EE C227C Convex Optimization and Approximation Lecture 8 - 02/10/2022

ℓ_p-ball Optimization and Graph Matching

Lecturer: Jiantao Jiao *Scribe: Newton Cheng and Xin Lyu*

1 Optimization in ℓ_p balls

Up to this point, we have mainly considered two cases for a constrained optimization domain K in our studies:

1. $K_2 = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$
2. $K_1 = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 1\}$

In these cases, we can employ our toolbox of (sub)gradient descent and mirror descent methods to perform the optimization, and as long as the functions or their gradients are appropriately continuous and/or bounded, we obtain convergence guarantees. A natural question is to ask whether similar properties hold for other norms; it turns out that in the case of the ℓ_∞ norm, we may not necessarily be able obtain such guarantees (at least with our current techniques). Let us see where things can go wrong.

We define our domain as the ℓ_∞ ball $K_\infty = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$. Recall that in the setup for mirror descent, we required that the mirror map $R : \Omega \to \mathbb{R}$ (where Ω contains K_∞ as a subset) be strongly convex – for our purposes, we will assume strong convexity condition is with respect to ℓ_∞ norm and modulus 1:

$$D_R(x, y) = R(x) - R(y) - \langle \nabla R(y), x - y \rangle \geq \frac{1}{2} \|x - y\|_\infty^2 , \tag{1}$$

Then one has the following result:

Theorem 1. *For any continuous differentiable map $R : \Omega \to \mathbb{R}$ with $K_\infty \subseteq \Omega$ satisfying the strong convexity condition in Equation (1), we have*

$$\sup_{x,y \in K} D_R(x, y) \geq \frac{1}{2} n. \tag{2}$$

Proof Without loss of generality, assume R and ∇R vanish at the origin. We will show that there always exists a point v such that $R(v)$ is at least $n/2$ greater than its linear approximation at the origin. Now, take a unit step in the direction of the first standard basis vector e_1, arriving at the point $v_1 = (1, \ldots, 0)$. Then by the strong convexity condition $R(v_1) \geq \frac{1}{2}$. Now, we want to take a second unit step along e_2. If the partial derivative with respect to the second coordinate x_2 at v_1 is negative $\partial_2 R(v_1) < 0$, then we will step in the negative direction. If $\partial_2 R(v_1) \geq 0$, then we step in the positive direction. In either case, we find that $R(v_2) \geq 1$ by strong convexity, i.e. R has increased by at least 0.5 again. We proceed in this fashion for a third step and all the way to the nth step. After n steps, we will have arrived at a point v on the unit box such that $R(v) \geq n/2$, but with $R(0) = 0$. Correspondingly, we find $\sup_{x,y \in K} D_R(x, y) \geq D_R(v, 0) \geq \frac{1}{2} n$. □

This is a problem, because it is not *dimension-free* – recall that the only place the dimension featured in previously discussed algorithms was computing the gradient. Otherwise, the runtimes of the various algorithms depended only on finite norm bounds and the target suboptimality gap. Hence, as long as we had a first-order oracle that could return the gradient in an efficient manner, our algorithms would function independent of the dimension. This is no longer the case if we have the above situation: we may need a number of oracle calls that scales polynomially in n. Hence, we get the heuristic that the "ℓ_∞-ball is hard to optimize."

While the problem is serious for the ℓ_∞-ball, it turns out to be a generic feature that the ℓ_p-ball is "hard" for $p \geq 2$:

Theorem 2. *Let $K_p = \{x \in \mathbb{R}^n \mid \|x\|_p \le 1\}$ be the ℓ_p-ball for $p \ge 2$. Consider a family of functions $\mathcal{F}_{n,p}(L) = \{f : \mathbb{R}^n \to \mathbb{R} \mid f \text{ convex and } L\text{-Lipschitz w.r.t. } \|\cdot\|_p\}$. Assume we have access to a first-order oracle for any $f \in \mathcal{F}_{n,p}(L)$. Then for any $t \le n$ and any algorithm $\mathcal{B}$ that makes at most t queries, there exists $f \in \mathcal{F}_{n,p}(L)$ such that $f(\bar{x}_{\mathcal{B}}(f)) - f^\star \ge \frac{0.5L}{t^{1/p}}$, where $\bar{x}_{\mathcal{B}}(f)$ is the output of $\mathcal{B}$ and $f^\star$ is the minimal value of f on K_p.*

