2, \ldots, x^t$. This is to avoid confusion with coordinates of these vectors.

In general, when the functions we deal with are not differentiable, we might not be able to construct such good quadratic approximations. However, we might still use first order approximations of f at x^t. Convexity implies that if we define

$$f_t(x) := f(x^t) + \left\langle \nabla f(x^t), x - x^t \right\rangle$$

then

$$\forall_{x \in K} \quad f_t(x) \leq f(x),$$

and moreover we expect f_t to be a decent approximation of f in a small neighborhood around x^t. One could thus try to apply the resulting update rule

$$x^{t+1} = \operatorname*{argmin}_{x \in K} \left\{ f(x^t) + \left\langle \nabla f(x^t), x - x^t \right\rangle \right\}. \tag{2}$$

An easy to spot downside of the above is that it is way too aggresive – in fact the new point x^{t+1} might be very far away from x^t. This is easily illustrated by an example in one dimension, when $K = [-1, 1]$ and $f(x) = x^2$. Then the algorithm (2) jumps between -1 and 1 indefinitely since one of these two points is always a minimizer of a linear lower bound of f over K. Thus the sequence $\{x^t\}_{t \in \mathbb{N}}$ never reaches 0 – the unique minimizer of f.

The situation is even worse when the domain K is unbounded: the minimum is not attained at any finite point and hence the above update rule is not well defined! This issue can be easily countered when the function is α-strongly convex[2] for some $\alpha > 0$, since then we can use a stronger – quadratic lower bound on f at x^t, i.e. use

$$f_t(x) = f(x^t) + \left\langle \nabla f(x^t), x - x^t \right\rangle + \frac{\alpha}{2} \left\| x - x^t \right\|_2^2.$$

Then, the minimizer of $f_t(x)$ is always attained at a point x which is not too far away from x^t.

This observation leads to the following idea: Even if the function f is not strongly convex, as in some of the examples mentioned earlier on, we can still add a function to f_t, to make it behave nicer! More specifically, we add a term involving a distance function $D : K \times K \to \mathbb{R}$ that does not allow the new point x^{t+1} to land far away from the previous point x^t; and D is referred to as a *regularizer*. More precisely, instead of minimizing $f_t(x)$ we minimize $D(x, x^t) + f_t(x)$. To vary the importance of these two terms we also introduce a positive parameter $\eta > 0$ and write the revised update strategy as:

$$x^{t+1} = \operatorname*{argmin}_{x \in K} \left\{ D(x, x^t) + \eta \left(f(x^t) + \left\langle \nabla f(x^t), x - x^t \right\rangle \right) \right\}.$$

[2]We say that a twice-differentiable function $f : K \to \mathbb{R}$ is α-strongly convex if its Hessian matrix $\nabla^2 f(x)$ satisfies $\alpha I \preceq \nabla^2 f(x)$ for every $x \in K$.

By ignoring constant term, this can be also simply written as

$$x^{t+1} = \operatorname*{argmin}_{x \in K} \left\{ D(x, x^t) + \eta \left\langle \nabla f(x^t), x \right\rangle \right\}. \tag{3}$$

Note that by picking large η, the significance of the regularizer $D(x, x^t)$ is reduced and thus it does not play a big role in choosing the next step. By picking η very small we force x^{t+1} to stay in a close vicinity of x^t.[3]

Before we go any further with these general considerations, let us consider one important example, where the "right" choice of the distance function $D(\cdot, \cdot)$ provides a quite interesting result – the Exponential Gradient Descent algorithm.

2 Mirror Descent for the Simplex

2.1 Exponential Gradient Descent

Consider a convex optimization problem over the probability simplex

$$\Delta_n := \{p \in [0,1]^n : \sum_{i=1}^n p_i = 1\},$$

i.e., the set of all probability distributions over n elements

$$\min_{p \in \Delta_n} f(p), \tag{4}$$

where $f : \Delta_n \to \mathbb{R}$ is a convex function. Recall the general form of an algorithm we would like to construct

$$p^{t+1} = \operatorname*{argmin}_{p \in \Delta_n} (D(p, p^t) + \eta \left\langle \nabla f(p^t), p \right\rangle), \tag{5}$$

where D is a certain distance function on Δ_n. The choice of D is up to us, but ideally it should allow efficient computation of x^{t+1} given x^t and $\nabla f(x^t)$ and, intuitively, should be a "natural" metric – compatible with the geometry of the feasible set Δ_n, so as to guarantee quick convergence. For the simplex, perhaps the most natural choice of such a metric is the relative entropy, or the Kullback-Leibler (KL) divergence.

Definition 1. *For two probability distributions $p, q \in \Delta_n$, their Kullback-Leibler divergence is defined as*

$$D_{KL}(p, q) := -\sum_{i=1}^n p_i \log \frac{q_i}{p_i}.$$

[3] A slightly different, yet related way to unsure that the next point x^{t+1} does not move too far away from x^t is to first compute a candidate $\tilde{x}^{t+1}$ according to the rule (2) and then to make a small step from x^t towards $\tilde{x}^{t+1}$ to obtain x^{t+1}. This is the main idea of the Frank-Wolfe algorithm.

While not being symmetric, D_{KL} satisfies several natural distance-like properties. For instance, from convexity it follows that $D_{KL}(p,q) \geq 0$. In the the next section we derive several other properties of D_{KL}.

When specialized to this particular distance function the update rule takes the form

$$p^{t+1} = \operatorname*{argmin}_{p \in \Delta_n} (D_{KL}(p, p^t) + \eta \left\langle \nabla f(p^t), p \right\rangle). \tag{6}$$

As we prove in the lemma below, the vector p^{t+1} can be computed using an explicit formula involving p^t and $\nabla f(p^t)$ only.

