15-451/651 Algorithms, Spring 2019

Homework #7 **Due: Tuesday–Friday, April 23–26, 2019**

(100/3 pts) 1. **(Can you Hear me Now?)** There are two providers in town, called 1 and 2. You start off with a phone from 1. The rules are simple: every day you receive exactly one call. If its from someone with the same provider as your current provider, it's free, else it costs \$1. You can change providers on any day for the (low low) cost of \$$X$.

Consider the algorithm: maintain a counter K (initially zero), and whenever you get a call from someone with the other provider, increment K. When the counter reaches X, reset K to zero and then switch providers.

Show this algorithm is $O(1)$-competitive. For full credit, make sure this constant is at most 5. (Doing better than 5 is possible, but not required.)

Hint: think of a good potential function. Just like in the MTF analysis from lecture, consider the various actions you and OPT can take, and how each of them changes the potential.

Solution: Let the current counter be K. If OPT is co-located with you, let $\Phi = 4 \cdot K$, else let it be $3X - K$. There are a few cases. We show that in each case the amortized cost is at most 5 times OPT's cost, i.e.,

$$\Delta ALG + \Delta\Phi \leq 5\Delta OPT.$$

How did we find this potential: the general form is easy to guess. When we are at the same location, both us and OPT pay at the same time. So the potential has a positive coefficient for the K term and rises when OPT pays, since OPT's payment gives us money that we use to pay for our cost, and the rest we can store in the "bank". For the case when we are in different places, OPT does not pay when we pay and hence we need the potential drop to pay for our cost. (We have to "withdraw money" from the "bank" to pay for our costs.) So the coefficient of K is negative in this case. Now we need to make sure that at the transition points (when either of us changes providers), the values match up. Playing with this gives these precise forms.

When a request comes:

- If the request is at ALG's location, then $C_{ALG} = \Delta\Phi = 0$, so the amortized cost $\leq C_{OPT}$.
- If the request is at the other location:
  - If OPT and ALG are co-located, then OPT pays 1, ALG pays 1 and $\Delta(Phi) = 4$. So $C_{ALG} + \Delta\Phi \leq 5C_{OPT}$.
  - Else, $\Delta\Phi = -1$, so $C_{ALG} + \Delta\Phi = 0 = C_{OPT}$.

When OPT moves:

- If he moves to ALG's location, the potential goes from $2X - K$ to $4K$, so $\Delta\Phi = 4K - (3X - K) = 5K - 3X \leq 2X \leq 2C_{OPT}$.

- If he moves away, the potential goes to $2X - K$ from $4K$, so $\Delta\Phi = (2X - K) - 4K \leq 2X \leq 2C_{OPT}$.

When ALG moves (and resets the counter):

- if ALG and OPT are co-located, $C_{ALG} = X$, the original potential is $4K_{old} = 4X$ and the final is $3X - K_{new} = 3X$. So $\Delta\Phi = -X$, so $C_{ALG} + \Delta\Phi \leq 0 = C_{OPT}$.
- If not, the potential goes from $3X - K_{old} = 2X$ to 0, so $C_{ALG} + \Delta\Phi = X - 2X \leq 0 = C_{OPT}$.

Finally, the initial potential is zero, and the final potential is always non-negative. So this shows that the total cost of ALG is at most 5 times the total cost of OPT.

(100/3 pts) 2. **(Fickle Experts.)** In class you saw the randomized weighted majority theorem, in which we were given n experts. Then over any sequence of T rounds, and any expert i, we had

$$\mathbf{E}[\text{number of mistakes by RWM}] \leq (1+\varepsilon)m_i + \frac{\ln n}{\varepsilon}.$$

Here m_i is the number of mistakes made by expert i until time T . In Section 4 of the notes, we observed that $m_i \leq T$, so dividing the above by T and choosing $\varepsilon := \sqrt{\frac{\ln n}{T}}$ we get that for any i

$$\mathbf{E}[\text{rate of mistakes by RWM}] \leq \frac{m_i}{T} + 2\sqrt{\frac{\ln n}{T}}.$$

I.e., for any expert i (which includes the best expert at time T) the *average regret (versus that expert)*, which is our mistake rate minus that of the expert's mistake rate, goes to zero as $T \to \infty$.

(a) Above, we assumed we knew the time horizon T and hence could set $\varepsilon = \sqrt{\frac{\ln n}{T}}$. What if we don't know T? Here's one algorithm: for $s = 1, 2, \ldots$, play 2^s rounds of RWM (starting from scratch) with $\varepsilon = \sqrt{\frac{\ln n}{2^s}}$.

