

6.1 Overview

In Lecture 5, we showed that for any DNF formula f with n variables and s terms (with s polynomial in n), there exists a polynomial threshold function (PTF) p of degree $n^{\frac{1}{3}} \log s$ which computes f . In Section 6.3, we will show that there exists such a DNF formula that *requires* a polynomial threshold function of degree at least $n^{\frac{1}{3}}$.

In Section 6.4, we will examine an algorithm for learning disjunctions that learns with mistake bound polylogarithmic in the number of variables, if many of those variables are irrelevant to the underlying target function. And, in Section 6.5 we will give an algorithm for learning from expert advice whose mistake bound differs from the best expert's by an additive factor logarithmic in the total number of experts.

We will begin by showing in Section 6.2 that the parity function requires a polynomial threshold function of degree at least n .

6.2 Bounding the degree for a PTF computing PARITY

Recall that $\text{PARITY} : \{0, 1\}^n \mapsto \{0, 1\}$ such that

$$\text{PARITY}(x) = \begin{cases} 1 & \text{if the number of activated bits in } x \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

Claim 1 *Any PTF Q computing PARITY must have degree at least n .*

If Q is a polynomial threshold function computing PARITY, then $Q(x) = \text{sgn}(q(x))^1$ for some degree- d polynomial $q(x)$. This implies that $q(x) \geq 0$ for all x such that $\text{PARITY}(x) = 1$ and $q(x) < 0$ for all x such that $\text{PARITY}(x) = 0$.

We wish to show that $d \geq n$, where n is the length of the bit string input to PARITY. We shall begin by symmetrizing the polynomial $q(x)$.

Definition 1 *A **permutation** π on a set S is a one-to-one mapping from S to itself.*

¹Note that we have dropped the threshold parameter θ from our earlier formulation for convenience; it is assumed that θ is “absorbed” into q to account for this.

Definition 2 A polynomial $p(x_1, x_2, \dots, x_n)$ is said to be **symmetric** if

$$p(x_1, x_2, \dots, x_n) = p(\pi(x_1, x_2, \dots, x_n))$$

for any permutation π .

Informally, a permutation is a rearrangement of elements in a list, and a symmetric polynomial is one which is invariant under rearrangements to its list of inputs. Let $\hat{q}(x) = q(\pi(x_1, x_2, \dots, x_n)) = q(\pi(x))$. Since PARITY is invariant under the ordering of its inputs, it is clear that $\text{sgn}(q(x)) = \text{sgn}(\hat{q}(x))$. Moreover, it is easy to verify that $\text{sgn}(q(x)) = \text{sgn}(q(x) + \hat{q}(x))$. Thus, $\text{sgn}(q(x) + \hat{q}(x))$ is also a polynomial threshold function of degree d computing PARITY, where the degree follows from the construction of the polynomial as the sum of two degree- d polynomials.

Lemma 1 If p is a degree- d symmetric polynomial, there exists a degree- d polynomial p^* such that

$$p(x_1, x_2, \dots, x_n) = p^*(x_1 + x_2 + \dots + x_n).$$

Proof: We will prove Claim 1 using the observation that $\text{sgn}(q(x) + \hat{q}(x))$ is a polynomial threshold function computing PARITY. We extend the idea to all unique permutations, and define q' by

$$q'(x) = \sum_{\pi \in S_n} q(\pi(x)).$$

Since our definition of q' sums over all possible permutations of x , we can see that $q'(x) = q'(\pi(x))$ for any permutation π ; q' is therefore a symmetric polynomial. Furthermore, since q' is defined as a sum of degree- d polynomials, we can say that q' is a degree- d symmetric polynomial.

By Lemma 1, there exists a degree- d polynomial q^* such that

$$q'(x_1, x_2, \dots, x_n) = q^*(x_1 + x_2 + \dots + x_n).$$

Since

$$\begin{aligned} \text{sgn}(q(x_1, x_2, \dots, x_n)) &= \text{sgn}(q(\pi(x_1, x_2, \dots, x_n))) \\ &= \text{sgn}(q'(x_1, x_2, \dots, x_n)) \\ &= \text{sgn}(q^*(x_1 + x_2 + \dots + x_n)), \end{aligned}$$

$\text{sgn}(q^*(x_1 + x_2 + \dots + x_n))$ is a polynomial threshold function computing PARITY.

Upon inspecting the values of q^*

$$\begin{aligned} q^*(0) &< 0 \\ q^*(1) &\geq 0 \\ q^*(2) &< 0 \\ &\vdots \end{aligned}$$

it is clear that q^* has n roots. Therefore, the degree of q^* must be at least n and, by implication, so must the degree of Q .

