CS 395T Computational Learning Theory

Lecture 9: September 27, 2005 Lecturer: Adam Klivans

Scribe: Aman Rathi

9.1 The PAC Learning Model

9.1.1 Some Background & Motivation

Till now, in the past lectures, we have worked with the Mistake Bounded(MB) Learning model. In this lecture, we are going to study about another significant learning model which in fact was the precursor of the MB model. We are talking about the PAC model i.e. *Probably Approximately Correct* Learning Model that was introduced by L.G Valiant, of the Harvard University, in a seminal paper [1] on Computational Learning Theory way back in 1984.

MB models may not always capture the learning process in a useful manner. For example, they require that the learning algorithm must yield the exact target concept within a bounded number of mistakes. But in many settings, we may only require an approximately correct answer. Also, it would be interesting to comment on the reliability of the learner (*i.e.* the learning algorithm) after it has seen a certain number of steps. These issues are addressed by the PAC model which because of the above mentioned relaxations can cater to a larger number of settings than the MB model

Hence, in general, the MB model enforces more stringent requirements on the learning algorithms. Thus, learning in the MB model is harder than the PAC model. We won't be formally proving this fact here as you are required to prove it as a Homework Problem in one of the problem sets.

9.1.2 An Intuitive Approach to PAC

The PAC model belongs to that class of learning models which is characterized by learning from examples. In these models, say if f is the target function to be learnt, the learner is provided with some random examples (actually these examples may be from some probability distribution over the input space, as we will see in a short while) in the form of (X,f (X)) where X is a binary input instance, say of length n, and f(X) is the value (boolean TRUE or FALSE) of the target function at that instance. Based on these examples, the learner must succeed in deducing the target function f which we can now express as $f : \{0,1\}^n \rightarrow \{0,1\}$. Before moving on, lets consider an important question. What actually should we mean by 'succeeded' ? Please agree that the notion of success better be reasonable enough so as to enable us to find interesting algorithms meeting its criteria. Hence,(see figure 9.1)

1. We may allow a small probability that the learning algorithm fails. This would be in some kind of worst cases where the examples that are picked up from the distribution and provided to the learner dont contain much information to help deduce the target function. In fact, as

you would be able to see later in the illustrations, PAC Learning actually tries to reduce the probability of getting such unlucky examples by considering a sufficient number of examples for learning a particular concept. In other words, the PAC learning strategy places an upper bound on the probability of committing an error by placing a minimum bound on the number of examples/samples it takes to learn a target concept.

2. We may spare the learner from learning the exact concept. If the learner can find a function h which is close to f i.e. if h is an approximates f, we would still be interested in it.

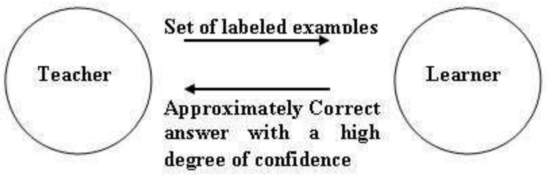
Teacher
Set of labeled examples
Learner
Approximately Correct answer with a high degree of confidence

Figure 9.1: PAC learning

The above two factors that we used to explain the meaning of success are quantified as ϵ and δ in the formal PAC literature. ϵ gives an upper bound on the error in accuracy with which h approximated f and δ gives the probability of failure in achieving this accuracy. Using both these quantities, we can express the definition of a PAC Algorithm with more mathematical clarity. Consequently, we can say that, to qualify for PAC Learnability, the learner must find with probability of at least $1-\delta$, a concept h such that the error between h and f is at most ϵ.

To make more mathematical sense of the above ideas, we define a probability distribution $\mathcal{D}$ over the input domain $X : \{0,1\}^n$. So when we say that we pick an example at random, we mean that we are picking an example from this distribution. Also, its imperative to mention that the all the examples that learner is provided with are independent of each other. So, if we are going to consider the total probability of encountering a sequence of examples over $\mathcal{D}$, we will use the product rule *i.e.* we would be multiplying the probabilities.

A major role played by $\mathcal{D}$, in our setting is that it conveniently and mathematically quantifies the error with which h approximates f. *i.e.* we can write

$$error(h,f) \equiv Pr_{x\in\mathcal{D}}(h(x) \neq f(x)) \tag{1}$$

Note, that the notation $Pr_{x\in\mathcal{D}}$ emphasizes that the probabilities are taken with respect to random draws over $\mathcal{D}$ only. If we view this error expression from the perspective of the set theory, we should call it the symmetric difference between the sets corresponding to h and f over X, as shown by the shaded region in the figure 9.2.

