10-806 Foundations of Machine Learning and Data Science

Homework # 3 **Due: November 4, 2015**

Groundrules:

- Your work will be graded on correctness, clarity, and conciseness. You should only submit work that you believe to be correct; if you cannot solve a problem completely, you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) proof summary that describes the main idea.
- You may collaborate with others on this problem set and consult external sources. However, you must *write your own solutions* and *list your collaborators/sources* for each problem.

Problems:

1. [25 pts] **Halving with a Prior.** Suppose we are in the realizable case, but we believe that certain functions in $\mathcal{C}$ are more likely to be the target than others. In particular, suppose we have a probability distribution p over $\mathcal{C}$ where $p(h)$ denotes the likelihood, according to our belief, that the target is h. In this case, a natural prediction strategy is to run the halving algorithm, but where each h in the version space (i.e., each $h \in \mathcal{C}$ that has not yet made any mistakes) is weighted according to p.

   (a) Prove that if the target is some $c^* \in \mathcal{C}$, then the total number of mistakes we make will be at most $\lg \frac{1}{p(c^*)}$.

   (b) The *Kolmogorov complexity* $K(h)$ of a function h is defined as the length in bits of the shortest program to compute h. Using the fact that at most 2^k functions can have Kolmogorov complexity k, define a prior p such that Halving run with that prior would make at most $2K(c^*)$ mistakes to learn a target function c^*. It is fine if your p has total probability mass less than 1 (this only improves the bound from part (a)).

   Interestingly, note that part (a) also implies that if indeed the target really was chosen randomly according to p, then the expected number of mistakes (expectation taken over the random draw of the target function) is at most the binary entropy of p, namely $\sum_{h \in \mathcal{C}} p(h) \lg \frac{1}{p(h)}$.

2. [25 pts] **Mistake-bound lower bound.** We saw that the class of threshold functions on the real line has VC-dimension 1. However, show that this class cannot be learned with any finite mistake bound in the mistake-bound model.

   Specifically, define $h_a(x) = 1$ if $x \geq a$ and $h_a(x) = 0$ if $x < a$, and let $\mathcal{C} = \{h_a : a \in R\}$. Show that for any deterministic prediction algorithm A, and any finite integer M, there exists a value a and a sequence of examples $x_1, x_2, \ldots, ..., x_{M+1}$ such that if these examples are presented in order to algorithm A and labeled by h_a then A makes mistakes on all of them.

   Hint: note that it is fine if your example x_i depends on A's answers to $x_1, \ldots, x_{i-1}$, and if the correct label of x_i depends on A's prediction on x_i, so long as in the end these are all consistent after the fact with some function h_a. That is because algorithm A is deterministic, so you could now fix a and the sequence $x_1, \ldots, x_{M+1}$ and re-run A on that.

3. [25 pts] **FTRL with an entropic regularizer.** In this problem you will see how FTRL with the appropriate regularizer produces the randomized weighted majority algorithm in the "combining expert advice" setting.

   Specifically, let $\mathcal{C} = \{p \in \mathcal{R}^n : \sum_i p_i = 1, \text{ and } p_i \geq 0 \text{ for all } i\}$. I.e., $\mathcal{C}$ is the set of all probability distributions over n experts. Each day t we will get a loss vector $\ell^{(t)}$ and pay the dot-product of the loss vector with our hypothesis vector (using superscripts to index time since we are using subscripts to index coordinates). Given a regularizer R (defined below), FTRL chooses $h^{(t)} = \text{argmin}_{p \in \mathcal{C}}[R(p) + \langle p, \ell^{(1)} + \ldots + \ell^{(t-1)} \rangle]$.

   Define regularizer $R(p) = \frac{1}{\epsilon}\sum_i p_i \ln p_i$, where by convention $0 \ln 0 = 0$. Note that $R(p) \leq 0$.

   (a) What is $h^{(1)}$? I.e., what p minimizes R subject to lying in the convex set $\mathcal{C}$. One way to solve this is to add a penalty term $\lambda(1 - \sum_i p_i)$ to R that is constant on $\mathcal{C}$, then take derivatives in each direction and set them to 0, and then adjust λ so that the minimum actually occurs in $\mathcal{C}$.

   (b) Now, show that the FTRL algorithm is equivalent to a version of the randomized weighted majority algorithm in which, when expert i experiences a loss of ℓ_i, we penalize it by multiplying its weight by $e^{-\epsilon \ell_i} \approx (1 - \epsilon \ell_i)$.