Lemma 2. *Consider any vector $q \in \mathbb{R}^n_{\geq 0}$ and a vector $g \in \mathbb{R}^n$.*

1. *Let $w^\star := \operatorname{argmin}_{w \geq 0} \{D_{KL}(w,q) + \eta \left\langle g, w \right\rangle\}$, then[4] $w^\star_i = q_i \exp(-\eta g_i)$ for all $i = 1, 2, \ldots, n$.*
2. *Let $p^\star = \operatorname{argmin}_{p \in \Delta_n} \{D_{KL}(p,q) + \eta \left\langle g, p \right\rangle\}$, then $p^\star = \frac{w^\star}{\|w^\star\|_1}$ for all $i = 1, 2, \ldots, n$.*

Proof. In the first part we are given an unconstrained optimization problem of the form

$$\min_{w \geq 0} \quad \sum_{i=1}^n w_i \log w_i + \sum_{i=1}^n w_i (\eta g_i - \log q_i - 1). \tag{7}$$

The above problem is in fact convex, hence one just needs to find a zero of the gradient to find the minimum. By computing the gradient, we obtain the following optimality condition

$$\log w_i = -\eta g_i + \log q_i,$$

and hence $w^\star_i = q_i \exp(-\eta g_i - 1)$.

Let us now proceed with the second part. To incorporate the constraint $\sum_{i=1}^n p_i = 1$ we introduce a Lagrange multiplier $s \in \mathbb{R}$ to (7) and obtain

$$\min_{p \geq 0} \quad \sum_{i=1}^n p_i \log p_i + \sum_{i=1}^n p_i (\eta g_i - \log p_i) + s \left(\sum_{i=1}^n p_i - 1 \right). \tag{8}$$

Then, the optimality condition becomes

$$p_i = q_i \exp(-\eta g_i - 1 - s),$$

and thus we just need to pick s so that $\sum_{i=1}^n p_i = 1$, which leads us to $p^\star = \frac{w^\star}{\|w^\star\|_1}$. □

[4]Here we consider the KL-divergence between arbitrary nonnegative vectors $x, y \geq 0$, $D_{KL}(x,y) := -\sum_{i=1}^n x_i \log \frac{y_i}{x_i} + \sum_{i=1}^n (y_i - x_i)$. See the subsequent section for more details.

We provide details and summarize our algorithm in the the form of pseudocode.

Exponential Gradient Descent (EGD):
Input: First order oracle to a convex function $f : \Delta_n \to \mathbb{R}$, a parameter $\eta > 0$, and an integer $T > 0$.
Output: A point $\bar{p} \in \Delta_n$.

1. Initialize $p^1 = \frac{1}{n}\mathbb{1}$ (the uniform distribution).
2. For each iteration $t = 1, 2, \ldots, T$:
   - Obtain $g^t = \nabla f(p^t)$.
   - Let $w^{t+1} \in \mathbb{R}^n$ and $p^{t+1} \in \Delta_n$ be defined as
   $$w_i^{t+1} := p_i^t e^{-\eta g_i^t}, \qquad p_i^{t+1} := \frac{w_i^{t+1}}{\sum_{j=1}^n w_j^{t+1}}.$$
3. Output $\bar{p} := \frac{1}{T}\sum_{t=1}^T p^t$.

In the above we have introduced an auxiliary (weight) vector w^t at every iteration. While it is not necessary to state the algorithm, it will be useful for us to refer to w^t in the proof of the convergence guarantee of the algorithm.

Note one interesting difference between the Exponential Gradient Descent algorithm and the variant of gradient descent studied in the previous lecture: the output of EGD is the **average** of all iterates $\bar{p}$ not the **last** iterate p^T. However, this is something to expect as the function f might not be even differentiable and it is rather hard to obtain a nontrivial approximation guarantee on a single iterate.

To illustrate the problem, note that if $f(x) = |x|$, then the gradient at every point is either[5] 1 or -1. Hence by just knowing that the gradient at a certain point x is 1 we still have no clue whether x is close to the minimizer (0) or very far from it. Thus, as opposed to the Lipschitz gradient case, the gradient at a point x does not provide us with a certificate that $f(x)$ is close to optimal. Therefore one naturally needs to gather more information, by visiting multiple points and average them in some manner.

Theorem 3. *Suppose that $f : \Delta_n \to \mathbb{R}$ is a convex function which satisfies $\|\nabla f(p)\|_\infty \leq G$ for all $p \in \Delta_n$. If we use $\eta = \Theta\left(\frac{\sqrt{\log n}}{\sqrt{T}G}\right)$ then after $T = \Omega\left(\frac{G^2 \log n}{\varepsilon^2}\right)$ iterations of the EGD algorithm, the point $\bar{p} := \frac{1}{T}\sum_{t=1}^T p^t$ satisfies*

$$f(\bar{p}) - f(p^\star) \leq \varepsilon,$$

[5]Except from the point $x = 0$ which is very unlikely to be hit by an iterative algorithm, hence we can ignore it.

where $p^\star$ is any minimizer of $\min_{p\in\Delta_n} f(p)$.

2.2 Properties of the KL-divergence

We present several important properties of the KL-divergence that will be useful in the proof of Theorem 3. Many of these are more general and hold for so called Bregman divergences of functions.

