Exercises

1. **Blocking Experts.** (due to Avrim Blum) Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

   (a) Each expert begins with weight 1 (as before).
   (b) We predict the result of a weighted-majority vote of the experts (as before).
   (c) If an expert makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least \( \frac{1}{4} \) of the average weight of experts.

   Prove that in any contiguous block of trials (e.g., the 51st day through the 77th day), the number of mistakes made by the algorithm is at most \( O(m + \log N) \), where \( m \) is the number of mistakes made by the best expert in that block, and \( N \) is the total number of experts.

2. **Lots of Flows.** Suppose you wanted to find an approximate solution to the following “multi-commodity” flow problem: given a digraph \( G = (V, E) \) with unit arc capacities, send \( F_i \) flow from node \( s_i \) to node \( t_i \) in the graph, for all \( i \in [k] \). You should imagine that the flow from \( s_i \) to \( t_i \) is of commodity \( i \) (e.g., oil, water, sand...) which are all distinct.

   (a) Suppose \( P_i \) is the set of all paths from \( s_i \) to \( t_i \): show that the following LP captures the problem we are trying to solve. The variables are \( f_P \), one for each path in \( \cup_i P_i \).

   \[
   \begin{align*}
   \sum_{P \in P_i} f_P &= F_i & \forall i \in [k] \\
   \sum_i \sum_{P \in P_i ; e \in P} f_P &\leq 1 & \forall e \in E \\
   f &\geq 0
   \end{align*}
   \]

   (b) Define an appropriate “easy” polytope \( K \) for this problem.

   (c) Given weights \( q \in \Delta_m \), how would you solve the oracle for this problem? Show you can find a flow that satisfies the demands, but uses at most \( (1 + \varepsilon) \) capacity on each edge, in time \( O\left(\frac{k(m+n \log n)}{\text{poly}(\varepsilon)}\right) \cdot \left(\sum_i F_i\right) = O\left(\frac{km(m+n \log n)}{\text{poly}(\varepsilon)}\right) \).

3. **Strength in Convexity.** A function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is called \( \ell \)-strongly-convex if

   \[
   f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\ell}{2} \| y - x \|^2.
   \]

   I.e., if the function is not just convex, but “locally it grows at least as fast as a quadratic”. Modify the basic gradient descent analysis to show that using the same update rule \( x_{t+1} \leftarrow x_t - \eta_t \nabla f(x_t) \) with suitably chosen \( \eta_t \), then we can find \( \hat{x} \in \mathbb{R}^n \) such that

   \[
   f(\hat{x}) - f(x^*) \leq O\left(\frac{G^2 \log T}{\ell \cdot T}\right)
   \]

   Again, assume that \( \| \nabla f(x) \| \leq G \). Note due to the assumption of strong convexity, we got better convergence (the dependence on \( T \) is better, there is no dependence on \( D = \| x_0 - x^* \| \).

   Show that this analysis also works in the online case, if each function is strongly convex. *(Bonus: remove the log \( T \) term in the numerator in the offline case. Why does this not extend to the online case?)*
Problems

1. Sherman-Morrison, and Dynamic Algorithms. Suppose graph $G_0$ has a perfect matching to start off, and someone starts adding and removing edges. Each update step adds or removes a single edge $e_i$ from $G_{i-1}$ to give $G_i$. At each time $i$, we want to answer “does $G_i$ still have a perfect matching?” Of course, we could simply answer the question for each $G_i$ independently, but let us try to be smarter.

The Sherman-Morrison(-Woodbury) formula\(^1\) says: for any non-singular $n \times n$ matrix $A$, and any $n$-dimensional vectors $u, v$,

\[
(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.
\]

(Assuming that $1 + v^\top A^{-1}u \neq 0$.) Moreover, the matrix determinant lemma says

\[
\det(A + uv^\top) = \det(A)(1 + v^\top A^{-1}u).
\]

These tell us how the inverse and determinant of a matrix change under rank-one updates, as long as these updates don’t make the matrices singular.

