CS 2416 - Machine Learning Theory

Lecture #12: 24 November 2005

Lecturer: Toniann Pitassi

Scribe Notes by: Toniann Pitassi

Hardness Results

Unconditional hardness results imply $P \neq NP$ so it is very unlikely that we'd get these.

0.1 Hardness based on worst-case complexity assumptions

This kind of hardness result is a reduction. That is, we start with some NP-hard problem, like graph colorability, and show how to reduce an instance of graph colorability to a PAC learning problem so that the instance is a "yes" instance if and only if bla-bla-bla. The advantage of this result is that the assumption is about as weak as possible, without proving $P \neq NP$. In addition, for some very restrictive concept classes (like DNFs and decision trees), the crypto-based hardness results do not apply.

The disadvantage is that this kind of result is much harder to get, and for many classes of interest (again DNF and decision trees), the best result currently known shows that the concept class is hard to learn over a restricted family of hypotheses classes.

To demonstrate this method, we will start with a warmup. We will prove that the class of monotone k-term DNF formulas cannot be proper PAC learned, unless $P = NP$.

We begin with graph colorability. The input is an undirected graph $G = (V, E)$, and a number k, and we want to accept the input if and only if G has a coloring of size k or less. (A coloring is an assignment of colors to vertices such that no two adjacent vertices get the same color. The coloring has size k if the total number of colors used in the coloring is k.) Graph colorability is a well-known NP-complete problem, even when $k = 3$. We will now describe our reduction.

Given an instance $G = (V, E)$, k of the colorability problem, $|V| = n$, we will define a partial function f_G on n underlying variables $x_1, \ldots, x_n$ as follows. For each x_i, let α_i be the truth assignment to $x_1, \ldots, x_n$ that is 0 on x_i and 1 on all other x_j, $j \neq i$. For all α_i, $f_G(\alpha_i) = 1$. For every pair (x_i, x_j) such that there is an edge (i, j) in G, we associate a truth assignment $\beta_{i,j}$ that is 1 on x_i and x_j and 0 elsewhere. For all such $\beta_{i,j}$, $f_G(\beta_{i,j}) = 1$.

Claim There is a k-term DNF consistent with the partial function f_G if and only if G is k-colorable.

Proof First, assume that G is k-colorable. This partitions the vertices of G into k groups, call them $G_1, \ldots, G_k$. For each $i \leq k$, we create a monotone term (associated with G_i). This term consists of the conjunction of all variables x' such that $x' \notin G_i$. Clearly this is a k-disjunction. To see that it is consistent with f_G, first consider any assignment α_i. Suppose that i is in group $G_{i'}$. Then the corresponding term will evaluate to true under α_i, and thus f_G evaluates to true under α_i as desired. Now consider an assignment $\beta_{i,j}$ such that (i, j) is an edge in G. Fix any term t in our disjunction. Since (i, j) is an edge, i and j must be in different partitions, and therefore at least one of them must lie outside of the group associated with t, and therefore t evaluated on $\beta_{i,j}$ will be 0, as desired.

In the other direction, suppose that we have a monotone k-term disjunction consistent with f_G. Then we can associate one equivalence class with each term as above. (For any term t, the set of variables not in t will be the equivalence class corresponding to this term.) Using an argument as above, it can be seen that this induces a partitioning of the vertices into k classes, and that this forms a valid coloring of G.

Now we will consider a distribution $\mathcal{D}$ over inputs where each α_i, $\beta_{i,j}$ is equally likely. Furthermore, we will set ϵ so that a learning algorithm ϵ, δ PAC learns f_G over $\mathcal{D}$ if and only if it is correct over all of the inputs α_i, $\beta_{i,j}$. (This is possible because the total number of inputs is polynomial, so we can set ϵ to be one over this polynomial.) Now suppose that we have a proper PAC-algorithm that learns the class of monotone k-term DNFs. Given $G = (V, E)$, k, we can simulate the algorithm on f_G over $\mathcal{D}$. At the end, if the algorithm returns a k-term hypothesis, then we know that G is k-colorable, and if not, then we know that G is not k-colorable.

