15-854 Machine Learning Theory

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

Solutions:

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 $\bar{x}_1x_3$ 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 $O(n^{2k})$.

   To learn this class, transform the input space by creating a new variable y for each k-tuple of the x_i's. So, the new space has $N = O(n^k)$ variables. A k-decision list over the original space can be viewed as a 1-decision list over the new space, and we know how to learn these with mistake bound $O(N^2) = O(n^{2k})$.

   (b) We showed in class.... 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 $1/\epsilon$.

   The same idea works here too.

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 ℓ leaves has rank at most $\log_2(\ell)$.

   Proof by induction on the depth of the tree. Base case: it's true for a single leaf by definition of "rank". General case: say the given tree has rank r and its left and right subtrees have rank r_L and r_R respectively. By induction, the number of leaves is at least $2^{r_L} + 2^{r_R}$, which is at least 2^r by definition of "rank".

Problems:

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

   Proof by induction on depth. Base case (a single leaf) is true by definition. General case: say the tree has rank k and its root contains variable x_i. We know one of the subtrees has rank at most $k-1$ (without loss of generality, say this is the left subtree) and the other has rank at most k. Inductively, say that L is a $(k-1)$-decision list equivalent to the left subtree and L' is a k-decision list equivalent to the right subtree. To create a k-decision list for the entire tree, begin with the list L but with x_i appended onto each rule, followed by the list L'. Notice that we do not need to append $\bar{x}_i$ onto the rules of L' because every example with $x_i = 0$ exits through one of the rules of L.

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 $n^{O(\log s)}$.