Proof We only prove for the case that $L = 1$. The same reasoning applies to every $L > 0$ by scaling the function family properly. Let $t \ge 1$ be the parameter in the statement of Theorem 2. We consider a subfamily of $\mathcal{F}_{n,p}(L)$, defined as:

$$\mathcal{F}_t = \left\{ f(x) = \max_{i \in [t]} [\varepsilon_i x_i + \delta_i] \right\}$$

which is given by 2^t collections of $\varepsilon_j \in \{\pm 1\}$ and all collections of $0 \le \delta_i \le \frac{1}{2t^{1/p}}$. It is straightforward to verify that functions in $\mathcal{F}_t$ are 1-Lipshitz w.r.t. $\|\cdot\|_p$. $\mathcal{F}_t$ is also convex, because every function in $\mathcal{F}_t$ is constructed by taking a point-wise maximum among a set of linear functions. Hence, we may conclude that $\mathcal{F}_t$ is a subfamily of $\mathcal{F}_{n,p}(L)$.

Now let $\mathcal{B}$ be an algorithm that operates on $\mathcal{F}_t$ and makes $t-1$ queries. We construct an adversarial function in $\mathcal{F}_t$ against $\mathcal{B}$ as follows: let x^1 be the first point generated and queried by $\mathcal{B}$. Let i_1 be the index of the largest (in absolute value) coordinate of x^1. Choose $\varepsilon^*_{i_1} \in \{\pm 1\}$ in such a way that $\varepsilon_{i_1} x^1_{i_1} = |x^1_{i_1}|$. Choose $\delta^*_{i_1} = \frac{1}{2t^{1/p}}$. Then we may consider the following subfamily of $\mathcal{F}_t$:

$$\mathcal{F}^{(1)} = \left\{ f(x) = \max_{i \in [t]} [\varepsilon_i x_i + \delta_i] : \begin{pmatrix} \varepsilon_j \in \{\pm 1\}, \forall j \ne i_1 \\ \varepsilon_{i_1} = \varepsilon^*_{i_1} \\ \delta_{i_1} = \delta^*_{i_1} > \max_{j \ne i_1} \delta_j \ge 0 \end{pmatrix} \right\}.$$

Intuitively, this means that we fix the parameters $\varepsilon_{i_1}, \delta_{i_1}$ to $\varepsilon^*_{i_1}, \delta^*_{i_1}$ respectively, and choose every other δ_j to be less than $\delta^*_{i_1}$. Note that every function in $\mathcal{F}^{(1)}$ behaves the same way in a neighborhood of x^1 (because the "max" operator always takes the $(x_{i_1}\varepsilon_{i_1} + \delta_{i_1})$ term). Therefore, after making one query, $\mathcal{B}$ cannot distinguish functions in $\mathcal{F}$.

Let us proceed. At each time step $\ell \in [t-1]$, suppose that the algorithm $\mathcal{B}$ has queried $\ell - 1$ points $x^1, \ldots, x^{\ell-1}$ and is going to query x^ℓ. Also suppose we have a subfamily of functions from the last step, denoted by $\mathcal{F}^{(\ell-1)}$. We then calculate:

- For every $s \le \ell$, let $i_s = \arg\max_{i \notin \{i_1, \ldots, i_{s-1}\}} |x^s_i|$. Choose $\varepsilon^*_{i_s}$ such that $\varepsilon^*_{i_s} \cdot x^s_{i_s} = |x^s_{i_s}|$.
- Assume we have constructed $\delta^*_{i_1}, \ldots, \delta^*_{i_{\ell-1}}$ before the start of the ℓ-th step. We choose $\delta^*_{i_\ell}$ such that it is slightly smaller than $\delta^*_{i_{\ell-1}}$. Here, we maintain an invariant that $\delta^*_{i_1} > \delta^*_{i_2} > \cdots > \delta^*_{i_s}$.