Using more complicated ideas, we can actually show that the class of DNF formulas of size s cannot be learned by DNF formulas of size polynomial in s.

1 Hardness based on Cryptographic Assumptions

In this section we show that certain classes of concepts are hard to PAC learn, not because of hypothesis class restrictions, but because of the inherent computational difficulty of prediction– that is, concept classes where the VC dimension is polynomial in n, and thus there is no information theoretic barrier to learning, and yet we will show that $\mathcal{C}$ cannot be PAC learned, regardless of the hypothesis class used by the learning algorithm.

First, we need to state something obvious but important. Namely, we cannot hope to prove a unconditional negative result concerning PAC learning, since such a result would imply $P \neq NP$! To see this, we will prove the contrapositive–that is, if $P = NP$, then any polynomially evaluatable concept class $\mathcal{C}$ is PAC learnable. Let $\mathcal{C}$ be some concept class. The problem of determining given any polynomial-sized sample set S, whether there is a hypothesis $h \in \mathcal{H}$ consistent with S is in NP. Thus if $P = NP$, then such a consistent hypothesis could be computed in polynomial time, and thus $\mathcal{C}$ is PAC learnable. Thus the best we can hope for in terms of a negative result is to show that some particular concept class $\mathcal{C}$ is not PAC learnable unless $P \neq NP$. Currently it is an open problem to show this. However, we can prove such negative results under stronger, cryptographic assumptions, which we will explain now.

The first hardness results for PAC learning can be traced to a 1986 paper of Goldreich, Goldwasser and Micali. In this paper they show that polysize boolean circuits are not PAC learnable. This holds even over the uniform distribution, and even if we require only weak learning, and even if we allow membership queries.

The high level idea is as follows. A one-way permutation f from strings of length n to strings of length n is a function that can be computed by a polynomial-size circuit, but such that the inverse map cannot be computed by a polynomial-size circuit. In fact, we require that the inverse map cannot be computed even some nonnegligible fraction of the time by a polynomial-size circuit. Let f be an alleged such one-way permutation, and assume that the class of polynomial-size circuits is PAC learnable. Then we will invert f as follows. Run the learning algorithm using examples of f taken over the uniform distribution. Note that since f is easy to compute and is a bijection, we can simulate randomly sampling from the uniform distribution by choosing random elements y in the domain of f, and then computing $f(y) = x$ to obtain a labelled pair (x, y). Since f is a bijection, this is equivalent to sampling over uniform x from the range. The concept output by the learning algorithm inverts f with nonnegligible probability!

In order to make this more precise, we will identify a particular permutation that is believed to be one-way, and give the details of this argument.

The group Z_N^* consists of the natural numbers $\{1, \ldots, N-1\}$ that are relatively prime to N, and the group operation is multiplication mod N. The order of the group Z_n^*, $\phi(N)$, is the size of the group. For example, for $N = 8$, Z_8^* consists of the elements $1, 3, 5, 7$, and the order is 4.

Claim For any N, if 3 does not divide $\phi(N)$, then $f_N(x) = x^3 mod N$ is a permutation of Z_n^*.

Proof Let d be such that $3d = 1 mod \phi(N)$. (Such a d exists since the gcd of 3 and $\phi(N)$ is 1, so by the extended Euclidean algorithm, one can compute integers c and d such that $\phi(N)c + 3d = 1$.) Now we claim that the inverse function $f_N^{-1}(y)$ of $f_N(x)$ is the mapping $f_N^{-1}(y) = y^d mod N$. To see this, note that since $3d = 1 mod N$, $3d = k\phi(N) + 1$ for some natural number k, so we have

$$\begin{aligned}
(f_N(x))^d mod N &= (x^3 mod N)^d mod N \\
&= x^{3d} mod N \\
&= x^{k\phi(N)+1} mod N \\
&= (x^{\phi(N)})^k x mod N \\
&= x mod N
\end{aligned}$$

Where the last equality holds because for any multiplicative group, and for any element x in the group, x raised to the q, where q is the order of the group, is equal to the identity element. We will call d the inverting exponent for N. The existence of this inverse mapping establishes that $f_N(x)$ is a bijection, as claimed.