   (c) Show that in fact, we can get an $O(\sqrt{T \ln n})$ regret bound for this algorithm using the FTRL analysis. Specifically,

      i. Show that $R(h^*) - R(h^{(1)}) \leq \frac{1}{\epsilon} \ln n$.

      ii. Show that so long as the loss vectors $\ell^{(t)} \in [0,1]^n$ (i.e., each expert has loss between 0 and 1), we have $\ell^{(t)}(h^{(t)}) - \ell^{(t)}(h^{(t+1)}) = \langle \ell^{(t)}, h^{(t)} - h^{(t+1)} \rangle \leq 1 - e^{-\epsilon}$.

      iii. Using the fact that $1 - e^{-\epsilon} \approx \epsilon$, we have a loss bound of $\epsilon T + \frac{1}{\epsilon} \ln n$. Now set ϵ in terms of T to get the desired bound.

4. [25 pts] **Tracking a moving target.** 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) We predict the result of a weighted-majority vote of the experts (as before).

   (c) If an expert 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 trials (e.g., the 51st example through the 77th example), 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.

5. [20 pts extra credit] **Decision List mistake bound.** Give an algorithm that learns the class of decision lists over n Boolean variables in the mistake-bound model, with mistake bound $O(n^2)$. The algorithm should run in polynomial time per example.

   Hint: think of using some kind of "lazy" version of decision lists as hypotheses that perhaps has several if-then rules at the same level.

6. [20 pts extra credit] **Generalization bounds for Boosting.** Recall that the final predictor produced by Adaboost looks like $\text{sgn}(f(x))$ where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ and the h_t are the hypotheses produced by the weak-learning algorithm in the boosting process. Let $\mathcal{H}$ be the class used by the weak-learner (i.e., $h_1, \ldots, h_T \in \mathcal{H}$).

   (a) Let $d = VCdim(\mathcal{H})$. Show that the class of functions of the form $\text{sgn}(\sum_{t=1}^{T} \alpha'_t h'_t(x))$ for $h'_t \in \mathcal{H}$ has VC-dimension $O(dT \log(dT))$. Since Adaboost's predictor is in this class, this implies that whp the gap between its empirical error and true error is $\tilde{O}(\sqrt{\frac{Td}{m}})$, where m is the number of examples in the training set S.

   The above bound depends on T, suggesting that if we run Adaboost longer, we will overfit more. However, in practice often Adaboost does *not* have this problem. Here we will see an explanation using Rademacher complexity and the notion of L_1 *margins* (defined below).

   Let $\hat{R}_m(\mathcal{H})$ be the empirical Rademacher complexity of $\mathcal{H}$. To define margin, let's scale the α_t used in the final predictor $\text{sgn}(f(x))$ produced by Adaboost so that $\sum_{t=1}^{T} \alpha_t = 1$. For a labeled example (x, y), define the *margin* of the prediction as $yf(x)$; that is, this is the strength of the vote on example x (and is positive if the vote is correct). What you will prove is that for any value θ, with probability $\geq 1 - \delta$:

$$\Pr_D(yf(x) \leq 0) \leq \Pr_S(yf(x) \leq \theta) + O(\tfrac{1}{\theta}\hat{R}_m(\mathcal{H})) + O(\sqrt{\tfrac{\ln(1/\delta)}{m}}).$$

   (b) Let $conv(\mathcal{H})$ be the set of all convex combinations of functions in $\mathcal{H}$; so $f \in conv(\mathcal{H})$. Prove that $\hat{R}_m(conv(\mathcal{H})) = \hat{R}_m(\mathcal{H})$.

   (c) Now, define the function

$$\phi(z) = \begin{cases} 1 & \text{if } z \leq 0 \\ 1 - z/\theta & \text{if } 0 \leq z \leq \theta \\ 0 & \text{if } z \geq \theta \end{cases}$$

   Argue why $\Pr_D(yf(x) \leq 0) \leq \mathbf{E}_D[\phi(yf(x))]$ and $\mathbf{E}_S[\phi(yf(x))] \leq \Pr_S(yf(x) \leq \theta)$.

   (d) The *contraction lemma* states that if ϕ is a function with Lipschitz constant ρ (changing its input by Δ can change the value of ϕ by at most $\rho\Delta$) then for any class of functions $\mathcal{F}$, the set of functions of the form $\phi(f(x))$ for $f \in \mathcal{F}$ has empirical Rademacher complexity at most $\rho\hat{R}_m(\mathcal{F})$.

   Now, let $\mathcal{G}$ be the set of functions of the form $\phi(yf(x))$ for $f \in conv(\mathcal{H})$. In class we showed that for any class $\mathcal{G}$, with probability $\geq 1 - \delta$ we have that for every $g \in G$, $\mathbf{E}_D[g(x,y)] \leq \mathbf{E}_S[g(x,y)] + O(\hat{R}_m(\mathcal{G})) + O(\sqrt{\frac{\ln(1/\delta)}{m}})$. Using this fact along with parts (b) and (c) and the contraction lemma, prove the desired guarantee.