Definition 4. *Let $F : K \to \mathbb{R}$ be a convex, differentiable function over a convex subset K of $\mathbb{R}^n$. For $x, y \in K$ we define the Bregman divergence*

$$D_F(x,y) := F(x) - F(y) - \langle \nabla F(y), x - y\rangle .$$

Note that $D_F(x,y)$ measures the distance between the value of F at the point x and the 1st order approximation of F at x. In particular, $D_F(x,y) \geq 0$ and $D_F(x,y) \to 0$ when, say x is fixed and $y \to x$.

To recover the KL-divergence we can consider the negative entropy function $H : \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ given by

$$H(x) = \sum_{i=1}^{n} x_i \log x_i.$$

For every $x, y \geq 0$ we have

$$D_H(x,y) = -\sum_{i=1}^{n} x_i \log \frac{y_i}{x_i} + \sum_{i=1}^{n} (y_i - x_i),$$

hence $D_H(x,y) = D_{KL}(x,y)$ for all $x, y \in \Delta_n$. In fact, the function D_H is the generalized KL-divergence, and we will use $D_{KL}(x,y)$ to denote it, for all non-negative vectors (even outside of the simplex).

For a different, simple example consider $F(x) = \|x\|_2^2$. We have

$$D_F(x,y) = \|x - y\|_2^2 ,$$

in this case the Bregman divergence is simply the squared Euclidean distance between vectors x, y.

2.2.1 Properties of the Bregman divergence

We start by a simple, yet useful identity involving the Bregman divergence.

Lemma 5 (Law of cosines)**.** *Let $F : K \to \mathbb{R}$ be a convex, differentiable function and let $x, y, z \in K$, then*

$$\langle \nabla F(y) - \nabla F(z), y - x\rangle = D_F(x,y) + D_F(y,z) - D_F(x,z).$$

The proof of the above identity is a direct calculation and hence we omit it. Note that for the case when $F(x) = \|x\|_2^2$ the above says that for three points $a, b, c \in \mathbb{R}^n$ we have

$$2\langle c-a, c-d\rangle = \|d-c\|_2^2 + \|c-a\|_2^2 - \|d-a\|_2^2,$$

which is the familiar *law of cosines*.

Next we state the following generalized Pythagorean Theorem.

Lemma 6 (Pythagorean Theorem)**.** *Let $F : K \to \mathbb{R}$ be a convex, differentiable function and let $S \subseteq K$ be a closed, convex subset of K. Let $x, y \in S$ and $z \in K$ such that $y = \operatorname{argmin}_{u \in S} D_F(u, z)$, then*

$$D_F(x, y) + D_F(y, z) \leq D_F(x, z).$$

It is an instructive exercise to consider the special case of this lemma for the $F(x) = \|x\|_2^2$. It says that if we project z onto a convex set S and call the projection y, then the angle between the vectors $\vec{yx}$ and $\vec{yz}$ is obtuse (larger than 90 degrees).

Proof of Lemma 6: Let x, y, z be as in the statement. We write down optimality conditions for the optimization problem $\min_{u \in S} D_F(u, z)$ at the minimizer y (see Lecture 1), they say that for every point $w \in S$ it holds

$$\langle \nabla_u D_F(y, z), w - y\rangle \leq 0,$$

which translates to

$$\langle \nabla F(y) - \nabla F(z), w - y\rangle \leq 0.$$

By plugging in $w = x$ and using the Law of cosines (Lemma 5) we obtain

$$D_F(x, y) + D_F(y, z) - D_F(x, z) \leq 0.$$

□

2.2.2 Pinsker's Inequality

The following bound known as Pinsker's inequality asserts that the negative entropy function $H(x) = \sum_{i=1}^n x_i \log x_i$ is 1-strongly convex with respect to the ℓ_1 norm, when restricted to the probability simplex Δ_n.

Lemma 7 (Pinsker's Inequality)**.** *For every $x, y \in \Delta_n$ we have*

$$D_{KL}(x, y) \geq \frac{1}{2}\|x - y\|_1^2.$$

2.3 Convergence proof of EGD

Given the properties of KL-divergence stated in the previous section we are ready to proceed with the proof of Theorem 3.

Proof of Theorem 3:

Step 1. Bounding $f(\bar{p}) - f(p)$ by $\frac{1}{T}\sum_{t=1}^{T}\langle g^t, p^t - p\rangle$.

Start by noting that a simple consequence of convexity of f is that

$$\begin{aligned} f(\bar{p}) - f(p) \leq & \left(\frac{1}{T}\sum_{t=1}^{T} f(p^t)\right) - f(p) \\ = & \frac{1}{T}\sum_{t=1}^{T}\left(f(p^t) - f(p)\right) \\ \leq & \frac{1}{T}\sum_{t=1}^{T}\langle \nabla f(p^t), p^t - p\rangle \\ = & \frac{1}{T}\sum_{t=1}^{T}\langle g^t, p^t - p\rangle . \end{aligned} \tag{9}$$

Thus, from now on we focus on the task of providing an upper bound on the sum $\sum_{t=1}^{T}\langle g^t, p^t - p\rangle$.

Step 2. Writing $\langle g^t, p^t - p\rangle$ in terms of KL-divergence.

