CS 395T Computational Complexity of Machine Learning

Lecture 3: January 27, 2005 Lecturer: Adam Klivans

Scribe: Kevin Liu

3.1 Decision Trees Are Less Expressive Than DNFs

3.1.1 Recap

Recall the discussion in the last lecture that formally introduced DNFs and a solution due to Bshouty 1995 that gave an algorithm that learned DNFs in n variables with size polynomial in n. The algorithm had running time $n^{\tilde{O}(\sqrt{n})}$ and mistake bound $n^{\tilde{O}(\sqrt{n})}$. (Recall that the notation $\tilde{O}(f(n))$ is equivalent to the traditional asymptotic notation of $O(f(n))$ while hiding log factors.)

We would like to find an algorithm for learning DNFs with an improved running time $n^{\tilde{O}(n^{\frac{1}{3}})}$ and running time $n^{\tilde{O}(n^{\frac{1}{3}})}$. It remains an open research question whether or not there exists a polynomial time algorithm for learning DNFs.

3.1.2 Decision Trees Can Be Expressed As DNFs

A polynomial size decision tree can be computed by a polynomial size DNF as folllows. Take each satisfying path (path with leaf value 1) from root to leaf in the decision tree and form a term in the DNF with the conjunction of all variables on that path. Finally taking the disjunction of all such terms gives an equivalent DNF.

Consider the example in Figure **??**.

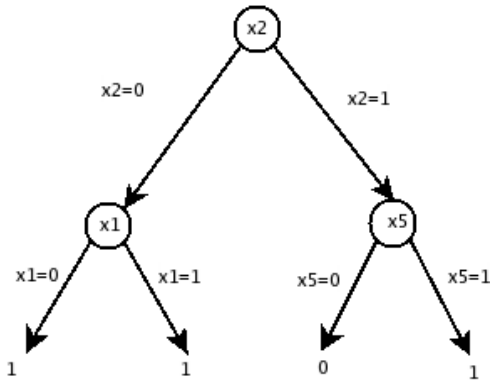

Figure 3.1: Example decision tree.

There exists an equivalent DNF of 3 variables, 3 terms, and a term length of 2, which is $\overline{x_2 x_1} + \overline{x_2} x_1 + x_2 x_5$.

3.2 Learning Halfspaces

3.2.1 Halfspaces Defined

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

Definition 1 *A* HALFSPACE *is the function* $f(\bar{x}) = \mathsf{SIGN}(\sum_{i=1}^{n} a_i x_i - \Theta)$ *for* $a_1, a_2, \ldots, a_n \in \mathbf{Z}$, $\Theta \in \mathbf{Z}$, *and* $\bar{x} \in \{0,1\}^n$.

$f(\bar{x})$ evaluates to either $+$ or $-$.

Definition 2 *A* LINEAR THRESHHOLD FUNCTION, *or LTF, is a halfspace* f.

3.2.2 A Halfspace Learning Scenario

We see examples of the form $\langle 0110001, + \rangle, \ldots, \langle 000111011, - \rangle$

We wish to specify a halfspace that classifies these examples accordingly, producing the sketch in Figure **??**.

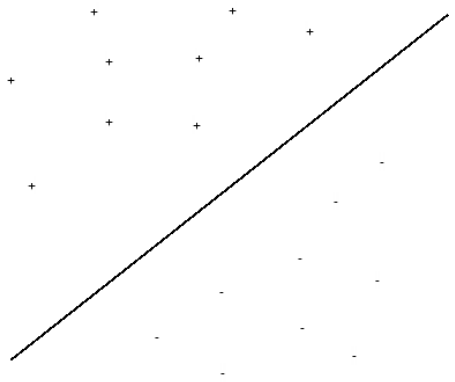

Figure 3.2: Halfspace sketch.

3.2.3 An Algorithm For Learning Halfspaces

Maass and Turan 1990 give a polynomial time algorithm with a polynomial mistake bound for learning halfspaces. Their result is not discussed here; the algorithm would make a good paper presentation.

3.2.4 Another Algorithm For Learning Halfspaces

Instead we give a PAC learning model algorithm for learning halfspaces, due to Blumer, Ehrenfeucht, Haussler, and Warmuth 1989.

1. Encode each example as a halfspace inequality. To do this, for all i substitute the value for x_i into the expression $\sum_{i=1}^{n} a_i x_i$. Then set this expression as greater than or equal to Θ for positive examples, and less than Θ otherwise.
2. Take $O(\frac{n}{\epsilon})$ examples.
3. Feed the encoded inequalities to an linear programming solver.
4. Get a solution for the a_i terms.

