CS 395T Computational Learning Theory

Lecture 4: September 12, 2007 — Lecturer: Adam Klivans

Scribe: Michael Glass

4.1 DNF Formulas

Our goal is to learn polynomial sized DNF formulas with runtime and mistake bound $2^{\tilde{\mathcal{O}}(\sqrt{n})}$.

4.1.1 The Algorithm

Use the demotion algorithm to learn t-decision lists where $t = 2\sqrt{n}\log s + 1$.

4.1.2 Runtime and Mistake Bound

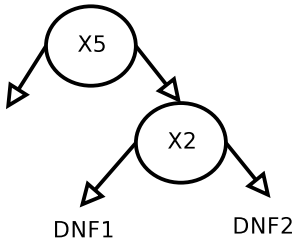
The runtime and mistake bound follows easily since the demotion algorithm for decision lists has time and mistake bound n^t.

$$
\begin{aligned}
n^t &= n^{2\sqrt{n}\log s+1} \\
&= n^{\mathcal{O}(\sqrt{n}\log s)} \\
&= 2^{\tilde{\mathcal{O}}(\sqrt{n}\log s)} && \text{Since } \tilde{\mathcal{O}} \text{ hides log factors} \\
&= 2^{\tilde{\mathcal{O}}(\sqrt{n})} && \text{Since the term length is polynomial in } n
\end{aligned}
$$

4.1.3 Correctness

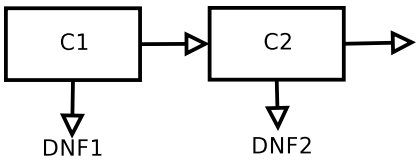
The correctness of this algorithm is less clear. We will demonstrate that for any DNF with n variables and s terms there is a t-decision list that computes the same function where $t = 2\sqrt{n}\log s + 1$. We begin by introducing the concepts of augmented decision trees and augmented decision lists.

Definition 1 *An augmented decision tree is a decision tree with DNF formulas at the leaves.*



When evaluating a bit string it proceeds as a normal decision tree until reaching a leaf. Then it evaluates the DNF formula on the bit string and returns the result.

Definition 2 *An augmented t-decision list is a t-decision list with DNF formulas at the output of a basic block.*



As in an augmented decision tree, an augmented t-decision list proceeds as normal but on output provides the result of computing the DNF rather than a zero or one.

Recall from previous lectures that we can transform a decision tree into a decision list. The same procedure can be applied to transform an augmented decision tree into an augmented decision list.

Lemma 1 *An augmented decision tree of rank r can be transformed into a augmented r-decision list*

We now prove that an augmented t-decision list with DNF formulas containing terms of length no greater than l can be transformed into an equivalent $t + l$-decision list.

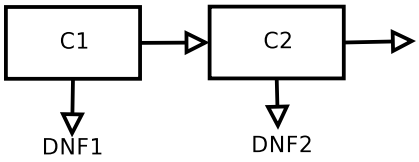
Lemma 2 *For every augmented t-decision list with term length l there is an equivalent $t+l$-decision list*

Consider a augmented t-decision list. Every basic block has a conjuction (call it C_i) of length at most t and a DNF formula at its output with each conjunctive term of length at most l (call them T_{ij}). For each basic block in the augmented decision list we will create a $t + l$-decision list. We will then append these decision lists together to make a $t + l$-decision list equivalent to the augmented decision list.

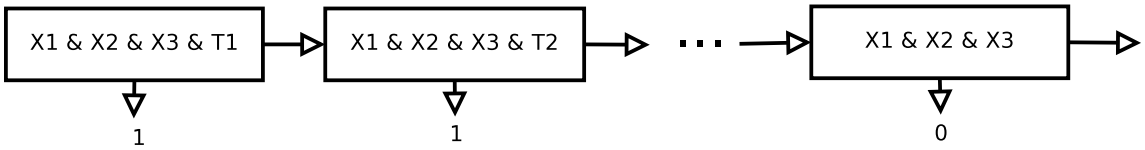
For each basic block we create a decision list as follows. For each term in the DNF of the basic block we create a new basic block. The conjunction in each basic block will be the conjuction of C_i and T_{ij} and the output will be 1. This decision list will output a 1 in exactly those cases where the original basic block would have output a 1. The original *ith* basic block would output a 1 if and only if C_i is true and for some j, T_{ij} is true. This same requirement holds in our $t + l$-decision list.

The concatenation of all such decision lists will compute the same function as the original augmented decision list.

Illustration: Given the augmented decision list:



Suppose $C_1 = X_1 \wedge X_2 \wedge X_3$ and $DNF_1 = T_1 \vee T_2 \vee T_3$ The resulting $t + l$-decision list for the first basic block:



So all that remains is to show an equivalence between DNF formulas and augmented decision trees.

Lemma 3 *For any s-term DNF formula and for any $t \geq 1$ there is an equivalent augmented decision tree of rank $\leq \frac{2n}{t} \log s + 1$ with DNF formulas of term length $\leq t$ at the leaves*

We prove this by demonstrating how to construct an augmented decision tree from a DNF formula.

First consider the base case. If all terms in the DNF formula are of length $\leq t$ we are done; place the DNF formula at the leaf. If exactly one term in the DNF formula has length $> t$ create a long branch

Otherwise, find the most common literal among terms of length $> t$. We choose this literal as the root of our augmented decison tree. On the branch of the decision tree where this literal is set to zero, we eliminate all terms that contained it. On the other branch we eliminate the terms that contain the negation of the literal and eliminate the literal from terms that contain it.

