CS446: Machine Learning **Fall 2015**

Problem Set 4

Handed Out: October 13^{th}, 2015 *Due: October* 22^{th}, 2015

- Feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- Please, no handwritten solutions. You will submit your solution manuscript as a single pdf file.
- Please present your algorithms in both pseudocode and English. That is, give a precise formulation of your algorithm as pseudocode and *also* explain in one or two concise paragraphs what your algorithm does. Be aware that pseudocode is much simpler and more abstract than real code.
- The homework is due at 11:59 PM on the due date. We will be using Compass for collecting the homework assignments. Please submit your solution manuscript as a pdf file via Compass (`http://compass2g.illinois.edu`). Please do NOT hand in a hard copy of your write-up. Contact the TAs if you are having technical difficulties in submitting the assignment.
- You cannot utilize the late submission credit hours for this problem set. We will release the solution immediately after it is due so that you can have time to go over it before the mid-term on Oct. 27th.
- No code is needed for any of these problems. You can do the calculations however you please. You need to turn in only the report.

1. **[VC Dimension - 30 points]**

   (a) **[15 points]** Assume that all examples are points in a two-dimensional space, i.e. $\mathbf{x} = \langle x_1, x_2 \rangle \in \mathbb{R}^2$. Consider the concept space of triangles in the plane. Hence, a concept $h \in \mathbf{H}$ is specified by three vertices $\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \langle x_3, y_3 \rangle$ (that are not on the same line). An example $\mathbf{x} \in \mathbb{R}^2$ is labeled as positive by h if and only if $\mathbf{x}$ lies inside or on the border of the triangle. Give the VC dimension of $\mathbf{H}$ and prove that your answer is correct. [Hint: When proving the upper bound of the VC dimension, use the concept of *convex hull* to distinguish among different situations. Provide illustrative pictures if necessary.]

   (b) **[15 points]** Consider the concept space of the union of k disjoint intervals in a real line. Hence, a concept $h \in \mathbf{H}$ is represented by $2k$ parameters $a_1 < b_1 < a_2 < b_2 < \ldots < a_k < b_k$. An example $x \in \mathbb{R}$ is labeled as positive by h if and only if x lies in one of the intervals $[a_p, b_p], p \in \{1, 2, \ldots, k\}$. Give the VC dimension of $\mathbf{H}$ and prove that your answer is correct. [Hint: Proving the lower bound of the VC dimension becomes easier if you divide it into simpler sub-problems.]

   Grading note: You will not get any points without proper justification of your answer.

2. **[Decision Lists - 40 points]**

   In this problem, we are going to learn the class of k-decision lists. A decision list is an ordered sequence of if-then-else statements. The sequence of if-then-else conditions are tested in order, and the answer associated with the first satisfied condition is returned. The class of k-decision lists is a subset of the class of all decision lists, where the statements in each rule are of a bounded size. Formally, for a fixed k, we define:

Definition 1 *A* **k-decision list** *over the variables* $x_1, \ldots, x_n$ *is an ordered sequence* $L = (c_1, b_1), \ldots, (c_\ell, b_\ell)$ *and a bit* b*, in which each* c_i *is a conjunction of at most* k *literals over* $x_1, \ldots, x_n$*. The bit* b_i *is referred to as the bit* associated *with condition* c_i*, and* b *is called the* default *value. For any input* $\mathbf{x} \in \{0,1\}^n$*,* $L(\mathbf{x})$ *is defined to take the value* b_j*, where* $j \in \{1, \ell\}$ *is the smallest index satisfying* $c_j(\mathbf{x}) = 1$*; if no such index exists, then* $L(\mathbf{x}) = b$*.*

We denote by **k-DL** *the class of concepts that can be represented by a* k*-decision list.*

Figure 1 shows an example of a 2-decision list over six variables, $x_1, \ldots, x_6$. For this decision list, $L(\mathbf{x}_1 = \langle 0,1,1,0,0,1 \rangle) = 1$ and $L(\mathbf{x}_2 = \langle 1,0,0,1,0,0 \rangle) = 0$, where in the case of $\mathbf{x}_1$ no condition is satisfied, so the labels is determined by the default bit, and for $\mathbf{x}_2$, the second condition is satisfied.

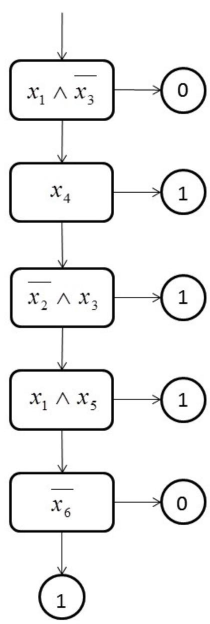

Figure 1: A 2-decision list.

(a) [**5 points**] Show that if a concept c can be represented as a k-decision list so can its complement, $\neg c$. You can show this by providing a k-decision list that represents $\neg c$, given $c = \langle (c_1, b_1), \ldots, (c_\ell, b_\ell), b \rangle$.

(b) [**10 points**] Show that

$$k\text{-DNF} \cup k\text{-CNF} \subseteq k\text{-DL}$$

Here, k-DNF is the disjunctive normal form with each conjunctive clause containing at most k literals. Similarly, k-CNF is the conjunctive normal form with each disjunctive clause containing at most k literals. [Hint: Use the relationship between k-DNF and k-CNF to help your proof.]

(c) **[15 points]** Let S be a sample data set that is *consistent* with some k-decision list. Provide an algorithm to construct a k-decision list that is consistent with S. Prove the correctness of your algorithm. [Hint: Consider the set of all possible conjunctions. When proving the correctness, argue that your algorithm will terminate, and when it does, it indeed produces a k-decision list, which is consistent with the sample S.]

(d) **[10 points]** Use Occam's Razor to show:
For any constant $k \geq 1$, the class of k-decision lists is efficiently PAC-learnable. [Hint: Consider the number of different k-DL functions.]

3. **[Constructing Kernels - 30 points]**

For this problem, we wish to learn a Boolean function represented as a **monotone** DNF (DNF without negated variables) using kernel Perceptron. For this problem, assume that the size of each term in the DNF is bounded by k, i.e., the number of literals in each term of the DNF is between 1 to k. In order to complete this task, we will first define a kernel that maps an example $\mathbf{x} \in \{0, 1\}^n$ into a new space of conjunctions of **upto** k different variables from the n-dimensional space. Then, we will use the kernel Perceptron to perform our learning task.

(a) **[5 points]** Define a kernel $K(\mathbf{x}, \mathbf{z}) = \sum_{c \in C} c(\mathbf{x})c(\mathbf{z})$, where C is a family of monotone conjunctions containing **upto** k different variables, and $c(\mathbf{x}), c(\mathbf{z}) \in \{0, 1\}$ is the value of c when evaluated on example $\mathbf{x}$ and $\mathbf{z}$ separately. Show that $K(\mathbf{x}, \mathbf{z})$ can be computed in time that is linear in n. [Hint: It would be useful to think about the case where each term of the DNF is *exactly* of k literals and then generalize.]

(b) **[10 points]** Write explicitly the kernel Perceptron algorithm that uses your kernel. [Hint: Think about the kernel based method.]

(c) **[15 points]** State a bound on the number of mistakes the kernel Perceptron algorithm will make on a sample of examples consistent with a function in this class. [Hint: In class, we proved a theorem by Novikoff; use this bound and estimate the two constants in it for this specific case. When doing it, observe that you are no longer learning in the original n dimensional space, but rather in a blown-up space that has a different dimensionality.]