■

6.3 Existence of a DNF requiring PTF degree $O(n^{\frac{1}{3}})$ [1]

Let the DNF formula f be the “one-in-a-box” function:

$$f = T_1 \vee T_2 \vee \cdots \vee T_m$$

with each T_i defined by

$$T_i = \bigwedge_{j=1}^{4m^2} x_{i,j}.$$

Note that f has $4m^3$ variables.

Claim 2 *Any PTF Q computing f must have degree at least m .*

If Q is a polynomial threshold function computing f , then $Q(x) = \text{sgn}(q(x))$ for some degree- d polynomial $q(x)$. This implies that $q(x) \geq 0$ for all x such that $f(x) = 1$ and $q(x) < 0$ for all x such that $f(x) = 0$.

We wish to prove that $d \geq m$, where m is the total number of disjuncts T_i .

Proof: Let π be a permutation on exactly $4m^2$ variables. Since the conjunctions for each T_i are invariant under permutations of their inputs, $Q(\pi(x_{1,1}, x_{1,2}, \dots, x_{1,4m^2}), x_{2,1}, \dots, x_{m,4m^2})$ is a degree- d polynomial threshold function computing f . Now we can symmetrize Q by defining Q' as

$$Q' = \text{sgn}\left(\sum_{\pi_1, \dots, \pi_m \in S_{4m^2}} Q(\pi_1(X_1), \pi_2(X_2), \dots, \pi_m(X_m))\right)$$

where

$$X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,4m^2}).$$

Note that Q' is a polynomial of degree d , is a polynomial threshold function computing f , and is symmetric within each set of variables X_i .

By Lemma 1, there exists a degree- d polynomial Q^* such that

$$Q'(X_1, X_2, \dots, X_m) = Q^*\left(\sum_{j=1}^{4m^2} x_{1,j}, \sum_{j=1}^{4m^2} x_{2,j}, \dots, \sum_{j=1}^{4m^2} x_{m,j}\right).$$

Note that $Q^*(x) \geq 0$ if and only if $\sum_{j=1}^{4m^2} x_{i,j} = 4m^2$ for some $i \in \{1, \dots, m\}$.

Now consider the following univariate polynomial:

$$Q^*(t) = Q^*(4m^2 - (t-1)^2, 4m^2 - (t-3)^2, 4m^2 - (t-5)^2, \dots, 4m^2 - (t-2m+1)^2).$$

Upon inspecting $Q^*(t)$ for values of $t = 1, 2, \dots$

$$\begin{aligned} Q^*(1) &\geq 0 \\ Q^*(2) &< 0 \\ Q^*(3) &\geq 0 \\ &\vdots \end{aligned}$$

it is clear that Q^* has at least m roots. Therefore, the degree of Q^* must be at least m and, by implication, so must the degree of Q .

■

6.4 An $O(k \log n)$ -mistake bound algorithm for learning disjunctions [2]

Let f be a monotone disjunction² over n variables:

$$f = x_1 \vee x_2 \vee \dots \vee x_n.$$

In Lecture 1, we discussed the Elimination Algorithm, which has a mistake bound of $O(n)$ for learning disjunctions over n variables. Now, we will assume the disjunction has length k , $k \leq n$.

Claim 3 *Any disjunction f over n variables with at most k terms can be learned with mistake bound $O(k \log n)$.*

We will prove that WINNOW guarantees the mistake bound by showing that the number of mistakes for false positives and for false negatives (denoted M_{fp} and M_{fn} , respectively) are both bounded above by $O(k \log n)$.

Proof:

Bounding M_{fn}

Consider any w_i . The value of w_i is doubled in Line 11 of WINNOW only if $w_i < n$, because if we had $w_i \geq n$ (and since from Line 11 we know that $x_i = 1$) the sum in Line 3 would be at least n , in which case $h(x^{(i)})$ would be set to 1 and we could not have a false negative. Since each execution of Line 11 doubles w_i and w_i cannot be doubled if it exceeds n , w_i can be doubled at most $\log n$ times. In the worst case, WINNOW will have to double each w_i independently, resulting in at most $k \log n$ doublings, i.e. $M_{fn} \leq k \log n$.

Bounding M_{fp}

Define the *weight* of the hypothesis at a point in time to be $w = \sum_{j=1}^n w_j$.

