15-451/651 Algorithm Design & Analysis, Spring 2026

Homework #6 Solutions

---

1. Similar to the binary case, let's drop the weight of an expert every time it's wrong.

   WTS: We have the same kind of bounds as in Lecture 19 when there are more than two possible outcomes.

   Then if we used the same strategy:

   - initialize all expert weights to 1
   - each iteration, set the decrease factor per mistake to $exp(-\beta)$
   - still play fractional strategies based on the probability distributions

   Then we can note that we would only make a mistake when a majority votes incorrectly ($> \frac{1}{k}\Phi_t$ expert weight). ... similar math to earlier

   a self-contained writeup:

   Let the experts be $i = 1 \dots n$. We assign weights to the experts, $w_1 \dots w_n$, all initialized to 1. After each turn, we half the weights of the experts who predicted incorrectly. Let the weight of expert i at time t be $w(t)_i$.

   At each step, we compute the total weight of the experts that produce each outcome. That is, for each outcome $1 \le j \le k$, we compute the total weight of the experts that predict that outcome

   $$w(t,j) = \sum_{i:\text{expert } i \text{ predicts outcome } j \text{ at step } t} w(t)_i.$$

   Then we predict the outcome with the maximum weight, $\arg\max_j w(t,j)$.

   Consider a step where we made a mistake. That is, at step t, we predicted outcome $\widehat{j}(t)$, while the correct outcome is $j^*(t)$. Then because the outcome we chose has the maximum weight, we have $w(t,\widehat{j}(t)) \ge w(t,j^*(t))$, and therefore

   $$w\left(t,j^*(t)\right) \le \frac{1}{2}\left(w\left(t,j^*(t)\right) + w\left(t,\widehat{j}(t)\right)\right) \le \frac{1}{2}\sum_{j=1}^{k} w(t,k) = \frac{1}{2}\sum_{i=1}^{n} w(t)_i\,.$$

   After this step, because we half the weight of all experts other than those who predicted $j^*(t)$, the total weight decrease is

   $$\frac{1}{2}\left(\sum_j w\left(t,j\right) - w(t,j^*(t))\right) \ge \frac{1}{4}\sum_{1\le i\le n} w(t)_i$$

   So for each mistake, the total potential decreases by a fact of 3/4. Once the algorithm makes M mistakes, the total potential is at most

   $$n \cdot \left(\frac{3}{4}\right)^M$$

   and the minimum number of mistakes made by some expert is at least

   $$\log_{\frac{1}{2}}\left(n\left(\frac{3}{4}\right)^M\right) \ge \log_{\frac{4}{3}} 2M - O\left(\log n\right)$$

2. With a cache size of $n-1$, there is always one of the n elements in the cache. So worst case scenario, after our cache is full, an adversary would always pick the element not in the set. Now compared to an optimal/omniscient algorithm, it would eject the element furthest in the future sequence. So after after $n-1$ iterations, the optimal algorithm would have a cost of 1 and any online algorithm would have a worst case cost of $n-1$, making it $n-1$ competitive.

3. Run regret minimization with weights initialized as $w_e^{(0)} = 1$.

   Each step, add a minimum weight spannning tree to the collection, and double the weights of the edges in the tree.

   We first show that for any assignment of weights to the edges of the graph, the minimum weight spanning tree has weight at most $(2/k) \cdot \sum_e w_e$.

   The key fact that we use is that any contraction of G is still k-edge connected. In particular, since each vertex has degree at least k, any contraction onto y vertices has at least $yk/2$ (possibly duplicate) edges.

   Consider running Kruskal's algorithm for building min-weight spanning tree. By the above argument, when there are $x+1$ connected components left, the number of edges remaining must be at least $xk/2$. The weight of the x edges remaining to be picked can then be charged to the weights of the $xk/2$ edges. So the amount charged to each edge is at most $2/k$, and we get that the total weight of all edges in the graph is at least $(k/2)$ times the weight of MST, or that $w(MST) \leq (2/k)w(G)$.

   Formally, we can write this as an integral:

$$w(T) = \int_{w=-\infty}^{\infty} \text{number of edges in } T \text{ with weights at least } w \, dw$$

   while

$$w(G) = \int_{w=\infty}^{\infty} \text{number of edges in } G \text{ with weights at least } w \, dw,$$

   and it suffices to show that for any w, there are at least $(k/2)$ times as many edges in G with weights at least w, than the number of edges with weight at least w in T.

   This is because by cut rule, once we contract all edge in T with weight $< w$, all edges remaining in the graph have weight at least w.

   If x edges remain to be picked, the contracted graph has at least $x+1$ vertices. So since each vertex has degree at least k, we get that the total number of edges is at least $(x+1)k/2 \geq xk/2$.

   Applying this to the multiplicative weight update procedure gives that the total potential grows by at most $1 + 10/k \leq \exp(10/k)$ per step, or that the total potential after $k \log n$ steps is at most

$$n \cdot \exp\left(\frac{10}{k} \cdot k \log n\right) \leq 2^{20 \log n},$$

   which in turn implies that the number of times an edge is used is at most $20 \log n$.