COMS 6998-2 Advanced Topics in Computational Learning Theory **Spring 2005**

Lecture 3 : February 3, 2005

Lecturer: Rocco A. Servedio *Scribe: Aaron Roth*

1 Today's Lecture

In this lecture we cover the following:

- Finish a $2^{\sqrt{n \log n \log s}}$ time algorithm for learning DNFs
- Learning Linear Threshold Functions (LTFs) including the perceptron, Winnow, and MT algorithms
- Polynomial Threshold Functions (PTFs) including a lower bound.

2 Learning DNFs

Theorem 1 *Any n-decision list (DL) of length s is expressible as a $O(\sqrt{n \log n \log s})$-DL.*

Lemma 1 *Let $h = O(\sqrt{n \log n \log s})$. Any n-DL of length s is equivalent to a decision tree (DT) with the following properties:*

1. *Internal nodes are variables $x_1, \ldots, x_n$*
2. *Each leaf is an h-DL*
3. *The tree has at most 2^h leaves*

Proof: Let f be a decision list. Say that a term T_i in f is *long* if it has more than h literals. Let $t =$ the number of long terms in the DL. By definition, there must be at least th occurances of literals in all long terms, and so by the pigeonhole principle, some literal occurs in at least $\lceil \frac{th}{2n} \rceil$ terms (Since there are only $2n$ literals). Let x_i be the variable that this literal refers to, and place x_i at the root of a DT. Place as its children the DL's resulting from setting the value of x_i to 1 or 0. That is, the two children of x_i will be $f|_{x_i \leftarrow 0}$ and $f|_{x_i \leftarrow 1}$. Recursively repeat this process on $f|_{x_i \leftarrow 0}$ and $f|_{x_i \leftarrow 1}$, halting when the resulting DL has no long terms. This by definition yields a DT satisfying the first two conditions of the lemma (since, with no more long terms, the leaves are h-DLs). It remains to be shown that it also satisfies the third:

For the upcoming analysis, note that for a literal included in a term, if it is set to 1, the term shrinks by one, and if it is set to 0, the term vanishes.

In each of $f|_{x_i \leftarrow 0}$ and $f|_{x_i \leftarrow 1}$ the number of variables is $n-1$, and in one of these, the number of long terms decreases by at least $\lceil \frac{th}{2n} \rceil$ (since this many terms have vanished when the selected literal is set to 0). Let $\phi(n,t)$ be the number of leaves in any DT our recursive procedure generates from a DL on n variables with at most t long terms. We have: $\phi(n,t) \leq \phi(n-1,t) + \phi(n-1, t(1-\frac{h}{2n}))$, $\phi(1,*) \leq 0$ and $\phi(*,0) = 1$. Thus:

$$\phi(n,t) \leq \phi(n-1,t) + \phi(n-1, t(1-\frac{h}{2n}))$$

$$\leq \phi(n-1, t(1-\frac{h}{2n})) + \phi(n-2,t) + \phi(n-2, t(1-\frac{h}{2n}))$$

$$\vdots$$

$$\leq \phi(n-1, t(1-\frac{h}{2n})) + \ldots + \phi(1, t(1-\frac{h}{2n}))$$

each time expanding the $\phi(n,t)$ term. There are $n-1$ terms in the last inequality, but we can assume even more (and ignore the fact that each term is smaller than the last) with the following very loosely bounded inequality:

$$\phi(n,t) \leq n \cdot \phi(n-1, t(1-\frac{h}{2n})) \leq n \cdot \phi(n, t(1-\frac{h}{2n}))$$

$$\phi(n,t) \leq n^k \cdot \phi(n, t(1-\frac{h}{2n})^k)$$

Take $k = \frac{2n}{h} \cdot \ln(t)$ motivated by the fact that $(1-\frac{1}{x})^x \leq \frac{1}{e}$. Then,

$$\phi(n,t) \leq n^{\frac{2n}{h}\ln(t)} \cdot \phi(n,0) = n^{\frac{2n}{h}\cdot\ln(t)}$$

