18.858 Machine Learning	**Lecture 5 : September 26, 1994**
Lecturer: Ron Rivest	*Scribe: Hugh B. Morgenbesser*

Outline

- PAC learning 3-term DNF by 3-term DNF is intractable.
- Occam's algorithms (simple case).
- Learning 3-term DNF by 3-CNF.

5.1 Intractability of PAC learning 3-term DNF by 3-term DNF

After talking and thinking about PAC learning, it is interesting to try to determine what types of concept classes are PAC learnable. In the last lecture, we saw that the concept class consisting of conjunctions is PAC learnable. Here, we will show that the class of 3-term DNF is probably not PAC learnable [2]. The "probably not" comes from a reliance on the assumption that NP-complete problems cannot be solved efficiently by a randomized algorithm. In other words, we assume here that $RP \neq NP$.

The general idea of this method is to choose a known NP-complete problem which we can reduce to the problem of PAC learning 3-term DNF formulae. This will prove that we cannot PAC learn 3-term DNF formulae by 3-term DNF formulae unless $RP = NP$.

In this learning problem, we are targeting 3-term DNF formulae. An example of a 3-term DNF formula is:

$$x_1\overline{x}_2x_3x_4 \vee x_2\overline{x}_4\overline{x}_6 \vee x_7x_8\overline{x}_9.$$

5.1.1 Graph 3-colorability

The problem of determining whether a given graph G with vertices V and edges E can be colored using 3 or less colors (where no edge connects two vertices of the same color) is a known NP-complete problem. For more information on this and other known NP-complete problems, see the Garey and Johnson text [1].

We desire some procedure to map these graphs $G = (V, E)$ to 3-term DNF target formulae. In other words, given an instance G we would like to compute a set S of labelled examples such that G is 3-colorable if and only if S is consistent with some 3-term DNF formula.

Note that in our mapping from G to S, S will be the size of $V + E$. Note also that the method for determining S does not require that we know whether or not G is 3-colorable.

5.1.2 What does learner get?

We give the learner a source of labelled examples, and values for ϵ and δ.

- oracle for labelled examples: the examples are chosen uniformly at random from S (defines D).
- choose $\epsilon = \frac{1}{|S|+1}$ so that hypothesis must be consistent with all examples in S.
- and choose an arbitrary small δ, say $\delta = 0.00001$.

We will now prove that there is a transformation that gives the desired property that: G is 3-colorable $\iff$ S is consistent with some 3-term DNF formula. This will prove the equivalence of PAC learning 3-term DNF formula and the existence of an *RP* algorithm for determining graph 3-colorability.

Note that if G is not 3-colorable we are asking the learner to do the impossible. It can't produce a 3-term DNF formula, because we have ensured that it doesn't exist.

Furthermore, if we have an algorithm for learning 3-term DNF, we have a randomized algorithm for graph 3-colorability. The algorithm would be to convert G to S and to try to learn S. The algorithm would conclude that the graph was 3-colorable if and only if it was able to learn S.

If the graph is 3 colorable, the learner will come up with a 3-term DNF formula consistent with S. If not, the learner will do something else (either halt, return bad

