CS 395T Computational Complexity of Machine Learning

Lecture 2: January 25, 2005 — Lecturer: Adam Klivans

Scribe: YF Sit

2.1 Polynomial Size Decision Trees and Decision Lists with Constant Term Length

From the last lecture we know that any decision tree of size polynomial in n is a $\log(n)$-decision list. Since any k-decision list on n variables can be learned with a $O(n^k)$ mistake bound, any decision tree of size polynomial in n can be learned with a mistake bound of $O(n^{\log n})$ and a time bound of $O(n^{\log n})$.

If there exists a constant m such that any polynomial size decision tree is a m-decision list, the mistake bound will be improved to $O(n^{\log m})$, i.e. polynomial. However, it can be proved that no such constant exists. Consider the boolean function $f(x_1, x_2, \ldots, x_n)$ that outputs the parity of the first $\log n$ x_i's, where parity is defined to be 1 if the number of 1's in the input bits is even and 0 otherwise. A decision tree for f can be constructed by first picking x_1 as the root node. Then x_2 is used as the root of the two subtrees from x_1. This process continues until $x_{\log m}$ is used. The outputs are then filled out. This forms a complete binary tree and its depth is $\log n$. The number of nodes is therefore some polynomial in n. On the other hand, to built a decision list that computes f, every node in the list has to be a conjunction of the $\log n$ variables $(x_1, x_2, \ldots, x_{\log n})$. Therefore, there does not exist a constant m such that any a polynomial size decision tree has a m-decision list.

2.2 Learning DNF

We will derive the mistake bound of learning DNF by first transforming a DNF into a more general form of decision tree. Then the decision tree is transformed into a decision list, for which we know the mistake bound.

2.2.1 Some Definitions

A Disjunctive Normal Form (DNF) formula is a disjunction of conjunctions: $(x_{11} \wedge x_{12} \wedge \ldots) \vee (x_{21} \wedge x_{22} \wedge \ldots) \vee \ldots \vee (x_{s1} \wedge x_{s2} \wedge \ldots)$. Each conjunction $(x_{11} \wedge x_{12} \ldots \wedge)$ is called a term. The length of a term is the number of literals x_{ij} it contains. A k-DNF is a DNF whose terms have length at most k.

An augmented decision tree is the same as a decision tree except that its leaves are DNF's instead of values. The rank of an augmented decision tree is defined the same way as an ordinary decision tree. The rank of the leaf DNF is 0.

2.2.2 From DNF to Augmented Decision Tree

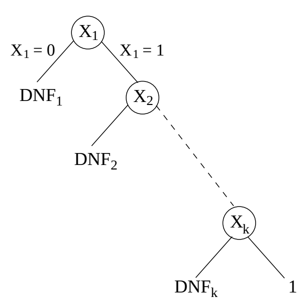
Theorem 1 *Any DNF f of s terms over n boolean variables has an augmented decision tree of rank at most $\frac{2n}{t} \ln s + 1$ in which all the the leaves are t-DNF's, $1 \leq t \leq n$.*

Proof: Let $T_1, T_2, \ldots, T_p$ be the terms of f that have length more than t. There are more than $p \times t$ literals in these terms and we can therefore always find some variable x^* that appears (either negated or non-negated) in at least pt/n of these T_i's. We can construct an augmented decision tree by putting x^* to the root. The left and right subtrees correspond to the conditions $x^* = 0$ and $x^* = 1$, respectively. This process is carried out recursively to expand the subtrees. An expansion is stopped when a t-DNF is obtained. Since a variable x^* is taken out from the DNF's at each subtree, a t-DNF can be obtained eventually and hence an augmented decision tree with t-DNF leaves is constructed.

It remains to show that the augmented decision tree has rank at most $\frac{2n}{t} \ln s + 1$. Let $R(n,p)$ be the maximum rank over all the augmented decision trees constructed by the above algorithm from any DNF on n variables with at least p terms of length more than t. There are at least pt/n terms that contains x^*. Either x^* or $\bar{x^*}$ (or both) appears in at least half of these terms. According to the algorithm, terms that have x^* are put in the same subtree and terms that contains $\bar{x^*}$ will be in the other subtree. One of the subtrees will therefore have at most $p - pt/2n = p(1-t/2n)$ terms of length more than t. By the definition of $R(n,p)$, the rank of this subtree S_0 is $\leq R(n-1, p(1-t/2n))$. The other subtree, S_1, has a rank $\leq R(n-1,p)$. Let the parent of S_0 and S_1 be S. The rank of S is $\leq R(n,p)$. Consider three cases for $R(n,p)$:

- rank of S_0 > rank of S_1: rank of $S \leq R(n,p) \leq R(n-1, p(1-t/2n))$, i.e. $R(n,p) \leq R(n-1, p(1-t/2n))$.
- rank of S_0 < rank of S_1: $R(n,p) \leq R(n-1,p)$.
- rank of S_0 = rank of S_1: $R(n,p) \leq R(n-1, p(1-t/2n)) + 1$.

Since we can construct an augmented decision tree with t-DNF leaves, p will be decreased in the above recursion. We can compute $R(n,p)$ from the above recurrent relation if we know the initial condition $R(m,1), 1 \leq m \leq n$. Assume the term in f that has more than t term is $x_1 \wedge x_2 \wedge x_3 \wedge \ldots \wedge x_k, k > t$, where x_i's are literals (can be positive or negative). We can construct the following augmented decision tree for f in which all the leaves are t-DNF's:



It is easy to see that the rank of this tree is one and hence $R(m,1) = 1$. Solving the recurrence relation, we have

$$R(n,p) \leq R(m,1) + \frac{-\ln p}{\ln(1-\frac{t}{2n})}$$

Since $\ln(1-x) = -(x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots)$,

$$\begin{aligned} R(n,p) &\leq 1 + \frac{\ln p}{t/2n + \frac{(t/2n)^2}{2} + \ldots} \\ &\leq 1 + \frac{\ln p}{t/2n} \\ &= (2n/t)\ln p + 1 \end{aligned}$$

Since $p \leq s$, $R(n,p) \leq (2n/t)\ln s + 1$. ■

2.2.3 From Augmented Decision Tree to Decision List

We can extend our notion of k-decision list so that each output is a DNF instead of 0 or 1. A k-decision can therefore be denoted as $(T_1, f_1), (T_2, f_2), \ldots (T_m, f_m)$, where each T_i is a term of length at most k and each f_i is a DNF.

Since the definition of rank for an augmented decision tree is the same as a decision tree, for an augmented decision tree T of rank R with t-DNF at its leaves, there is a R-decision list that is equivalent to T. Let D_{ij} be the jth term of the DNF f_i. We can convert the R-decision list of T into an $(R+t)$-decision list that has outputs 0 or 1 only by replacing each (T_i, f_i) with $(T_i \wedge D_{i1}, 1), (T_i \wedge D_{i2}, 1), \ldots, (T_i \wedge D_{ij}, 1)$.

For a DNF of s terms over n variables, the rank of the decision list is therefore at most $(2n/t)\ln s + 1 + t$, which attains its minimum when $t = \sqrt{2n \ln s}$. This gives us a mistake bound of $O(n^{\sqrt{n}})$ for a fixed s.