6.858/18.428 Machine Learning	**Lecture 4: September 21, 1994**
Lecturer: Ron Rivest	*Scribe: Rob Miller*

Outline

- PAC Model
- PAC-learning of Conjunctions
- Intractability of PAC-learning 3-term DNF (introduction only)

4.1 PAC Model

So far in this course we've studied the mistake-bound model of learning, which puts an upper bound on the total number of mistakes that the learner will ever make. Unfortunately, mistake bounds do not capture everything one might want from a learning model. For instance, it might be desirable to prove something about the learner's reliability after it has seen a certain number of examples. This would be useful in practice to model learning algorithms that go through a period of training before they are actually used in the field. Clearly it would be better for the algorithm to make its mistakes during training, rather than in operation. With the mistake-bound model, however, we can only say *how many* mistakes the algorithm will make, not *when* they will be made.

Another problem with the mistake-bound model is its assumption that the examples are chosen arbitrarily — in the worst case, by an intelligent adversary. In practice, this assumption may be unnecessarily strong. Often it is satisfactory to assume that examples are drawn randomly, from an arbitrary probability distribution on the instance space. These problems are the motivation for the *Probably Approximately Correct* (PAC) model of learning, first defined by L.G. Valiant [1].

4.1.1 Parts of the model

Like our other learning models, the PAC model includes a domain of instances X, a class of concepts $\mathcal{C}$ containing functions mapping X to $\{0,1\}$, and a target concept $c \in \mathcal{C}$. The learner's goal is to identify the target concept c.

The model also includes a probability distribution $\mathcal{D}$ on X. The learner has access to a random oracle $EX(c, \mathcal{D})$, which at the learner's request will return a labelled example $< x, c(x) >$ chosen randomly and independently according to the probability distribution $\mathcal{D}$. The label of x is given by the target concept c.

We now make some remarks about the distribution $\mathcal{D}$. First, $\mathcal{D}$ is completely arbitrary. For instance, some examples may have probability zero under $\mathcal{D}$, in which case the learner will never see them. Second, $\mathcal{D}$ is assumed to be *stationary*, fixed over time, so that the probability of seeing an example during the training period is the same as the probability of seeing it afterwards. In essence, training and testing must use the same source of examples, $EX(c, \mathcal{D})$. Thirdly, $\mathcal{D}$ is unknown to the learner. One approach to handling this arbitrary distribution might be to try to learn $\mathcal{D}$ by sampling from $EX(c, \mathcal{D})$. Typically, however, the learner does not have to learn $\mathcal{D}$ to generate a satisfactory hypothesis.

Using its oracle of random examples, the learner is expected to compute a hypothesis h which closely approximates the target concept c, where the accuracy of a hypothesis is defined as follows:

Definition 1 *The* **error rate** *$error(h)$ for hypothesis h with respect to a distribution $\mathcal{D}$ and target concept c is*

$$error(h) = \Pr_{x \in \mathcal{D}}(c(x) \neq h(x))$$

Note that we couldn't express the accuracy of a hypothesis without the distribution $\mathcal{D}$, which specifies which areas of disagreement between the hypothesis and the true concept are more important than others. The learner takes as inputs an accuracy parameter ϵ and a confidence parameter δ, and is expected to output a hypothesis h such that with probability at least $1 - \delta$,

$$error(h) \leq \epsilon$$

This is the source of the name "Probably Approximately Correct" — the learned function is probably (within δ) a good approximation (with an error rate of at most ϵ) of the true concept.

4.1.2 Definition of PAC-learning

The following definition sums up this discussion of PAC-learning.

Definition 2 *$\mathcal{C}$ is* **PAC-learnable** *(using $\mathcal{C}$) if there exists a learning algorithm L such that for all $c \in \mathcal{C}$, all $\mathcal{D}$ on X, all ϵ $(0 < \epsilon < 1)$, and all δ $(0 < \delta < 1)$, then running L with access to $EX(c, \mathcal{D})$ and inputs ϵ and δ, produces with probability at least $1 - \delta$ an output $h \in \mathcal{C}$ such that $error(h) \leq \epsilon$.*

Note that this definition says nothing about what happens in the case where L fails to output a sufficiently accurate hypothesis. L may output an arbitrary hypothesis, or it may not even terminate! The definition of PAC-learning guarantees only that the probability of this pathological case will be less than δ, where δ can be made as small as desired.

To guarantee that L always terminates within polynomial time (even when it fails to produce an accurate hypothesis), we will prefer to use the following definition.

Definition 3 *$\mathcal{C}$ is* **efficiently PAC-learnable** *(using $\mathcal{C}$) if $\mathcal{C}$ is PAC-learnable and there exists a PAC-learning algorithm L for $\mathcal{C}$ that runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n (the size of an instance of X), and $size(c)$.*

4.1.3 Remarks

- Typically L will be polynomial in $\ln \frac{1}{\delta}$, not just $\frac{1}{\delta}$.
- Defining PAC-learning with two oracles, one for positive examples and one for negative examples, results in an equivalent model. See problem 1.3 in the Kearns/Vazirani text. The solution may be found in [2].
- This is a worst-case model: L must meet its accuracy and confidence requirements even for the worst possible choices of $\mathcal{D}$ and c. As a result, many of the results about PAC-learning in the literature are negative results, showing that a certain class $\mathcal{C}$ is *not* PAC-learnable.
- As defined, the model requires the learner to choose its hypothesis from $\mathcal{C}$. It is frequently useful to extend the definition, allowing the user to choose h from some other class of concepts $\mathcal{H}$, in which case $\mathcal{C}$ is said to be "PAC-learnable using $\mathcal{H}$." Usually $\mathcal{C} \subseteq \mathcal{H}$, but not necessarily. We will see that $\mathcal{H}$ is often chosen to make computation easier, so that a class of concepts that was not

efficiently PAC-learnable using itself becomes efficiently PAC-learnable when a larger class of concepts are permitted as hypotheses.