Show that the average regret of this algorithm after time T is $O\left(\sqrt{\frac{\ln n}{T}}\right)$.

Solution: Suppose $T \in (2^1 + 2^2 + \ldots + 2^k, 2^1 + 2^2 + \ldots + 2^k + 2^{k+1}]$ for integer k. When we run RWM for 2^s rounds, call that *epoch* s. Let $m_i(s)$ be the number of mistakes expert i makes in epoch s. Moreover, in epoch s, we use parameter $\varepsilon = \varepsilon_s = \sqrt{(\ln n)/2^s}$. Finally, time T falls in epoch $k+1$.
So by the RWM guarantee, for each s,

$$\begin{aligned}\mathbf{E}[\#\text{mistakes by RWM in epoch } s] &\leq (1+\varepsilon_s)m_i(s) + \frac{\ln n}{\varepsilon_s} \\ &\leq m_i(s) + \varepsilon_s 2^s + \frac{\ln n}{\varepsilon_s}.\end{aligned}$$

And in epoch $k+1$, let I denote the interval of times $[2^1 + 2^2 + \ldots + 2^k \ldots T]$. Then

$$\begin{aligned}\mathbf{E}[\#\text{mistakes by RWM at times in } I\ ] &\leq (1+\varepsilon_{k+1}) m_i(\text{times in } I) + \frac{\ln n}{\varepsilon_{k+1}} \\ &\leq m_i(\text{times in } I) + \varepsilon_{k+1} 2^{k+1} + \frac{\ln n}{\varepsilon_{k+1}}\end{aligned}$$

Summing these together:

$$\mathbf{E}[\#\text{mistakes by RWM until time } T] \leq m_i + \sum_{s=1}^{k+1} \left(\varepsilon_s 2^s + \frac{\ln n}{\varepsilon_s} \right).$$

But the two terms in the parentheses are both equal to $\sqrt{2^s \ln n}$, and hence the sum gives $O(\sqrt{2^k \ln n}) = O(\sqrt{T \ln n})$. Dividing through by T completes the proof.

(b) Here's a different extension. Now you don't just want to compare yourself to the best you could have done by choosing a single expert and sticking with them. Call an deterministic algorithm K-*fickle* if over the time horizon T, it follows the advice of some expert i_1 for the first t_1 steps, then i_2 for the next t_2 steps, etc, and then i_K for the last t_K steps, where each $i_j \in [n]$, $t_j \geq 0$ and $\sum_{j=1}^K t_j = T$. (Assume you know T, else you can use the "guess-and-double" idea from part (a).) Give an algorithm such that for any K-fickle (deterministic) algorithm A,

$$\mathbf{E}[\#\text{ mistakes by your algo }] \leq (\#\text{ mistakes by } A)(1+\varepsilon) + \frac{O(K \log(nT))}{\varepsilon}.$$

Your algorithm is allowed to run in time $(nT)^{O(K)}$.

Solution: First, observe there are $N \leq (nT)^K$ K-fickle algorithms: indeed each such algorithm can be described by writing down the name of the original expert they follow first, the length of time they follow that expert, then the name of the next expert, etc.

Create a new prediction-from-experts problem, where the experts are now the N different K-fickle algorithms, and run RWM on this set of new experts. The guarantee is that our mistakes are no worse than $(1+\varepsilon)$ times those of any K-fickle algorithm A by $\frac{\log N}{\varepsilon} = \frac{O(K \log(nT))}{\varepsilon}$.

(100/3 pts) 3. **(Let's Eliminate Gauss!)** Given an $n \times n$ *symmetric* matrix A and an $n \times 1$ vector b, our goal is to solve the equation $Ax = b$ to high accuracy. We will use gradient descent to solve this problem quickly given some assumptions about A; see the lecture notes for background on gradient descent. *(The analysis here is independent of the one from lecture, but you should be comfortable with the ideas there.)*

Recall from linear algebra that every symmetric $n \times n$ matrix A can be written as $V \Lambda V^T$, where V is an $n \times n$ matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix whose entries are the eigenvalues of A. Recall that for $x \in \mathbb{R}^n$, $\|x\|^2 = \sum_{i=1}^n x_i^2$.

(a) Consider the function $f(x) = \frac{1}{2}\|Ax - b\|^2$. Prove that f is convex and the gradient $\nabla f(x) = A(Ax - b)$. *(Hint: if $g(y)$ is a convex function, what about $f(x) = g(Ax - b)$?)*

Solution: $f(x) = \frac{1}{2}(Ax-b)^\intercal(Ax-b)$. You can expand out the product as $\frac{1}{2}x^\intercal A^2 x - b^\intercal Ax + \frac{1}{2}b^\intercal b$, where we used symmetry of A. Now differentiate with respect to each x_i.

