CS 395T Computational Learning Theory

Lecture 3: September 13, 2006 Lecturer: Adam Klivans

Scribe: Marc Alban

3.1 Learing DNF Formuals

Recall from the last lecture the introduction of polynomial sized DNF(Disjunctive Normal Form) formulas and augmented decision trees.

We would like to describe an algorithm for learning DNF formulas with a running time of $n^{\tilde{O}(\sqrt{n})}$ and a mistake bound of $n^{\tilde{O}(\sqrt{n})}$ [2]. Later we would like to find an algorithm for learning DNF formulas with an improved running time and mistake bound of $n^{\tilde{O}(n^{1/3})}$ [5].

Our approach to learning DNF formulas will be to construct a decision list that computes the formula and show that the mistake bound and running time of learning the decision list are both $n^{\tilde{O}(\sqrt{n})}$

3.1.1 DNF Formula to Augmented Decision Tree

Definition 1 *An augmented t-decision tree is a decision tree that has DNF formulas with term length $\leq t$ at the leaves.*

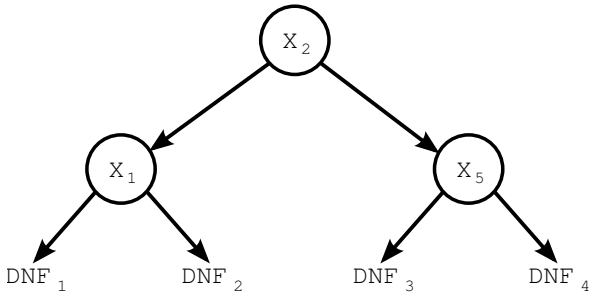

Figure 3.1: Augmented Decision Tree

Theorem 1 *Any polynomial size DNF formula 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 leaves are DNFs with term length at most t.*

Proof: Let $T_1, T_2, \ldots, T_p$ be the terms of f that have length $> t$. Choose variable $x*$ that occures most often in these terms. We can construct an augumented decision tree by putting $x*$ at the root. The left and right subtrees correspond to the conditions $x* = 0$ and $x* = 1$, respectivly. This process

is carried out recursivly to expand the subtrees. An exression is stopped when a DNF formula with term length at most t is obtained. Since a variable $x*$ is taken out from the DNF formulas at each subtree, a DNF formula with term length at most t will eventually be obtained. The result is an augmented decision tree with DNF formula leaves of term length $\leq t$.

Definition 2 *Let $R(n,p)$ be the maximum rank over all the augmented decision trees contructed by the above algorithm from any DNF formula on n variables with at least p terms of length at least t.*

There are at least $p \times t$ literals in the terms of the DNF, therefore $x*$ must appear in at least $\lceil \frac{p \times t}{2n} \rceil$ of them due to the pigeon hole principal. According to the algorithm, terms with $x*$ are put into one subtree and terms with $\overline{x}*$ are put into the other. One subtree will therefore have $\leq \frac{p-p \times t}{2n} = p \times \left(1 - \frac{t}{2n}\right)$ terms of length $\geq t$. By the definition of $R(n,p)$, the rank of this subtree, S_0, is $\leq R\left(n-1, p\left(1-\frac{t}{2n}\right)\right)$. 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 the following cases:

- $RANK(S_0) > RANK(S_1) \rightarrow R(n,p) \leq R\left(n-1, p\left(\frac{1-t}{2n}\right)\right)$
- $RANK(S_0) < RANK(S_1) \rightarrow R(n,p) \leq R(n-1,p)$
- $RANK(S_0) = RANK(S_1) \rightarrow R(n,p) \leq 1 + R\left(n-1, p\left(\frac{1-t}{2n}\right)\right)$

We can compute $R(n,p)$ from the above recurrent relation given the initial condition $R(m,1)$ where $1 \leq m \leq n$. An augmented decision tree with 1 term of length at least t would look like Figure 3.2.