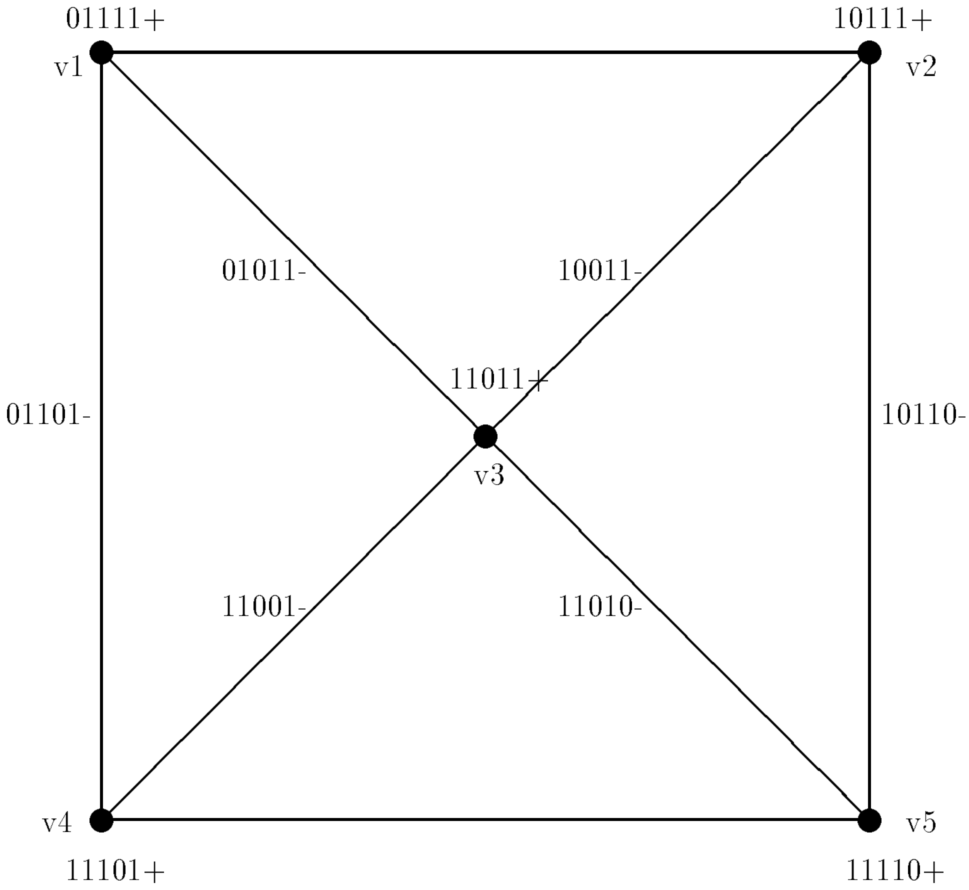

Figure 5.1: Example of the mapping from a graph to a set of labelled examples.

formula, etc.). So we can test the resultant formula against S. If we do not find a consistent hypothesis, then G is not 3-colorable (with probability $\geq 1-\delta$).

Construction: The transformation from a graph G to a set of labelled examples S is done as follows (as shown in Figure 5.1). For each vertex v_i, a positive example $x^{(i)}$ is created, where $x^{(i)} = x_1x_2x_3\ldots x_n$, and $x_i = 0$ and $x_j = 1$ for all $j \neq i$. For each edge, (v_i, v_j), a negative example is created $e_{ij} = x_1x_2x_3\ldots x_n$, where $x_i = x_j = 0$ and $x_k = 1$ for all $k \neq i, j$.

This is clearly constructible in polynomial time.

First we show the $\Rightarrow$ direction. Assume G is 3-colorable. Then, by the definition of 3-colorability, there exist sets R, B, and Y of vertices that are disjoint and color the graph. For example, in the graph in the figure, sets of $R = \{v_1, v_5\}, B = \{v_2, v_4\}$, and $Y = \{v_3\}$ represent a valid 3-coloring. That is, in our example, by coloring vertices v_1 and v_5 red, vertices v_2 and v_4 blue, and v_3 yellow, no two adjacent vertices have the same color.

In this case, the following terms of a 3-term DNF will have the desired property.

$$T_R = \bigwedge_{x \in B \cup Y} x = x_2 x_4 x_3$$

$$T_B = \bigwedge_{x \notin B} x = x_1 x_3 x_5$$

$$T_Y = \bigwedge_{x \notin Y} x = x_1 x_2 x_4 x_5$$

Claim 1 *$F = T_R \vee T_B \vee T_Y$ is consistent with S.*

1. F is *true* on all *positive* examples: If $v_i \in R$ then $x^{(i)}$ satisfies T_R. This implies that F is true.

2. F is *false* on all *negative* examples: Say (v_i, v_j) is an edge. Then v_i and v_j aren't both red since this a valid 3-coloring. Assume that v_i isn't colored red. Then, by definition, literal x_i is in T_R And thus the negative example corresponding to edge (v_i, v_j) doesn't satisfy T_R. Similarly we can show it doesn't satisfy T_Y or T_B.