Let us fix $t \in [1, T]$, we provide an expression for g^t in terms of w^{t+1} and p^t. Note that since $w_i^{t+1} = p_i^t \exp(-\eta g_i^t)$ (for $i \in \{1, 2, \ldots, n\}$), we have

$$g_i^t = \frac{1}{\eta}\left(\log p_i^t - \log w_i^{t+1}\right).$$

This can be also written in terms of the gradient ∇H of the entropy function $H(x) = \sum_{i=1}^{n} x_i \log x_i$, specifically

$$g^t = \frac{1}{\eta}\left(\log p^t - \log w^{t+1}\right) = \frac{1}{\eta}\left(\nabla H(p^t) - \nabla H(w^{t+1})\right), \tag{10}$$

where in the above log is applied coordinatewise to the appropriate vectors. Thus we obtain (from the above and the law of cosines – Lemma 5)

$$\begin{aligned} \langle g^t, p^t - p\rangle &= \frac{1}{\eta}\left\langle \nabla H(p^t) - \nabla H(w^{t+1}), p^t - p\right\rangle \\ &= \frac{1}{\eta}\left(D_{KL}(p, p^t) + D_{KL}(p^t, w^{t+1}) - D_{KL}(p, w^{t+1})\right). \end{aligned} \tag{11}$$

Step 3. Using Pythagorean theorem to get a telescoping sum.

Now, since p^{t+1} is the projection with respect to D_{KL} of w^{t+1} onto Δ_n (see Lemma 2), the generalized Pythagorean theorem (Lemma 6) says that

$$D_{KL}(p, w^{t+1}) \geq D_{KL}(p, p^{t+1}) + D_{KL}(p^{t+1}, w^{t+1}).$$

Thus when we can bound the expression $\sum_{t=1}^{T} \left\langle g^t, p^t - p \right\rangle$ as follows:

$$\begin{aligned}
\eta \sum_{t=1}^{T} \left\langle g^t, p^t - p \right\rangle &= \sum_{t=1}^{T} D_{KL}(p, p^t) + D_{KL}(p^t, w^{t+1}) - D_{KL}(p, w^{t+1}) \\
&\leq \sum_{t=1}^{T} D_{KL}(p, p^t) + D_{KL}(p^t, w^{t+1}) - \left[D_{KL}(p, p^{t+1}) + D_{KL}(p^{t+1}, w^{t+1}) \right] \\
&= \sum_{t=1}^{T} \left[D_{KL}(p, p^t) - D_{KL}(p, p^{t+1}) \right] + \left[D_{KL}(p^t, w^{t+1}) - D_{KL}(p^{t+1}, w^{t+1}) \right] \\
&\leq D_{KL}(p, p^1) + \sum_{t=1}^{T} \left[D_{KL}(p^t, w^{t+1}) - D_{KL}(p^{t+1}, w^{t+1}) \right].
\end{aligned} \tag{12}$$

In the last step we used the fact that the first term of the sum is telescoping to $D_{KL}(p, p^1) - D_{KL}(p, p^{T+1})$ and that $D_{KL}(p, p^{T+1}) \geq 0$.

Step 4. Using Pinsker's ineq. and bounded gradient to bound the remaining terms.

To bound the second term we first apply the law of cosines

$$\begin{aligned}
D_{KL}(p^t, w^{t+1}) - D_{KL}(p^{t+1}, w^{t+1}) &= \left\langle \nabla H(p^t) - \nabla H(w^{t+1}), p^t - p^{t+1} \right\rangle - D_{KL}(p^{t+1}, p^t) \\
&= \eta \left\langle g^t, p^t - p^{t+1} \right\rangle - D_{KL}(p^{t+1}, p^t)
\end{aligned} \tag{13}$$

And finally we apply Pinsker's inequality (Lemma 7) to obtain

$$D_{KL}(p^t, w^{t+1}) - D_{KL}(p^{t+1}, w^{t+1}) \leq \eta \left\langle g^t, p^t - p^{t+1} \right\rangle - \frac{1}{2} \left\| p^{t+1} - p^t \right\|_1^2. \tag{14}$$

Further, since $\left\langle g^t, p^t - p^{t+1} \right\rangle \leq \left\| g^t \right\|_\infty \left\| p^t - p^{t+1} \right\|_1$ we can write:

$$\begin{aligned}
D_{KL}(p^t, w^{t+1}) - D_{KL}(p^{t+1}, w^{t+1}) &\leq \eta \left\| g^t \right\|_\infty \left\| p^t - p^{t+1} \right\|_1 - \frac{1}{2} \left\| p^{t+1} - p^t \right\|_1^2 \\
&\leq \eta G \left\| p^{t+1} - p^t \right\|_1 - \frac{1}{2} \left\| p^{t+1} - p^t \right\|_1^2 \\
&\leq \frac{(\eta G)^2}{2}.
\end{aligned} \tag{15}$$

Where the last inequality follows by simply maximizing the quadratic function $x \mapsto \eta G x - \frac{1}{2}x^2$.

Conclusion of the proof.

By summing up all the terms, we arrive at

$$\sum_{t=1}^{T} \langle g^t, p^t - p \rangle \leq \frac{1}{\eta} \left(D_{KL}(p, p^1) + T \frac{(\eta G)^2}{2} \right).$$

By observing that $D_{KL}(p, p^1) \leq \log n$ and optimizing the choice of η, the theorem follows. □

2.4 The Multiplicative Weights Update framework

An attentive reader might have realized that in the proof of Theorem 3 the only place we used the fact that the vectors g^t are gradients of the function f at points p^t (for $t = 1, 2, \ldots, T$) is the bound in (9). Afterwards g^ts were treated as arbitrary vectors, and we proved a guarantee that in order to reach

$$\frac{1}{T} \sum_{t=1}^{T} \langle g^t, p^t \rangle - \min_{p \in \Delta_n} \frac{1}{T} \sum_{t=1}^{T} \langle g^t, p \rangle \leq \varepsilon,$$

it is enough to take $T = O\left(\frac{G^2 \log n}{\varepsilon^2}\right)$. Let us now state this observation as a theorem. Before doing so, we first state this general Meta-Algorithm – known under the name *Multiplicative Weights Update* method (in the Hedge form).

