

# 15-859(B) Machine Learning Theory

Take-home final

Allotted time: 48 hours

---

*Groundrules:* You can use the book and lecture notes, but you shouldn't look through research papers (even those handed out) for solutions. You should work alone. Send me email if questions are confusing or unclear. If necessary, I will post an errata on the web page.

Do 3 of the following 4 questions. This test is worth 75 points.

## Questions:

- Decision Lists.** One natural approach to trying to learn a Decision List is to ask  $n$  Statistical Queries “what is the correlation of the target function  $c$  with variable  $x_i$ ” and to use the variable  $x_i$  of highest correlation (or the negation  $\bar{x}_i$  of the variable of lowest correlation) as a weak hypothesis, plugging into Boosting. (Formally, by “correlation” we mean  $\Pr(c(x) = x_i) - \Pr(c(x) \neq x_i)$ .)
  - Give an example of a distribution  $D$  over  $\{0, 1\}^n$  and a target decision list  $c$  where *all* variables have exponentially small correlation with  $c$  and furthermore  $\Pr(c(x) = 1) = 1/2$ . Thus, the above approach will fail at the outset.
  - Your example also illustrates a common mistake on problem #1 of homework 5, which was to use SQs of the form: “what is  $\Pr(c(x) = \ell_1 | x_i = \ell_2)$ ,” for all  $i, \ell_1, \ell_2$  and to include the rule for which this is *highest*. Explain why your example from part (a) would be problematic for such an algorithm. In particular, all statistics of this form that can be accurately estimated from a polynomial-size sample (i.e.,  $\Pr(x_i = \ell_2)$  is non-negligible) have values close to  $1/2$ .
- Models of learning.** Consider the following four learning models. In each of these we are given a class  $C$ , and the goal of the learning algorithm is to *exactly* recover the target  $c \in C$ .

**Equivalence Query:** In this model the algorithm can propose a hypothesis  $h$  and is told either that  $h$  is correct, or else is given a counterexample  $x$  such that  $h(x)$  is incorrect. This is really the same as the Mistake-Bound model.

**Restricted Equivalence Query:** Same as above except  $h$  must be from class  $C$ .

**Membership Query Only:** In this model the algorithm can propose examples  $x$  and is told their labels.

**Teacher-directed:** Like the membership query model, but now a teacher who knows the target function proposes the examples to be queried. The examples must *uniquely identify* the target concept, in the sense that no other  $c' \in C$  is consistent with this set of data.

For each class of functions below, state in which of the above models it can or cannot be learned/taught in a polynomial number of queries, along with a brief explanation.

- (a) The class of all functions over  $\{0, 1\}^n$  having *exactly one* positive example.
- (b) The class of all functions over  $\{0, 1\}^n$  having *at most one* positive example.
- (c) The class of monotone conjunctions over  $\{0, 1\}^n$  (including the conjunction of nothing, which is always positive).
- (d) The class of decision lists over  $\{0, 1\}^n$ .
- (e) The class of linear separators over  $\{0, 1\}^n$ .

3. **Kernels.** Car-talk statistician Marge Innovera proposes the following simple kernel function:

$$K(x, x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise.} \end{cases}$$

- (a) Prove this is a legal kernel. You may assume the instance space  $X$  is finite. Specifically, describe an implicit mapping  $\Phi : X \rightarrow R^m$  (for some value  $m$ ) such that  $K(x, x') = \Phi(x) \cdot \Phi(x')$ .
- (b) Marge likes this kernel because in the  $\Phi$ -space, any labeling of the points in  $X$  will be linearly separable. So, this should be perfect for learning any target function you want to: just run a kernelized version of Perceptron or SVM.
  - i. Why is any assignment of labels to points linearly separable?
  - ii. Nonetheless, what is the problem with her reasoning?

4. **PNF.** The class of  $k$ -term PNF is just like  $k$ -term DNF except that we use a *parity function* in place of the *OR* function. For instance, the following is a 2-term PNF:

$$x_1x_3\bar{x}_5 \oplus x_2x_4.$$

This function is positive on examples 11100 and 11011, and is negative on examples 00000 and 11110.

Give an algorithm that learns the class of 2-term PNF over  $\{0, 1\}^n$  in the mistake-bound model. Your algorithm should have a mistake bound polynomial in  $n$  and should be efficient (running in polynomial time per example) too. Your algorithm need *not* use 2-term PNF as its hypothesis representation. If you get stuck, then for partial credit give an *inefficient* algorithm.