

Lecture 2 : 9.03.08

Lecturer:

Scribe:

Disclaimer: *These are rough notes, with some exercises.*

2.1 Last Time.

Given n experts, with mistakes (payoffs) in the interval $[0, 1]$, the multiplicative yields an expected loss of at most

$$m_i(1 + \epsilon) + \log n/\epsilon,$$

where m_i is the loss of the i th expert. Recall, proof follows from

$$(1 - \epsilon)^{m_i} \geq \Phi_n \geq \Phi_0(1 - \epsilon)^m,$$

since the potential Φ_t decreases by $(1 - \epsilon)^{l_t}$ where l_t is the expected loss in iteration i , and the potential is at least the potential of any individual expert.

Can be recast as gains, to yield at least

$$m_i(1 - \epsilon) - \log n/\epsilon.$$

Question: Independent of number of iterations?

Yes. But for a good additive bound. Balance ϵT and $\log n/\epsilon$, and one gets a loss of $\sqrt{T \log n}$. Here, T is the number of iterations or the max loss.

Question: Shall we philosophize?

If random world, experts with same average loss and random losses, can likely do no worse than average less $\sqrt{\log n}$ deviations (square root of the average) from the average, by simply choosing any person. Multiplicative updates gets to the random world. There is a recent paper that shows something like this in a very elegant and nice way [?].

The multiplicative updates algorithm is not just to deal with an “adversary” but to deal with a computation that you may design (with no extra properties.) Since it can deal with any adversary the requirements for the computation can be minimal. We’ll see two examples.

2.2 Learning.

Suppose that for any given a set of examples, there is an algorithm to classify the examples correctly with probability $1/2 + \gamma$. Indeed, suppose that for any distribution on the examples the algorithm can predict correctly with probability $1/2 + \gamma$. (This has been called a weak learner against any distribution.)

Can you learn with probability $1 - \epsilon$? on the examples?

Question: How do we set this up as an experts problem?

Start with N examples. And the uniform distribution. Make a hypothesis that learns the function with probability $1/2 + \gamma$ on this distribution.

Question: Then what?

Then, reweight the current distribution according to whether the hypothesis classified it correctly.

That is, if hypothesis was right, downweight the example.

Question: In each iteration, what is the expected penalty?

At least $1/2 + \gamma$ since the hypothesis is right on this fraction of the distribution.

Question: Final hypothesis?

Weighted average of all hypothesis.

Question: What is analysis?

S is incorrectly labelled examples. Penalty is at most $T/2$ since majority gets it wrong.

In a step, the potential changes as follows..

$$\Phi_{t+1} = \Phi_t^{incorrect} + (1 - \epsilon)\Phi_t^{correct}.$$

Since the correct fraction is at least $(1/2 + \gamma)$, this results in the following bound.

$$\Phi_{t+1} \leq (1 - \epsilon(1/2 + \gamma))\Phi_t \leq (1 - \epsilon)^{1/2+\gamma}$$

The potential function at time T , Φ^T is at most

$$\phi^1(1 - \epsilon)^{T(1/2+\gamma)} \leq ne^{-\epsilon T(1/2+\gamma)}.$$

The last following from $(1 - \alpha) \leq e^{-\alpha}$.

The potential function is at least

$$|S|(1 - \epsilon)^{T/2} \geq |S|e^{-(\epsilon+\epsilon^2)T/2}.$$

The last following from $(1 - \alpha) \geq e^{-(\alpha+\alpha^2/2)}$.

Thus,

$$|S|/n \leq e^{-\epsilon(\gamma-\alpha/2)T}.$$

Choosing $\epsilon = \gamma$ and $T = \frac{2^2}{\gamma} \ln \frac{1}{\delta}$, we get that

$$|S|/n \leq \delta.$$

Question: Is there any intuition here?

Majority vote of γ biased coins. After approximately $(1/\gamma)^2$ coin tosses, you get a coin with constant bias. To get the error, rate down to δ , one needs $\log(1/\delta)$ rounds with a constant biased coin.

Question: Anything for complexity folks?

See the AHK notes on hard-core sets. (The idea being if any circuit of size S is for a function is wrong on ϵ fraction of its inputs, than any circuit of some smaller size is wrong on almost half the inputs. This also implies there is some subset of inputs for which even large circuits are often wrong.)

2.3 An aside

Eventually we will discuss cuts. One approach to finding small cuts in a graph relies on the notion that one should embed a complete graph into an underlying graph. One can hope (as in the max-flow, min-cut theorem) that the congested edges lie in a good cut. For example, routing traffic in the Manhattan typically congests the bridges.

