

## CMSC 39600: Online Algorithms

Lecture 7

Course Instructor: Adam Kalai

Date: October 13, 2004

### Hannan-type algorithms

Today we're going to consider a class of online optimization problems solvable by expert type of algorithms.

#### 0.1 Example: online shortest paths

This is also known as the “online driving to work problem.” One has a fixed directed graph with a fixed pair of nodes  $(s, t)$ . Each day, one has to pick a path from  $s$  to  $t$ , and then the times on all the edges are revealed. The cost that day is the sum of the times on the edges of the chosen path. This repeats. At the end of the year, we look back and figure out which path would have been the best in hindsight. (To do this computation, we simply sum the times over the year for each edge and then compute the shortest path in the totals graph.)

Let's say we normalize things so that all times are in  $[0, 1]$  on each edge. We can consider this as an “expert” problem where each path is an expert recommending to take it as the path.

Let us summarize the results of the previous lecture. For online maximization problems, with  $n$  experts and maximum costs of  $R$ , we have an algorithm that guarantees,

$$E[\text{cost of WM}] \leq (1 + \epsilon)\text{cost of best expert} + R \frac{\log n}{\epsilon}.$$

Here we have simply scaled everything by  $R$  which was assumed to be 1. For online minimization problems, we have,

$$E[\text{reward of WM}] \leq (1 - \epsilon)\text{reward of best expert} - R \frac{\log n}{\epsilon}.$$

In a graph with  $n$  nodes and  $m$  edges, there are certainly at most  $n^m$  paths that we might choose (we're never going to visit a node more than once). Thus if we think of each path as an expert that incurs cost equal to the time on that path, we have  $R \leq n$ , and we get,

$$E[\text{cost of WM paths}] \leq (1 + \epsilon)\text{cost of best path} + n \frac{\log m^n}{\epsilon} = (1 + \epsilon)\text{cost of best path} + \frac{n^2 \log m}{\epsilon}.$$

Here the cost of the best path is the total cost of the best single path, if we had to use the same path every day, chosen with the benefit of hindsight.

Note: one reason for the  $(1 + \epsilon)\text{blah} + \frac{\text{blah}}{\epsilon}$  notation, rather than solving for the best  $\epsilon$ , is that it maintains the form of a competitive ratio with an additive term. In this case, any constant competitive ratio larger than 1 can be achieved.

Interestingly, we have completely ignored the structure of this problem, its paths, etc., and have been able to get competitive guarantees. The problem is that the algorithm is *inefficient*, a weight has to be maintained for each path.

Also note that this problem, like the experts problem itself, can be viewed as a convex optimization problem. Each path can be viewed as a point  $x$  in  $\{0, 1\}^m$ , where  $m$  is the number of edges and a 1 in the  $i$ th coordinate  $x_i = 1$  indicates that the  $i$ th edge is included in the path. The set of legal paths is a subset  $L \subset \{0, 1\}^m$ . We can consider the convex hull of a  $L$ ,  $S \subset [0, 1]^n$ . Each point in  $S$  is a *flow*.

Each period there is a linear (and hence convex) cost function which is  $f^t(x) = \sum w_i^t x_i$ , where  $w_i^t \in [0, 1]$  is the time on edge  $i$  on period  $t$ . Using Zinkevich's algorithm, we can choose a flow each period. From a flow, it is easy to choose a random path so that the expected cost of the path equals the cost of the flow, and Zinkevich's algorithm applies.

## 0.2 Example: binary search trees

Again, the (static) binary search tree problem can be thought of as an experts problem. Each tree can be thought of as an expert, whose cost on a particular period is the depth of the requested item. Thus the maximum cost is  $n$ , the number of nodes in the tree. The number of trees is  $2^{O(n)}$  (fewer than the number of lists!). Thus we get immediately,

$$\begin{aligned} E[\text{cost of WM trees}] &\leq (1 + \epsilon)\text{cost of best static tree} + n \frac{\log 2^{O(n)}}{\epsilon} \\ &= (1 + \epsilon)\text{cost of best static tree} + \frac{O(n^2)}{\epsilon}. \end{aligned}$$

Wow, that was a lot easier than analyzing Splay Trees! The problem is, it's a bit tedious to maintain a weight for every possible tree and then choose a random one with probability proportional to its weight.

