Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
Ankur Moitra

Wednesday, April 13, 2016

Problem Set 8

Due: Wednesday, April 20, 2016 – 7 pm
Dropbox Outside Stata G5

Collaboration policy: collaboration is *strongly encouraged*. However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so you should keep your collaboration groups limited to 3 or 4 people in a given week.
4. Write each problem in a separate sheet and write down your name on top of every sheet.
5. **No bibles. This includes solutions posted to problems in previous years.**

1 Multiplicative Weights with Switching Experts

Multiplicative weights can compete with the "best expert in hindsight". If f_i^t is the loss of expert i in round t, multiplicative weights will select experts from the distribution p^t in round t such that:

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle \leq \min_i \sum_{t=1}^{T} f_i^t + O(\sqrt{T \ln n})$$

For this problem, instead of simply picking the best expert in hindsight, consider a more complex set of strategies that switch between experts, *but only up to k times*. Let S_k be the loss of the best switching strategy in hindsight:

$$S_k = \min_{1=t_1<t_2<\ldots<t_k\leq t_{k+1}=T} \left[\sum_{j=1}^{k} \min_i \sum_{t=t_j}^{t_{j+1}} f_i^t \right]$$

(a) Show how to modify the multiplicative weights algorithm to compete with S_k. Specifically, your algorithm should pick an expert from distributions $p^1, \ldots, p^T$ in each round so that:

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle \leq S_k + O(\sqrt{Tk \ln(nT)}).$$

As in class, you may assume that you know T ahead of time.

(b) How high can we set k so that our average loss still goes to 0 as $T \to \infty$?

2 Multiclass AdaBoost

AdaBoost can combine many weak *binary* classifiers into a strong *binary* classifier. It turns out the reweighting approach is useful for *multiclass* classification as well.

Specifically, consider the k class problem where we are given a sample set of data examples and labels:

$$(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_m, f(x_m))$$

where $f : X \to \{1, \ldots, k\}$ is the classification function that we're trying to learn.

Let U be the discrete uniform distribution on $\{x_1, \ldots, x_m\}$. Our ultimate goal is to output a classifier H that satisfies:

$$err(H, U) = \mathop{\mathbb{E}}_{x \sim U} \mathbb{1}[H(x) \neq f(x)] < \epsilon$$

where $\mathbb{1}[c]$ denotes an indicator random variable for the event c.

For binary boosting, we started with a weak classifier and then reweighted points to increase the "importance" of the ones which we classified incorrectly. For multiclass boosting we will do something a bit more complicated: we maintain a weight for every (point, wrong class) pair. For example, if $f(x_1) = 3$ we'll have $(k-1)$ weights for:

$$(x_1, 1), (x_1, 2), (x_1, 4), \ldots, (x_1, k).$$

This allows use to discourage specific wrong answers when boosting. Formally, we'll scale our weights to form a distribution D over all $m(k-1)$ (point, wrong answer) pairs. We use the notation $D(x_i, j)$ to denote the weight of the pair containing x_i and the j^{th} incorrect class for x_i – i.e. the j^{th} smallest value from $\{1, \ldots, k\} - \{f(x_i)\}$.

For a distribution D, we say a weak classifier h has "edge" η if:

$$\mathop{\mathbb{E}}_{x,j \sim D} [\mathbb{1}[h(x) = f(x)] - \mathbb{1}[h(x) = j]] \geq \eta.$$

In other words, if we pick a (point, wrong answer) pair from D, the point is more likely (by η) to be classified correctly by h than to be assigned that particular incorrect answer.

Consider the following boosting procedure:

- Set D_1 to be the uniform distribution – i.e. $D_1(x_i, j) = \frac{1}{m(k-1)}$ for all i, j.
- For $t = 1, 2, \ldots, T$:
  - Find a weak learner h_t with edge η_t over D_t.
  - Set $\alpha_t = \frac{1}{2} \ln \frac{1+\eta_t}{1-\eta_t}$
  - For each i, j set $D_{t+1}(x_i, j) = \frac{D_t(x_i,j) \cdot e^{-\alpha_t [\mathbb{1}[h(x_i)=f(x_i)] - \mathbb{1}[h(x_i)=j]]}}{z_t}$ where z_t is a normalization constant.

- Output classifier H with $H(x_i) = \arg\max_{\ell \in 1,\ldots,k} \sum_{t=1}^{T} \alpha_t \mathbb{1}[h_t(x_i) = \ell]$. In other words, output the label for x_i that was output with the highest (weighted) frequency by our intermediate weak classifiers.

To argue that this procedure works we'll use a proof similar to the binary setting.

(a) Show that:

$$err(H, U) = \mathop{\mathbb{E}}_{x \sim U} \mathbb{1}[H(x) \neq f(x)] \leq (k-1) \prod_{t=1}^{T} z_t$$

(b) Show that, as in the binary classification case,

$$z_t \leq e^{-O(\eta^2)}$$

There are many ways to do this, but one approach uses that, for any values $u_{i,j} \in [-1, 1]$,

$$\sum_{i=1}^{m} \sum_{j \neq f(x_i)} D(x_i, j) e^{-\alpha u_{i,j}} \leq \sum_{i=1}^{m} \sum_{j \neq f(x_i)} D(x_i, j) \left(\frac{1 + u_{i,j}}{2} e^{-\alpha} + \frac{1 - u_{i,j}}{2} e^{\alpha} \right).$$

This fact just follows from the convexity of $e^{-\alpha x}$.

(c) Conclude a final bound on $err(H, U)$. If η_t is always greater than some constant, what dependence do we expect to incur on k in the number of iterations required to reach ϵ error?

