Online Convex Optimization & Complexity
Problem Set 6 – CS 6515/4540 (Fall 2025)

This problem set is due on **Monday November 17th**. Submission is via Gradescope. Your solution must be a typed pdf (e.g. via LaTeX) – no handwritten solutions.

22 Online Gradient Descent

Online gradient descent is frequently utilized with stochastic gradient descent. Suppose that we want to run online gradient descent but cannot query the gradient of our convex function f_t, however we can construct a random vector variable G_t that estimates ∇f_t at any x_t:

$$\nabla f_t(x_t) = \mathbb{E}[G_t]$$

Show that if we run Online Gradient Descent using G_t as our gradient of f_t at time t, then we have

$$\mathbb{E}\left[\sum_{t=0}^{T} f_t(x_t) - f_t(x^*)\right] \leq DG\sqrt{T}$$

where G in this case will satisfy $\mathbb{E}[||G_t||^2] \leq G^2$ for all t.

(Sketch):
We proceed analogously as in class. Define $\Phi(t) = \mathbb{E}\left[\frac{1}{2\eta}||X_t - x^*||^2\right]$ where X_t is the r.v. representing x_t. Then just like in lecture but using linearity of expectation, we have

$$\Phi(t+1) = \Phi(t) + \frac{\eta}{2}\mathbb{E}[||G_t||^2] - \mathbb{E}[\langle G_t, X_t - x^*\rangle]$$

Note by the problem assumptions,

$$\mathbb{E}[||G_t||^2 | X_t = x_t] \leq G^2$$

$$\mathbb{E}[\langle G_t, (X_t - x^*)\rangle | X_t = x_t] = \langle \mathbb{E}[G_t | X_t = x_t], (x_t - x^*)\rangle = \langle \nabla f_t(x_t), x^t - x^*\rangle \geq f(x_t) - f(x^*)$$

It can be proven (we won't do it here) that for any random variable Y and functions g and h that take in Y and possibly other r.v.'s as input, that if

$$\mathbb{E}(g(Y, ...)|Y = y] \leq h(y)$$

then

$$\mathbb{E}[g(Y, ...)] \leq \mathbb{E}[h(Y)]$$

(and the same kind of statement holds for $\geq$)
Hence it is easy to see that

$$\mathbb{E}[f(x_t) - f(x^*)] \leq \Phi(t) - \Phi(t+1) + \frac{\eta}{2}G^2$$

and now we can finish off our proof analogous to lecture by summing over all t and using linearity of expectation.

23 Missing Proof Details

1. In Lecture 20 we assumed that the following convex body K has diameter $O(\sqrt{n})$. Prove that.

$$K = \{x \in \mathbb{R}^{n \times n}_{\geq 0} \mid \sum_i x_{ij} = 1 \quad \forall j \quad \text{and} \quad \sum_j x_{ij} = 1 \quad \forall i\}.$$

> Note that for each column of x, the norm is at most 1 since the sum of all elements in that column sum to 1. Hence, the Frobenius norm of the entire x is at most $\sqrt{1^2 \times n} = \sqrt{n}$ since the (Frobenius) norm of x is the square root of the sum of the squared norms of each of its columns. Hence by triangle inequality, for any two matrices X and Y, $||X - Y|| \leq ||X|| + ||Y|| \leq 2\sqrt{n}$ and so we're done.

2. In Lecture 21 while proving the minimax theorem, by applying OGD and taking T large enough, we obtained

$$\frac{1}{T}\sum_t M(x_t, a_t) \geq \frac{1}{T}\max_i \sum_t M(i, a_t) - \epsilon.$$

Prove why the LHS is at most expression B and why the RHS is at least expression A$-\epsilon$ to complete the proof of the minimax theorem.

