# 15-854 Machine Learning Theory

**Homework # 5** **Due: March 18, 1998**

---

In this assignment we will derive an algorithm for learning decision trees that learns trees of size $s$ in time $n^{O(\log s)}$ in the PAC prediction model (or with $n^{O(\log s)}$ mistakes and time-per-stage in the mistake-bound model). In fact, we will be able to do this with a statistical query algorithm. This matches the lower bound of $n^{\Omega(\log n)}$ for learning size-$n$ decision trees with SQ algorithms that we proved in class.

**Exercises:**

1. A $k$-decision list is a decision list where each rule has as its precondition a conjunction of up to $k$ literals. For instance, the function: "if $x_1\bar{x}_2$ then $+$, else if $\bar{x}_1 x_3$ then $-$, else if $x_4$ then $+$, else $-$" is a 2-decision list.

   (a) Say (in at most one paragraph) why the class of $k$-decision lists can be learned with mistake bound $O(n^{2k})$.

   (b) We showed in class that 1-decision lists can be learned in the statistical query model, where the number of queries and $1/\tau$ are polynomial in $n$ and $1/\epsilon$. Say (in at most one paragraph) why this implies that $k$-decision lists can be learned in the SQ model where the number of queries and $1/\tau$ are polynomial in $n^k$ and $1/\epsilon$.

2. 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 $r_R$. For example, the decision tree in Figure 2.2 in the book has rank 3.

   Prove that a decision tree with $\ell$ leaves has rank at most $\log_2(\ell)$.

**Problems:**

3. Show that the class of rank-$k$ decision trees is a subclass of $k$-decision lists.

   Hint: Use induction on the depth of the tree. (Actually, there are several different ways of proving this.)

Thus, we conclude that we can learn constant rank decision trees in polynomial time, and 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.)