Definition 3 *$R(n, p)$ is the maximum rank produced by running this algorithm on a DNF with n variables and p terms with length $> t$*

Lemma 4 $R(n, p) \leq 2n/t \log p + 1$

Note that $R(n, 0) = 0$ since this corresponds to our base case.

Although more subtle, note also that $R(n, 1) = 1$. This will result in a tree that has a long arm with null DNFs (zeros) on the zero branch of the arm and eventually a leaf with a DNF on the one side. Such a tree has rank 1.

Recall that in the zero branch of the decision tree we eliminated all terms that contained the most common literal. There are n variables which can each be negated or not, giving $2n$ distinct literals. With p terms of length greater than t there are at least pt total literals in these terms. The most common literal therefore occurs at least $\frac{pt}{2n}$ times. And therefore at least $\frac{pt}{2n}$ terms will be eliminated from the zero branch.

Additionally both branches will have one fewer variable.

Recall that the rank of a tree is either the maximum rank of its subtrees or, if its subtrees are of equal rank, the rank of the tree is one greater than the rank of its subtrees.

So there are three cases to consider, the rank of the one subtree is greater or the rank of the zero subtree is greater or they have equal rank.

$$\begin{aligned} R(n,p) &= R(n-1,p) \\ &\text{or} \\ R(n,p) &= R(n-1,p-\frac{pt}{2n}) \\ &\text{or} \\ R(n,p) &= 1+R(n-1,p-\frac{pt}{2n}) \end{aligned}$$

In the first two cases the rank doesn't increase since we are trying to show an upper bound on the rank we will examine only the third case. Each time the rank increases by 1 the number of terms with length $> t$ decreases by $\frac{pt}{2n}$. So each time p is being multiplied by $1-\frac{t}{2n}$. In order to get to our base case of $p=1$ this multiplication will need to occur $\log_{1-\frac{t}{2n}} 1/p$ times. The base case where $p=1$ gives us a rank of 1. Therefore $R(n,p) = 1+\log_{1-\frac{t}{2n}} 1/p$.

$$\begin{aligned} R(n,p) &= 1+\log_{1-\frac{t}{2n}} 1/p \\ &= 1-\log_{1-\frac{t}{2n}} p \\ &= 1-\frac{\log p}{\log(1-\frac{t}{2n})} \\ &= 1-\frac{\log p}{-(\frac{t}{2n}+(\frac{t}{2n})^2/2)} \\ &= 1+\frac{\log p}{\frac{t}{2n}+(\frac{t}{2n})^2/2} \\ &\leq 1+\frac{\log p}{\frac{t}{2n}} \\ &= 1+\frac{2n}{t}\log p \\ &\leq \frac{2n}{t}\log s+1 \end{aligned}$$

Since we can construct a $r = \frac{2n}{t}\log s+1$ rank augmented decision tree with DNF formulas with term length t at the leaves, we can construct an augmented r-decision list with DNF formulas of the same term length. And since we can construct such an augmented r-decision list, we can construct

a $r+t$-decision list. We now choose t to give the required bound. Let $t = 2\sqrt{n}$.

$$\begin{aligned} r + t &= \frac{2n}{t}\log s + 1 + t \\ &= \frac{2n}{2\sqrt{n}}\log s + 1 + 2\sqrt{n} \\ &= \sqrt{n}\log s + 1 + 2\sqrt{n} \\ &\leq 2\sqrt{n}\log s + 1 \qquad \text{Assuming } s \geq 4 \end{aligned}$$

4.2 Linear Threshold Functions

Definition 4 *A linear threshold function is a binary function f of the form $f = SIGN(\sum_{i=1}^{n} \alpha_i X_i - \theta)$ where $\alpha_i \in \mathbb{R}$ and $X_i \in \{0,1\}$*

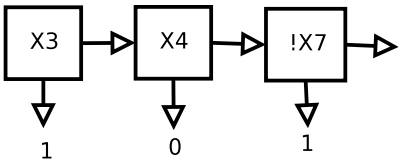
LTFs define half spaces in n dimensional space. LTFs are efficiently learnable in the Mistake Bound model. The proof is a refinement of the proof that they are efficiently learnable in the PAC model. Consider a bit string and label $(0110110, 1)$ from this we know that $\alpha_2 + \alpha_3 + \alpha_5 + \alpha_6 \geq \theta$. Each example presented will give another such inequality. These inequalities can be solved efficiently with linear programming.

The fact that LTFs are efficiently learnable also implies that decision lists are efficiently learnable.

Lemma 5 *For any decision list we can construct an equivalent LTF.*

Without loss of generality consider a decision list with n basic blocks. Index them by position p from the head to tail such that the index of the head is 0 and the tail is n. Let the literal in the pth basic block be L_p and the output be O_p (either 0 or 1). For each variable X_i in position p set $\alpha_i = (2O_p - 1)2^{n-p}$. If the literal is negated then this coefficient is multiplied by $(1 - X_i)$ rather than X_i. Set $\theta = 0.5$.

Example:



$$2^n X_3 - 2^{n-1} X_4 + 2^{n-2}(1 - X_7) + ... - 0.5$$

4.3 Polynomial Threshold Functions

Definition 5 *A polynomial threshold function is a total degree d polynomial in n variables with real coefficients.* $f(x) = SIGN(P(x) - \theta)$

Polynomial threshold functions can be learned by using variable subsitution to map them to linear threshold functions. Let each monomial be named Y_i there are n^d such monomials. So by learning a LTF with n^d variables we can learn a PTF with n variables and degree d.