CSE 522: Algorithms and Uncertainty Spring 2017

Lecture 7 — April 17, 2017

Lecturer: Nikhil R. Devanur

In this class, we will continue to discuss applications of the experts problem.

1 Boosting

Recall the PAC learning setting: there is a distribution $\mathcal{D}$ over examples of feature and label pairs (x, y). The goal is to accurately predict the label y given the feature x. We define a "weak learner" and a "strong learner" as algorithms whose errors are slightly better than a random guess, and almost perfect, respectively.

Definition 1. *A* weak *(resp.* strong*) learner is an algorithm that given any distribution $\mathcal{D}$ (explicit, or with sample access), outputs a hypothesis h_{ALG} such that for some $\gamma > 0$ (resp. $\epsilon > 0$),*

$$err_{\mathcal{D}}(h_{\text{ALG}}) \leq \frac{1}{2} - \gamma \quad (\textbf{\textit{resp.}} \quad \leq \epsilon).$$

In particular, suppose that we have a given training set of examples,

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\},$$

and the distribution given to the strong learner is the empirical distribution over these examples.

For the weak learner we will assume that h_{ALG} is in some given set of hypothesis $\mathcal{H}$, and for the strong learner,

$$h_{\text{ALG}} = \text{WT-MAJORITY}(\mathcal{H}, \cdot, \alpha(\cdot)),$$

for some set of weights $\alpha(\cdot)$.

1.1 Adaboost

In the last lecture we saw how to do boosting using the experts algorithm. This has a couple of disadvantages.

- You need to know the value of γ ahead of time.
- The error of the strong learner depends on γ, which is the worst case error of the weak learner.

An alternate boosting algorithm, called Adaboost, addresses both these drawbacks. You don't need to know γ (although you still need to make the same assumption about the worst case error), and the error of the strong learner depends on the actual errors observed during the run of the algorithm. Improving any one of these errors will improve the error of the strong learner. The algorithm is similar to the one we saw for the experts problem. In each round $t \in [T]$,

1. it maintains a weight $w_t(i)$ for each example in the training set (initialized to 1 for $t = 1$),
2. runs the weak learner on the probability distribution $\mathcal{D}_t = w_t(\cdot)/W_t$, to find a hypothesis h_t,
3. with $\text{err}_{\mathcal{D}_t}(h_t) =: \epsilon_t < \frac{1}{2}$,
4. and updates the weights as $w_{t+1}(i) := w_t(i)\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{\mathbf{1}(h(x_i)\neq y_i)}$.

The main difference from applying the algorithm for the experts problem is the update. The update depends on the error ϵ_t of the hypothesis h_t found by the weak learner in round t. If the error is $\frac{1}{2}$, then the update doesn't change the weights and we will be stuck. This is why we need the error to be bounded away from $\frac{1}{2}$. As the error decreases, the term $\frac{1-\epsilon_t}{\epsilon_t}$ increases, so the update is more aggressive. At the other extreme, if the error is 0, then we just use the hypothesis found in that round since it correctly labels all the examples. Notice that $\frac{1-\epsilon_t}{\epsilon_t}$ is greater than 1, therefore the weights of the examples that are wrongly labeled increases. This ensures that these examples are more important, and are more likely to be correctly labeled by the weak learner in the next round. The eventual strong learner is once again a weighted majority, but this time the weights of the h_ts also depend on the ϵ_ts. Let $\alpha(h_t) := \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$. The strong learner then predicts

$$\text{Adaboost-predict}(x) := \text{WT-MAJORITY}(\{h_1, h_2 \ldots, h_T\}, x, \alpha(\cdot)).$$

Theorem 1. *Let $\gamma_t = \frac{1}{2} - \epsilon_t$. Let $\mathcal{D}_{emp}$ be the empirical distribution over the training set. Then*

$$err_{\mathcal{D}_{emp}}(\textit{Adaboost-predict}) \leq \exp(-2\sum_{t\in[T]} \gamma_t^2).$$

While the training error reduces with more rounds, the generalization error increases with T. Hence, (in theory as well as in practice) we should pick a T that optimizes the sum of the two errors.

