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 $C$ are more likely to be the target than others. In particular, suppose we have a probability distribution $p$ over $C$ 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 $h \in C$ that has not yet made any mistakes) is weighted according to $p$.

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

   Solution: The total weight of all $h$ in the version space begins at 1. Each mistake cuts down the weight of the version space by at least a factor of 2. The weight can never get below $p(c^*)$, so the total number of mistakes is 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)).

   Solution: Use a $p$ defined as $p(h) = 1/2^{2K(h)}$. The total sum of probabilities of all functions of Kolmogorov complexity $k$ is at most $2^k/2^{2k} = 1/2^k$. So the total probability mass summed over all $k$ is at most $1/2 + 1/4 + 1/8 + ... \leq 1$, as desired.

   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 $\sum_{h \in C} p(h) \lg \frac{1}{p(h)}$.

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 \( C = \{ h_a : a \in R \} \). 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.

Solution: Here is one solution (there are many). For convenience, let's look at the smaller class \( C' = \{ h_a : a \in [0, 1] \} \). We begin with \( x_1 = 1/2 \) and give it the opposite label of whatever \( A \) predicts. If the label is positive, we use \( x_2 = 1/4 \), else if it is negative we use \( x_2 = 3/4 \). We again give \( x_2 \) the opposite label of whatever \( A \) predicts. We continue to do this, defining the next example to be in the midpoint of the version space. So, the length of the version space gets cut by a factor of 2 on each example, but we can continue to do this forever.

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 \( C = \{ p \in R^n : \sum p_i = 1, \text{ and } p_i \geq 0 \text{ for all } i \} \). I.e., \( C \) is the set of all probability distributions over \( n \) experts. Each day 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 \( h(t) = \text{argmin}_{p \in C} [R(p) + \langle p, \ell^{(1)} + \ldots + \ell^{(t-1)} \rangle] \).

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 \( C \). One way to solve this is to add a penalty term \( \lambda(1 - \sum p_i) \) to \( R \) that is constant on \( C \), then take derivatives in each direction and set them to 0, and then adjust \( \lambda \) so that the minimum actually occurs in \( C \).

Solution: We add the term as in the hint and set partial derivatives to 0. This gives us \( \frac{1}{\epsilon}(\ln(p_i) + 1) - \lambda = 0 \), or \( \ln(p_i) = \lambda \epsilon - 1 \). This means that all \( p_i \) are equal, and we can adjust \( \lambda \) so that we get a legal probability distribution and we have \( p = (1/n, \ldots, 1/n) \).

(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 \( \ell_i \), we penalize it by multiplying its weight by \( e^{-\epsilon \ell_i} \approx (1 - \epsilon \ell_i) \).

Solution: Let \( p = h^{(t)} \). Using the same approach as above, we set all partials to 0 and get \( \ln(p_i) = \epsilon(\lambda - \sum_{r=1}^{t-1} \ell_i^{(r)}) - 1 \). This implies \( p_i = ce^{-\epsilon \sum_{r=1}^{t-1} \ell_i^{(r)}} \) where \( c = e^{\epsilon \lambda - 1} \) is a scaling constant that makes the probabilities sum to 1. This is exactly the probabilities in the RWM algorithm where when expert \( i \) experiences a loss of \( \ell_i \) we penalize it by multiplying its weight by \( e^{-\epsilon \ell_i} \).

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

Solution: This just comes from the fact that $R(h^*) \leq 0$ and $R(h^{(1)}) = \frac{1}{\epsilon} \ln n$ for $h^{(1)} = (1/n, \ldots, 1/n)$.

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 $\ell(t) (h^{(t)}) - \ell(t) (h^{(t+1)}) = \langle \ell(t), h^{(t)} - h^{(t+1)} \rangle \leq 1 - e^{-\epsilon}$.

Solution: For each coordinate $i$ we have $h^{(t+1)}_i \geq h^{(t)}_i e^{-\epsilon \ell(t)_i}$, because we first multiply $h^{(t)}_i$ by exactly this quantity and then we re-normalize which can only increase the value. Since $\ell(t)_i \leq 1$ we have $h^{(t+1)}_i \geq h^{(t)}_i e^{-\epsilon}$ which means that $\langle \ell(t), h^{(t)} - h^{(t+1)} \rangle \leq (1 - e^{-\epsilon}) \langle \ell(t), h^{(t)} \rangle \leq 1 - e^{-\epsilon}$.

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 $\epsilon$ in terms of $T$ to get the desired bound.

Solution: We just use $\epsilon = \sqrt{\ln n / T}$.