3.2.5 The Halfspace Scenario Revisited

The two specified examples in the previous scenario are encoded into inequalities as follows:

$\langle 0110001, + \rangle$ becomes $a_1 \cdot 0 + a_2 \cdot 1 + a_3 \cdot 1 + a_4 \cdot 0 + a_5 \cdot 0 + a_6 \cdot 0 + a_7 \cdot 1 \geq \Theta$.

Similarly, $\langle 000111011, - \rangle$ becomes $a_4 + a_5 + a_6 + a_8 + a_9 < \Theta$.

3.2.6 Remarks On Another Algorithm For Learning Halfspaces

What if the unknown halfspace has enormous coefficients a_i, e.g. something like $2^{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}}$? (If $a_i = 2^{2^n}$, just outputting a_i would require 2^n bits and the algorithm would not be polynomial time.)

Actually, we can assume for all a_i that $a_i \leq 2^{n \log n}$. The theorem is not proven here, but would make a good paper presentation.

3.2.7 Decision Lists Can Be Expressed As Halfspaces

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 2^{n+1}.
2. Construct a similar term for the child of the head of the list, except that final factor is 2^n.
3. Construct a similar term for the child of the child of the head of the list, except that final factor is 2^{n-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 **??**.

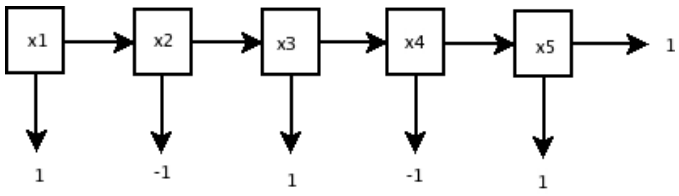

Figure 3.3: Decision tree expressed as LTF example.

The equivalent LTF is $f(\bar{x}) = \mathsf{SIGN}(2^{5+1} \cdot x_1 - 2^5 \cdot x_2 + 2^{5-1} \cdot x_3 - 2^{5-2} \cdot x_4 + 2^{5-3} \cdot x_5 + 1)$.

Note how the factors allow individual variables to dominate in decision list order.

The Maass and Turan mistake bounded model algorithm for learning LTFs can then be applied to learn the original decision lists. Such an application has identical running time and mistake bound to the decision list learning algorithm given two lectures ago, since $\mathsf{poly}(n)$ examples are needed.

3.3 Learning Polynomial Threshhold Functions

3.3.1 Polynomial Threshhold Functions Defined

Definition 3 *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 $\forall x \in \{0,1\}^n$ $f(x) = \mathsf{SIGN}(p(x) - \Theta)$ 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.)

3.3.2 An Algorithm For Learning Polynomial Threshhold Functions/PTFs can Be Expressed As LTFs

We wish to learn polynomial threshhold functions of degree d. This is done in much the same way as the method for learning LTFs above, except that we need to convert the PTF to an equivalent, but larger, LTF.

Consider a PTF $f = \mathsf{SIGN}(p(x))$.

Notice that we can express f as a sum over its monomials and their coefficients:

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

α_i is the coefficient of a monomial and $M_i(\bar{x})$ is a single monomial in f.

There are $n^{O(d)}$ such distinct monomials $M_i(\bar{x})$ 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 $\sum_{i=0}^{d} \binom{n}{i}$. It is easy to verify by induction on d that $\sum_{i=0}^{d} \binom{n}{i} \leq cn^d$ for some constant c, giving the stated bound.)

The following algorithm learns PTF f:

1. For each monomial, introduce a new variable $y_i = M_i(\bar{x})$. This gives the following equivalent expression:

$$f = \mathsf{SIGN}(\sum_{i=1}^{n^{O(d)}} \alpha_i y_i)$$

2. Similarly expand each example into a linear inequality over $n^{O(d)}$ variables.
3. We now have an equivalent LTF learning problem. Apply Maass and Turan's LTF learning algorithm to this problem.

This algorithm has running time $n^{O(d)}$ and mistake bound $n^{O(d)}$.

3.3.3 PTF to LTF Conversion: An Example

Consider the PTF $x_0 + x_1 + x_2 + x_1x_2 + x_0x_1x_2$. A possible LTF that is equivalent would be $y_0 + y_1 + y_2 + y_5 + y_6$, where we have taken $x_0 = y_0, x_1 = y_1, x_2 = y_2, x_0x_1 = y_3, x_0x_2 = y_4, x_1x_2 = y_5, x_0x_1x_2 = y_6$.