Multiplicative Weights Update (MWU):
Input: An oracle providing (adaptively) vectors $g^t \in \mathbb{R}^n$ at every step $t = 1, 2, \ldots$, a parameter $\eta > 0$, and an integer $T > 0$.
Output: A sequence of probability distributions $p^1, p^2, \ldots p^T \in \Delta_n$.

1. Initialize $p^1 = \frac{1}{n}\mathbb{1}$ (the uniform distribution).
2. For each iteration $t = 1, 2, \ldots, T$:
   - Obtain $g^t \in \mathbb{R}^n$ from the oracle.
   - Let $w^{t+1} \in \mathbb{R}^n$ and $p^{t+1} \in \Delta_n$ be defined as

$$w_i^{t+1} := p_i^t e^{-\eta g_i^t}, \qquad p_i^{t+1} := \frac{w_i^{t+1}}{\sum_{j=1}^{n} w_j^{t+1}}.$$

By reasoning exactly as in the proof of Theorem 3 we obtain.

Theorem 8. *Consider the MWU algorithm as defined above. Assume that all the vectors g^t provided by the oracle satisfy $\|g^t\|_\infty \leq G$. Then, taking $\eta = \Theta\left(\frac{\sqrt{\log n}}{\sqrt{T}G}\right)$, after $T = \Omega\left(\frac{G^2 \log n}{\varepsilon^2}\right)$ iterations we have*

$$\frac{1}{T}\sum_{t=1}^{T}\langle g^t, p^t\rangle - \min_{p\in\Delta_n}\frac{1}{T}\sum_{t=1}^{T}\langle g^t, p\rangle \leq \varepsilon.$$

At this point it might not be clear what is the purpose of stating such a theorem. However, as we will see – this theorem allows, very generally, proving convergence bounds for numerous different algorithms based on the idea of maintaining weights and changing them multiplicatively. In the example we provide, we design an algorithm for checking if a bipartite graph has a perfect matching. This can be further extended to linear programming or even semidefinite programming.

3 General Mirror Descent

3.1 Generalizing the EGD algorithm

In this section we take inspiration from the just introduced exponential gradient descent algorithm to derive a general method for convex optimization called *Mirror Descent* (see [5, 1]). The main idea follows the intuition provided at the very beginning of the lecture. Recall that our update rule is (3), i.e. at a given point x^t we construct a linear lower bound $f(x^t) + \langle \nabla f(x^t), x - x^t\rangle \leq f(x)$ and move to the next point, being the minimizer of this lower bound "regularized" by a distance function $D(\cdot,\cdot)$, thus we get (3)

$$x^{t+1} := \operatorname*{argmin}_{x\in K}\left\{\langle D(x, x^t) + \eta\nabla f(x^t), x\rangle\right\}.$$

When deriving the exponential gradient descent algorithm we used the relative entropy function $D_{KL}(\cdot,\cdot)$ for this purpose. In general we will use the *Bregman Divergence* $D_R(\cdot,\cdot)$ of a convex regularizer $R : \mathbb{R}^n \to \mathbb{R}$. For the KL-divergence $R(x)$ was the negative entropy function $R(x) = \sum_{i=1}^n x_i \log x_i$. In general, by denoting the gradient at step t by g^t we have

$$\begin{aligned} x^{t+1} &= \operatorname*{argmin}_{x\in K}\left\{D_R(x, x^t) + \eta\langle g^t, x\rangle\right\} \\ &= \operatorname*{argmin}_{x\in K}\left\{\eta\langle g^t, x\rangle + R(x) - R(x^t) - \langle \nabla R(x^t), x - x^t\rangle\right\} \\ &= \operatorname*{argmin}_{x\in K}\left\{R(x) - \langle \nabla R(x^t) - \eta g^t, x\rangle\right\}. \end{aligned} \tag{16}$$

For now, ignoring certain technical difficulties, let w^{t+1} be a point such that

$$\nabla R(w^{t+1}) = \nabla R(x^t) - \eta g^t.$$

This corresponds to the same w^{t+1} as we had in the EGD algorithm (the "unscaled" version of p^{t+1}). We have then

$$\begin{aligned} x^{t+1} &= \operatorname*{argmin}_{x \in K} \left\{ R(x) - \left\langle \nabla R(w^{t+1}), x \right\rangle \right\} \\ &= \operatorname*{argmin}_{x \in K} \left\{ D_R(x, w^{t+1}) \right\}. \end{aligned} \tag{17}$$

Note again the analogy with the EGD algorithm: there p^{t+1} was obtained as a KL-divergence projection of w^{t+1} onto the simplex $\Delta_n = K$, exactly as above. We are ready to state the Mirror Descent algorithm in its general form.

Mirror Descent Algorithm:
Input: First order oracle access to a convex function $f : K \to \mathbb{R}$. Oracle access to ∇R mapping and its inverse. Access to the projection operator with respect to $D_R(\cdot, \cdot)$. An initial point $x^1 \in K$, a parameter $\eta > 0$, and an integer $T > 0$.
Output: A point $\bar{x} \in K$.

1. Repeat for $t \in \{1, \ldots, T\}$
   - Let w^{t+1} be such that $\nabla R(w^{t+1}) = \nabla R(x^t) - \eta \nabla f(x^t)$.
   - Let $x^{t+1} = \operatorname{argmin}_{x \in K} D_R(x, w^{t+1})$.
2. Output $\bar{x} := \frac{1}{T} \sum_{i=1}^{T} x_i$.