> Note that
>
> $$\max_t M(x_t, a_t) \geq \frac{1}{T}\sum_t M(x_t, a_t)$$
>
> . Let $t = \arg\max_t M(x_t, a_t)$. Note $a_t = \arg\min_s M(x_t, s)$. Hence the
>
> $$LHS \leq \min_s M(x_t, s) \leq \max_X \min_s M(X, s) = B$$
>
> For the RHS, note that the left term we can view it as which action can player 1 best respond with assuming player 2 plays the fixed strategy of choosing any action a_t w.p. $\frac{1}{T}$, hence it must be more than A and hence $RHS \geq A - \epsilon$

3. Suppose you run online gradient descent in n-dimensions where for all t, $f_t(x) = \sum_{i=1}^n a_i^t x_i$ for some non-negative reals $a_i^t \in [0, A]$ and where convex body $K = \{x \in \mathbb{R}^n_{\geq 0} \mid \sum_i x_i = B\}$ for some positive B. After T iterations of online gradient descent, what is the best possible upper bound on the regret of the algorithm? (Give the answer in Big-O notation in terms of parameters A, B, and n.)

> Note $D = \sqrt{2}B$ (the diameter of a convex body can be realized at two vertices) and $||G|| = ||a^t|| \leq A\sqrt{n}$. Hence our best possible upper bound on regret is $O(AB\sqrt{nT})$ by the guarantees of OGD being $DG\sqrt{T}$

24 Zero Sum Games

Consider a zero-sum game between players A and B with A having actions $\{A_1, A_2\}$ and B having actions $\{B_1, B_2\}$. The entries in this table correspond to the rewards of player A (i.e., the costs of player B):

	B_1	B_2
A_1	2	-1
A_2	-1	1

1. If player A has to commit a strategy before player B (i.e., B will choose their best strategy/action knowing A's randomized strategy), what strategy would they adopt and how much expected reward would it get?

   (Hint: First solve for a general randomized strategy playing A_1 with probability p and A_2 with probability $1-p$, and then optimize p.)

   Let A play A_1 with probability p and A_2 with probability $1-p$.
   If B plays B_1, A's expected payoff is

   $$u(B_1) = 2p + (-1)(1-p) = 3p - 1.$$

   If B plays B_2, A's expected payoff is

   $$u(B_2) = (-1)p + 1(1-p) = 1 - 2p.$$

   Since A commits first and B best-responds, A solves

   $$\max_{0 \le p \le 1} \min(3p - 1,\ 1 - 2p).$$

   The optimum occurs when the expressions are equal:

   $$3p - 1 = 1 - 2p \quad \Rightarrow \quad 5p = 2 \quad \Rightarrow \quad p = \frac{2}{5}.$$

   At $p = \frac{2}{5}$,

   $$3p - 1 = 1 - 2p = \frac{1}{5}.$$

   Thus A plays A_1 w.p. $\frac{2}{5}$ and A_2 w.p. $\frac{3}{5}$, and the expected reward is $\frac{1}{5}$.

2. Now answer the above question when player B has to first commit a strategy (and then player A chooses their best strategy/action): what strategy would B adopt and how much would be the expected cost?

   Let B play B_1 with probability q and B_2 with probability $1-q$.
   If A plays A_1, expected payoff is

   $$u(A_1) = 2q + (-1)(1-q) = 3q - 1.$$

   If A plays A_2, expected payoff is

   $$u(A_2) = (-1)q + 1(1-q) = 1 - 2q.$$

   Since B commits first and A best-responds, B solves

   $$\min_{0 \le q \le 1} \max(3q - 1,\ 1 - 2q).$$

   The optimum occurs when

   $$3q - 1 = 1 - 2q \quad \Rightarrow \quad 5q = 2 \quad \Rightarrow \quad q = \frac{2}{5}.$$

   At $q = \frac{2}{5}$,

   $$\max(3q - 1,\ 1 - 2q) = \frac{1}{5}.$$

   Thus B plays B_1 w.p. $\frac{2}{5}$ and B_2 w.p. $\frac{3}{5}$, and the expected cost (A's payoff) is $\frac{1}{5}$.