Recall that $t \leq s$ and so the total number of leaves in the tree is less than $n^{\frac{2n}{h}\ln(s)} = 2^h$ by our choice of h, demonstrating (3) and completing the proof. ■

Lemma 2 *Any DT as described in the previous lemma is equivalent to some $2h$-DL*

Proof: By Blum's proof converting DT's to $\log(s)$-DLs, we know that our 2^h leaf DT from the lemma above must have a leaf with a path with length at most h. Let C be the conjunction corresponding to the path of the shallow leaf, and let $T_1, T_2, \ldots, T_s$ be the terms in the DL at the leaf that path C leads to. Construct decision list L with terms $C \wedge T_1, C \wedge T_2, \ldots C \wedge T_s$, and with final term T' where T' is the reduced decision tree as in Blum's earlier proof. This computes the same function as the original DT, and so we can simply recurse on T'. Note that since our original DT has leaves that are DL's with no long terms, each T_i has length at most h, and similarly C has length at most h, so we have constructed a $2h$-DL. ■

These two lemmas complete the proof of the theorem.

3 Learning Linear Threshold Functions (LTFs)

Low weight LTFs are "easy" to learn:

Algorithm 1 (Perceptron Algorithm) *This algorithm is more fully discussed in the reading:*
The algorithm maintains a weight vector $\vec{v}$, a threshold τ, and predicts using the boolean function $\vec{v} \cdot \vec{x} \geq \tau$ For a false positive, the algorithm updates:

$$\vec{v} \leftarrow \vec{v} - \vec{x}$$

$$\tau \leftarrow \tau + 1$$

For a false negative, the algorithm updates similarly, but with addition and subtraction switched.

Example: If we have a perceptron hypothesis hyp $= 3x_1 + 4x_2 + x_3 \geq 6$, and a data point $(1,0,1)$ which is a positive example, we calculate hyp $= 3 + 1 \leq 6$, producing a false negative. The new hypothesis will be hyp $= 4x_1 - 4x_2 + 2x_3 \leq 5$.

Theorem 2 (Perceptron Convergence Theorem) *If the perceptron algorithm is run on a sequence of examples labelled according to some weight-w LTF (each example in $\{0,1\}^n$) then the mistake bound is $O(n \cdot w^2)$*

Thus, we can learn weight-w LTFs in poly(n,w) time.

Algorithm 2 (Winnow Algorithm) *Like a "Multiplicative Perceptron"*
The initial hypothesis hyp $= \overrightarrow{w} \cdot \overrightarrow{x} \geq \theta$, *with* $w = (1, \ldots, 1)$. *There is also an update parameter* $\alpha > 1$.

If $\overrightarrow{w} \cdot \overrightarrow{x} \geq \theta$ *but the true label is 0, for each* x_i *such that* $x_i = 1$ *set* $w_i \leftarrow \frac{w_i}{\alpha}$. *Similarly, if* $\overrightarrow{w} \cdot \overrightarrow{x} < \theta$ *but the label is 1, then for each* x_i *such that* $x_i = 1$ *set* $w_i \leftarrow w \cdot \alpha$.

Theorem 3 (Main Theorem) *We can use Winnow to learn any weight-w LTF over* $\{0,1\}^n$ *with a poly*$(w) \cdot \log n$ *mistake bound, and runtime = poly*(n, w).

Theorem 4 (MT) *There is an algorithm to learn* any *LTF over* $\{0,1\}^n$ *in poly*(n) *time, with mistake bound* $\approx O(n^2 \log n)$ *and time per step poly*(n).

Details about this algorithm are in the reading. In general, it works by considering the coordinate space of each data point $(w_1, w_2, \cdots, w_n, \theta)$ where w_i's are LTF coefficients and θ is a threshold. Each example provides a half space, so the set of all m examples received thus far defines a convex region that is an intersection of m halfspaces. By making sure that the current hypothesis vector is "close to the center" of this convex region, the algorithm ensures that when ever an error occurs (a half space is encountered inconsistent with the hypothesis), the convex region decreases substantially (intuitively by some multiplicative factor less than 1), and thus the space of possible hypotheses decreases exponentially with the number of mistakes. It is analogous to an n-dimensional binary search.

