TTIC 31250 An Introduction to the Theory of Machine Learning

Homework # 2 **Solutions**

Exercises:

1. **Kernels.** Recall that $K : X \times X \to \mathbb{R}$ is a legal kernel if there exists an implicit function ϕ such that $K(x, x') = \phi(x) \cdot \phi(x')$. (Here, X is our space of examples, such as $\{0, 1\}^n$.)

   Often the easiest way to prove that some function is a legal kernel is to build it out of other legal kernels. In particular, suppose $K_1(x, x') = \phi_1(x) \cdot \phi_1(x')$ and $K_2(x, y) = \phi_2(x) \cdot \phi_2(x')$, where $\phi_1 : X \to R^{N_1}$ and $\phi_2 : X \to R^{N_2}$ for some N_1, N_2. (Let's not worry about infinite-dimensional implicit feature spaces.)

   (a) Show that for any constant $c \geq 0$, cK_1 is a legal kernel. That is, show how to define ϕ in terms of ϕ_1 such that $\phi(x) \cdot \phi(x') = cK_1(x, x')$.

   (b) Show that the sum, $K_1 + K_2$, is a legal kernel. That is, show how to define ϕ in terms of ϕ_1 and ϕ_2 such that $\phi(x) \cdot \phi(x') = K_1(x, x') + K_2(x, x')$.

   (c) Show that the product, $K = K_1 K_2$, is a legal kernel. That is, show how to define ϕ in terms of ϕ_1 and ϕ_2 such that $\phi(x) \cdot \phi(x') = K_1(x, x')K_2(x, x')$. Hint: you will want to have $\phi : X \to R^{N_1 N_2}$.

   For instance, this implies that $(1 + x \cdot x')^d$ is a legal kernel. (Using the fact that "1" is itself a legal kernel, viewed as a function that always outputs 1).

   Solution:

   (a) Use $\phi(x) = \sqrt{c}\phi_1(x)$. So $\phi(x) \cdot \phi(x') = c\phi_1(x) \cdot \phi_1(x') = cK_1(x, x')$.

   (b) Use $\phi(x) = \phi_1(x) \circ \phi_2(x)$ where "$\circ$" is concatenation (i.e., $\phi : X \to R^{N_1+N_2}$). Then $\phi(x) \cdot \phi(x') = \phi_1(x) \cdot \phi_1(x') + \phi_2(x) \cdot \phi_2(x') = K_1(x, x') + K_2(x, x')$.

   (c) Use $\phi(x)$ as the outer-product of $\phi_1(x)$ and $\phi_2(x)$, so $\phi : X \to R^{N_1 N_2}$. In particular, $\phi(x)_{ij} = \phi_1(x)_i \phi_2(x)_j$. Now we get:

   $$\begin{aligned} \phi(x) \cdot \phi(x') &= \sum_{ij} \phi_1(x)_i \phi_2(x)_j \phi_1(x')_i \phi_2(x')_j \\ &= (\sum_i \phi_1(x)_i \phi_1(x')_i)(\sum_j \phi_2(x)_j \phi_2(x')_j) \\ &= K_1(x, x')K_2(x, x'). \end{aligned}$$

2. **About δ.** Suppose we changed the definition of PAC-learning to only require an algorithm succeed in finding a low-error hypothesis with probability at least $1/2$ (rather than with probability $1 - \delta$). It turns out that this does not change what is learnable.

Specifically, suppose we had an algorithm $\mathcal{A}$ with at least a $\frac{1}{2}$ chance of producing a hypothesis of error at most $\epsilon/2$ when given m labeled examples drawn from distribution $\mathcal{D}$ and labeled by a target function $f \in \mathcal{C}$. We can convert such an algorithm into an algorithm $\mathcal{B}$ that has at least a $1-\delta$ probability of producing a hypothesis of error at most ϵ. The reduction is that we first run $\mathcal{A}$ on $N = \lg \frac{2}{\delta}$ different sets of m random examples (so with probability at least $1-\delta/2$, at least one of the N hypotheses produced has error at most $\epsilon/2$). Then we test the N hypotheses produced on a new test set, choosing the one that performs best. Use Chernoff bounds to analyze this second step and finish the argument. That is, assuming that at least one of N hypotheses has error at most $\epsilon/2$, give an explicit bound (without O notation) on a size for the test set that is sufficient so that with probability at least $1-\delta/2$, the hypothesis that performs best on the test set has error at most ϵ.

Solution: It is enough to have a test set such that with probability $1-\delta/2$, any hypotheses of true error at most $\epsilon/2$ has empirical error at most $3\epsilon/4$, and any of true error greater than ϵ has empirical error greater than $3\epsilon/4$.