Then we construct

$$\mathcal{F}^{(\ell)} = \left\{ f(x) = \max_{i \in [t]} [\varepsilon_i x_i + \delta_i] : \begin{pmatrix} \varepsilon_j \in \{\pm 1\}, \forall j \notin \{i_1, \ldots, i_\ell\} \\ \varepsilon_{i_1} = \varepsilon^*_{i_1}, \varepsilon_{i_2} = \varepsilon^*_{i_2}, \ldots, \varepsilon_{i_\ell} = \varepsilon^*_{i_\ell} \\ \delta_{i_1} = \delta^*_{i_1} > \delta_{i_2} = \delta^*_{i_2} > \cdots > \delta_{i_\ell} = \delta^*_{i_\ell} > \max_{j \notin \{i_1, \ldots, i_\ell\}} \delta_j \ge 0 \end{pmatrix} \right\}.$$

Intuitively, this means that we fix one coordinate of ε and δ in each step. We fix ε_i in such a way that its sign always agrees with the query variable. We choose δ_i in a decreasing order, with arbitrarily small gap between every two consecutive steps.

It is not hard to verity that (1) $\mathcal{F}^{(\ell)}$ is a subfamily of $\mathcal{F}^{(\ell-1)}$; and (2) the algorithm $\mathcal{B}$ cannot distinguish functions in $\mathcal{F}^{(\ell)}$ after making ℓ queries. The second claim follows by observing that for every function in $\mathcal{F}^{(\ell)}$, the local maximizers around points $x^1, \ldots, x^\ell$ always appear among the ℓ chosen coordinates $\{i_1, \ldots, i_\ell\}$, whose associated parameters ε_i, δ_i are fixed inside $\mathcal{F}$.

After $t-1$ steps of query, let x^t denote the final output of the algorithm. We apply the construction above again and get the final function family $\mathcal{F}^{(t)}$. Choose an arbitrary function f_t from $\mathcal{F}^{(t)}$. On the one hand, for every x^q, $q \in [t]$, we have $f_t(x^q) \ge \varepsilon^*_{i_q} x^q_{i_q} + \delta_{i_q} \ge 0$. On the other hand, consider an adversarial

input x^* such that $x^* = -\frac{1}{t^{1/p}}\varepsilon^*$ (we interpret ε^* as a vector, and take coordinate-wise operation on it). Note that $\|x\|_p = 1$ and

$$f_t(x^*) \leq -\frac{1}{t^{1/p}} + \max_{i\in[t]} \delta_i \leq -\frac{1}{2t^{1/p}}.$$

This shows that $f_k(\overline{x}) - f^* \geq \frac{1}{2t^{1/p}}$ as desired. □

2 Perfect matching in bipartite graphs

Much of this section is derived from Section 7.6 in [1]. We stressed in the beginning of our study of mirror descent that one of its most powerful properties is the flexibility by which we can choose our updates, in the sense that the g^t appearing in the algorithm can be *anything*, not just a gradient. We have the following general statement:

Theorem 3. *Consider exponential gradient descent (EGD) with the modification that we no longer interpret g^t as a gradient, but still requiring $\|g^t\|_\infty \leq G$. Then*

$$\frac{1}{T}\sum_{i=0}^{T-1} \langle g^t, p^t - p\rangle \leq \epsilon, \tag{3}$$

if $\eta = \Theta\left(\sqrt{\frac{\log n}{TG^2}}\right)$ and $T = \Theta(\frac{G^2 \log n}{\epsilon^2})$, where $p = \arg\min_{p\in\Delta_n} \frac{1}{T}\sum_{i=0}^{T-1} \langle g^t, p\rangle$ and Δ_n is the n-dimensional simplex.