(b) Suppose $x^* = \text{argmin}_x \frac{1}{2}\|Ax - b\|^2$. State why $A^2x^* = Ab$.

Solution: The minimizer is where the gradient is zero.

(c) Suppose we set $x^{(0)} = 0^n$, and

$$x^{(t+1)} \leftarrow x^{(t)} - \nabla f(x^{(t)}).$$

Argue for any $i \geq 0$, $A(x^{(i+1)} - x^*) = (I - A^2)(A(x^{(i)} - x^*))$.

Solution: Substitute the results of (a) and (b) into the update rule, and simplify.

(d) Argue that $\|Ax^{(t)} - b\|^2 = \|A(x^{(t)} - x^*)\|^2 + \|Ax^* - b\|^2$. Hint: for $x, y \in \mathbb{R}^n$, if $\langle x, y\rangle = 0$, then $\|x + y\|^2 = \|x\|^2 + \|y\|^2$. You may also find part (b) useful.

Solution: $Ax^{(t)} - b = (Ax^{(t)} - Ax^*) + (Ax^* - b) = A(x^{(t)} - x^*) + (Ax^* - b)$. Observe that $\langle Ay, z\rangle = (Ay)^\intercal z = y^\intercal A^\intercal z = y^\intercal Az = \langle y, Az\rangle$ for symmetric matrices A. So

$$\langle A(x^{(t)} - x^*), (Ax^* - b)\rangle = \langle x^{(t)} - x^*, A(Ax^* - b)\rangle = 0$$

using part (b). Hence we can use the Pythagoras **equality** here to get what we want.

The above parts should all be proven for any symmetric matrix A, regardless of whether it is invertible or not.

For the next parts, you may find the following statements helpful: (1) for a symmetric matrix B and a vector y, $\|By\| \leq \max(|\lambda_{max}|, |\lambda_{min}|) \cdot \|y\|$ where λ_{max} is the maximum eigenvalue of B and λ_{min} is the minimum eigenvalue of B, and (2) for a symmetric matrix B, the eigenvalues of $I - B$ are in the range $[1 - \lambda_{max}, 1 - \lambda_{min}]$. Please try to prove these facts about eigenvalues yourself for practice, though you will not need to prove these to us in the oral presentation.

For the following parts, assume that all eigenvalues of A are in the range $[.9, 1.1]$; such an A is called *well-conditioned*. (Although you need not use this fact: is such a matrix invertible, i.e., does A^{-1} exist?)

(e) Show that $\|A(x^{(i+1)} - x^*)\| \leq \frac{1}{2}\|A(x^{(i)} - x^*)\|$.

Solution: Define $e_i := A(x^{(i)} - x^*)$. So $e_{i+1} = (I - A^2)e_i$. Now the eigenvalues of A^2 are the squares of the eigenvalues of A, and hence lie in $[.81, 1.21]$. So those of $B := (I - A^2)$ lie in $[-0.21, 0.19] \subseteq [-.5, 0.5]$. Hence $\|e_{i+1}\| \leq 0.5\|e_i\|$.

(f) Prove that there exists a constant c such that for any $\epsilon \in (0,1)$, if $t \geq c\log(1/\epsilon)$), then

$$\|A(x^{(t)} - x^*)\|^2 \leq \epsilon \|b\|^2.$$

Solution: Using the above repeatedly, $\|e_t\|^2 \leq 2^{-2t}\|e_0\|^2 \leq \epsilon\|A(0-x^*)\|^2 = \epsilon\|b\|^2$, where we used that $t = \frac{1}{2}\log_2(1/\epsilon)$ ensures that $2^{-2t} = \epsilon$.

(g) Assuming that A has m non-zero entries, what is the overall running time of the algorithm for outputting an $x^{(t)}$ for which $\|Ax^{(t)} - b\|^2 \leq \|Ax^* - b\|^2 + \epsilon\|b\|^2$? Give an answer in terms of m, n, ϵ. Assume the non-zero entries of A are represented in such a way so that for any vector z, $A \cdot z$ can be computed in $O(m)$ time.

Solution: Each iteration you have to compute $x^{(i)} - A(Ax^{(i)} - b)$, which takes $m+n$ time if you store the matrix reasonably to be able to skip over the zero-entries. We have $O(\log(1/\epsilon))$ iterations so $(m+n)\log(1/\epsilon)$.