3 Approximating "Light" Max Cuts

Let's consider an undirected graph $G = (V, E)$ for simplicity. Recall that the Semidefinite Programming relaxation for max cut outputs n unit vectors $v_1, \ldots, v_n \in \mathbb{R}^n$ maximizing:

$$OPT = \sum_{(i,j) \in E} \frac{1 - \langle v_i, v_j \rangle}{2}$$

These vectors are then "rounded" by choosing a unit vector $x \in \mathbb{R}^n$ uniformly at random from the sphere and placing node i into S if $x^T v_i \geq 0$ and into $V \setminus S$ otherwise.

Goemans and Williamson showed that, in general, this rounding scheme returns a maximum cut in G with expected value within $\alpha \approx 0.87856$ of optimal. Specifically, let $\theta_{i,j}$ be the angle between vectors v_i and v_j (so $\langle v_i, v_j \rangle = \cos \theta_{i,j}$), then the expected size of the cut returned is:

$$\sum_{(i,j) \in E} \mathbb{P}[i \in S, j \notin S \text{ or } i \notin S, j \in S] = \sum_{(i,j) \in E} \frac{\theta_{i,j}}{\pi}.$$

This sum was shown to be $\geq OPT$ by showing that, for each term,

$$\frac{\theta_{i,j}}{\pi} \geq 0.87856 \frac{1-\cos(\theta_{i,j})}{2} = 0.87856 \frac{1-\langle v_i, v_j \rangle}{2},$$

which follows from the fact that, for all θ, $\frac{2}{\pi}\frac{\theta}{1-\cos(\theta)} \geq 0.87856$. Let θ_0 be the unique θ minimizing $\frac{2}{\pi}\frac{\theta}{1-\cos(\theta)}$.

(a) Show that, when $OPT/|E| > \frac{1-\cos(\theta_0)}{2}$ this term-by-term analysis cannot be tight – i.e. it cannot be that $\theta_{i,j} = \theta_0$ for all i, j. Goemans and Williamson actually take advantage of this fact to give better approximation ratios for "heavy" max cut instances, where the optimum of the SDP relaxation is large compared to the number of edges in G.

(b) On the other hand, there are known instances of graphs where $OPT/|E| < \frac{1-\cos(\theta_0)}{2}$ but the Goemans-Williamson rounding scheme will not achieve an approximation of better than 0.87856 in expectation. Explain why this is possible. Why does your argument from part (a) not work for the less than case?

Note: We're not asking you to actually come up with a bad case graph. Simply explain what values for $\theta_{i,j}$ would lead to the Goemans-Williamson approximation being tight.

Nevertheless, there are several algorithms that *do* achieve better approximations when $OPT/|E|$ is small. These approaches are inspired by noting that a *random cut* (i.e. assign each vertex independently at random to one side of the cut) can actually give a decent approximation to "light" maximum cut instances, since a random cut will have expected value $|E|/2$. We will look at one such approach.

(c) Construct a set of unit vectors $u_1, \ldots, u_n \in \mathbb{R}^n$ that, when rounded using the random scheme, outputs a uniformly random cut. In other words, $u_1, \ldots, u_n$ should be chosen such that the procedure assigns each node i to S with probability exactly $1/2$ and the choices are independent amongst nodes.

Consider appending a weighted copy of u_i to each v_i obtained from solving the SDP relaxation. Specifically, for some $\lambda \in [0, 1]$ construct:

$$v_i' = [\sqrt{\lambda} v_i, \sqrt{1-\lambda} u_i].$$

(d) Let $\theta'_{i,j}$ be the angle between v_i' and v_j'. For what values of $\theta_{i,j}$ is $\theta'_{i,j} > \theta_{i,j}$? For what values is it less than $\theta_{i,j} < \theta'_{i,j}$? Use your answers to justify how rounding with $\{v_1', \ldots, v_n'\}$ instead of $\{v_1, \ldots, v_n\}$ could improve your bad case from part (b).

We won't prove it formally, but it can be shown that when $OPT/|E| < \frac{1-\cos(\theta_0)}{2}$, it's always possible to obtain an improvement over Goemans-Williamson using this scheme.

4 Log-Determinant Barrier

Like linear programs, to solve semidefinite programs using interior point methods, we need a self-concordant barrier function. One choice is the log determinant function:

$$-\log \det(\mathbf{X}) = -\log(\text{determinant}(\mathbf{X})).$$

Using the definitions from Problem Set 7, Question 1, show that $-\log \det$ is an n-self-concordant barrier for the positive definite cone, which contains any symmetric $n \times n$ matrix $\mathbf{A}$ with $\mathbf{A} \succ 0$. Here, we are viewing the positive definite cone as a subset of the space of *symmetric* matrices, rather than arbitrary matrices.

You may use the fact that, for *any pair* of positive definite $\mathbf{A}$ and symmetric $\mathbf{B}$, there is some full rank $\mathbf{C}$ and diagonal matrices $\mathbf{D_A}$ and $\mathbf{D_B}$ such that:

$$\mathbf{A} = \mathbf{C}^T\mathbf{D_A}\mathbf{C} \text{ and } \mathbf{B} = \mathbf{C}^T\mathbf{D_B}\mathbf{C}.$$

Hint: Recall that proving self-concordance requires restricting a function to a line. Since $-\log \det$ is defined over the positive definite cone, its restriction to a line in that cone takes the form $f_{\mathbf{A},\mathbf{B}}(t) = -\log \det(\mathbf{A} + t\mathbf{B})$ where $\mathbf{A}$ is positive definite and $\mathbf{B}$ is symmetric.