We might try to use Zinkevich's algorithm for this problem as well. Each tree can be thought of as a point  $x \in \mathbb{R}^n$ , where  $x_i$  is the depth of  $i$  in the tree. Again, we could take the legal set of tree vectors  $L \subset \{1, 2, \dots, n\}^n$ , and take its convex hull  $S \subset \mathbb{R}^n$ . The cost function on a period with a lookup to element  $i$  would again be a linear function  $f^t(x) = x^t \cdot r^t$  where  $r^t$  is a vector of all 0's except in the position of the most recent request.

To try to be efficient, we could then use Zinkevich's algorithm on this sequence. The problem is: what the heck is  $S$ ? In the previous problem, it was a set of flows. Now, it is the convex hulls of a bunch of trees. How do we actually pick our tree?

# 1 Hannan's algorithm

Let's first describe Hannan's algorithm for the experts problem, say with rewards, so our goal is to achieve a reward not much smaller than the best expert. Arguably the most natural approach to this problem (and any static online problem) is the Frequency Count (FC) algorithm, i.e. each period choose the best single expert so far. For the experts problem, this amounts to keeping counts for each expert of its total reward, and then choosing the best expert. For simplicity, let's say that each expert receives a reward of 0 or 1 each period, and that the frequency count algorithm always chooses the expert of maximal reward, in the case of ties by choosing the expert of least index.)

Again, the frequency count algorithm could be quite bad. In fact, any deterministic algorithm can be really bad. Each period, every expert except the one it chooses may receive a reward. With  $n$  experts, the average expert reward each period is  $(n - 1)/n$  while FC receives 0. 0 is not a very good competitive ratio.

Hannan's general solution is to add perturbations. In other words, to imagine a pretend period 0 where the rewards are chosen randomly according to some distribution and then use FC.

Let's recall from our previous analysis of FC, that,

$$E[\text{reward of FC}] \geq (\text{reward of best expert}) - (\# \text{ times FC changed}). \quad (1)$$

To see this, note that each time FC doesn't change, that means that the best expert so far hasn't changed and the cost we incur is the same as the cost incurred by the best expert. Each time FC changes, i.e. the best expert changes, can increase by at most 1.

Let us choose as a perturbation distribution, a pretend period 0 with a discrete geometric distribution with mean  $1/\epsilon$ . In other words, for each expert, we flip a coin with  $P(H) = \epsilon$  until we get a  $H$ , and count the number of  $T$ 's before we see a  $H$ . In other words, we choose a perturbation equal to  $i$  with probability  $(1 - \epsilon)^i \epsilon$ , and we do this independently for each expert  $i$ , once at the beginning. This is its imaginary period 0 reward. Hannan used a different distribution, and a more complicated analysis, which is probably why his algorithm was overlooked for 50 years.

Define the imaginary total reward of an expert to be its period 0 reward plus the rewards during the real periods. Call the algorithm which does these perturbations, Perturbed Frequency Count (PFC). Now, we argue that the probability that PFC changes is low. In order for PFC to change, it must have been on one of those experts that received 0 reward that period, and must have switched to a different expert that received reward 1.

We want to bound the probability of a switch with perturbations. We say that it is much more likely that we are at a given expert and stay than that we switch to that expert.

$$P[\text{switch to expert } i \text{ at period } t + 1] \leq \frac{\epsilon}{1 - \epsilon} P[\text{are already at expert } i \text{ on period } t] \quad (2)$$

Essentially, this is happening because for a geometrically distributed random variable  $X$  with mean  $1/\epsilon$ ,

$$P[X = v] = \frac{\epsilon}{1 - \epsilon} P[X > v] \quad (3)$$

Furthermore, if we fix everyone else's perturbations and all rewards. Claim: there is a value  $v \geq 0$  such that *if and only if* expert  $i$ 's perturbation is at least  $v$  then expert  $i$  will be chosen by PFC at period  $t + 1$ . To see this, note that for sufficiently large  $v$ , expert  $i$  will certainly be chosen. As we decrease  $i$ 's perturbation to 0, there will be at most one value where  $i$  no longer is leader. In order for PFC to switch to expert  $i$  on period  $t + 1$  it must be that its perturbation is exactly  $v$  (even in this case, it may not switch). On the other hand, if its perturbation is greater than  $v$ , PFC will already choose it on period  $t$ . So, by (3), we have (2).

So, now imagine that we run PFC on the sequence including period 0. Since we're imagining period 0 anyway, let's imagine that we happened to start with the expert with largest perturbation. In this case, we can bound our regret by the number of switches, as in eq. (1).

We will finish in the next lecture... The basic idea is to combine equations (2) and (3) with the expected maximum perturbation.