2 Yao's minmax principle

An important application of zero sum games is to the analysis of algorithms. In particular, suppose we are interested in understanding the fastest run time required to solve a particular problem. Suppose we fix the size of the input to say n bits, and we want to understand which algorithm achieves the fastest worst-case running time for this problem. Similarly, we could consider any performance measure, such as the approximation factor, or the regret. Hence the experts problem itself can be thought of as a zero sum game!

The statement naturally has a min-max flavor to it, so it's not surprising that this is a zero sum game. We need to define a finite set of algorithms. Once we have a trivial upper bound on the running time, one can consider all Turing machines of a certain size, since the size is bounded by some function of the running time. Let A_D denote the set of all such Turing machines, i.e., the set of all deterministic algorithms, and let A_R denote the set of all distributions over A_D, i.e., the set of all randomized algorithms. Let I denote all n bit strings. Then the quantity we are interested in is

$$\min_{a\in A_R} \max_{i\in I} \text{Perf}(a, i),$$

where $\text{Perf}(a, i)$ is the (expected) performance measure we are interested in, such as the running time or the regret, of the algorithm a on the input i. This is a zerosum game with the row player strategy set being A_D and the column player strategy set being I, and the payoff is $T(a, i)$. From the min-max theorem, we have that

$$\min_{a \in A_R} \max_{i \in I} T(a, i) = \max_{\mathcal{D} \in \Delta^I} \min_{a \in A_D} \mathbb{E}_{i \sim \mathcal{D}}[T(a, i)].$$

The RHS in the above has the following interpretation: it is the *average case* running time when the inputs are drawn from a distribution $\mathcal{D}$, and a deterministic algorithm is tailored to this distribution.

The main application of this is to prove lower bounds. It is often easier to come up with a particular distribution $\mathcal{D}$, for which one can argue that every deterministic algorithm a must have an average performance measure higher than something. This implies that the same lower bound holds for any randomized algorithm in the worst case.

The less common application is to show existential upper bounds/algorithms. By designing an algorithm that works well in the average case, for any given distribution over the inputs, one can conclude that there is some algorithm that has the same performance bound in the worst-case. However, this doesn't actually give us one such algorithm.

3 Experts for combinatorial problems

Suppose every day, we travel from work to home, and have the option of using one of many routes. Since we don't like to sit in traffic, we wish to minimize the time we spend in commuting. Suppose we don't have any idea what the traffic on any road is going to be. How should we then choose our daily route? This is formalized in the following problem. There is a given graph $G = (V, E)$, and two special nodes s and $t \in V$. Let $\mathcal{P}$ be the set of all simple paths in G from s to t. In every round $t \in [T]$, we do

- Pick a path $P_t \in \mathcal{P}$.
- Observe the delays on all the edges, $\ell_t(e)$, forall $e \in E$. We observe the delays on edges we don't take as well.

We wish to minimize the total delay over all the rounds, and compare it to the delay experienced by the best fixed path on hindsight.

$$\text{REGRET} = \sum_{t \in [T]} \sum_{e \in P_t} \ell_t(e) - \min_{P \in \mathcal{P}} \sum_{e \in P} \ell_t(e).$$

This is an instance on the experts problem, where the set of experts is $\mathcal{P}$ and $\ell_t(P) = \sum_{e \in P} \ell_t(e)$. The problem with this reduction is that the number of paths maybe exponential in the size of the graph, therefore running the experts problem will take exponential time. The same issue arises with any combinatorial problem such as the minimum spanning tree, or a min-cost matching problem. One would ideally like an algorithm with regret comparable to that of the experts problem, while running in time polynomial in the graph size.

