Homework 6
Due: Tuesday, Dec 6.

Ground rules: *same as for Homework 1.* If you have scribed twice, do 2 out of 5 problems. If you will only scribe once, do 5 out of 5 problems.

---

1. You've Got the Power. Take some psd $n \times n$ matrix D with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, and unit eigenvectors v_i corresponding to λ_i. Recall that the vectors $\{v_i\}_{i=1}^n$ form an orthonormal basis for $\mathbb{R}^n$. Start with a unit vector $x \in \mathbb{R}^n$. Let $\langle x, v_i \rangle = a_i$, so you get $\sum_i a_i^2 = 1$. Consider the following iterative process to approximate λ_1.

> Set $x^{(0)} \leftarrow x$. For $i = 1, 2 \ldots, t$ steps, iteratively define the unit vector
>
> $$x^{(i)} = \frac{Dx^{(i-1)}}{\|Dx^{(i-1)}\|_2}.$$
>
> Let $y \leftarrow x^{(t)}$, and return $\mu := D \bullet yy^\top$.

(a) Show that if $t = \Omega(\varepsilon^{-1} \log(\varepsilon^{-1} a_1^{-2}))$, then $\mu \in [\frac{\lambda_1}{1+\varepsilon}, \lambda_1]$.

(b) (Extra Credit.) To choose the initial vector x, pick $X_1, X_2, \ldots, X_n \sim N(0,1)$, set $\vec{X} = (X_1, X_2, \ldots, X_n)$, and set $x \leftarrow \vec{X}/\|\vec{X}\|_2$. Show that $a_1 \geq 1/\text{poly}(n)$ with probability $1 - 1/\text{poly}(n)$.

(c) Conclude that the algorithm above returns returns a $(1+\varepsilon)$-approximation to $\lambda_{\max}(D)$ with high probability, in time $O(\varepsilon^{-1} \log(n/\varepsilon))$.

2. Live and Learn. Consider a set of points $X = \{x_1, x_2, \ldots, x_n\}$, each with a "correct" label $f(x_i) \in \{0, 1\}$. We are given an algorithm $\mathcal{A}$ that takes a probability distribution $\mathcal{D}$ over $[n]$, and outputs a labeling $h : X \to \{0, 1\}$ such that

$$\Pr_{i \sim \mathcal{D}}[f(x_i) = h(x_i)] \geq \frac{1}{2} + \gamma$$

for some $\gamma > 0$. ($\mathcal{A}$ is called a "weak learner".) Consider the following algorithmic idea:

> Start with $w_i^{(1)} = 1$ for all $i \in [n]$. Repeat the following $T = O(\gamma^{-2} \log \delta^{-1})$ times: use the weak learner $\mathcal{A}$ on the probability distribution $p_i^{(t)} = w_i^{(t)}/(\sum_i w_i^{(t)})$ to get a labeling $h^{(t)}$. For $i \in [n]$ such that $f(x_i) \neq h^{(t)}(x_i)$, change the weight of i by some factor. Output the function $H : X \to \{0, 1\}$ where $H(x_i)$ is the majority of $\{h^{(1)}(x_i), h^{(2)}(x_i), \cdots, h^{(T)}(x_i)\}$.

How should you change the weights so that the number of points in X misclassified by $H()$ is at most δn? Give a full proof.

3. Epsilon Nets and Hitting Sets. Let $(U, \mathcal{F})$ be a set system, where $|U| = n$, and $\mathcal{F} = \{S_1, S_2, \ldots S_m\}$ where $S_i \subseteq U$. Define the two quantities:

- *Hitting Set.* The set $H \subseteq U$ is a *hitting set* for $\mathcal{F}$ if $H \cap S_i \neq \emptyset$ for all $i \in [m]$. Let Z^* be the size of the smallest hitting set for $\mathcal{F}$; this is NP-hard to compute.
- *ε-net.* Given weights w_e for each element $e \in U$, define the weight of a setset as $w(A) = \sum_{e \in A} w_e$. The set $N \subseteq U$ is an *ε-net* for $(\mathcal{F}, w)$ if for all sets S_i such that $w(S_i) \geq \varepsilon\, w(U)$, we have $N \cap S_i \neq \emptyset$. (In other words, N hits all the "high-weight" elements.)

Suppose you are given an algorithm that for any setting of the element weights and any $\varepsilon > 0$, finds an ε-net of size $T(\varepsilon)$. Consider the following algorithm:

i. set $w_e = 1$ for all $e \in U$.
ii. find an $(\frac{1}{2Z^*})$-net N for $(\mathcal{F}, w)$.
iii. if N is not a hitting set for $\mathcal{F}$, pick a set S_i not hit by N, double the weight of all elements in S_i, and goto step ii.

(a) Show that this algorithm terminates after $O(Z^* \log n)$ iterations.

(b) Give a randomized algorithm that given any set system $\mathcal{F}$ with m sets, finds an ε-net of size $T(\varepsilon) = O(\varepsilon^{-1} \log m)$ in expected polynomial time.

(c) Infer the existence of a randomized $O(\log m)$-approximation to the hitting set problem in general set systems.

4. Missing the Forest for the Trees? Given a connected graph $G = (V, E)$, we can write down the spanning tree polytope $P_{tree}(G) \subseteq R^{|E|}$ (see, e.g., the polytope P in Lecture 7), such that the vertices of P_{tree} correspond to spanning trees in G. So, given a point $x \in P_{tree}$ we can write x as a convex combination of spanning trees of G. More precisely, we know there exist $\lambda_T \geq 0$, $\sum_T \lambda_T = 1$ (one for each spanning tree T) such that

$$x = \textstyle\sum_T \lambda_T\, \chi^T \tag{1}$$

where $\chi^T \in \mathbb{R}^{|E|}$ is the characteristic vector of the tree $T \subseteq G$. But how do you find this convex combination (in poly time)? One way is to write the following LP with variables y_T:

$$\begin{array}{rll} \text{maximize} & \sum_T y_T & \\ \text{subject to} & \sum_T \chi^T_e\ y_T = x_e & \forall e \in E \\ & y_T \geq 0 & \end{array}$$

We know that the optimal value of this LP is exactly 1 (if x is indeed in the spanning tree polytope). So a solution to this LP is exactly what we are looking for. But the number of variables in this LP is potentially exponential!

(a) Can you solve this LP in time poly(n)? For full points, give an algorithm to find $\{\lambda_T\}$ satisfying (1) such that only $|E|$ of these λ's are nonzero. (Assume that the size of the numbers in x are also poly(n).)

(b) Show a family of graphs $\{G^n\}_n$ and solutions $x^n \in P_{tree}(G_n)$ for which you need at least $|E|$ trees in the convex combination.

5. The Path Less Traveled. Given a edge-weighted complete graph $G = (V, E)$ where the edge lengths $c : \binom{V}{2} \to \mathbb{R}_{\geq 0}$ satisfy the triangle inequality. Let T^* be the minimum-spanning tree on this graph. For an edge set S, let $c(S) = \sum_{e \in S} c_e$. Fix vertices s, t: a $TSPP$ is a Hamilton path in G that starts at s and ends at t.

(a) Use this tree T^* to find a TSPP of length at most

$$2c(T^*) - c(s, t).$$

(b) Use an argument similar to Cristofides' (again using T^* and OPT) to find a TSPP of length at most

$$\frac{3}{2}OPT + \frac{1}{2}c(s, t).$$

(c) Use the above parts to give a 5/3-approximation to TSPP. (I.e., a poly-time algorithm to find an TSPP in G of length at most $\frac{5}{3}$ times the shortest TSPP.)