Assignment 2
CS289: Algorithmic Machine Learning, Spring 2022
Due: April 27, 10PM

Guidelines for submitting the solutions:

- The assignments need to be submitted on Gradescope. Make sure you follow all the instructions - they are simple enough that exceptions will not be accepted.
- Start each problem or sub-problem on a separate page even if it means having a lot of white-space and write/type in large font.
- The solutions need to be submitted by 10 PM on the due date. No late submissions will be accepted.
- Please adhere to the code of conduct outlined on the class page.

1. Consider the hypothesis class $\mathcal{H}$ consisting of functions $f : \{0,1\}^d \to \{0,1\}$ of the form $AND_S(x) = \wedge_{i \in S} x_i$ for subsets $S \subseteq [d]$. Now, suppose you are given as input $(example, label)$ pairs $(x^1, y_1), \ldots, (x^m, y_m), \ldots$, where $x^\ell \in \{0,1\}^d$ and $y^\ell = AND_{S_*}(x^\ell)$ for some unknown non-empty S_*. Give an efficient (the algorithm should run time at most polynomial in d) online learning algorithm in the *mistake bounded* model. For full-credit your algorithm should run in polynomial time for each example, and must make at most a polynomial number of mistakes in its entire run-time (no matter how long). You don't have to prove that your algorithm works. [4 points]

   Hint: Start with the full set of features and think of weeding out coordinates that you know *cannot* be in S_*.

   Solution. The algorithm is as follows.

   1. Start with $S = \{1, 2, \ldots, d\}$.

   2. For each example (x, y): if correct, do nothing. If a false negative (i.e., $AND_S(x) = 0$, but $y = 1$), remove from S all those coordinates that are 0 in x. In other words, set $S = S \setminus \{i : x_i = 0\}$.

   Claim: The above algorithm makes at most d mistakes.

   Proof: We first argue that $S_* \subseteq S$ at all times. This is because, we only remove a coordinate i from S when $x_i = 0$ but $And_{S_*}(x) = 1$. Which in turn implies that the removed coordinates are not in S_*. Thus, the only possible mistakes the algorithm makes are false negatives (there can't be a mistake of the form $AND_S(x) = 1$, but $AND_{S_*}(x) = 0$). Further, whenever we make a mistake we remove at least one coordinate so $|S|$ decreases by at least 1. As it is never empty (it contains S_*), the number of mistakes is at most d.

2. Show that no *deterministic algorithm* can do better than a factor two approximation to the loss of the best expert for the *learning with experts* problem. That is, show that for any deterministic algorithm, there exists a sequence of losses where the algorithm is off by at least a factor of two from the performance of the best expert. [4 points]

   Hint: You can come with an example even with just two experts.

   Solution. Suppose you have a deterministic algorithm $\mathcal{A}$. Consider the following loss sequence, for each round t, if the algorithm $\mathcal{A}$ asks to follow expert 1, then set $L(t,1) = 1$ and $L(t,2) = 0$ and if $\mathcal{A}$ asks to follow expert 2, then set $L(t,1) = 0$ and $L(t,1) = 1$. Note that the choice of $\mathcal{A}$ can be complicated and depend on the entire history of the game so far; nevertheless, as it is deterministic, we know the decision to be made before the t'th day and hence can fix the loss functions adversarially as above.

   In the above sequence of losses, the algorithm $\mathcal{A}$ makes a mistake every round so the total loss is T. On the other hand, in each round at least one of the experts is right. So overall, the total-loss of E_1 + the total-loss of E_2 is exactly T. So the best expert achieves a loss of at most $T/2$.

3. The goal here is to solve a variant of the learning with experts setup where the experts incur *losses* that take values in $[0,1]$ (instead of $\{0,1\}$ as we did in class). Let us suppose that on each day, we are trying to guess some number between $[0,1]$ (instead of just UP vs DOWN as in class) based on the guesses of the experts.

   Concretely, consider a learning with experts setup where you have n experts $E_1, \ldots, E_n$ and the interaction proceeds as follows:

   - At the beginning of each day, the experts make a prediction which is a number between $[0,1]$.
   - Our algorithm then makes a prediction which is also a number between $[0,1]$.
   - At the end of the day, the true number (for that day) is revealed to us and each expert and our algorithm incur a loss that is the absolute-value of the difference between their prediction and the true value.

   Given $0 < \epsilon < 1$, design a randomized algorithm whose expected total-loss after T days is at most $(1+\epsilon)\mathcal{A}_*(T) + (\ln n)/\epsilon$, where $\mathcal{A}_*(T)$ is the loss of the best-expert after T days. Give a complete proof that the algorithm achieves the above bound. [5 points]

   Hint: You basically have to slightly change the multiplicative weights algorithm we did in class (in how you update the weights). The analysis also does not change much; you may use the following fact: For $0 \le \epsilon \le 1/2$, $(-\ln(1-\epsilon))/\epsilon \le (1+\epsilon)$.

   Solution. You have to change the algorithm to use the update $w(t,i) = (1-\epsilon L(t,i))w(t-1,i)$. The analysis remains identical, the only necessary change is to use the inequality $e^{-\epsilon(1+\epsilon)\ell} \le (1-\epsilon\ell)$ to lower bound the weight function with the loss of the best expert. Nothing else really changes.