The solution to all these different problems is to generalize the experts problem to allow arbitrary convex sets of experts, just like we generalized the zero sum game strategies to allow arbitrary

convex sets of strategies. We define this generalization now, and in the rest of the lecture, see an algorithm for it. The key property of the problems we consider is that

- each solution (a path, a spanning tree, or a matching) can be thought of as a vector in a lower dimensional space. E.g., a path P is represented by an indicator vector $x_P \in \mathbb{R}^E$, where $x_P(e) = 1$ if $e \in P$, and is 0 otherwise. The set of randomized strategies is simply the convex hull of all the corresponding vectors.
- With this representation, the loss is a linear function, i.e., $\ell_t(P) = \ell_t \cdot x_P$, where $\ell_t \in \mathbb{R}^E$ is the vector of losses for each edge.

We therefore consider the following generalization of experts. The set of experts is an arbitrary convex set $K \subseteq \mathbb{R}^n$, which is given. For $t = 1..T$,

- Pick $x_t \in K$.
- See $\ell_t \in \mathbb{R}^n$.

The loss of the algorithm in round t is $\ell_t(\text{ALG}) = \ell_t \cdot x_t$. The goal is to minimize the regret, defined as

$$\text{REGRET} = \ell_{1..T}(\text{ALG}) - \min_{x \in K} \ell_{1..T} \cdot x.$$

We will assume the following boundedness property

$$\forall x \in K, \forall t \in [T], \ell_t \cdot x \in [0, 1].$$

Also, as in the zero sum games, we will assume that the following optimization problem is easy to solve for K (and define a new notation, $M(\cdot)$,):

$$\forall \ell \in \mathbb{R}^n, \text{solve } M(\ell) := \arg\min_{x \in K} \ell \cdot x.$$

When the minimizer is not unique, we break ties arbitrarily, for the definition of $M(\cdot)$.

The experts problem is indeed a special case of this more general problem, by letting $K = \Delta^n$. Each corner of the simplex represents an expert. $M(\ell)$ is simply the expert corresponding to the smallest co-ordinate.

3.1 Be the leader

We will first analyze a hypothetical algorithm, which we cannot actually implement. The reason is that the algorithm uses the knowledge of ℓ_t to define x_t, which is actually not allowed; this is just for the sake of analysis.

Algorithm Be the leader (BTL) :

$$\text{Pick } y_t = M(\ell_{1..t}).$$

The loss of this algorithm is denoted by $\ell_t(\text{BTL}) = \ell_t \cdot y_t$. The main insight we need from this algorithm is that this has no regret.

Lemma 1.

$$\ell_{1..T}(\text{BTL}) \leq \min_{x \in K} \ell_{1..T} \cdot x = \ell_{1..T} \cdot y_T.$$

Proof. The proof is by induction on T. Note that for $T = 1$, the statement is trivial. Suppose that the statement of the theorem is true for $T - 1$ rounds, i.e.,

$$\ell_{1..T-1}(\text{BTL}) \leq \ell_{1..T-1} \cdot y_{T-1}.$$

We will now argue that it holds for T rounds as well. Since y_{T-1} is the minimizer of $\ell_{1..T-1}$, the above inequality implies that

$$\ell_{1..T-1}(\text{BTL}) \leq \ell_{1..T-1} \cdot y_T.$$

The inductive step now follows by adding $\ell_T \cdot y_T$ to both sides of this inequality. □

3.2 Follow the leader

A more reasonable algorithm is the following algorithm, which does the same as BTL, except for the unreasonable part of knowing ℓ_t.

Algorithm Follow the leader (FTL) :

$$\text{Pick } x_t = M(\ell_{1..t-1}).$$

The loss of this algorithm is denoted by $\ell_t(\text{FTL}) = \ell_t \cdot x_t$. Notice that this algorithm is always one step behind BTL, i.e.,

$$x_t = y_{t+1}.$$

This implies that the difference in losses between the two algorithms is bounded by the number of times the leader changes.

Lemma 2.

$$\ell_{1..T}(\text{FTL}) - \ell_{1..T}(\text{BTL}) \leq \sum_{t \in [T]} \mathbf{1}(y_t \neq y_{t+1}).$$

This suggests that this algorithm would be good if the leader didn't change too many times. But this is not in our control. In any case, we already saw that no deterministic algorithm can get a sublinear regret even for the special case of the experts problem. Since this algorithm when specialized to the experts case is a deterministic algorithm, there is no hope of this working.

