CS 787: Advanced Algorithms	**Homework 3**
Out: 11/17/16	**Due: 12/08/16**

Guidelines

- This homework consists of 6 problems. You are required to solve and turn in all of the problems.
- Some of the problems are difficult, so please get started early. Late submissions do not get any credit.
- Please typeset your solutions.
- Homework may be done in pairs. Please write your names clearly on your homework.

Problems

1. The probabilistic method is a way of asserting deterministic statements by using randomness. One of its main approaches is to assert the existence of a certain object by giving a probability distribution under which the object has non-zero probability. You will use this approach in this question.

   Call two vectors near-orthogonal if their inner product has small absolute value compared to the product of their lengths; in this problem we will show that while there are at most d orthogonal vectors in $\Re^d$, there can be exponentially more near-orthogonal vectors.

   Given a parameter $\epsilon > 0$, two vectors $\vec{x}, \vec{y} \in \Re^d$ are called ϵ-orthogonal if

   $$|\vec{x} \cdot \vec{y}| \leq \epsilon \, \|x\| \, \|y\|.$$

   Show that there exist $N = \exp(\Omega(\epsilon^2 d))$ vectors in $\{-1, 1\}^d$ that are mutually ϵ-orthogonal.

   Hint: appropriately define a random set of vectors, and show that the probability that any two vectors in the set are ϵ-orthogonal is strictly positive, i.e., non-zero.

2. Recall that the singular vectors of a matrix A are defined as follows:

   $$v_1 = \underset{|v|=1}{\operatorname{argmax}} |Av|$$
   $$v_i = \underset{|v|=1, v \perp v_1, \cdots, v_{i-1}}{\operatorname{argmax}} |Av|$$

   Let $\sigma_i = |Av_i|$ and $u_i = \frac{1}{\sigma_i} A v_i$ for all i.

   (a) Prove that the vectors u_i are orthonormal. That is, they are pairwise orthogonal unit vectors.

   (b) Let $B = A^T A$. Note that B is a symmetric square matrix. Prove that the vectors v_j are right eigenvectors of B, that is, $Bv_j = \gamma_j v_j$ for some scalar γ_j for all j. What are the eigenvalues γ_j?

   (c) Write and simplify an expression for B^k.

   (d) Assume that $\sigma_i \ll \sigma_1$ for all $i > 1$. Prove that the first column of the matrix B^k converges to a multiple of v_1 as k goes to infinity.

3. The **online bin-packing** problem is a variant of the knapsack problem. We are given an unlimited number of bins, each of size 1. We get a sequence of items one by one, each of a certain size no more than 1, and are required to place them into bins as we receive them. Our goal is to minimize the number of bins we use, subject to the constraint that no bin to should be filled to more than its capacity. In this question we will consider a simple online algorithm for this problem called **First-Fit** (FF). FF orders the bins arbitrarily, and places each item into the first bin that has enough space to hold the item.

   Recall that the competitive ratio of an online algorithm for a minimization problem is the maximum over all arrival sequences of the cost of the algorithm (in this case, the number of bins used by the algorithm) to the cost of the hindsight OPT (in this case, the minimum number of bins necessary to pack the items).

   (a) Give an instance of bin-packing for which FF does not obtain the optimal packing.

   (b) Prove that FF has competitive ratio no more than 2, that is, on every instance, it uses no more than twice as many bins as necessary.

4. Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

   (a) Each expert begins with weight 1 (as before).

   (b) At every step, we predict the result of a weighted-majority vote of the experts (as before).

   (c) At every step, for every expert that makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least $1/4$ of the average weight of experts.

   Prove that in any contiguous block of steps (e.g., the 51st step through the 77th step), the number of mistakes made by the algorithm is at most $O(m + \log n)$, where m is the number of mistakes made by the best expert in that block, and n is the total number of experts.

   (Recall that the original weighted majority algorithm only gives a guarantee over a block of steps starting at step 1.)

5. Consider a standard experts setting and suppose that at any time step, only a subset of the experts are available to make a prediction. Can we modify the Hedge algorithm from class to give a low regret bound in this case? More precisely, in this setting, at every time step we get to see which experts are "awake" to make a prediction. We choose one of those experts. Then we get to see the cost vector for that step. The total cost of expert i is the sum of its costs over the steps that i was awake in. Naturally, if an expert is awake for very few steps, we cannot hope to prove that our total cost over all the steps is not much larger than the expert's cost over the steps that it was awake in. So we will aim for a slightly different guarantee.

