CS 395T Computational Learning Theory

Lecture 2: September 11, 2006 Lecturer: Adam Klivans

Scribe: Raghu Meka

2.1 Learning Conjunctive Decision Lists

Definition 1 *(t-decision Lists) A t-decision list is a decision list where in each box instead of a literal we are allowed to have a conjunction of length at most t.*

Theorem 1 *Any t-decision list can be learnt in the mistake bound model with $n^{O(t)}$ mistakes.*

Proof: Note that there are at most $(2n)^t = n^{O(t)}$ conjunctions of length t over n variables. For each such conjunction create a new variable Y_i. Then the number of these new variables is $n^{O(t)}$. Also, we can assume without loss of generality that no conjunction is repeated in the list. So length of a t-conjunctive decision list is $n^{O(t)}$. We can now use the algorithm for learning decision lists over variables Y_i. Giving us a mistake bound of $O(n^{O(t)} \cdot n^{O(t)}) = n^{O(t)}$. ■

2.2 Learning Decision Trees

We'll first, informally, define what decision trees are. A decision tree is a rooted tree with the following properties:

- Each internal node is labeled with a literal.
- Each internal node has exactly two children. The edges to the children are labeled with distinct labels from $\{0, 1\}$.
- Each leaf is labeled with one of 0 or 1.

On a particular input we start from the root, look at the literal at the root and move along the edge labeled 0 if the literal's value is false, and move along the edge labeled 1 else. We repeat this process untill we get to a leaf node at which point we output the label of the leaf.

Note that decision trees are more powerful than decision lists. In fact, decision trees can compute any boolean function where as decision lists cannot, for instance, compute parity. To see this first note that for any function computed by a decision list, there exists a variable whose value if fixed to one of $\{0, 1\}$ the function's value is fixed. And as there can exist no such variable for parity, parity cannot be computed by a decision list.

The best known algorithm for learning polynomial size (size of a trees is the number of nodes in the tree) decision trees in the mistake bound model has a mistake bound of $n^{O(\log n)}$. We will now see an algorithm, due to A .Blum(91), that achieves this bound. Actually we won't be giving a new algorithm but will reduce the problem of learning decision trees to that of learning decision lists - by showing that every polynomial size decision tree has an equivalent $O(\log n)$-decision list.

We will first need to define the rank of a tree.

Definition 2 *(Rank of a decision tree) Rank of a decision tree T is defined recursively as follows:*

- *If T is a leaf then, $Rank(T) = 0$.*
- *If T is not a leaf and T_1 and T_2 are the left and right sub-trees of T, then*

$$Rank(T) = \begin{cases} max(Rank(T_1), Rank(T_2)) & \text{if } Rank(T_1) \neq Rank(T_2) \\ Rank(T_1) + 1 & \text{if } Rank(T_1) = Rank(T_2) \end{cases}$$

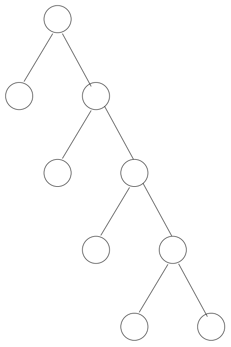
Figure 2.1: Example: The above tree has rank 1.

Lemma 1 *Every decision tree of size s has rank at most $\log s$.*

Proof: We'll prove this by induction on the number of nodes in a decision tree. For a tree with one node the claim is obvious (Rank(leaf) = 0). Now suppose that the lemma is true for all trees with at most $s-1$ nodes. For a tree with more than two nodes let T_1, and T_2 be the left and right sub-trees of the root of T. Suppose that $Rank(T_1) \neq Rank(T_2)$. Since both T_1, T_2 have less than s nodes, by the induction hypothesis $Rank(T_1) \leq \log s$ and $Rank(T_2) \leq \log s$. From which we get that $Rank(T) = max(Rank(T_1), Rank(T_2)) \leq \log s$.

Now suppose that $Rank(T_1) = Rank(T_2)$. Without loss of generality suppose that T_1 has at most $s/2$ nodes. Then, by induction hypothesis and definition of $Rank(T)$ we get that

$$Rank(T) = Rank(T_1) + 1 \leq \log(\frac{s}{2}) + 1 = \log s.$$

■

Theorem 2 *Every decision tree of rank t can be computed by a t-decision list.*

Proof: We'll use induction on the size of the decision tree. If the tree has one node, there's nothing to prove. Now, suppose that for all trees of size less than s and rank k, there is an equivalent k-decision list.