4 Perturbation/Regularizer

The next idea we introduce is to add a dummy loss vector at time 0. We can then define the analog of BTL and FTL where we include this loss at time 0 in our calculations. Let ℓ_0 be some given vector in $\mathbb{R}^n$.

Algorithm Be the perturbed leader (BTPL) :

$$\text{Pick } y_t = M(\ell_{0..t}).$$

Algorithm Follow the perturbed leader (FTPL) :

$$\text{Pick } x_t = M(\ell_{0..t-1}).$$

Applying Lemma 1 to include ℓ_0, we get the following corollary. The proof of this lemma is an easy exercise.

Lemma 3.

$$\ell_{1..T}(\text{BTPL}) \leq \min_{x \in K} \ell_{1..T} \cdot x + \max_{x,y \in K} \ell_0 \cdot (x - y).$$

In the rest of this section, we will show how to pick ℓ_0 in order to get a multiplicative version of Lemma 2. Pick ℓ_0 randomly as follows. For each coordinate i, independently pick a random variable $z(i)$ from the exponential distribution with parameter ϵ, i.e.,

$$\mathbb{P}[z(i) > a] = e^{-\epsilon a}.$$

Then set $\ell_0(i)$ to be z_i or $-z_i$ with equal probability. It is easy to see that the density μ of this distribution is as follows (by simply multiplying the densities of the individual coordinates, since they are independent):

$$d\mu(\ell) = \left(\frac{\epsilon}{2}\right)^n e^{-\epsilon \|\ell\|_1}.$$

Lemma 4.

$$\ell_{1..T}(\text{FTPL}) \leq (1 + O(\epsilon A))\, \ell_{1..T}(\text{BTPL}),$$

where $A := \max_{t \in [T]} \|\ell_t\|_1$, *and* $\epsilon A \leq \frac{1}{2}$.

Proof. Fix a round t. The proof follows by showing that the distribution of ℓ_0 and $\ell_0 + \ell_t$ are similar to each other. Let μ' be the distribution of the random variable $\ell_0 + \ell_t$. Then,

$$d\mu'(\ell) = d\mu(\ell - \ell_t) = \left(\frac{\epsilon}{2}\right)^n e^{-\epsilon \|\ell - \ell_t\|_1}.$$

We now claim that

$$d\mu(\ell) \leq d\mu'(\ell) e^{\epsilon \|\ell_t\|_1} \leq d\mu'(\ell)\,(1 + O(\epsilon A)). \tag{1}$$

The second inequality follows from the fact that $e^x \leq 1 + O(x)$ for $x \leq \frac{1}{2}$. For the first inequality, subsituting the values of $d\mu$ and $d\mu'$, it is equivalent to

$$\left(\frac{\epsilon}{2}\right)^n e^{-\epsilon \|\ell\|_1} \leq \left(\frac{\epsilon}{2}\right)^n e^{-\epsilon \|\ell - \ell_t\|_1} e^{\epsilon \|\ell_t\|_1}.$$

Canceling the common terms and taking logs, it reduces to

$$-\|\ell\|_1 \leq -\|\ell - \ell_t\|_1 + \|\ell_t\|_1,$$

which is just the triangle inequality for the $\|\cdot\|_1$ norm, rearranged. Now note that

$$\ell_t(\text{FTPL}) = \mathbb{E}_{\ell_0 \sim \mu}[\ell_t \cdot M(\ell_0 + \ell_{1..t-1})], \text{ and}$$

$$\ell_t(\text{BTPL}) = \mathbb{E}_{\ell_0 \sim \mu}[\ell_t \cdot M(\ell_0 + \ell_{1..t})] = \mathbb{E}_{\ell \sim \mu'}[\ell_t \cdot M(\ell + \ell_{1..t-1})].$$

Notice that the only difference between $\ell_t(\text{FTPL})$ and $\ell_t(\text{BTPL})$ is the distribution used, either μ or μ'. Now from (1) we get that

$$\ell_t(\text{FTPL}) \leq (1 + O(\epsilon A))\, \ell_t(\text{BTPL}).$$

The lemma follows by summing this over all t. □

Now it remains to bound the term $\max_{x,y \in K} \ell_0 \cdot (x - y)$. We do that in the following lemma.

