CSE 513T Theory of Artificial Intelligence & Machine Learning September 13, 2016

Lecture 5

Lecturer: Brendan Juba *Scribe: Bradley Flynn, Editor: Samuel Leonard*

Recall:

- Relation Problems

  For sets X, Y ("inputs" and "outputs"), and a relation $R \subseteq X \times Y$, the corresponding relation problem is "given $x \in X$, find $y \in Y$ such that $(x, y) \in R$."

- NP Search Problems

  A relation problem $R \subseteq X \times Y$ is an NP Search Problem if

  1. R is "polynomially balanced" there is a polynomial $p(n)$ such that for every $(x, y) \in R$, there exists a y' with $|y'| \leq p(|x|)$ and $(x, y') \in R$, where $|y'|$ is its length in bits.
  2. R is "efficiently verifiable" there is a polynomial-time algorithm for computing the Boolean function
  $$R(x, y) = \begin{cases} 0, & (x, y) \notin R \\ 1, & (x, y) \in R \end{cases}$$

- NP Completeness

  We will say that a relation problem R is NP-Complete if:

  1. R is an NP-search problem
  2. for every NP-search problem S, there is an algorithm that uses an algorithm for solving R as a subroutine, and assuming that this subroutine runs in polynomial time, solves S in polynomial time.

- Proposition regarding boolean circuits and computing functions

  Suppose there is a polynomial-time algorithm that computes $f = \{f_n : \{0,1\}^n \to \{0,1\}^{k(n)}\}$. Then there is an algorithm that when given an integer n as input, runs in polynomial time in n and outputs a representation of a Boolean circuit that computes f on inputs of length n, i.e., the circuit c_n that satisfies $c_n(x_1, \ldots, x_n) = f_n(x_1, \ldots, x_n)$ for all $x \in \{0,1\}^n$.

1 CircuitSAT and Cook-Levin

Theorem 1 (Cook-Levin) CircuitSAT *is NP complete.*

Proof We saw that CircuitSAT is an NP search problem.

Idea: Use efficient verifiability of arbitrary NP-Search Problem S and the proposition to obtain a circuit S_x computing $S(x, y)$ ("hard wire" input $\bar{x}$). Once we have $S_x(y) = S(x, y) \forall y$, CircuitSAT on circuit S_x returns y such that $(x, y) \in S$.

But we have a couple technical problems ...

1. Input length. We're given an x of length n. We can use polynomial balance to guarantee $|y| \leq p(n)$. So, overall, the input S only needs length $\approx n + p(n)$.
2. The circuit must have a fixed number of input wires, for y, but the length of a solution y could vary. We need to know that there is a y of length exactly l, for given arbitrary x.

Idea: we allow "padding" at the end of the solutions so that there always exists a solution y of length exactly $p(n)$ (to satisfy S_x).

For example - use a "null terminated string" to encode y ($\{0,1\} \to \{10,11\}$), or similarly, use 2 bits to encode each bit of y, and then extend it with all 0's.

We then solve CircuitSAT for S_x and then strip our padding in the solution. ■

2 "Improper" Learning, Representation-dependent Hardness

So far we have defined PAC-learning as follows:

Definition 2 (Proper Learning) *For a given representation class $\mathcal{C}$, we assume the data is labeled by some $c \in \mathcal{C}$, and we want to find $h \in \mathcal{C}$ so that $Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon$. We call this "proper learning."*

Today we will broaden our definition of PAC-learning:

Definition 3 (Improper Learning) *In "improper learning," we may return h in some other class of representations $\mathcal{H}$. (If we regard these to efficiently evalulatable, the we could take $\mathcal{H}$ to be Boolean circuits without loss of generality)*

We will show that there are classes $\mathcal{C}$ for which there is no efficient **proper learning** algorithm, but there *is* an efficient **improper learning** algorithm! (Note that we can still use such algorithms to make predictions) Specifically, we will show this to be true about the class of 3-term-DNFs.

2.1 Solving consistency problems with PAC-learning algorithms

Lemma 4 *If there is a (proper) PAC-learning algorithm for a class of representations $\mathcal{C}$, then there is a randomized polynomial-time algorithm for solving*

$$Consis_{\mathcal{C}}^{B(n)} = \left\{ \left([\overbrace{(x^{(1)}, b^{(1)}), \ldots, (x^{(n)}, b^{(n)})}^{\bar{x}}], c \right) : \ c \in \mathcal{C}, \ |c| \leq B(n), \ c(x^{(i)}) = b^{(i)}) \forall i \right\}$$

(Note: the last condition can be stated as c is consistent with x)

Proof If $\mathcal{C}$ has a proper PAC-learning algorithm, let such a (polynomial-time) algorithm be given. Let's run this algorithm on D, where D is a uniform distribution on the examples drawn from $\bar{x}$ (i.e. each $(x^{(i)}, b^{(i)})$ is chosen uniformly, with probability $1/m$). We can sample D easily.