Let T be a tree of size $s > 1$, T_1, T_2 be the sub-trees of the root of T and the literal at the root of T be y. Note that both T_1, T_2 have size strictly less than s. We'll consider two cases:

Case 1: $Rank(T_1) = Rank(T_2) = t - 1$. By the induction hypothesis there exists $(t-1)$-decision lists L_1 and L_2 which are equivalent to T_1, T_2 respectively. Also, we can assume without loss of generality that for all inputs the decision lists L_1 and L_2 produce a output before reaching the end of the lists. Let the lists be as shown in the figure below:

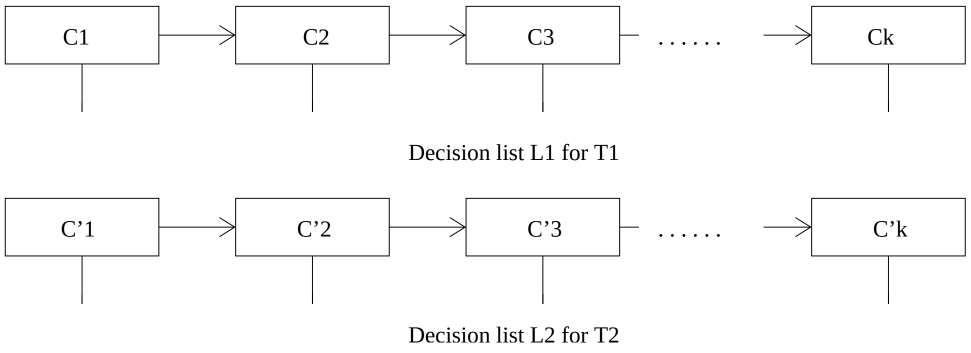

Figure 2.2: Decision lists for T_1, T_2

Then, it can be seen easily that the following is a decision list for T.

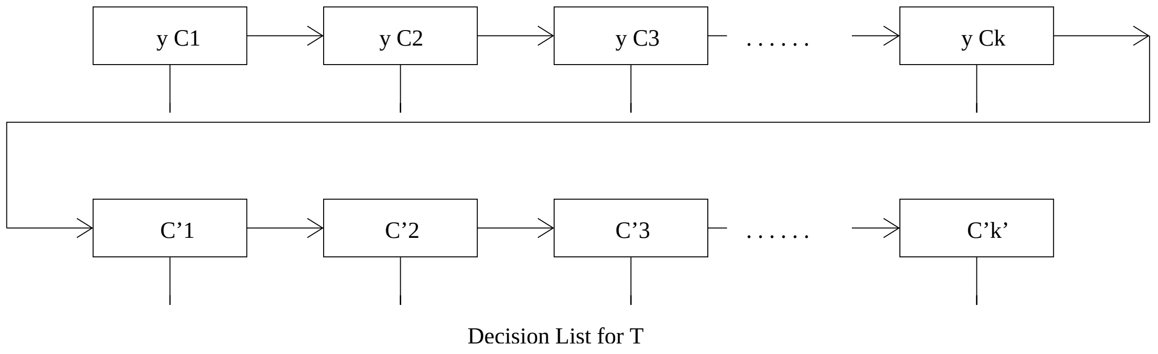

Figure 2.3: Decision list for T

Further, since L_1, L_2 are $(t-1)$-decision lists L above is a t-decision list.

Case 2: $Rank(T_1) \neq Rank(T_2)$. In this case one of T_1, T_2 must have rank strictly less than t. Without loss of generality suppose that $Rank(T_1) < t$ and $Rank(T_2) = t$. Then, there exist decision lists L_1, L_2 equivalent to T_1, T_2 as above, where L_1 is a $(t-1)$-decision list and L_2 is a t-decision list. So that L as constructed above is a t-decision list equivalent to T. This completes the proof of the theorem. ■

Now, by lemma (1), the above theorem and the learning algorihtm for learning decision lists (theorem (1)) we get a learning algorithm for decision trees with a mistake bound of $n^{O(\log s)}$, where s is the size of the tree.

Theorem 3 *A decision tree of size s can be learnt in the mistake bound model with a mistake bound of $n^{O(\log s)}$.*

2.3 Learning DNF formulas

Though both decision trees and DNF formulas can compute all boolean functions, DNF formulas are more powerful complexity wise - i.e., there exist polynomial size DNF formulas which require exponentially large decision trees. In this and the next lecture we will look at the best known learning algorithm for polynomial size DNF formulas in mistake bounded model. This has a mistake bound of $n^{\tilde{O}(n^{1/3})}$ ($\tilde{O}$ means we are ignoring $O(\log)$ factors). As a first step we will show a mistake bound of $n^{\tilde{O}(n^{1/2})}$. We will need the following definition.

Definition 3 *(t-augmented decision trees) A t-augmented decision tree is a decision tree where each leaf instead of being labeled with 0 or 1 is labeled by a DNF formula with term length at most t.*

Lemma 2 *Any t-augmented decision tree of rank k can be computed by a $(k+t)$-decision list.*

Proof: Replace the DNF's in the t-augmented decision tree, T, with new variables $Y_1, Y_2, \ldots$. Then, by theorem (2) we know that T has a k-conjunctive decision list, with some of the variables possibly being Y_i's. It is easy to see that if we now expand the Y's, and break up the resulting boxes, we get a $(k+t)$-decision list. ■

We will contine from here in the next lecture.