4 Polynomial Threshold Functions (PTFs)

Definition 1 *Let* $f : \{0,1\}^n \rightarrow \{-1,1\}$ *be any boolean function. A polynomial* $p(x_1, \ldots, x_n)$ *is a* polynomial threshold function *(PTF) for f if for all* $x \epsilon \{0,1\}^n$, *sign*$(p(\overrightarrow{x})) = f(x)$. *f has a* degree-d PTF *if the smallest degree polynomial as above has degree d.*

Example: $f = x_1 x_2 \vee x_3 x_4$, and $p(x) = x_1 \cdot x_2 + x_3 \cdot x_4 - \frac{1}{2}$.

As in the above example, any k-DNF has a degree-k PTF.

Additionally, we can consider only *multilinear* polynomials, since $0^n = 0$ and $1^n = 1$ for all n.

Note that we can view degree-d PTF's as LTF's over $\sum_{i=1}^{d} \binom{n}{i} \approx n^d$ "variables" (monomials of degree at most d)

Corollary 1 *The MT algorithm can learn any degree-d PTF in $n^{O(d)}$ time.*

This will be our main learning algorithm for DNF formulas and intersections of halfspaces. It remains to show that the algorithm actually works, i.e. that DNF formulas and intersections of halfspaces actually have low-degree PTFs. First we'll look at some lower bounds to establish limits of the technique.

4.1 Lower Degree Bounds on PTF's

Claim 1 *Any $f : \{0,1\}^n \to \{-1,1\}$ has a PTF of degree $\leq n$*

Proof: View f as an n-DNF.

$$f(x) = \bigvee_{z:f(x)=1} T_z$$

where T_z is a length n term that is true on exactly the input z. For example, for $n = 5$, $z = 11001$, $T_z = x_1 x_2 \overline{x_3 x_4} x_5$.

$$P(x) = \sum_{z:f(z)=1} M_z(x)$$

where for example, $M_{11001} = x_1 x_2 (1 - x_3)(1 - x_4) x_5$ We thus have:

$$f(x) = 1 \Leftrightarrow p(x) = 1$$

$$f(x) = -1 \Leftrightarrow p(x) = 0$$

Thus, $2p(x) - 1 = f(x)\ \forall x \epsilon \{0,1\}^n$ ■

History Digression: PTFs were studied by Minsky/Papert in their '68 book. They proved the following:

1. Parity$(x_1, \ldots, x_n)$ has no PTF of degree less than n.

2. There is an $n^{\frac{1}{3}}$-term DNF that has no PTF of degree $< n^{\frac{1}{3}}$.
3. There is an intersection of half spaces $h_2(x) \cap h_1(x)$ that has no $\theta(1)$-degree PTF.

Recall that $\mathrm{PAR}(x_1, \ldots, x_n) = (-1)^{x_1+\ldots+x_n}, x_i \epsilon \{0,1\}^n$.

Theorem 5 (MP) *Any PTF $p(\overrightarrow{x})$ for parity must have degree at least n.*

Proof: Let $p(\overrightarrow{x})$ be a PTF for PAR, and let $d = \deg(p)$. Let π be a permutation on $\{1, 2, \ldots, n\}$. If for some x we have $p(x_1, \ldots, x_n) > 0$ then $p(x_{\pi(1)}, \ldots, x_{\pi(n)}) > 0$. Thus, for any $x \epsilon \{0,1\}^n$

$$Q(\overrightarrow{x}) = \sum_{\text{all permutations } \pi} p(x_{\pi(1)}, \ldots, x_{\pi(n)})$$

$$Q(\overrightarrow{x}) = \mathrm{sign}(\mathrm{PAR}(X))$$

i.e. $Q(\overrightarrow{x})$ is also a degree-d PTF for PAR.

Moreover, $Q(\overrightarrow{x})$ is a symmetric multilinear polynomial. i.e., every term in the polynomial of equal degree have coefficients equal to one another.

To be continued... $\blacksquare$