So, since 1 and 2 are both true, we see that if G is 3-colorable, then S is consistent with some DNF formula.

For reverse direction $\Leftarrow$, we assume that some 3-term DNF formula is consistent with S, i.e. $T_R \vee T_B \vee T_Y$. Call vertex v_i red if example $x^{(i)}$ satisfies T_R, blue if it satisfies T_B, and yellow if it satisfies T_Y. (Ties may be broken arbitrarily).

Claim 2 *No two adjacent vertices have the same color.* Suppose that vertices v_i and v_j are both colored red, then the examples $x^{(i)}$ and $x^{(j)}$ satisfy T_R by definition. If v_i and v_j are adjacent, then $e_{ij} = x^{(i)} \wedge x^{(j)}$ (bitwise). But $x^{(j)}$ and e_{ij} differ only in

i^{th} bit so then e_{ij} must also satisfy T_R. But e_{ij} is a negative example, so this is a contradiction. So no two adjacent vertices have the same color, and we have colored the graph using 3-colors.

From this we see that PAC learning of 3-term DNF is as "tough" as an NP-complete problem [1].

This result is important, but we won't give up yet. We still want efficient PAC learning algorithms. But how? One idea is explored below. The basic idea is to learn a concept class $\mathcal{C}$ using a larger hypothesis class, $\mathcal{H}$.

But first, let us talk about Occam algorithms.

5.2 Occam Algorithms

An Occam algorithm is just a general structure for how to apply a PAC learning algorithm. Simply stated it is two steps:

- draw a sample of size $m = poly(n, size(\mathcal{H}))$
- return any $h \in \mathcal{H}$ consistent with sample.

We define h to be "bad" if error$(h) \geq \epsilon$. Then the probability h survives m examples is $\leq (1-\epsilon)^m$. The probability that some bad hypothesis survives m examples is then $\leq |\mathcal{H}|(1-\epsilon)^m$. We want that probability to be $\leq \delta$, so we require:

$$|\mathcal{H}|(1-\epsilon)^m \leq \delta.$$

We can get this inequality into another form by taking the natural logarithm of both sides:

$$\ln|\mathcal{H}| + m\ln(1-\epsilon) < \ln(\delta)$$

or equivalently:

$$m > \frac{\ln(\delta) - \ln(|\mathcal{H}|)}{\ln(1-\epsilon)}$$

[1] Note to the curious: k-term DNF is hard for any $k \geq 2$.

and since $\ln(1-\epsilon) < -\epsilon$, we see that:

$$m > \frac{1}{\epsilon}(\ln(|\mathcal{H}|) + \ln(\frac{1}{\delta}))$$

.

5.3 PAC Learning 3-term DNF by 3-CNF

3-CNF is Conjunctive Normal Form with at most 3 literals per clause.

$$\bigwedge_i (x_{i_1} \vee x_{i_2} \vee x_{i_3})$$

For example, $(a \vee b)(\overline{c} \vee b \vee d)(\overline{a} \vee e)$, is a 3-CNF formula. Every 3-term DNF formula is equivalently representable by a 3-term CNF formula, i.e.

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{x \in T_1, y \in T_2, z \in T_3} (x \vee y \vee z)$$

.

The converse is not true. Many 3-CNF formulae cannot be expressed in 3-term DNF. 3-CNF is more expressive than 3-term DNF.

How big are these classes? Well, $\ln(|\text{3-term DNF}|) = \theta(n)$, and $\ln(|\text{3-CNF}|) = \theta\left(\binom{2n}{3}\right) = \theta(n^3)$. So we see that 3-CNF is bigger than 3-term DNF.

To learn 3-term DNF by 3-CNF, we use an Occam style algorithm.

1. Pick enough samples (given by formula for m in Section 5.2).
2. To find a hypothesis consistent with labeled examples, reduce the problem to learning conjunctions. This is done by creating a new set of labelled examples on $\theta(n^3)$ variables, one for each possible 3-CNF clause. Then the problem is as easy as learning conjunctions.

References

[1] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the theory of NP-Completeness.* Freeman, 1979.

[2] L. Pitt and L. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35:965-984, 1988.