4. Let us now apply MWM to an online learning problem like the perceptron. Suppose we are given some examples $(x_1, y_1), \ldots, (x_t, y_t), \ldots$ in an online manner where $x_i \in [-1,1]^d$ and

$y_i \in \{1, -1\}$. We know that $y_i = sign(\langle w_*, x_i \rangle)$ where w_*, the *unknown true coeffecient* vector satisfies:

(a) w_* has non-negative coordinates with $\|w_*\|_1 = 1$ (i.e., has sum of entries exactly 1).

(b) For all i, $y_i \langle w_*, x_i \rangle > \gamma$.

Use MWM to come up with an online learning algorithm where the number of mistakes (for any arbitrary sequence of points (x_i, y_i)) after T days is $O(\sqrt{T \log d}/\gamma)$.

Hint: Try to keep things simple (the solution can be reasonably short). Note that there is no *learning with experts* problem here to start with! But we can define a thought experiment where the features of x correspond to experts. The entire problem and solution then comes down to defining a suitable (and perhaps somewhat magical) choice of losses for each day so that applying the regret bound for MWM (the more general version with losses in $[0, 1]$ as in the previous problem) gives you the claim above. You don't have to redo the analysis of MWM.

In particular, imagine there are d experts corresponding to each coordinate of the input (just as we did for the boosting problem). Note that the true predictor w_* can be interpreted as a distribution on d experts. Imagine running a game where you use the distribution induced by the weights assigned to the experts to make a prediction of the sign in each round. Now, you have to update the weights. The key here is to define a 'loss' so that the MWM update makes the new distribution more aligned with the unknown w_*. The crux of the problem is to come up with the definition of this 'fictional' loss to use in the MWM framework to get the new distribution to be used for the next day. [5 points]

(Remark: The second condition above on w_* is similar in spirit to the margin condition we had in analysis of perceptron **but** is different. The normalizations are different - the x vectors are not set to be on the unit sphere any more. If you apply the perceptron algorithm, you end up with a guarantee like $O(d/\gamma^2)$ which could be much worse than the above if the dimension is big.)

Solution. As suggested in the hint above, consider learning with experts game where we imagine there are d experts corresponding to the d features (d-dimensional data).

1. Set $w(0, i) = 1$, and $p^0 = (1/d, 1/d, \ldots, 1/d)$.

2. For each $t = 1, \ldots T$ for example (x^t, y^t):

a. Predict using p^{t-1}: Guess the label as $sign(\langle p^{t-1}, x^t \rangle) = \hat{y^t}$.

b. If correct, define losses $L(t, i) = 0$ for all i. Set $w(t, i) = w(t-1, i)$ and $p^t = p^{t-1}$.

c. Else, define losses $L(t, i) = -y^t x_i^t$. Update weights $w(t, i)$ using the corresponding MWM update. Update the prediction vector as $p_i^t = w(t, i) / \sum_{j=1}^d w(t, j)$

Let us bound the number of mistakes made. Note that when there are no mistakes, all losses are 0. When our algorithm makes a mistake in round t, the expected loss $L(t) = \sum_{i=1}^d Pr[E_i \text{ is picked}] L(t, i) = -y^t \langle p^{t-1}, x^t \rangle$. But as we made a mistake in this round, $sign(\langle p^{t-1}, x^t \rangle) \neq y^t$, so $L(t) \geq 0$! Thus, the expected loss of the algorithm after T rounds is non-negative: $0 \leq A(T)$.

For brevity, let $L^t \in [-1,1]^d$ be the losses vector in each round, i.e, $L^t_i = L(t,i)$. Let $A'(T) = \sum_{t=1}^{T} L^t$ denote the vector of total losses incurred by the experts (so the first coordinate is the total loss of first expert and so on). We will use the following fact about any vector $a \in R^d$:

$$\min((a_i : i \in [d])) \leq \langle a, w_* \rangle.$$

This is true mainly because w_* is a non-negative vector whose coordinates add up to 1 (i.e., the expectation of a random variable is always at least its minimum possible value.) Concretely, we have

$$\langle a, w_* \rangle = \sum_{i=1}^{d} a_i (w_*)_i \geq \sum_{i=1}^{d} (\min_i a_i)(w_*)_i = (\min_i a_i) \sum_{i=1}^{d} (w_*)_i = (\min_i a_i).$$

Using the above inequality to $a = A'(T)$, we get

$$\text{Loss of best expert} = \min((A'(T)_i : i \in [d])) \leq \langle A'(T), w_* \rangle.$$

Therefore, by combining with the regret bound, we get

$$A(T) \leq \sum_{t=1}^{T} \langle L^t, w_* \rangle + O(\sqrt{T \ln d}).$$

However, when we make a mistake, $\langle L^t, w_* \rangle = -y^t \langle x^t, w_* \rangle \leq -\gamma$, by assumption (b) in the problem. Therefore, $\sum_t \langle L^t, w_* \rangle \leq -\gamma(\text{num of mistakes})$. So we get,

$$0 \leq (-\gamma)(\text{num of mistakes}) + O(\sqrt{T \ln d}).$$

This gives the bound in the problem.