Line 9 of WINNOW demotes all w_i for all x_i set to 1 only if the sum in Line 3 is at least n . This implies that the sum of those demoted w_i is at least n and so w decreases by at least n . Note that

²Monotone disjunctions are those disjunctions with no variables negated.

Algorithm 1: Winnow

Input: examples $((x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}), y^{(i)})$, for $i = 1, 2, \dots$ **Output:** weights w_1, w_2, \dots, w_n

```

1 Initialize  $w_i \leftarrow 1 \forall i = 1, \dots, n$ 
2 foreach  $i$  in examples do
    /* calculate hypothesis */
3   if  $\sum_{j=1}^n w_j x_j^{(i)} \geq n$  then
4      $h(x^{(i)}) = 1$ 
5   else
6      $h(x^{(i)}) = 0$ 
7   end
    /* demote weights on false positives */
8   if  $h(x^{(i)}) = 1$  and  $y^{(i)} = 0$  then
9      $w_i \leftarrow 0 \forall i$  such that  $x_i = 1$ 
    /* promote weights on false negatives */
10  else if  $h(x^{(i)}) = 0$  and  $y^{(i)} = 1$  then
11     $w_i \leftarrow 2w_i \forall i$  such that  $x_i = 1$ 
12  end
13 end

```

in the false negative case above, w increases by at most n in Line 11 (this is the case in which the sum falls just short of n).

If we have each demotion step decreasing w by at least n and each promotion increasing it by at most n (and since w is clearly non-negative), we see that the number of demotions cannot exceed the number of promotions: $M_{fp} \leq M_{fn}$. From above we had $M_{fn} \leq k \log n$; therefore $M_{fn} + M_{fp} \leq 2k \log n = O(k \log n)$.

■

6.5 An $O(\text{opt} + \log k)$ algorithm for learning from a heterogeneous pool of experts[3]

We consider a setup in which the learner has access to k experts E_1, E_2, \dots, E_k , each of which outputs $E_i(x) \in \{-1, +1\}$ when given an example input x .

The learner combines the k estimates to form a hypothesis for an example. Denote the mistake bound of the best expert by opt .

Claim 4 *WEIGHTED MAJORITY has mistake bound $O(\text{opt} + \log k)$ when predicting $y = f(x)$ using experts E_1, E_2, \dots, E_k .*

Algorithm 2: Weighted Majority

Input: examples (x_i, y_i) , for $i = 1, 2, \dots$

- 1 experts E_1, E_2, \dots, E_k
- Output:** weights w_1, w_2, \dots, w_k
- 2 Initialize $w_i \leftarrow 1 \forall i = 1, \dots, k$
- 3 **foreach** (x_i, y_i) *in examples* **do**
 - /* calculate hypothesis */
 - 4 **if** $\sum_{j=1}^k w_j E_j(x_i) \geq 0$ **then**
 - 5 $h(x_i) = 1$
 - 6 **else**
 - 7 $h(x_i) = 0$
 - 8 **end**
 - /* demote weights on experts' mistakes */
 - 9 **foreach** *expert* E_j **do**
 - 10 **if** $E_j(x_i) \neq y_i$ **then**
 - 11 $w_j \leftarrow \frac{w_j}{2}$
 - 12 **end**
 - 13 **end**
- 14 **end**

Proof:

Define the *weight* of the hypothesis at a point in time to be $w = \sum_{j=1}^k w_j$ and denote the initial weight $w_0 = k$.

If the learner outputs a mistake, at least half the weight is on experts E_j whose estimate $E_j(x_i) \neq y_i$. Because we halve all w_j corresponding to such E_j (Line 11), it follows that the updated weight after the first mistake $w \leq \frac{3}{4}w_0$. Similarly, after M mistakes $w \leq (\frac{3}{4})^M w_0$.

Because the best expert makes at most opt mistakes, his weight is at least $(\frac{1}{2})^{\text{opt}}$. Moreover, since the best expert's weight is included in w ,

$$\left(\frac{1}{2}\right)^{\text{opt}} \leq w \leq \left(\frac{3}{4}\right)^M w_0.$$

Some algebra reveals that:

$$\left(\frac{4}{3}\right)^M \leq 2^{\text{opt}} w_0$$

$$M \leq \frac{\text{opt} + \log(w_0)}{\log \frac{4}{3}} = O(\text{opt} + \log k),$$

where we've substituted k back for w_0 .

■

References

- [1] M.L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.
- [2] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285-318, 1988.
- [3] N. Littlestone and M.K. Warmuth. The Weighted Majority Algorithm. *IEEE Symposium on Foundations of Computer Science*, 256-261, 1989.