TTIC 31250 An Introduction to the Theory of Machine Learning

Homework # 2 **Due: April 18, 2018**

Groundrules:

- Homeworks will generally consist of *exercises*, easier problems designed to give you practice, and *problems*, that may be harder, and/or somewhat open-ended. You should do the exercises by yourself, but you may work with a friend on the problems if you want. (Working together doesn't mean "splitting up the problems" though.) If you work with a friend, then write down who you are working with.
- If you've seen a problem before (sometimes I'll give problems that are "famous"), then say that in your solution. It won't affect your score, I just want to know. Also, if you use any sources other than material handed out, write that down too. It's fine to look up a complicated sum or inequality or whatever, but don't look up an entire solution.

Exercises:

1. **Kernels.** Recall that $K : X \times X \to R$ is a legal kernel if there exists an implicit function ϕ such that $K(x, y) = \phi(x) \cdot \phi(y)$. (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 $K(x, y) = \phi(x) \cdot \phi(y)$ and $K'(x, y) = \phi'(x) \cdot \phi'(y)$, where $\phi : X \to R^N$ and $\phi' : X \to R^{N'}$ for some N, N'. (Let's not worry about infinite-dimensional implicit feature spaces.)

   (a) Show that for any constant $c \geq 0$, cK is a legal kernel.

   (b) Show that the sum, $K + K'$, is a legal kernel.

   (c) Show that the product, KK', is a legal kernel.

   For instance, this implies that $(1 + x \cdot y)^d$ is a legal kernel. (Using the fact that "1" is itself a legal kernel).

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.

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.

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.

   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).