Similarly, an example $\langle 110, + \rangle$ might become $\langle 1101000, + \rangle$, using the new variables above.

(Aside: it's useful to think this one through - it's not always possible to turn to the trick of introducing a new set of variables to reduce one learning problem into another.)

3.3.4 PTFs For Different Learning Problems

Question: For a boolean function f, what is the lowest degree PTF computing f?

1. Decision lists $\Longleftrightarrow$ PTFs of degree 1. (Recall the decision list to LTF argument.)
2. Decision trees $\Longleftrightarrow$ PTFs of degree $\lg n$. (Recall the argument in last lecture on the maximum rank of any decision tree, and its use in converting decision trees into equivalent decision lists.)
3. DNFs $\Longleftrightarrow$ PTFs of degree $n^{\frac{1}{3}} \log n$. This is optimal (modulo the $\log n$ factor).

3.3.5 Chebyshev Polynomials

Certain properties of Chebyshev polynomials will be used in the following lemma. For more details on Chebyshev polynomials, please see the appendix.

A Chebyshev polynomial $P_d(x)$ of order d is a univariate polynomial of degree d with the following properties:

$$P_d(1) = 1$$

$$\forall x \in [-1, 1], |P_d(x)| \leq 1$$

$$\forall x \geq 1, P'_d(x) \geq d^2$$

$$P_d(1 + \frac{1}{d}) \geq 2$$

The first and second property indicates that the Chebyshev polynomial may oscillate or have some generally unspecified behavior for domain values in $[-1, 1]$, but the function range is always contained in $[-1, 1]$.

However, outside of domain $[-1, 1]$ the function explodes: the third property above concerning the derivative of $P_d(x)$ suggests this explosive growth.

The fourth property suggests that moving a little bit ($\frac{1}{d}$ distance past 1) along the domain of the Chebyshev polynomial will yield a value of 2.

Consider the plot of five example Chebyshev polynomials in Figure **??**.

In the plot, P_1 is plotted in red, P_2 is plotted in yellow, P_3 is plotted in green, P_4 is plotted in blue, and P_5 is plotted in violet. Notice how all example ploynomials are within $[-1, 1]$ for domain values in $[-1, 1]$, and they all have value 1 at domain value 1. Also notice how all polynomials have value 2 at domain value $1 + \frac{1}{d}$. It can also be seen that the polynomials grow quickly with increasing domain values past 1.