4.2 PAC-Learning of Conjunctions

To demonstrate a PAC-learning algorithm, we return to a class of concepts that we studied under the mistake-bound model: conjunctions.

$$X = \{0,1\}^n$$

$$\mathcal{C} = \text{conjunction of literals}$$

$$c = \text{(for example) } x_1\bar{x_2}x_5$$

The PAC-learning algorithm for conjunctions follows the same essential form as most of the PAC-learning algorithms we will study:

1. Draw a sufficiently large sample from the oracle (m examples).
2. Find some $h \in \mathcal{C}$ consistent with all m examples.
3. Output h.

To finish describing the algorithm, we must show how to find an h consistent with the sample, and we must calculate how large m needs to be to make h sufficiently accurate. The procedure for finding h is the same as in the mistake-bound algorithm. Start with $h = x_1\bar{x_1}x_2\bar{x_2}...x_n\bar{x_n}$. Ignore negative examples, and consider only the positive examples. For each positive example, delete from h every literal v such that v is false in the example. For instance, if the positive example is 0100, then $x_1, \bar{x_2}, x_3$, and x_4 would be deleted from h. This has the effect of making h the most specific hypothesis consistent with all positive examples seen so far.

Literals in the target concept are never deleted from h — they must be true in all positive examples, since the target concept is a conjunction. However, h may include some additional literals not present in the target conjunction, making it *too specific* and thus excluding some instances that should be positive examples. We want to be confident that the combined probability of these excluded positive instances is no greater than ϵ.

Definition 4 *Let z be a literal. Then $p(z)$ is the probability that $EX(c, \mathcal{D})$ returns a positive example in which z is false. In other words, $p(z)$ is the probability that z is deleted from h by a randomly-chosen positive example.*

There are $2n$ different $p(z)$: $p(x_1), p(\bar{x_1}), ..., p(x_n)$, and $p(\bar{x_n})$. For all literals z in the target concept, $p(z) = 0$, since z is never removed from h.

Claim 1 $error(h) \leq \sum_{z \in h} p(z)$

To understand Claim 1, take a literal z appearing in h, and suppose we draw a positive example x in which z is false. Since z appears in h and is false, $h(x) = 0$, so h makes an error on x. Thus $p(z)$ is the probability that z causes h to make an error on a randomly-drawn positive example. The sum of all $p(z)$ is therefore an upper bound on the error rate of h.

Definition 5 *A literal z is* **bad** *if $p(z) \geq \frac{\epsilon}{2n}$.*

Claim 2 *If h has no bad literals, then h is sufficiently accurate: $error(h) \leq \epsilon$.*

If h has no bad literals, then by Claim 1, $error(h) \leq 2n\frac{\epsilon}{2n} = \epsilon$. To finish the proof, we need to show that the probability that a bad literal remains in h after m examples is is at most δ. Let z be a bad literal. Then

$$\begin{aligned}
\Pr(z \text{ survives one example}) &= 1 - \Pr(z \text{ is deleted by one example}) \\
&= 1 - p(z) \\
&\leq 1 - \tfrac{\epsilon}{2n} \\
\Pr(z \text{ survives } m \text{ examples}) &\leq (1 - \tfrac{\epsilon}{2n})^m
\end{aligned}$$

There are at most $2n$ bad literals, so the probability that *some* bad literal survives m examples is bounded by $2n(1 - \frac{\epsilon}{2n})^m$. We need to choose m large enough to make this probability smaller than δ. Appealing to the inequality $1 - x \leq e^{-x}$ (which holds for $0 \leq x \leq 1$), we can say that the probability of failure is at most $2ne^{-\frac{m\epsilon}{2n}}$. So

$$m \geq \frac{2n}{\epsilon}(\ln 2n + \ln \frac{1}{\delta})$$

suffices to make the probability of failure less than δ.

Since m is polynomial in n, $\frac{1}{\epsilon}$, and $\ln \frac{1}{\delta}$, the algorithm is efficient. Thus, we have the following theorem:

Theorem 1 *Conjunctions are efficiently PAC-learnable.*

4.3 Intractability of PAC-learning 3-term DNF

Having seen that conjunctions are efficiently learnable under the PAC model, we might ask whether a larger concept class is also learnable. For instance, are disjunctions of two or three (or more generally some fixed k) conjunctions efficiently PAC-learnable? We will see next lecture that this type of concept class, called k-term DNF, is probably not efficiently learnable using itself, for computational reasons. In particular, we will show that PAC-learning 3-term DNF is NP-complete by reducing graph 3-colorability to it. As a result, unless NP=RP, this concept class can't be learned in polynomial time.

References

[1] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134-1142, 1984.

[2] D. Haussler, M. Kearns, N. Littlestone, and M.K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129-161, 1991.