(a) Use these to show that for $i \geq 1$, if $G_0, G_1, \ldots, G_{i-1}$ all have perfect matchings, we can answer the question for $G_i$ correctly with probability $1 - O(1/poly(n))$, but in time $\tilde{O}(n^2)$. (Be sure to mention what field $\mathbb{F}_q$ you are working over.)\(^2\) You may assume that the graph $G_i$ are all bipartite.

(As soon as $G_{i-1}$ has no perfect matching, your algorithm may be incorrect for all subsequent $G_j$, $j \geq i$. Bonus: suggest ways to handle this shortcoming.)

This general idea of using Sherman-Morrison for rank-one updates gives dynamic algorithms for several other problems. Here is another example, where we need to count the number of paths between vertices in DAGs.

(b) For a square matrix $M$ such that $M^k = 0$ for some positive integer $k$, show that

\[
(I - M)^{-1} = \sum_{i=0}^{k-1} M^i.
\]

(c) For a DAG $G$, let $A$ be the $n \times n$ adjacency matrix: $A_{ij} = 1$ if and only if $(i, j)$ is an arc in $G$. Show that the $(i, j)^{th}$ entry of $(I - A)^{-1}$ is the number of paths between vertices $i, j$ in $G$. Hence, infer that you can compute the number of paths from $i$ to $j$, for all $i, j$, in time $O(n^\omega)$.

(d) Show how to maintain these path counts under edge additions and deletions to the graph, as long as it remains a DAG. (Make sure you argue that you don’t perform any operations that cause matrices to become singular, etc.) The time for each update should be $O(n^2)$.

---

\(^1\)See this stackexchange discussion for intuition, or the wikipedia article for other pointers.

\(^2\)The naive way would be to use the Tutte-Lovasz approach for each timestep independently, which would require $O(n^{\omega}) \gg O(n^2)$ time per update: your goal is to do better.
2. Zero-Sum Games using LP Duality. Recall the zero-sum game setup: we’re given a matrix $M \in \mathbb{R}^{m \times n}$; if the row player plays a strategy $x \in \Delta_m$ and the column player plays strategy $y \in \Delta_n$, the payoff to the row player is $x^\top My$.

If we define $C(x) = \min_{y \in \Delta_n} x^\top My$, and $R(y) = \max_{x \in \Delta_m} x^\top My$, the minimax theorem proves that (a) for all $x, y$, $C(x) \leq R(y)$, and moreover (b) there exist $x^*, y^*$ such that $C(x^*) = R(y^*)$.

(a) Show an LP to compute $\max_x C(x)$, the optimal strategy for the row player. (Hint: be careful, the definition of $C()$ has a min sitting in there, so you’re looking to find $\max_x \min_y x^\top My$, which certainly does not look like a linear program.)

(b) Show an LP to compute $\min_y R(y)$, the optimal strategy for the column player.

(c) Show that you can, in fact, find LPs for both the above parts, such that the dual of the first LP is a solution to the second part.

Now use weak duality to infer the first part of the minimax theorem, and strong duality to infer the second part.

3. Boost Your Score. Suppose you have a set of points $S \subseteq \mathbb{R}^d$, and for each point $x \in S$, you are also given a weight $w(x) \geq 1$ and a label $\ell(x) \in \{0, 1\}$. A function $f : \mathbb{R}^d \to \{0, 1\}$ is called a hypothesis. If $W := \sum_{x \in S} w(x)$, the error of the hypothesis on the set $S$ is

$$\text{err}_{S,\ell,w}(f) := \frac{1}{W} \sum_{x \in S} w(x) \cdot 1(f(x) \neq \ell(x)),$$

i.e., the fraction of weight on points in $S$ that $f$ labels incorrectly.\(^3\)

Suppose you know an algorithm $A$ that is “weak learner” for class $\mathcal{H}$: given any set $S \subseteq \mathbb{R}^d$ (along with its labeling $\ell()$), it will return a function $f \in \mathcal{H}$, such that $\text{err}_{S,\ell,w}(f) \leq \frac{1}{2} - \eta$ for some $\eta > 0$. If $\eta$ is small, this means the returned hypothesis is only slightly better than a random coin toss.