For the above algorithm to be well defined we need to make sure that w^{t+1} always exists. Formally we assume that the regularizer $R : \Omega \to \mathbb{R}$ has a domain Ω which contains K as a subset. Moreover, as in the case of the entropy function, we assume that the map $\nabla R : \Omega \to \mathbb{R}^n$ is a bijection – this is perhaps more than what we really need, but such an assumption makes the picture much more clear. In fact ∇R is typically referred to as the *mirror map*.

Note that in order for the algorithm to be useful the mirror map ∇R (and its inverse) should be efficiently computable. Similarly, the projection step $\operatorname{argmin}_{x \in K} D_R(x, w^{t+1})$ should be also easy to perform. The efficiency of these two operations determines then the time required to perform one iteration of Mirror Descent.

3.2 Statement of the algorithm and convergence proof

We are now ready to state the iteration bound guarantee for Mirror Descent.

Theorem 9 (Mirror Descent guarantee)**.** *Let $f : K \to \mathbb{R}$ and $R : \Omega \to \mathbb{R}$ be convex functions with $K \subseteq \Omega \subseteq \mathbb{R}^n$ and suppose the following assumptions hold*

1. *The gradient map* $\nabla R : \Omega \to \mathbb{R}^n$ *is a bijection.*

2. *The function* f *has bounded gradients in a norm* $\|\cdot\|$*:*

$$\forall_{x \in K} \quad \|\nabla f(x)\| \leq G.$$

3. R *is* α*-strongly convex with respect to the dual*[6] *norm* $\|\cdot\|_\star$*:*

$$\forall_{x \in \Omega} \quad D_R(x, y) \geq \frac{\alpha}{2} \|x - y\|^{\star 2}.$$

Then after $T = \Omega\left(\frac{G^2 \cdot D_R(x^\star, x^1)}{\alpha \varepsilon^2}\right)$ *iterations of the Mirror Descent Algorithm the point* $\bar{x}$ *satisfies*

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

where $x^\star$ *is any minimizer of* $\min_{x \in K} f(x)$*.*

Proof. The proof of the above theorem follows exactly the one we gave for Theorem 3 by replacing the entropy function H by a regularizer R and replacing the KL-distance terms D_{KL} by D_R.

We now go step by step through the proof of Theorem 3 and emphasize what properties of R and D_R are being used.

In **Step 1.** the reasoning in (9) used to obtain

$$f(\bar{x}) - f(x^\star) \leq \frac{1}{T} \sum_{t=1}^{T} \left\langle g^t, x^t - x^\star \right\rangle.$$

is general and relies on convexity of f only.

In **Step 2.** the facts used in equations (10) and (11) to prove that

$$\left\langle g^t, x^t - x^\star \right\rangle = \frac{1}{\eta} \left(D_R(x^\star, x^t) + D_R(x^t, w^{t+1}) - D_R(x^\star, w^{t+1}) \right)$$

are the definition of w^{t+1} and the law of cosines which is valid for any Bregman divergence (see Lemma 5). Subsequently, in **Step 3.** to arrive at the conclusion of (12)

$$\eta \sum_{t=1}^{T} \left\langle g^t, x^t - x^\star \right\rangle \leq D_R(x^\star, x^1) + \sum_{t=1}^{T} \left[D_R(x^t, w^{t+1}) - D_R(x^{t+1}, w^{t+1}) \right],$$

only the Pythagorean theorem (Lemma 6) is necessary.

[6] For a norm $\|\cdot\|$ on $\mathbb{R}^n$, its dual norm $\|\cdot\|_\star$ on $\mathbb{R}^n$ is defined as $\|y\|_\star := \sup\{|\langle x, y\rangle| : \|x\| \leq 1\}$. For instance, the dual norm to $\|\cdot\|_p$ is $\|\cdot\|_q$ such that $\frac{1}{p} + \frac{1}{q} = 1$.

Finally in **Step 4.**, an analogue of equation (15) which can be proved under our assumptions is

$$D_R(x^t, w^{t+1}) - D_R(x^{t+1}, w^{t+1}) \leq \|g^t\| \left\|x^t - x^{t+1}\right\|_\star - \frac{\alpha}{2}\left\|x^{t+1} - x^t\right\|_\star^2.$$

The above follows from the strong convexity assumption with respect to $\|\cdot\|$ (which we use in place of the Pinsker's inequality) and the Cauchy-Schwarz inequality

$$\langle u, v\rangle \leq \|u\| \|v\|_\star,$$

which directly follows from the definition of the dual norm. The remaining part of the reasoning is generic and does not rely on any specific properties of R or D_R.

□

4 Application: Bipartite Matching

The goal of this section is to give an application of the MWU framework and specifically show how Theorem (8) can be employed.

4.1 Problem statement

Here is a definition of the problem we would like to solve: the Perfect Matching problem on bipartite graphs

Perfect Matching Problem on Bipartite Graphs

Input: An undirected bipartite graph $G = (V, E)$.

Goal: Find a subset of edges $M \subseteq E$ such that every vertex $v \in V$ is contained in exactly one of the edges in M.

Recall that a graph is bipartite if there are two disjoint sets of vertices A, B with $A \cup B = V$ such that there are no edges within A or within B. We assume that in the bipartition (A, B) we have $|A| = |B| = n$, so that the total number of vertices is $2n$ (note that if the cardinalities of A and B are not the same, then there is no perfect matching in G!).

