10-806 Foundations of Machine Learning and Data Science

Homework # 1 **Due: September 28, 2015**

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. [20 pts] **Expressivity of Decision Lists.** Read the notes for Lecture 2, which describe the class of *decision lists*. Here we examine their expressive power.
   (a) Explain why any conjunction (like $x_1 \wedge \bar{x}_2 \wedge x_3$) can be written as a decision list.
   (b) Explain why any disjunction (like $x_1 \vee \bar{x}_2 \vee x_3$) can be written as a decision list.
   (c) Give an example of a decision list that is not a conjunction or a disjunction.
2. [20 pts] **Learning Decision Lists.** Give a PAC algorithm for learning the class of decision lists. This is in the notes, but we want you to describe it and then explain why it is correct in your own words. Give a bound on the sample size and running time needed for finding a rule of error at most ϵ with probability at least $1 - \delta$.
3. [20 pts] **Learning k-Decision Lists.** A "k-decision list" is just like a regular decision list, except the conditions ℓ_i are conjunctions of up to k literals, rather than just being single literals. For example, the function "if $x_1 \wedge \bar{x}_2$ then positive, else if $x_2 \wedge x_3$ then negative, else positive" is a 2-decision list. So a regular decision list is a 1-decision list.
   Give a PAC algorithm for learning k-decision lists whose running time and sample size is polynomial in n^k (and so is polynomial in n when k is a constant). Hint: do it by reduction.
4. [40 pts] **Expressivity of Decision Lists, contd.** Show that decisions lists are a special case of linear threshold functions. That is, any function that can be expressed as a decision list can also be expressed as a linear threshold function "$f(x) = +$ iff $w_1x_1 + \ldots w_nx_n \geq w_0$".
5. [extra credit: 20 pts] **Decision Tree Rank.** The *rank* of a decision tree is defined as follows. If the tree is a single leaf then the rank is 0. Otherwise, let r_L and r_R be the ranks of the left and right subtrees of the root, respectively. If $r_L = r_R$ then the rank of the tree is $r_L + 1$. Otherwise, the rank is the maximum of r_L and r_R.
   Prove that a decision tree with s leaves has rank at most $\log_2(s)$.
6. [extra credit: 20 pts] **Expressivity of Decision Lists, contd., contd.** Show that the class of rank-k decision trees is a subclass of k-decision lists.

Thus, by Problem 3, we conclude that we can learn rank-k decision trees using the hypothesis class of rank-k decision lists in time polynomial in n^k, and using Problem 5 we can learn arbitrary decision trees of size s in time and number of examples $n^{O(\log s)}$. (So this is "almost" a PAC-learning algorithm for decision trees.)