(a) Give an algorithm that, given $S$, along with the weights $w()$ of the points, and their labeling $\ell()$, runs $A$ for $T = O\left(\frac{1}{\eta^2} \log \frac{1}{\varepsilon}\right)$ times, and outputs a function $\hat{f}$ of the form:

$$\hat{f}(x) = \text{majority}(f_1(x), f_2(x), \ldots, f_T(x))$$

for $f_i \in \mathcal{H}$, such that $\text{err}_{S,\ell,w}(\hat{f}) \leq \varepsilon$. (Hint: As an easier problem, you may try to show that $T' = O\left(\frac{1}{\eta^2} \log W\right)$ suffices. Start off with the case where the weights of the points in $S$ are $w(x) = 1$ for all $x \in S$. Also, you may need to revisit the analysis of Hedge.)

\(^3\)N.b. One perfect hypothesis is of course the original labeling $\ell()$, but the goal usually is to find a hypothesis in some “nice” class $\mathcal{H}$ of functions. E.g., you may like the class of all half-spaces $\mathcal{H}_{lin} = \{f_{a,b}(x) := 1_{(a^\top x \geq b)} \mid a \in \mathbb{R}^d, b \in \mathbb{R}\}$.
4. **Capacitated Max-Flow and Width Reduction.** Consider the directed s-t max-flow problem: in Lecture #14, we defined \( K = \{ f \mid f_P \geq 0, \sum_{P \in P} f_P = F \} \), with constraints

\[
f_e/c_e \leq 1 \quad \forall e \in E,
\]

where define \( f_e := \sum_{P : e \in P} f_P \). We considered \( c_e = 1 \) in lecture; now we consider the general case. Given the weights \( q \in \Delta_m \), the “average” constraint looks like

\[
\sum_{e} q_e \cdot (f_e/c_e) = \sum_{e} f_e(q_e/c_e) \leq 1.
\]

(a) (Do not submit.) Suppose the oracle sends the entire \( F \) units of flow along a shortest path w.r.t. \( q_e/c_e \). Show that there are capacitated networks, and possible \( q \in \Delta_m \), where all \( F \) flow is routed along a path using the least-capacity edge. Hence the width of this oracle is at least \( (F/c_{\min}) \).

Note that \( F \) may be as large as \( O(mc_{\max}) \), so with general capacities, this ratio could be \( \gg m \). Now, we’ll investigate the idea of adding a little bit to the edge weights (e.g., setting \( w_e := q_e + \varepsilon/m \)), and how it reduces the width of the problem at the expense of giving slightly approximate solutions.

For the rest, assume \( \varepsilon \leq 1/10 \), say. You may also assume that the instance is feasible; i.e., there exists a flow \( f^* \in K \) that satisfies all the edge capacities.

(b) Set the edge weights to be \( w_e := q_e + \varepsilon/m \), compute the shortest path w.r.t. edge lengths \( w_e/c_e \), and send all \( F \) flow along it. If this shortest path is \( P^* \), show

\[
F \cdot \sum_{e \in P^*} \frac{w_e}{c_e} \leq \min_{f \in K} \sum_{e \in E} \frac{f_e}{c_e} \leq 1 + \varepsilon.
\]

(c) Show that

\[
\max_{e \in P^*} \frac{F}{c_e} \leq O\left( \frac{m}{\varepsilon} \right).
\]

Hence the oracle width is \( O(m/\varepsilon) \). (Hint: if we define “thin edges” to be those with capacity \( \leq \frac{\varepsilon F}{2m} \), would \( P^* \) ever use a thin edge?)

(d) (Do not submit.) Using this oracle and the MW algorithm guarantee, give an \( \tilde{O}(m^2/\varepsilon^3) \)-time \( (1 + \varepsilon) \)-approximate max-flow algorithm for the capacitated case.

(e) Bonus: use these ideas to get an algorithm for the multi-commodity case from exercise #2 that works for capacitated graphs, but whose runtime does not depend on the magnitude of the capacities.

Note: This idea was used in the Christiano et al. paper, combined with the electrical flows, to bring the width down to \( O(\sqrt{m}/\varepsilon) \).