A small cut clearly lower bounds the congestion; we define here the “size” as the *sparsity* to be

$$\frac{c(S, \bar{S})}{|S||\bar{S}|},$$

where $c(S, \bar{S})$ is the capacity of the flow between S and \bar{S} . This is basically the same as edge expansion

$$\frac{c(S, \bar{S})}{\min(|S|, |\bar{S}|)},$$

which cuts off the most per unit edge cost.

Now, for a cut, the congestion is at least $|S||\bar{S}|/c(S, \bar{S})$.

Indeed, from this intuition, and using the dual of this linear program, one can indeed show that the congestion is at most an $O(\log n)$ factor larger than this lower bound. Indeed, one can find a cut whose cost is at most $O(\log n)/C$ where C is the optimal value of the dual linear program (which is at least the inverse of the sparsity of the best cut.)

Thus, we wish to solve the min congestion problem (and its dual) on embedding the complete graph into a graph. (In fact, one needs only embed a much smaller (say $\Theta(n)$) number of randomly chosen pairs to get essentially the same result. Perhaps, more later.)

2.4 Path Selection

The problem is given a set of pairs $(s_1, t_1), \dots, (s_k, t_k)$ we wish to route paths in a graph to minimize congestion.

Question: Experts?

We could set this up as a two person game. Each column is a flow that satisfies all the constraints; one unit of flow is routed between s_i and t_i . Each row is an edge. The payoff to the row for the usage of an edge is the amount of flow sent on the edge.

Question: Let's look at the value of the column player?

The column player is minimizing the payoff to the row player of its best response.

$\min C$

$$\forall e \sum_f x_f * f(e) \leq C$$

Here x_f are the variables and $f(e)$ are the inputs. The congestion of the final flow!!!

Question: What is the row player trying to do?

Maximizes the payoff of the column player's best response.

$\max D$

$$\forall \text{ flows } f, \sum_e w(e)f(e) \geq D.$$

Here, $w(e)$ is the variables and $f(e)$ are inputs.

Examining this bound. We note that for a flow f , $\sum_e w(e)f(e) = \sum_{\text{flow path } p} w(p)$. Uncountably many constraints! Note, for example, the minimizers will always be a flow that routes along shortest paths along $w(e)$ for each commodity.

Question: Experts.

In an algorithm setting, rather than the on-line setting, we do all the work. That is, we give a method for the "event" as well as the expert.

Question: How?

Play the optimal column strategy! Route everyone along shortest paths with respect to the current $w(e)$ assigned by the row player.

Question: Which one is optimal?

Minimize payoff to row which is weighted sum of edge congestions: $\sum_e w(e)c(e)$. Noted above as routing along the shortest paths.

Question: Now what?

At each step, the column strategy pays at most C_{opt} at any step.

Moreover, the row player receives at most C_{opt} on average and receives at least $C * (1 - \epsilon) - V \log n / T \epsilon$ on average, where C^* is the best that any row strategy can receive.

Thus, no row player gets more than $C^* \leq (C_{opt} + V \log n / \epsilon) / (1 - \epsilon)$ against this column distribution. This upper bounds the congestion!

Here, V is the max payoff possible in any one round. This could be k (all the paths could use the same edge.)

Thus, this takes time $\tilde{O}(k(km))$ to converge to something that is a reasonably good approximation. That is $k \log n / T \epsilon < \epsilon C$. Or $T > k / C (\log n) / \epsilon^2$.

Question: Is this just analysis? Could the bad case happen?

Sure. An example is a network consisting of k parallel arcs, and identical source sink pairs for the two nodes.

Question: How do we fix this?

Perhaps update weights, as you add paths inside the column player.

Question: How about using the multiplicative weights framework?

Use a different game. Payoff for the row is the congestion of the column player. The column player chooses a random path to commodity and routes it along its shortest path. (The game does not directly model a linear program, but can perhaps be used similiarly.)

Exercise 1: Show how to convert this explanation into an algorithm that runs in time $\tilde{O}(km)$ (where the \tilde{O} notation ignores logarithmic and ϵ factors.) Note, the maximum payoff to the row player in this game is 1. Moreover, each round can be done in $\tilde{O}(m)$ time. Unfortunately, the flow in a round is not feasible in that it does not route one unit of flow for every pair. One observation (to prove) is that after $\Omega(k \log k)$ rounds each path is routed approximately the same number of times. Another has to do with the expected cost of the column player.