3. Are the answers to both the above parts the same? Why does it agree with the minimax theorem?

Both parts (1) and (2) give the same value:

$$\frac{1}{5}.$$

This is exactly the statement of the minimax theorem, which guarantees that in zero-sum games,

$$\max_{\text{A strategies}} \min_{\text{B actions}} \mathbb{E}[\text{payoff}] = \min_{\text{B strategies}} \max_{\text{A actions}} \mathbb{E}[\text{payoff}].$$

25 Reductions: Dominating Set

Given a graph $G = (V, E)$, dominating set $S \subseteq V$ is a subset of vertices such that every vertex $v \in V$ has at least one neighbor in S. Dominating set problem asks to find the minimum dominating set of G.

1. Prove that an α-approx algorithm for set cover implies an α-approx algorithm for dominating set.

Given a graph $G = (V, E)$, we create a Set Cover instance as follows.
The universe is $U = V$. For every vertex $v \in V$, we make a set that contains v and all of its neighbors. Call this set S_v.
A dominating set D in G has the property that every vertex is either in D or has a neighbor in D. This means the sets $\{S_v : v \in D\}$ cover all of V. So any dominating set gives a valid set cover of the same size.
Conversely, if we have a set cover made of sets $\{S_{v_1}, \ldots, S_{v_k}\}$, then choosing the corresponding vertices $\{v_1, \ldots, v_k\}$ gives a dominating set in G. So both problems have the same optimal value.
Now run the α-approximation algorithm for Set Cover on this instance. If it selects some sets S_v, we simply output the corresponding vertices v as our dominating set.
Since the two problems have the same optimum, and we get an α-approximate set cover, we also get an α-approximate dominating set.

2. Prove that an α-approx algorithm for dominating implies an α-approx algorithm for set cover.

Given a Set Cover instance with universe U and sets $S_1, \ldots, S_n$, we build a graph so that dominating sets correspond to set covers.
We create one vertex s_i for each set S_i, and one vertex x for each element of U. We connect s_i to x whenever the element x belongs to the set S_i. Now, add an edge between s_i and s_j for all i, j. This is the entire construction.
Now observe:

- If C is a set cover, say it includes sets $S_{i_1}, \ldots, S_{i_k}$, then the vertices $s_{i_1}, \ldots, s_{i_k}$ dominate every element-vertex x (because each x belongs to at least one chosen set). So a set cover directly gives a dominating set of the same size.
- Conversely, if D is a dominating set in this graph, each element-vertex x must be adjacent to some $s_i \in D$. Note that if some $x \in U$ is also in D, we can instead replace x with one of its neighbor vertices to create a new dominating set D' such that $|D'| = |D|$, hence we can transform any dominating set into a dominating set using only the vertices s_i where $1 \leq i \leq n$. Since we get a dominating set using only such vertices, this means the corresponding sets S_i cover all elements. So a dominating set gives a set cover of the same size.

Thus the two problems have the same optimum value.
Now we run the α-approximation algorithm for Dominating Set on this graph and get some dominating set D. We output the sets S_i for which $s_i \in D$.
Because dominating sets and set covers correspond one-to-one with equal sizes, and we obtained an α-approximate dominating set, this also gives an α-approximate set cover.

26 Fine-Grained Complexity

Consider two variants of orthogonal vectors problem.

- Problem A: Given set S of n vectors $s_1, \ldots, s_n$ in $\{0,1\}^d$, find two vectors s_i, s_j that are orthogonal to each other.
- Problem B: Given two sets V, W of $n/2$ vectors each in $\{0,1\}^d$, where $V = \{v_1, \ldots, v_{n/2}\}$ and $W = \{w_1, \ldots, w_{n/2}\}$, find two vectors v_i, w_j that are orthogonal to each other.