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.

Solution: First, notice that all weights are at least $1/8$ of the average. We can see this by induction: the average never increases, so you don’t have to worry about weights that weren’t lower in the last round. Also, if a weight was lowered, then it must have been at least $1/4$ of the old average, so it is now at least $1/8$ of the old average which is at least $1/8$ of the new average.

Let $W_{\text{init}}$ be the sum of weights of experts at the beginning of the block. So, the weight of best expert at end of the block is at least $(1/2)^m W_{\text{init}}/(8n)$

Also, on each mistake, at most $W/4$ of weight is fixed. So at least $(W/2 - W/4) = W/4$ gets cut in half. So, $W/8$ is removed. This means $W_{\text{final}} < W_{\text{init}} (7/8)^M$.

Then we solve.

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.

Solution: There are several ways to solve this problem. One is to create a hypothesis which is like a decision list but may have several rules at the same level. The semantics is that if any rule at
the top level fires, you pick one arbitrarily to use; else you go to the second level and see if any rule there fires and if so you pick one arbitrarily to use, and so on. We begin with all of the if-then rules at the top level. Any time we make a mistake, we move the rule that we used to make our prediction down by one level. Notice that the rule that is at the top level in the target function will never fire incorrectly and so will never be moved down. Therefore, the rule that is at level 2 in the target function will never be moved below level 2. More generally, the rule at level \( i \) in the target function will never be moved below level \( \ell(i) \). Since each mistake moves one rule down by one level, and no rule is moved past level \( n \), this means that the total number of mistakes must be \( O(n^2) \).

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 \( d = \text{VCdim}(\mathcal{H}) \). Show that the class of functions of the form \( \text{sgn}(\sum_{i=1}^{T} \alpha_i h_i(x)) \) for \( h_i \in \mathcal{H} \) 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 \( \tilde{O}(\sqrt{\frac{T d}{m}}) \), where \( m \) is the number of examples in the training set \( S \).

Solution: The solution to the question on the earlier homework about \( \text{MAJ}_k(\mathcal{H}) \) also solves this question as well, where \( k = T \).

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 \( \alpha_t \) 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 \( \theta \), with probability \( \geq 1 - \delta \):

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

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

Solution: We are interested in \( \frac{1}{m} \mathbb{E}_\sigma[\sup_{f \in \text{conv}(\mathcal{H})} \sum_i \sigma_i f(x_i)] \). Notice, however, that for any fixed \( \sigma \), the sup will occur at \( f \in \mathcal{H} \). That is because if \( f = \sum \alpha_t h_t \) then \( \sum \sigma_i f(x_i) = \sum \alpha_t(\sum \sigma_i h_t(x_i)) \), and the weighted average of a collection of numbers is never larger than the maximum of those numbers. So, \( \frac{1}{m} \mathbb{E}_\sigma[\sup_{f \in \mathcal{H}} \sum_i \sigma_i f(x_i)] = \frac{1}{m} \mathbb{E}_\sigma[\sup_{f \in \mathcal{H}} \sum_i \sigma_i f(x_i)] \) as desired.

(c) Now, define the function

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

4
Argue why \( \Pr_D(yf(x) \leq 0) \leq E_D[\phi(yf(x))] \) and \( E_S[\phi(yf(x))] \leq \Pr_S(yf(x) \leq \theta) \).

Solution: This follows from the definition of \( \phi \). When \( yf(x) \leq 0 \) we have \( \phi(yf(x)) = 1 \), and when \( yf(x) \geq 0 \) we have \( \phi(yf(x)) \geq 0 \), giving us the first inequality. Also, when \( yf(x) \leq \theta \) we have \( \phi(yf(x)) \leq 1 \) and when \( yf(x) \geq \theta \) we have \( \phi(yf(x)) = 0 \), giving us the second inequality.

(d) The contraction lemma states that if \( \phi \) is a function with Lipschitz constant \( \rho \) (changing its input by \( \Delta \) can change the value of \( \phi \) 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 \( f \in \text{conv}(H) \). In class we showed that for any class \( \mathcal{G} \), with probability \( \geq 1 - \delta \) we have that for every \( g \in \mathcal{G} \), \( E_D[g(x,y)] \leq E_S[g(x,y)] + O(\hat{R}_m(\mathcal{G})) + O(\sqrt{\frac{\ln(1/\delta)}{m}}) \). Using this fact along with parts (b) and (c) and the contraction lemma, prove the desired guarantee.

Solution: We use the fact that \( \phi \) has Lipschitz constant \( 1/\theta \) and then combine everything together.