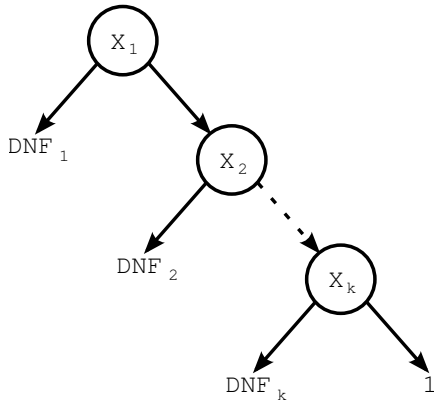

Figure 3.2: Decision tree of DNF with 1 term of length t or more.

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

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

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

$$R(n,p) \leq 1 + \frac{\ln p}{\left(\frac{t}{2n} + \frac{\left(\left(\frac{t}{2n}\right)^2\right)}{2} + \frac{\left(\left(\frac{t}{2n}\right)^3\right)}{3} + \ldots\right)}$$

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

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

Since $p \leq s$, $R(n,p)\, le1 + \left(\frac{2n}{t}\right) \ln s$

■

3.1.2 Augmented Decision Tree to Decision List

We can extend our notion of a k-decision list so that each output is a DNF instead of 0 or 1. A k-decision list 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 $\leq k$ and each f_i is a DNF formula.

For an augmented decision tree T of rank R with DNF formulas with term length at most t for leaves, there is an R-decision list that is its equivalent. The R-decision list of T can be converted into an $(R+t)$-decision list that has outputs 0 or 1 by replacing each (T_i, f_i) with $(T_i \wedge D_{i1}, 1), (T_i \wedge D_{i2}, 1), \ldots$ where D_{ij} is the j^{th} term of f_i.

3.2 Half Spaces

Halfspaces will be used in our task to create a DNF learning algorithm with time and mistake bound $n^{\tilde{O}(n^{1/3})}$. They are useful since almost all learning algorithms can be reduced to learning halfspaces, including those for learning decision trees, DNFs, and decision lists. Half spaces are efficiently learnable in the mistake bound model [3].

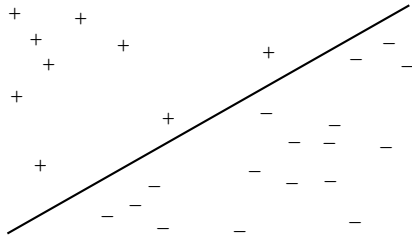

Figure 3.3: A Half Space

Definition 3 *A halfspace is the function* $f(x) = SIGN\left(\sum_{i=1}^{n} a_i x_i - \Theta\right)$ *for* $a_i \in Z$, $\Theta \in Z$, *and* $x \in \{0,1\}^n$

Definition 4 *A linear threshhold function, or LTF, is a halfspace f.*

In a learning scenerio, examples such as $(101101, +))$, $(101100, -))$ and $(100010, +))$ are given and we wish specify a half space which satisfies them. The examples can be written as the following inqualities:

$$\begin{aligned} a_1 + a_3 + a_4 + a_6 &\geq \Theta \\ a_1 + a_3 + a_4 &\leq \Theta \\ a_1 + a_5 &\geq \Theta \end{aligned}$$

and solved using a linear program solver [4] to find the a_i's and Θ's.

3.2.1 Decision List to Half Space

Consider a decision list with n variables that outputs either -1 or +1. A linear threshhold function can be specified that is equivalent to the decision list as follows.

1. Construct a term for the head of the list by multiplying together the output value by the variable at the head by $2n + 1$.
2. Construct a similar term for the child of the head of the list, except that final factor is 2n.
3. Construct a similar term for the child of the child of the head of the list, except that final factor is $2n - 1$. Continue in this fashion until a term is created for all nodes in the decision list.
4. Then add these terms together and add the final output value of the decision list.
5. The linear threshhold function is the sign function of the resulting expression.