Lemma 5. *Let $D := \max_{x,y \in K} \|x - y\|_1$. Then,*

$$\mathbb{E}_{\ell_0 \sim \mu}[\|\ell_0\|_\infty] \leq \frac{2 \log n}{\epsilon}.$$

$$\max_{x,y \in K} \mathbb{E}_{\ell_0 \sim \mu}[\ell_0 \cdot (x - y)] \leq \frac{2D \log n}{\epsilon}.$$

Proof. The second inequality follows from the first by Holder's inequality. The first one is a routine calculation. (See Exercise 1.) □

We can now put it all together to get the regret bound.

Theorem 2.

$$\ell_{1..T}(\text{FTPL}) \leq (1 + O(\epsilon A)) \min_{x \in K} \ell_{1..T} \cdot x + \frac{O(D \log n)}{\epsilon},$$

where $A := \max_{t \in [T]} \|\ell_t\|_1$, and $D := \max_{x,y \in K} \|x - y\|_1$, and $0 < \epsilon \leq \frac{1}{2A}$. By setting $\epsilon = \sqrt{\frac{D \log n}{AT}}$ we get the regret bound

$$\text{REGRET} \leq O\left(\sqrt{ADT \log n}\right).$$

Proof. From Lemma 4, we have that

$$\ell_{1..T}(\text{FTPL}) \leq (1 + O(\epsilon A))\, \ell_{1..T}(\text{BTPL}).$$

From Lemmas 3 and 5, we have

$$\ell_{1..T}(\text{BTPL}) \leq \min_{x \in K} \ell_{1..T} \cdot x + \max_{x,y \in K} \mathbb{E}[\ell_0 \cdot (x - y)] \leq \min_{x \in K} \ell_{1..T} \cdot x + \frac{2D \log n}{\epsilon}.$$

Combining the two inequalities above, we get

$$\ell_{1..T}(\text{FTPL}) \leq (1 + O(\epsilon A)) \min_{x \in K} \ell_{1..T} \cdot x + \frac{O(D \log n)}{\epsilon}.$$

This completes the first part of the Theorem. For the second part, we simply use the fact that $\ell_{1..T} \cdot x \leq T$ for all $x \in K$, and set ϵ as in the statement of the theorem. □

Exercise 1. *Let $n \geq 3$.*

$$\mathbb{E}_{\ell_0 \sim \mu}[\|\ell_0\|_\infty] \leq \frac{2 \log n}{\epsilon}.$$

Hint: Recall that the absolute values of the coordinates of ℓ_0 are independent exponential random variables with parameter ϵ. An exponential random variable y with parameter ϵ has the property that for any $a > 0$,

$$\mathbb{E}[y - a | y > a] = \frac{1}{\epsilon}.$$

Let $a = \log n/\epsilon$. Let $y_i = |\ell_0(i)|$. What's the probability that $y_i > a$? What is $\mathbb{E}[\|\ell_0\|_\infty - a | \|\ell_0\|_\infty > a]$? Also recall that $\max \leq \sum$.

4.1 Recovering the experts regret bound

The experts problem is the special case when $K = \Delta^n$. It is easy to check that $D = 2$ for this case. Also, if the loss vectors $\ell_t \in [0, 1]$, you can also check that $\ell_t \cdot x \in [0, 1]$ for all $x \in K$. However, A could be as large as n. This gives us an extra n factor in the regret bound. How do we get rid of this?

Exercise 2. *Show how to get a regret bound of $O(\sqrt{T \log n})$ for the experts problem using Theorem 2.*

Hint: Split the loss in one round to n rounds where each expert only incurs loss in one of these rounds. This should reduce A to 1. How does this affect the loss of the algorithm? What about the benchmark? Do you want to use the first or the second bound in the Theorem 2?