Definition Let two primes p and q be chosen such that $N = pq$ and 3 does not divide $\phi(N) = (p-1)(q-1)$. Then $x \in Z_n^*$ is chosen. An algorithm for the Discrete Cube Root Problem is given N and $y = f_N(x)$ as input, and must output x.

Note that the input to the above problem has length $O(\log N)$, so a polynomial time algorithm should run in time polynomial in $\log N$. Clearly if N could be factored, then it would be easy to solve the Discrete Cube Root Problem. (We would simply compute the inverting exponent d from p and q using the extended Euclidean algorithm, and then compute $f_N^{-1}(y) = y^d mod N = x mod N$. The Discrete Cube Root problem is believed to be intractable although this is an unproven assumption. The well-known RSA public key cryptosystem is based on the assumption that the Discrete Cube Root Problem is intractable.

Definition The Discrete Cube Root Assumption (DCRA) states that for every polynomial p, there is no algorithm that runs in time $p(n)$ and that on input N and $y = f_N(x)$ (where $|N| = n$, and $N = pq$ such that 3 does not divide $(p-1)(q-1)$ and x is chosen randomly from Z_n^*), outputs x with probability greater than $1/p(n)$.

Now suppose we are given a random N, and $y = f_N(x)$ for a random $x \in Z_N^*$ and we want to compute x from these inputs. The DCRA asserts that this is a difficult problem. Now suppose that in addition to these two inputs we have access to many already solved examples for the given N. That is, suppose we are also given a bunch of samples $(y_1, f_N^{-1}(y_1), \ldots (y_k, f_N^{-1}(y_k)$ where each $y_i \in Z_n^*$ is chosen randomly. It is not hard to see that even with these examples the problem of computing x is just as hard. To see this, note that since we have N, we can generate random pairs efficiently by ourselves by just picking a random $x_i \in Z_N^*$ and then obtaining the pair $(x_i, f_N(x_i))$. Because f_N is a bijection, this has the same distribution as a set of random samples.

For every natural number n, let $\mathcal{F}_n$ be the concept class of all inverse functions f_N^{-1} for the functions $f_N(x) = x^3 modN$, where $N = pq$, $|N| = n$ and where 3 is relatively prime with $(p-1)(q-1)$. Let f_N^{-1} be the target function and let the underlying distribution $\mathcal{D}$ be the set of all random input-output pairs where the input distribution is uniform on Z_N^*. A PAC learning algorithm, A, should output in time polynomial in n, a hypothesis h that agrees with f_N^{-1} on at least a $1/p(n)$ fraction of the distribution where p is some fixed polynomial.

If A exists, then the DCRA is false. Given N and y, we use N to generate examples of f_N^{-1} with respect to the uniform distribution on Z_N^*. Then we use these examples to simulate A, and then use the hypothesis h output by A to compute $f_N^{-1}(y)$. For a random input y, we compute $f_N^{-1}(y)$ correctly with probability at least $1/p(n)$, contradiction the DCRA.

The only issue is that the above concept class that we have shown is not PAC learnable (under the DCRA), is not a boolean concept class. However, using a standard trick of reducing a search problem to a decision problem, we can show that a related boolean concept class is also not PAC learnable, again under the DCRA. Namely, the class of boolean functions $\mathcal{C}$ is the set of all functions $f_{N,i}^{-1}(y)$, where $f_{N,i}^{-1}(y)$ outputs the i^{th} output bit of $f_N^{-1}(y)$. Clearly if $\mathcal{C}$ were PAC learnable, then so would $\mathcal{F}_n$, and thus $\mathcal{C}$ is not PAC learnable unless the DCRA is false.