Notice that a mistake on any of the $x^{(i)}$s incurs an error with with probability $\geq 1/m$. If we take $\epsilon = 1/2m$, then any h output by the algorithm cannot afford to label any $x^{(i)}$ inconsistently.

Note that the algorithm runs in time polynomial in

$$\underbrace{|x^{(i)}|}_{n}, \quad \underbrace{\frac{1}{\epsilon}}_{2m}, \quad \underbrace{\frac{1}{\delta}}_{\text{constant}}, \quad \underbrace{size(c)}_{\leq B(n), \text{which is poly-n}}$$

and it's all polynomially bound in $|\bar{x}|$, So there exists a consistent $c \in \mathcal{C}$. Our algorithm must run in polynomial-time, and it outputs an $h \in \mathcal{C}$ that is consistent. So, it solves $\text{Consis}_{\mathcal{C}}^{B(n)}$. ■

Now we'll use this lemma to show that $\text{Consis}_{\text{3-term-DNF}}^{3n}$ is NP complete.

2.2 NP-Completeness of 3-term-DNFs

A 3-term-DNF is an OR of at most 3 ANDs of literals, i.e., some boolean function $T_1 \vee T_2 \vee T_3$, where each T_i is a conjunction. Note, WLOG, that DNFs have a size $\leq 3n$.

Theorem 5 *: Unless NP has rondomized polynomial-time algorithms, there is no PAC-learning algorithm for 3-term-DNF*

Proof

Strategy: Show reduction from a known NP-complete problem (3-color) to $\text{Consis}_{\text{3-term-DNF}}^{3n}$, and then apply our new lemma.

The 3 coloring problem is defined as follows:

$$\{(G, \chi) \mid G = (V, E) \text{ is an undirected graph, } \chi : V \to \{R, G, B\}, \ \chi(u) \neq \chi(v) \ \forall (u, v) \in E\}$$

Given G as input, construct the following set of examples $n = |V|$ attributes. For each vertex i create a positive example v_i :

$$(1, 1, \ldots, 1, \underbrace{0}_{i^{\text{th}}}, 1, 1, \ldots, 1)$$

and for each edge (i, j) create a negative example $e_{i,j}$:

$$(1, 1, \ldots, 1, \underbrace{0}_{i^{\text{th}}}, 1, 1, \ldots, 1, \underbrace{0}_{j^{\text{th}}}, 1, 1, \ldots, 1)$$

We need to show

1. if G is 3-colorable, then some 3-term-DNF is consistent with this data set, and
2. using any 3-term-DNF that is consistent, we can "read off" a proper 3-coloring of G.

So, if there is a 3-coloring and a PAC-learning algorithm exists, then by (1) we can find a consistent 3-term-DNF, which by (2) we can use to find a 3-coloring.

1. Let a proper 3-coloring be given for G. Construct a 3-term-DNF where each color gets a term, and that term contains a literal x_i which is *not* that color.

   We'll show that this DNF is consistent with our examples.

   Consider the positive example v_i. We know that the vertex i has some color $\chi(i)$. Thus the term corresponding to $\chi(i)$ will not contain the literal x_i. Furthermore, each term only contains positive literals and v_i is positive in every attribute other than x_i. So v_i satisfies that term and thus the whole DNF.

   Now consider the negative example $e_{i,j}$. We know that $\chi(i) \neq \chi(j)$. The literal x_i appears in the term corresponding to $\chi(j)$ and the literal x_j appears in the term corresponding to $\chi(i)$. So, since x_i and x_j are false in this example, each of these terms is false. the third term (corresponding to the color other than $\chi(i)$ and $\chi(j)$) contains both x_i and x_j. So it is also false, which means the DNF is false as well.

2. Assume we have some 3-term-DNF that satisfies out data set. Let's arbitrarily assign a color to each of the three terms of h. Since h is consistent with our data set, every vertex example i is satisfied. Assign vertex i a color corresponding to one of the satisfied terms.

   To see the 3-coloring is proper, consider any two vertices (i, j) that got the same color. We know that the same term is satisfied by both v_i and v_j. Since the attribute x_i appears false in i's example, and true in j's example, neither of the literals x_i or $\neg x_i$ could appear in that term.

   If there were an edge (i, j) in G, the negative example $e_{i,j}$ and the positive example v_j would differ only in the value of x_i. Hence the term from before would be satisfied by $e_{i,j}$ as well. Since h was consistent with our data set, no such example $e_{i,j}$ could be in the data set. So, $(i, j) \notin E$. Therefore, (contrapositively) for every $(i, j) \in E$, $\chi(i) \neq \chi(j)$ (i.e. the coloring is proper).

■