(Note 1: actually, we just need one hypothesis of true error at most $\epsilon/2$ to have empirical error at most $3\epsilon/4$, so we don't really need to do a union bound on this side. Note 2: if h has true error strictly less than $\epsilon/2$, the chance that its empirical error is greater than $3\epsilon/4$ is less than or equal to the chance this would occur if its true error were exactly $\epsilon/2$. You can see this by defining a new hypothesis h' such that h' is wrong whenever h is wrong, but h' makes additional mistakes to reach true error exactly $\epsilon/2$. Since the empirical error of h is never greater than the empirical error of h', it suffices to show that whp the empirical error of h' will be at most $3\epsilon/4$.)

By Chernoff bounds, it is sufficient to have a number of examples m such that

$$N \cdot \max\{e^{-m(\epsilon/2)(1/2)^2/3}, e^{-m\epsilon(1/4)^2/2}\} \leq \delta/2.$$

So, $m = \frac{32}{\epsilon}\ln(2N/\delta)$ is sufficient.

Problems:

3. **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: Let W_{init} be the total weight at the beginning of the interval and W_{final} be the total weight at the end of the interval.

First, notice that all weights are at least $1/8$ of the average. We can see this by induction: the average never increases, so the statement holds for weights that were not lowered 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.

This means that the weight of the best expert at beginning of the interval is at least $W_{init}/(8n)$, and therefore by end of the interval it is at least $(1/2)^m W_{init}/(8n)$.

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

The bound results from solving $(1/2)^m W_{init}/(8n) \leq W_{init}(7/8)^M$.

4. **Decision lists revisited.** On Homework 1 you showed that any decision list over n Boolean variables can be written as a linear threshold function. This makes one wonder if the Perceptron algorithm would get a good mistake bound in learning them.[1]

Unfortunately, the Perceptron may make $2^{\Omega(n)}$ mistakes in learning a decision list over n Boolean variables. To prove this, give a set S of labeled examples in $\{0,1\}^n$ that (a) is consistent with some decision list L and yet (b) has the property that any LTF with integer weights that has zero error on S must have at least one weight that is exponentially large. Prove that your set S has properties (a) and (b).

Solution: There are multiple ways to do it. Here is one, that uses a set of $O(n)$ examples. Define the linear separator to be $w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq w_0$, and let e_i denote the unit vector with a 1 in the ith coordinate. The all-zeroes example will be negative to force $w_0 \geq 1$. Now, the construction is as follows. We begin with two positive examples: $(e_1, +)$ and $(e_2, +)$, which ensure that $w_1 \geq 1$ and $w_2 \geq 1$. For the remaining examples, for each $i = 1, \ldots, n/4 - 1$ we give two negative and two positive examples as follows:

$$\begin{aligned}
&(e_{4i-3} + e_{4i-2} + e_{4i-1}, -),\\
&(e_{4i-3} + e_{4i-2} + e_{4i}, -),\\
&(e_{4i-1} + e_{4i} + e_{4i+1}, +),\\
&(e_{4i-1} + e_{4i} + e_{4i+2}, +).
\end{aligned}$$

[1]In class, we only described the Perceptron algorithm for learning a homogeneous linear separator (a linear separator through the origin). The algorithm can be applied to learn general linear separators by having it add a fake coordinate x_0 equal to 1 in every example.

The first few examples in this sequence are shown below:

example	label	implication
0000000000	$-$	$w_0 \geq 1$
1000000000	$+$	$w_1 \geq 1$
0100000000	$+$	$w_2 \geq 1$
1110000000	$-$	$w_3 \leq -2$
1101000000	$-$	$w_4 \leq -2$
0011100000	$+$	$w_5 \geq 4$
0011010000	$+$	$w_6 \geq 4$
0000111000	$-$	$w_7 \leq -8$
0000110100	$-$	$w_8 \leq -8$
...	...	...

These examples are consistent with a decision list "...if x_8 then negative else if x_7 then negative else if x_6 then positive else if x_5 then positive else if x_4 then negative else if x_3 then negative else positive".

We now claim that these examples ensure that $w_{4i+1} \geq 4^i$ and $w_{4i+2} \geq 4^i$ for all $i \leq n/4-1$. In particular, assume inductively that $w_{4(i-1)+1}, w_{4(i-1)+2} \geq 4^{i-1}$ (base case $i = 1$ given by the first two positive examples). Then the negative examples $(e_{4i-3} + e_{4i-2} + e_{4i-1}, -)$ and $(e_{4i-3} + e_{4i-2} + e_{4i}, -)$ ensure that $w_{4i-1} \leq -2 \cdot 4^{i-1}$ and $w_{4i} \leq -2 \cdot 4^{i-1}$ respectively. The examples $(e_{4i-1} + e_{4i} + e_{4i+1}, +)$ and $(e_{4i-1} + e_{4i} + e_{4i+2}, +)$ then ensure in turn that $w_{4i+1}, w_{4i+2} \geq 4^i$ as desired.

Thus, at the end, the largest weight has magnitude at least $4^{n/4-1} = 2^{n/2-2}$ as desired.

Note that the above analysis proves that the Perceptron algorithm makes an exponential number of updates.