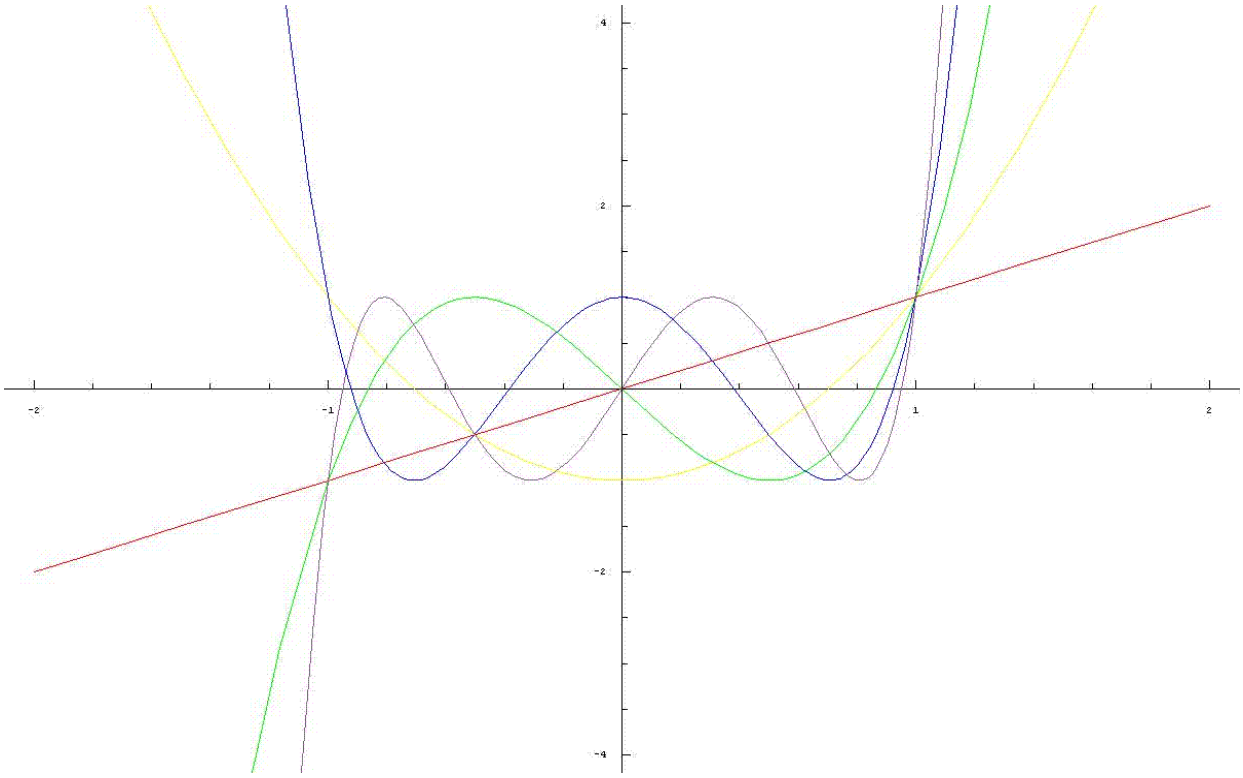

Figure 3.4: Plot of five example Chebyshev polynomials.

3.3.6 DNFs Can Be Expressed as PTFs

Lemma 1 *Any DNF with s terms each of length at most t has a PTF of degree $\sqrt{t}\log s$ computing that DNF.*

Proof: We are given a DNF $D = T_1 \vee \ldots \vee T_s$. For $1 \leq i \leq s$, term $T_i = x_1 \wedge \ldots \wedge x_t$.

Consider the function $s_i = \frac{x_1+\ldots+x_t}{t}$.

(Aside: notice that $s_i \notin \mathbf{Z}$. To deal with this we can proceed until the end where we will have a complicated rational polynomial and we can clear out denominators as desired.)

Now take the Chebyshev polynomial $Q_i = P_{\sqrt{t}}\left((1+\frac{1}{t})(s_i(\bar{x}))\right)$.

Notice that the degree of Q_i is $\sqrt{t}$. (The degrees multiply in the degree 1 argument to the function.)

For all x satisfying T_i, it must be that $Q_i \geq 2$. ($T_i = 1$, implying $S_i = 1$, and so $Q_i = P_{\sqrt{t}}(1+\frac{1}{t})(1) \geq 2$ by the fourth property above.)

For all x not satisfying T_i, it must be that $Q_i \leq 1$. This is since $s_i \leq \frac{t-1}{t}$, implying:

$$Q_i \leq P_{\sqrt{t}}(1 - \frac{1}{t})(1 + \frac{1}{t})$$

$$= P_{\sqrt{t}}(1 - \frac{1}{t^2})$$

$$< P_{\sqrt{t}}(1) = 1$$

Now we apply a similar dominating powers trick as in the LTF to PTF conversion method, using construction Q_i.

Using the above strategy to obtain a Q_i for each term in the DNF, we then consider the following inequality (noting that it can be put into PTF form):

$$Q_1^{\log 2s} + Q_2^{\log 2s} + \ldots + Q_s^{\log 2s} > s$$

Consider x:

For all x satisfying the DNF, there exists a term T_i that is set to 1. WLOG say that second term T_2 is set to 1. Then $Q_2^{\log 2s}$ contributes $+2s$ to the inequality above. ($Q_2 \geq 2$, and so $Q_2^{\log 2s} \geq 2^{\log 2s} = 2s$.) At worst all other terms in the DNF are not satisfied, and for them at worst $Q_i = -1$ since $-1 \leq Q_i(x) \leq 1$ for $x \in [-1, 1]$, and so for this case:

$$Q_1^{\log 2s} + Q_2^{\log 2s} + \ldots + Q_s^{\log 2s} \geq -1 + 2s \ldots - 1$$

$$= \sum_{j=1}^{s-1}(-1) + 2s$$

$$= 2s - s + 1 > s$$

Thus the inequality is satisfied.

For all x not satisfying the DNF, by similar reasoning the total sum is at most s. (For any $x \in [-1, 1]$, $|x^{\log 2s}| \leq 1$. Then $\sum_{j=1}^{s} Q_i^{\log 2s}(x) \leq \sum_{j=1}^{s} 1 \leq s$.)

Finally, the degree of the polynomial is $\sqrt{t}\log(2s)$. (Recall the comment above about the degree of Q_i.)