As an example, take the decision list as shown in Figure 3.4

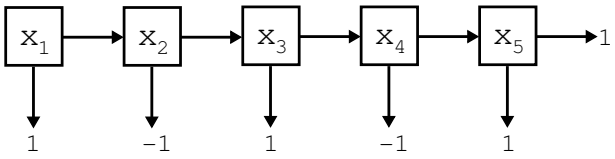

Figure 3.4: Decision List Expressed as an LTF.

The equivalent LTF is $f(\overline{x}) = SIGN\left(2^{5+1} \times X_1 - 2^5 \times X_2 + 2^{5-1} \times X_3 + 2^{5-2} * X_4 + 2^{5-3} \times X_5 + 1\right)$ Note how the factors allow individual variables to dominate in decision list order.

3.3 Polynomial Threshold Functions

Definition 5 *A function $f : \{0,1\}^n \to \{+,-\}$ is computed by a polynomial threshhold function, or PTF, of degree d if there exists a real, multivariate polynomial p of total degree at most d such that $\forall x \in \{0,1\}^n f(x) = SIGN\left(p\left(x\right) - \Theta\right)$ for some Θ.*

(Aside: recall that the total degree of a multivariate polynomial is the maximum degree of any monomial term, where the degree of a monomial term is the sum of the exponents of the variables in that term.)

3.3.1 Polynomial Threshold Function to Linear Threshold Function

PTFs can be learned by converting them into LTFs. Consider a PTF $f = SIGN\left(p\left(x\right)\right)$. Notice that we can express f as a sum over its monomials and their coefficients:

$f = SIGN\left(\sum_{i=1}^{n^{O(d)}} \alpha_i M_i\left(x\right)\right)$

where α_i is the coefficient of a monomial and $M_i\left(x\right)$ is a single monomial in f.

There are $n^{O(d)}$ such distinct monomials in f. (All these monomials are multilinear, i.e. all variables have at most an exponent of 1. Then the count of such monomials is $\sum_{i=0}^{d} \binom{n}{i}$. Induction on d will verify that $\sum_{i=0}^{d} \binom{n}{i} \leq cn^d$ for some constant c. Each monomial M_i can be represented by a new variable y_i resulting in a LTF that can be solved with a mistake bound and running time of $O\left(n^d\right)$.

3.3.2 DNF functions to Polynomial Threshold Functions

Minsky and Papert [5] showed that any polynomial sized DNF formula has a $\tilde{O}\left(n^{\frac{1}{3}}\right)$ PTF. Finding low degree PTFs for even relativly simple DNF formulas can be tricky. Consider the DNF formula $f = x_1x_2x_3 + x_4x_5x_6$. The following degree 3 PTF computes f.

$SIGN\left(2\left(x_1x_2x_3 + x_4x_5x_6\right) - 1\right) > 0$

Here is a slightly more tricky PTF of degree 2 which computes f.

$SIGN\left(3 - x_1 - x_2 - x_3\right)\left(3 - x_4 -_x 5 - x_6\right) - .5 > 0$

In the next lecture we will use the Chebuchev Polynomial [1] to show that a PTF of degree $\sqrt{t}\log s$ will compute a DNF formula with s terms of length t.

References

[1] Arfken. Chebyshev (tschebyscheff) polynomials. In *Mathematical Methods for Physicists*. Academic Press, 3rd edition.

[2] Bshouty. A subexponential exact learning algorithm for dnf using equivalence queries. *Information Processing Letters*, 1996.

[3] Maass and Turan. How fast can a threshold gate learn? *Computational Learning Theory and Natural Learning Systems*, 1994.

[4] Megiddo. Linear programming. In *Encyclopedia of Microcomputers*. 1991.

[5] Minsky and Papert. *Perceptrons: An introduction to Computational Geometry*. MIT Press, 1969.