Our approach is based on solving the following linear programming reformulation of the perfect matching problem.

$$\begin{aligned} \text{Find} \quad & x \in \mathbb{R}^E \\ \text{s.t.} \quad & \sum_{e \in E} x_e = n, \\ & \sum_{e:\, v \in e} x_e \leq 1 \qquad \forall v \in V, \\ & x_e \geq 0 \qquad \forall e \in E. \end{aligned} \tag{18}$$

A solution to the above linear feasibility program is called a *fractional perfect matching*. The question which arises very naturally is whether the "fractional" problem is equivalent to the original one?

Fact 1. *If G is a bipartite graph then G has a perfect matching if and only if G has a fractional perfect matching.*

We do not provide a proof of this fact as it is a classical exercise in linear programming. Note however that one direction is easy: if there is a perfect matching $M \subseteq E$ in G then its characteristic vector $x = 1_M$ is a fractional perfect matching. The opposite direction is a bit harder (and relies on the assumption that the graph is bipartite), it can be deduced from Hall's theorem, or from the fact that the constraint matrix defining the above linear program is totally unimodular.

Algorithmically, one can also convert fractional matchings into matchings in $\widetilde{O}(|E|)$ time using an idea introduced in [2] (see also [3] for a proof). Thus solving (18) is also sufficient to solve the perfect matching problem in its original form.

As our algorithm naturally produces approximate answers, for an $\varepsilon > 0$ we also define an **ε-approximate fractional perfect matching** to be an $x \in \mathbb{R}^E$ which satisfies

$$\begin{aligned} &\sum_{e \in E} x_e = n, \\ &\sum_{e:\, v \in e} x_e \leq 1 + \varepsilon \quad \forall v \in V, \\ &x_e \geq 0 \quad \forall e \in E. \end{aligned}$$

From such an approximate fractional matching one can (given the ideas discussed above) construct a matching in G of cardinality at least $(1-\varepsilon)n$ and thus also solve the perfect matching problem exactly by taking $\varepsilon < \frac{1}{n}$.

4.2 Past work

The perfect matching problem on bipartite graphs has been extensively studied and is perhaps the most classical problem in combinatorial optimization. It can be reduced to the max-flow problem and thus solved in $O(nm)$ using the Ford-Fulkerson method. It just requires to find n augmenting paths – every such iteration takes $O(m)$ as it requires to run the DFS algorithm.

A more refined variant of this algorithm was presented by Hopcroft and Karp which runs in $O(\sqrt{n}m)$ time. 40 years later, after the latter result came out, a partial improvement (for the sparse regime: when m is small) was obtained by Madry [4] and his algorithm, based on the interior point method for linear programming, which runs in $\widetilde{O}(m^{10/7})$ time.

4.3 Statement of the Theorem

We construct an algorithm based on the MWU framework to construct ε-approximate fractional perfect matchings. Below we formally state its running time guarantee.

Theorem 10. *There is an algorithm which given a bipartite graph G (containining a perfect matching) with $2n$ vertices and m edges, and an $\varepsilon > 0$, outputs an ε-approximate fractional perfect matching in G in time $\widetilde{O}(\varepsilon^{-2}n^2m)$.*

The running time of the above algorithm is certainly not comparable to the best known algorithms for this problem, but its advantage is for sure is overall simplicity. We remark that the running time of this algorithm can be also brought down to $\widetilde{O}(\varepsilon^{-2}m)$ by a rather simple improvement in one of its steps.

4.4 The Algorithm

We construct an algorithm for finding fractional solutions to the perfect matching problem, i.e., to (18). In every step of the algorithm we would like to construct a point $x^t \in \mathbb{R}^E_{\geq 0}$

which satisfies the constraint $\sum_{i=1}^{n} x_i^t$, but is not necessarily a fractional matching. However, x^t should (in a certain sense) brings us closer to satisfying all the constraints.

More precisely, we maintain positive weights $w^t \in \mathbb{R}^V$ over vertices V of the graph G. The value w_v^t corresponds to the importance of the inequality $\sum_{e:\, v \in e} x_e \leq 1$ at the current state of the algorithm. Intuitively, the importance is large whenever our "current solution" (which one should think of as the average of all x^t produced so far) violates this inequality, and more generally, the larger the violation of the vth constraints, the larger w_v^t is.

Given such weights, we then compute a new point x^t to be any point which satisfies $\sum_{e \in E} x_e^t = n$ and satisfies the single inequality being the **weighted average** (with respect to w^t) of all the inequalities $\sum_{e:\, v \in e} x_e \leq 1$. Next, we update our weights based on the violations of the inequalities by the new point x^t. We now provide a pseudocode of our method, based on the ideas discussed above.

Approximate Perfect Matching:
Input: A bipartite graph $G = (V, E)$, $\eta > 0$, $T \in \mathbb{N}$
Output: An approximate fractional perfect matching $x \in [0,1]^E$ or NO.

1. Initialize $w^1 = (1, 1, ..., 1) \in \mathbb{R}^V$,
2. Repeat for $t = 1, 2, ..., T$:
   - Find a point $x^t \in \mathbb{R}^E$ satisfying

$$\sum_{v \in V} w_v^t \left(\sum_{e \in E} x_e^t \right) \leq \sum_{v \in V} w_v^t,$$

$$\sum_{e: v \in e} x_e^t = n \quad \text{and} \quad x_e^t \geq 0 \quad \text{for all } e \in E.$$

     If there is no such point return "NO".
   - Construct a vector $g^t \in \mathbb{R}^v$

$$g_v^t = \frac{1 - \sum_{e: v \in e} x_e^t}{n},$$

   - Update the weights:

$$w_v^{t+1} = w_v^t \cdot \exp(-\eta \cdot g_v^t) \quad \text{for all } v \in V.$$

3. Output $x = \frac{1}{T} \sum_{i=1}^{T} x^t$.

The algorithm, as given above, is not completely specified, as we have not yet provided a method for finding x^t. In the analysis we specify one particular method for finding x^t and show that it gives us then an algorithm with a running time as in 10. When replaced by a "better" algorithm (we will specify what does it mean later) the running time can be brought down to $\widetilde{O}(\varepsilon^{-2}m)$.

