COMPUTATIONAL LEARNING THEORY: ANSWER TO HOMEWORK 1

VISHVAS VASUKI

1

Notation. Boolean function f is k closed := f can be written as both a DNF d and as a CNF c; each of which have max term length k.

Input x is of length n. Algorithm L evaluates f(x). L knows c and d.

Theorem 1.0.1. *$\exists$ L which evaluates f(x) by looking only at $O(k^2)$ variables.*

Proof. Don't know. Hint: Begin evaluating the variables DNF, use this to reduce the number of terms in the CNF and vice versa. $\square$

2

Theorem 2.0.2. *There exists an algorithm L which can learn DNF formalae on n variables with term length t with mistake bound and running time $O(n^t)$.*

Proof. This can easily be solved by using the canonical disjunction learning algorithm after feature expansion. We create new variables representing all possible combinations of n variables. These new variables are $O((2n+1)^t) = O(n^t)$ in number.

We know that disjunctions of $O(n^t)$ variables can be learnt with mistake bound and running time $O(n^t)$ using the algorithm which starts by including all literals in the hypothesis disjunction, and removes all literals evaluated to be true on an example rejected by the target concept. $\square$

3

p(x) and q(x) are polynomials. Rational function $R(x) = \frac{p(x)}{q(x)}$. Degree(R) = max(Degree(p), Degree(q)). Halfspace $f = sign(\sum_{i=1^n} a_i x_i - c) = sign(l_f(x))$, $a_i, c \in Z$. $g(x) = sign(l_g(x))$ is another halfspace. Weight of halfspace = $\sum |a_i| + |c|$. f and g are of weight W. $h = f \wedge g$, the AND of f and g, is a boolean function.

Remark 3.0.3. There exists R(x) of degree $O(l \log t)$:
If $x \in [1, 2^l]$, $R(x) \in (1, 1 + t^{-1})$. If $x \in [-2^l, -1]$, $R(x) \in (-1 - t^{-1}. - 1)$.

$R(x) = \frac{p(x)}{q(x)} > 0$ if and only if $p(x)q(x) > 0$. $R(x) = \frac{p(x)}{q(x)} < 0$ if and only if $p(x)q(x) < 0$.

Theorem 3.0.4. *h is learnable in the mistake bounded model in time and mistake bound $n^{(O(\log W))}$.*

Proof.

Remark 3.0.5. For the rest of the proof, let R(x), p(x), q(x) be degree $O(t \log W)$ functions useful in approximating the sign function of the numbers $|x| \in [1, W]$, with the accuracy $t = 1/10$.

Let p()q() be used in stead of the sign function in f(x) and g(x).

Consider the polynomial $p_h(x) = (f(x)+1)(g(x)+1) - 2^{-1}$. $p_h(x) > 1$ if and only if p()q() shows that both f(x) and g(x) are greater than 1; and $p_h(x) < 1$ if and only if p()q() shows that either f(x) or g(x) or both are lesser than 1.

$degree(f(x)) = degree(p(l_f(x))q(l_f(x))) = O(\log W)$.
$degree(g(x)) = degree(p(l_g(x))q(l_g(x))) = O(\log W)$.

So, $degree(p_h(x)) = O(\log W)$. $sign(p_h(x))$ is our PTF.

We know that PTF's of the degree $O(\log W)$ can be learnt in $n^{O(\log W)}$ time and mistake bound (by reduction to a halfspace learning problem, which can inturn be reduced to a linear programming problem). □

4

An algorithm L in the mistake bounded model is 'careful' if it does nothing after amking a correct prediction. It only updates itself after making a mistake. C is a concept class.

Theorem 4.0.6. *If C is efficiently learnable by algorithm A with mistake bound t, then there exists a careful algorithm L which learns C with mistake bound t.*

Proof. An algorithm A has mistake bound M(c) for learning some concept $c \in C$ if A makes at most M mistakes on any sequence that is consistent with concept c. A has mistake bound $M(C) = max_{c \in C} M(c)$. M(C) = t.

Let h be the hypothesis maintained by A. Let c be the target concept.

Let L be constructed as follows: Whenever L sees an example x, L calls A as a subroutine, returns h(x) and keeps a copy of h. Then L informs A of whether $c(x)? = h(x)$. Then, A may or may not update h to hypothesis h'. If $c(x) = h(x)$, L restores h. If $c(x) \neq h(x)$, L does allows A to update h.

Suppose that L makes a sequence of $m > t$ mistakes on examples $\{x_i\}$. Then, due to the construction of L, this implies that A would have made $m > t$ mistakes on that same sequence of examples. But this would contradict our initial assumption that the mistake bound of A is t. Hence, L cannot make more than t mistakes. Hence L has mistake bound t. □

5

Acknowledgement. The algorithm L' is due to Alex Tang.

Infinite feature model for learning decision lists is like the mistake bounded model, with the following changes: Infinite literals $\{x_i\}$ may appear in a decision list. But, any fixed decision list depends on only k of them. Any example is a list of at most n literals set to 1, the others are 0.

Remark 5.0.7. Hints: "Problem 4 may help you think about problem 5". "This problem has very little to do with decision llists: there is a general transformation that will work for almost any concept class."

Theorem 5.0.8. *If we have an efficient algorithm L, in the mistake bounded model, for learning decision lists of length k over a universe of n literals with mistake bound* *k log n: This implies that decision lists would be learnable in the infinite feature model with mistake bound polynomial in n and polynomial running time.*

Proof.

Notation. Let t(n) be the mistake bound of L. Let N be the set of literals used by L. Only k literals appear in a decision list. Let L' be the algorithm learning decision lists in the infinite feature model.

L' is constructed as follows:

L' uses L and N to respond to label the examples. Initially, N is empty.

When L makes more than $t(|N|)$ mistakes, we know that N does not include all the relevant variables. When L makes more than $t(|N|)$ mistakes, we would have seen atleast 1 relevant literal not already in N. (This is proved in the lemma below.) So, we add the $nt(|N|)$ literals, corresponding to the $t(|N|)$ mistakes, to N. So, N now has $n + nt(n) \leq cnt(n)$ literals, where c is a constant. We restart L with this new set of literals, N.

After each failure, we are adding at least 1 relevant literal. So, after k such iterations, we should have all the relevant literals in N. So, the total mistake bound is $mb = t(n) + t(cnt(n)) + t(cnt(cnt(n)))....$ As t(n) = O(klog n), we have a mistake bound of $O(k^2 \log kn)$ for the algorithm. In general, if t(n) = O(n), we have $mb = O(kn) + O(n^2)..O(n^k) = O(kn^k)$.

As L' uses L to process examples, the computation required per example remains the same. Hence, L' has running time polynomial in n and k.

□

Lemma 5.0.9. *Consider the algorithm L' described above. When L makes more than $t(|N|)$ mistakes, we would have seen atleast 1 relevant literal not already in N.*

Proof.

Notation. Let R be the set of relevent variables in N. Let c be the target decision list. Let c' be a 'truncated' decision list, where every literal not in R is removed from c. As the length of c is at most k, the length of c' is at most k.

Let M be the sequence of 'examples on which L makes mistakes'.

Proof by contradiction. Suppose that L made more than $t(|N|)$ mistakes, but did not see even 1 relevant literal not in R.

We assume without loss of generality that L is a 'careful' algorithm, as defined in the previous section.

So, the labels corresponding to M can be seen as being equal to c'(x). So, L, given a situation where c' is the target concept and the sequence of examples it is presented corresponds to M; would have made more than t(n) mistakes. This contradicts our assumption that L has mistake bound t(n), leading us to absurdity.

□