Prove that these two problems are equivalent in the sense that there exists $O(n^{2-\delta}d)$ runtime algorithm for one, where $\delta > 0$ is some constant, iff there exists $O(n^{2-\delta}d)$ runtime algorithm for the other one.

Problem B ⇒ Problem A
Assume we have an $O(n^{2-\delta}d)$-time algorithm for Problem A: given n vectors in $\{0,1\}^d$, it finds two orthogonal ones.
We are given an instance of Problem B: two sets V and W, each of size $n/2$, of vectors in $\{0,1\}^d$. We now build a single set S of n vectors in a slightly higher dimension, and then run the algorithm for Problem A on S.
For each $v \in V$, define a new vector

$$v' := (v, 1, 0) \in \{0,1\}^{d+2},$$

and for each $w \in W$, define

$$w' := (w, 0, 1) \in \{0,1\}^{d+2}.$$

Let S be the set of all these n vectors.
Now check what happens to dot products:

- For $v_1, v_2 \in V$, we have

  $$v_1' \cdot v_2' = v_1 \cdot v_2 + 1 \cdot 1 + 0 \cdot 0 = v_1 \cdot v_2 + 1 \geq 1.$$

  So no two vectors coming from V are orthogonal in the new instance.

- For $w_1, w_2 \in W$, we have

  $$w_1' \cdot w_2' = w_1 \cdot w_2 + 0 \cdot 0 + 1 \cdot 1 = w_1 \cdot w_2 + 1 \geq 1.$$

  So no two vectors coming from W are orthogonal either.

- For $v \in V$ and $w \in W$, we get

  $$v' \cdot w' = v \cdot w + 1 \cdot 0 + 0 \cdot 1 = v \cdot w.$$

  Thus v' and w' are orthogonal if and only if v and w are orthogonal.

So there is an orthogonal pair in S if and only if there is an orthogonal pair (v, w) with $v \in V$ and $w \in W$ in the original Problem B instance.
We can now run the Problem A algorithm on the set S of n vectors in dimension $d + 2$. Its running time is $O(n^{2-\delta}(d+2)) = O(n^{2-\delta}d)$. From the orthogonal pair it returns, we can recover an orthogonal pair (v, w) for Problem B. This gives an $O(n^{2-\delta}d)$ algorithm for Problem B.

Problem A $\Rightarrow$ Problem B.

Now assume we have an $O(n^{2-\delta}d)$-time algorithm for Problem B: given two sets V and W, each of size $n/2$, it finds orthogonal $v \in V$ and $w \in W$.
We are given an instance of Problem A: a single set S of n vectors in $\{0,1\}^d$.
Split S into two halves:

$$V = \{s_1, \ldots, s_{n/2}\}, \qquad W = \{s_{n/2+1}, \ldots, s_n\}.$$

There are three possibilities for an orthogonal pair in S:

- one vector in V and one in W,
- both vectors in V,
- both vectors in W.

We can cover all three cases using the Problem B algorithm as follows:

- Run the Problem B algorithm on (V, W) to find an orthogonal pair crossing the partition (if it exists).
- Run it on (V, V) to find an orthogonal pair both in V (if it exists).
- Run it on (W, W) to find an orthogonal pair both in W (if it exists).

Each call has input size $n/2$ (per side), so each runs in time $O((n/2)^{2-\delta}d) = O(n^{2-\delta}d)$. We only make a constant number of calls (three), so the total time is still $O(n^{2-\delta}d)$.
If any of these calls finds an orthogonal pair, that pair is also an orthogonal pair in the original Problem A instance. Therefore, we obtain an $O(n^{2-\delta}d)$-time algorithm for Problem A.

A Further Practice Problems

A.1 Experts Problem

Prove that the greedy algorithm, which in each round $t \in [T]$ plays the best expert based on performance in rounds $\{1, \ldots, t-1\}$, could incur $\Omega(T)$ total regret.

(Hint: There is an example already with $n = 2$ experts.)