4.5 Analysis

The analysis is divided into two steps: how to find x^t? and the correctness of the algorithm given in the previous section (assuming a method for finding x^t).

4.5.1 Step 1. How to find x^t?

In the lemma below we prove that x^t can be always found efficiently such that the infinity norm of g^t is bounded by 1. As we will see soon the bound on $\left\|g^t\right\|_\infty$ is crucial for the efficiency of our algorithm.

Lemma 11. *If G has a perfect matching then:*

1. *x^t always exists and can be found in $O(m)$ time,*
2. *we can guarantee that $\left\|g^t\right\|_\infty \leq 1$.*

Proof. If M is a perfect matching, then it is easy to see that $x^t = 1_M$ (the indicator vector of M) satisfies all conditions. However, we don't know M and cannot compute it easily, but still we would like to find such a point. Let us rewrite the condition

$$\sum_{v \in V} w_v^t \left(\sum_{e \in E} x_e^t \right) \leq \sum_{v \in V} w_v^t$$

in the form:

$$\sum_{e \in E} \alpha_e x_e^t \leq \beta,$$

where all the coefficients α_e and β are positive. It is easy to see that putting greedily $x_{e^*}^t = n$ for e^* being an edge with smallest coefficient α_{e^*} (and $x_{e'}^t = 0$ for $e' \neq e^*$) will be a valid solution. Such a choice of x^t guarantees that for every $v \in V$:

$$-1 \leq \sum_{e \in v} x_e^t - 1 \leq n - 1$$

which in particular implies that $\left\|g^t\right\|_\infty \leq 1$. □

4.5.2 Correctness and running time of the algorithm

Lemma 12. *Suppose that G has a perfect matching. Then for every $\varepsilon > 0$, if we choose $\eta = \Theta\left(\frac{\varepsilon}{n}\right)$ then after $T = \Omega(\varepsilon^{-2} n^2 \log n)$ steps the point $x := \frac{1}{T}\sum_{i=1}^{T} x^t$ is an ε-approximate fractional perfect matching.*

Proof. Let $p^t = \frac{w^t}{\sum_{v \in V} w_v^t}$. We use the Theorem 8 to a obtain guarantee on the underlying MWU scheme. We plug in $p := e_v$, for any fixed $v \in V$ to conclude

$$-\frac{1}{T}\sum_{t=1}^{T} g_v^t \leq -\frac{1}{T}\sum_{t=1}^{T} \left\langle p^t, g^t \right\rangle + \delta, \tag{19}$$

for $T = \Omega\left(\frac{\log n}{\delta^2}\right)$ (since $G = 1$ in this case). The fact that x^t satisfies the considered system of inequalities "on average" translates to the following condition:

$$\left\langle p^t, g^t \right\rangle \leq 0.$$

Therefore, from the guarantee (19), for a fixed $v \in V$ we get

$$\begin{aligned} \frac{1}{T} \cdot \frac{1}{n}\left(\sum_{e:e\in v} x_e^t - 1\right) &\leq \frac{1}{T} \cdot T \cdot 0 + \delta \\ \sum_{e:e\in v} x_e - 1 &\leq n\delta \\ \sum_{e:e\in v} x_e &\leq 1 + n\delta. \end{aligned}$$

Thus to make the above $1 + \varepsilon$ we need to pick $\delta = \frac{\varepsilon}{n}$ and hence T becomes $\Omega\left(\frac{n^2 \log n}{\varepsilon^2}\right)$. Further, since x is a convex combination of x^t for $t = 1, .., T$ it also satisfies $\sum_{e\in E} x_e = n$ and $x \geq 0$. □

We are now ready to prove Theorem 10.

Proof of Theorem 10: The correctness of the algorithm follows from Lemma 12. We only need to reason about the running time. From Lemma 12 the number of iterations is $O\left(\frac{n^2 \log n}{\varepsilon^2}\right)$ and each iterations is to find x^t and update the weights, which can be done in $O(m)$ time. □

It can be easily seen that the reason for the "n^2" factor in the running time is because of our bound $|\sum_{e\in v} x_e^t - 1| \leq n$. If one could always produce a point x^t with

$$\forall v \in V \quad \left|\sum_{e\in v} x_e^t - 1\right| \leq \rho,$$

then the running time would become $O(\varepsilon^{-2} m\rho^2 \log n)$. Note that intuitively, this should be possible to do, since $x^t = 1_M$ (for any perfect matching M) gives $\rho = 1$. However, at the same time we would like our procedure for finding x^t to be efficient, preferably it should run in nearly linear time.

It is an interesting exercise to design a nearly linear time algorithm for finding x^t with the guarantee $\rho = 2$, which yields a much better running time of only $O(\varepsilon^{-2} m \log^2 n)$.

References

[1] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, 2003.

[2] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in o(n log n) time in regular bipartite graphs. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 39–46, 2010.

[3] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 755–764, New York, NY, USA, 2013. ACM.

[4] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 253–262, 2013.

[5] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley Interscience, 1983.