Proof The proof is identical to that of exponential gradient descent. □

We now apply this result to the problem of perfect matching in bipartite graphs. The setup is as follows: consider an undirected, unweighted graph with vertex set $V = A \cup B$ such that A and B are disjoint, and every edge e in the edge set E has one end in A and one end in B – this is called a *bipartite graph*. A *perfect matching* of G is a subset of edges $M \subseteq E$ such that each vertex $v \in V$ is incident to exactly one edge in M. Notice that a perfect matching can exist only if $|A| = |B| = n$ for total vertices $|V| = 2n$. We set $m = |E|$. We will proceed by recasting the perfect matching problem as the following linear program:

$$\begin{aligned} \text{Find} \quad & x \in \mathbb{R}^m \\ \text{s.t.} \quad & \sum_{e\in E} x_e = n, \\ \forall v \in V, \quad & \sum_{e:v\in e} x_e \leq 1, \\ \forall e \in E, \quad & x_e \geq 0. \end{aligned} \tag{4}$$

This is a continuous relaxation of the perfect matching problem: its solutions are called *fractional perfect matchings*, and proving that this is equivalent to the original integral perfect matching is a non-trivial problem in combinatorial optimization. One direction is quite simple: if $M \subseteq E$ is a perfect matching, then its indicator vector $1_M \in \mathbb{R}^m$ is a fractional perfect matching. The indicator 1_M is defined as having 1 in the eth component if $e \in M$, and 0 otherwise.

Now, our goal will be to find an ϵ-optimal solution; we will slightly relax the condition that $\sum_{e:v\in e} x_e \leq 1$ to $\sum_{e:v\in e} x_e \leq 1 + \epsilon$ for all $v \in V$. That is, we will be looking for solutions that only approximately satisfy the fractional perfect matching constraints – indeed, all of our continuous optimization algorithms can only

guarantee ϵ-accuracy in finite time. It turns out that this relaxation allows us to find matching of cardinality at least $(1-\epsilon)n$ in G, so taking $\epsilon < \frac{1}{n}$ will be sufficient to get a matching in G.

Our algorithm to solve approximate fractional perfect matching will be as follows: we need to input a graph G, $T > 1$, and $\eta > 0$. Then we run the algorithm:

Algorithm 1 Approximate fractional perfect matching

Initialize:
$w^0 = (1, \ldots, 1) \in \mathbb{R}^{2n}$
for $t = 0, \ldots, T-1$ **do**
$\sum_{v \in V} w_v^t \left(\sum_{e: v \in e} x_e^t \right) \leq \sum_{v \in V} w_v^t$ and
$\sum_{e \in E} x_e^t = n$ and
$x_e^t \geq 0$ for all $e \in E$

Construct $g^t \in \mathbb{R}^{2n}$ as $g_v^t = \frac{1}{n}\left(1 - \sum_{e: v \in e} x_e^t\right)$

Update $w_v^{t+1} = w_v^t \cdot \exp(-\eta \cdot g_v^t)$ for all $v \in V$
end for
return $x = \frac{1}{T} \sum_{t=0}^{T-1} x^t$

We first get some intuition for how this algorithm works. We interpret the w vectors as weight vectors that, in some sense, quantify how much we violate the constraint that $\sum_{e: v \in e} x_e \leq 1 + \epsilon$. At every iteration, it tries to shrink the components of x^t to satisfy the constraint. We see that the algorithm automatically satisfies the other 2 constraints in the linear program, by virtue of a restricted search space. We now prove the following theorem:

Theorem 4. *Given a bipartite graph G with $2n$ vertices and m edges that has a perfect matching, for $\epsilon > 0$, Algorithm 1 outputs an ϵ-approximate fractional perfect matching for G in time $O(n^2 m/\epsilon^2)$.*

Proof Assume $\|g^t\|_\infty \leq 1$ and that a valid x^t exists at every step. We provide a lemma proving both of these statements at the end of the notes. For now, introduce the familiar notation:

$$p^t = \frac{w^t}{\|w^t\|_1}. \tag{5}$$

Recall the performance guarantee from EGD that $\frac{1}{T}\sum_{t=0}^{T-1} \langle g^t, p^t - p \rangle \leq \delta$ for $T = \Theta\left(\frac{\log n}{\delta^2}\right)$. Then choose $p = e_v \in \mathbb{R}^{2n}$ for some $v \in V$, where e_v is vector with a 1 in the v-th component, and 0s otherwise. Then plugging into the performance guarantee gives

$$-\frac{1}{T}\sum_{t=0}^{T-1} g_v^t \leq -\sum_{t=0}^{T-1} \langle p^t, g_v^t \rangle + \delta, \tag{6}$$

where $\langle g^t, p \rangle = \langle g^t, e_v \rangle = g_v^t$. Now, we know that $\sum_{v \in V} w_v^t \left(\sum_{e: v \in e} x_e^t \right) \leq \sum_{v \in V} w_v^t$, which implies

$$\sum_{v \in V} w_v^t \left(1 - \sum_{e: v \in e} x_e^t\right) \geq 0. \tag{7}$$