   Let T_i denote the set of steps when expert i was awake, $\text{cost}_i(\text{ALG})$ denote the expected cost of the algorithm over the steps T_i, and $\text{cost}_i(i)$ denote the cost of expert i over the steps T_i. Then, our goal is to say that $\text{cost}_i(\text{ALG}) \leq \frac{1}{1-\epsilon}\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$, and this should hold for every expert i.

   Consider the following variant of the Hedge algorithm for this problem:

   (i) Initialize $w_{i,0} = 1$ for all i.

   (ii) At every step t, let $p_{i,t} = w_{i,t}/W_t$ be the probability of picking an awake expert i, where W_t is the sum of the weights of all the experts awake at that step (not the total weight of all the experts).

   (iii) Let $c_{i,t}$ denote the cost to expert i at step t. Update weights for awake experts as follows:

   $$R_{i,t} = \frac{1}{1+\epsilon}\left(\sum_j p_{j,t}c_{j,t}\right) - c_{i,t}$$

   $$w_{i,t+1} = w_{i,t}(1+\epsilon)^{R_{i,t}}$$

   (a) Prove using induction that the total weight of all experts at any step is at most n. (In particular, although individual weights can go up or down, the total weight never goes up).

(b) Express the weight of expert i at time T in the form of the total expected cost of the algorithm and the total cost of the expert. Use part (a) to prove that $\text{cost}_i(\text{ALG}) \leq (1+\epsilon)\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$.

6. The "Follow The Perturbed Leader" (FTPL) algorithm for the online prediction problem is given as follows. Here $i \in [n]$ denotes an expert. For all times $t \in [T]$ and experts $i \in [n]$, $c_t(i)$ denotes the cost of expert i at time t; the cost can be 0 or 1. Fix some $\epsilon > 0$.

   - For every expert i independently, flip a coin with bias ϵ (that is, the probability of heads is ϵ) repeatedly until it comes up heads. Let X_i denote the number of coin flips.
   - For each $t = 1, 2, \cdots, T$:
     - Set $a^t = \text{argmin}_{i\in[n]} \left(X_i + \sum_{s<t} c_s(i)\right)$ breaking ties in favor of smaller indices i. a^t is the perturbed leader at time t.
     - Follow a^t's prediction.

   Recall that the algorithm's cost is $\sum_{t\in[T]} c_t(a^t)$. In this problem your goal is to prove that, in expectation over the choices of the X_is, this is not much more than $\min_i \sum_{t\in[T]} c_t(i)$, the total cost of the best expert.

   To prove this claim, we will consider a slight variant of the algorithm, where we can use a one-step lookahead: we get to see the costs of the current step when we decide which expert to choose. Of course this makes the problem trivial, but we will nevertheless use an FTPL style algorithm. Call the following algorithm FTPL with lookahead, or FTPLL for short. Observe that it is identical to FTPL, except that the sum in the definition of $\tilde{a}^t$ is over $s \leq t$ rather than over $s < t$.

   - Use the same X_i's as in FTPL.
   - For each $t = 1, 2, \cdots, T$:
     - Set $\tilde{a}^t = \text{argmin}_{i\in[n]} \left(X_i + \sum_{s\leq t} c_s(i)\right)$ breaking ties in favor of smaller indices i.
     - Follow $\tilde{a}^t$'s prediction.

   (a) Note that for every $t \in [T]$ and $i \in [n]$, we have $X_{\tilde{a}^t} + \sum_{s\leq t} c_s(\tilde{a}^t) \leq X_i + \sum_{s\leq t} c_s(i)$. Use this to prove that $\sum_{t\in[T]} c_t(\tilde{a}^t) \leq \min_i \sum_{t\in[T]} c_t(i) + \max_i X_i$.

   (b) Prove that $\text{E}[\max_i X_i] = O(\log n/\epsilon)$.

   (c) Use parts (a) and (b) to conclude that the regret of FTPLL is small.

   (d) Now we will compare the performance of FTPL and FTPLL. Fix any step t, and observe that the leader chosen by FTPL at step $t+1$, a^{t+1}, is identical to the leader chosen by FTPLL at step t, $\tilde{a}^t$. Prove that the probability, over the choices of X_is, that a^t is different from a^{t+1} is at most ϵ. Conclude that the probability that a^t is different from $\tilde{a}^t$ is at most ϵ.

   Hint: Suppose that a^t is different from a^{t+1}. What does this say about X_{a^t} and $X_{a^{t+1}}$?

   (e) Use part (d) to prove that the regret of FTPL is bounded by $\epsilon T + O(\frac{1}{\epsilon}\log n)$.