■

3.3.7 Another Algorithm For Learning DNFs

The above Lemma immediately gives a $\sqrt{n}\lg n$ degree PTF. Following the reasoning throughout the lecture immediately gives a $n^{O(\sqrt{n}\lg n)}$ algorithm for learning DNFs. (Recall the PTF to LTF conversion, and then apply Maass and Turan's result for learning LTFs with their specified time and mistake bounds.)

3.4 Remarks For Next Lecture

Notice that $\sqrt{t}\log s \in O(\sqrt{n}\log n)$ if s is polynomial in n. This matches Bshouty's algorithm for learning DNFs w.r.t running time and mistake bounds, as discussed in last lecture.

Contrast today's DNF learning algorithm and constructive proof with that lecture's discussion. They are completely different! No n appears anywhere in today's lecture.

"With two completely different algorithms (for the same problem), it is usually useful to combine them."

There will also be a problem set next time.

3.5 Appendix A - More About Chebyshev Polynomials

As taken from http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html:

"The Chebyshev polynomials of the first kind are a set of orthogonal polynomials defined as the solutions to the Chebyshev differential equation and denoted $T_n(x)$. They are used as an approximation to a least squares fit, and are a special case of the ultraspherical polynomial with $\alpha = 0$. They are also intimately connected with trigonometric multiple-angle formulas."

The following definitions are paraphrased from the same Mathworld source.

$$T_n(z) = \frac{1}{4\Pi i}\oint \frac{(1-t^2)t^{-n-1}}{1-2tz+t^2}dt$$

where the contour encloses the origin and is traversed in a counterclockwise direction (Arfken 1985, p. 416).

$$T_n(\cos\theta) = \cos(n\theta)$$

$$T_n(x) = \frac{n}{2}\sum_{r=0}^{\lfloor n/2\rfloor}\frac{(-1)^r}{n-r}\binom{n-r}{r}(2x)^{n-2r}$$

$$T_n(x) = \cos(n \cos^{-1} x) = \sum_{m=0}^{\lfloor n/2 \rfloor} \binom{n}{2m} x^{n-2m}(x^2-1)^m$$

$$T_n(x) = 2^{n-1}\Pi_{k=1}^{n}\left\{x - cos\left[\frac{(2k-1)\Pi}{2n}\right]\right\}$$

(Zwillinger 1995, p. 696).

The Chebyshev polynomial can also be obtained from the following generating functions:

$$g_1(t,x) \equiv \frac{1-t^2}{1-2xt+t^2} = T_0(x) + 2\sum_{n=1}^{\infty} T_n(x)t^n$$

and

$$g_2(t,x) \equiv \frac{1-xt}{1-2xt+t^2} = \sum_{n=1}^{\infty} T_n(x)t^n$$

for $|x| \leq 1$ and $|t| < 1$ (Beeler et al. 1972, Item 15).

3.6 Appendix B - References

Eric W. Weisstein. "Chebyshev Polynomial of the First Kind." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html.

Arfken, G. "Chebyshev (Tschebyscheff) Polynomials" and "Chebyshev Polynomials–Numerical Applications." 13.3 and 13.4 in Mathematical Methods for Physicists, 3rd ed. Orlando, FL: Academic Press, pp. 731-748, 1985.

Beeler et al. . Item 15 in Beeler, M.; Gosper, R. W.; and Schroeppel, R. HAKMEM. Cambridge, MA: MIT Artificial Intelligence Laboratory, Memo AIM-239, p. 9, Feb. 1972. http://www.inwap.com/pdp10/hbaker/hakmem/recurrence.html#item15.

Zwillinger, D. (Ed.). CRC Standard Mathematical Tables and Formulae. Boca Raton, FL: CRC Press, 1995.

W. Maass, Gy. Turin (1990): On the complexity of learning from counterexaanples and membership queries, 31. FOCS (1990), 203-210.

N. H. Bshouty. Exact Learning Boolean Functions via the Monotone Theory. Information and Computation, 123:146–153, 1995.

Rosenblatt F 1958 The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65: 386-408.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning, 2:285–318, 1988.

Anselm Blumer , A. Ehrenfeucht , David Haussler , Manfred K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, Journal of the ACM (JACM), v.36 n.4, p.929-965, Oct. 1989

Nimrod Megiddo. Linear Programming. The Encyclopedia of Microcomputers. June 1991.

Steven Rudich. Great Theoretical Ideas in Computer Science. http://www-2.cs.cmu.edu/ 15251/.