Noticing the similarity to g_v^t, we can just normalize the LHS above by dividing by n and $\|w^t\|_1$, and we arrive at

$$\langle p^t, g^t \rangle \geq 0. \tag{8}$$

Now returning to our performance guarantee, we can use this to get the new bounds

$$\frac{1}{Tn}\sum_{t=0}^{T-1}\left(\sum_{e:v\in e} x_e^t - 1\right) \leq -\sum_{t=0}^{T-1}\langle p^t, g_v^t\rangle + \delta \leq \delta. \tag{9}$$

Rearranging terms, we find

$$\sum_{e:v\in e} \bar{x}_e \leq 1 + n\delta. \tag{10}$$

Then choosing $\delta = \epsilon/n$ gives $T = \Theta\left(\frac{n^2 \log n}{\epsilon^2}\right)$, and the fact that $\bar{x}_e$ is a convex combination of each iterate implies that it satisfies the constraints enforced at every iterate: $\sum_{e\in E}\bar{x}_e = n$ and $\bar{x}_e \geq 0$. We hence see that $\bar{x}$ is an ϵ-approximate solution to fractional perfect matching on G.

Finally, we know that the number of iteration is $O\left(\frac{n^2 \log n}{\epsilon^2}\right)$, and each iteration takes $O(m)$ time (as we show in the existence lemma below), dominated by the search for an edge with minimal $\sum_{e:v\in e} w_v^t$. □

We remark that while this is not the most efficient known algorithm for perfect matching, it is remarkably simple, and requires essentially no modification to the EGD algorithm.

We finish off these notes with the promised existence lemma:

Lemma 5. *If G has a perfect matching, then x^t satisfying the requirements for Algorithm 1 exists and can be found in $O(m)$ time, and $\|g^t\|_\infty \leq 1$.*

Proof Begin by rewriting the weight condition $\sum_{v\in V} w_v^t \left(\sum_{e:v\in e} x_e^t\right) \leq \sum_{v\in V} w_v^t$ as

$$\sum_{e\in E} \alpha_e x_e \leq \beta, \tag{11}$$

where

$$\alpha_e = \sum_{e:v\in e} w_v^t, \quad \beta = \sum_{v\in V} w_v^t. \tag{12}$$

If G has a perfect matching M, then there exist edges $e^1, \ldots, e^n$ that do not share any vertices – this is the definition of a perfect matching. Hence, for these edges, it must the case that

$$\sum_{i=1}^{n} \alpha_{e^i} = \beta. \tag{13}$$

Now define $e^\star = \arg\min_{e\in E} \alpha_e$. Then we get the simple bound

$$n\alpha_{e^\star} \leq \sum_{i=1}^{n} \alpha_{e^i} = \beta. \tag{14}$$

Then if we choose $x_{e^\star}^t = n$ and $x_e^t = 0$ for all $e \neq e^\star$, the resulting x^t satisfies the weight condition in Equation 11. This takes $O(m)$ time to search through all edges on the graph to find $e^\star$. Finally, notice that for this choice of x^t, we have the following bound for all $v \in V$:

$$-1 \leq \sum_{e:v\in e} x_e^t - 1 \leq n - 1, \tag{15}$$

which follows from observing that $\sum_{e:v\in e} x_e^t \geq 0$ by the $x_e^t \geq 0$ condition, and $\sum_{e:v\in e} x_e^t \leq n$ follows from the condition that $\sum_{e\in E} x_e^t = n$ in combination with $x_e^t \geq 0$. This then gives us the desired bound that $\|g^t\|_\infty \leq 1$ by plugging in the definition of g^t. □

References

[1] N. K. Vishnoi, *Algorithms for Convex Optimization.* Cambridge University Press, 2021.