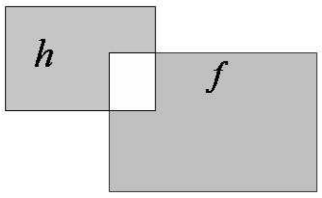

Figure 9.2: $ErrorRegion(shaded) = (h - f) \cup (f - h)$ /*Symmetric Difference*/

The shaded region shows the probabilistic region of disagreement between h and f and hence gives the probability of error. Note that a PAC Learning Algorithm would always be independent of $\mathcal{D}$ *i.e.* we can also say that the learning algorithm would be ignorant of $\mathcal{D}$ or that it must work for any $\mathcal{D}$ whatsoever. The only restriction on the setting is that we must calculate the probability of error over the same distribution $\mathcal{D}$ from which we draw our random examples. This is well justified because as we said earlier that the learner essentially tries to see some sufficient number of examples so that the probability of seeing an unlucky example *i.e.* an error over the same distribution reduces to some specified bounds. Hence, it is natural to calculate the error probability over the same distribution that was used during the learning phase. And as you can see, we have taken care of this fact in expression (1) which calculates the probability over D only.

9.1.3 The Formal Definitions

With an intuitive understanding of PAC from the previous section, we are well prepared to appreciate its formal definition as well. We present the definition to PAC in two stages. First we are going to define *PAC Learnability* and then graduate to *Efficient PAC Learnability* .

Definition 1 *(PAC learnability) Let $\mathcal{C}$ be a class of boolean functions $f : \{0,1\}^n \rightarrow 0,1$. We say that $\mathcal{C}$ is PAC-learnable if there exists an algorithm $\mathcal{L}$ such that*
for every $f \in \mathcal{C}$
for any probability distribution $\mathcal{D}$
for any ϵ $(where 0 \leq \epsilon < \frac{1}{2})$
for any δ $(where 0 \leq \delta < 1$
algorithm $\mathcal{L}$ on input ϵ and δ and a set of random examples picked from any probability distribution $\mathcal{D}$ outputs at least with a probability $1 - \delta$, concept h such that $error(h, f) \leq \epsilon$.

In the above definition, we want ϵ to be less than $\frac{1}{2}$. Well, this is pretty natural as we won't be interested in the algorithms which have more than fifty percent chances of giving an inaccurate answer. Also, observe the strict inequality i.e. $\epsilon < \frac{1}{2}$. This is because $\epsilon = \frac{1}{2}$ would be useless as it would be no better than generating the output with the toss of a coin.

Definition 2 *(Efficient PAC learnability) We say that $\mathcal{C}$ is efficiently PAC-learnable if $\mathcal{C}$ is PAC learnable as in Definition 1 and*

- *the number of examples that $\mathcal{L}$ takes is bounded by some polynomial in $n, 1/\epsilon and 1/\delta$*
- *$\mathcal{L}$ runs in time asymptotically bounded by some polynomial in $n, 1/\epsilon and 1/\delta$.*

9.2 Illustrative Examples for PAC

We now take up two examples and learn them using PAC strategy.

9.2.1 Learning Axis Aligned Rectangles

Consider a game to learn an unknown axis-aligned rectangle i.e. a rectangle in the Euclidean plane $\Re^2$ whose sides are parallel to the coordinate axes. Let R be the target rectangle. The examples are provided to the learner are in the form of random points p along with a label indicating whether p is contained in R(a positive example) or not contained in R(a negative example). Figure 9.3 shows the unknown rectangular region R along with some positive and negative examples.

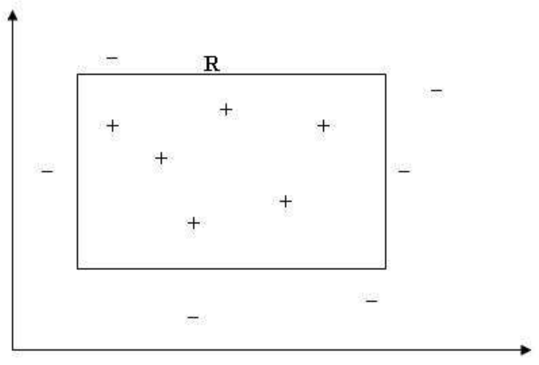

Figure 9.3: The target rectangle R along with some positive and negative points.

For the sake of motivation, [2] shows how a seemingly simple concept like learning axis-aligned rectangles can be readily adapted to realistic scenarios for eg. suppose we want to learn the concept of men of medium build. Assuming that a man is of medium build if his height and weight both lie in some prescribed ranges, then each man's build can be shown as a point in the Euclidean plane and the concept of medium-build would be an axis-aligned rectangle in that plane.

Coming back to our axis-aligned rectangle problem, we now intend to prove the following theorem :

Theorem 1 *The concept class of axis-aligned rectangles is efficiently PAC learnable.*

Proof:

Note that there is a simple and efficient way to come up with a hypothesis R' by taking a large number of examples and fitting the tightest rectangle around the positive ones so that all the given positive points lie inside it. (as shown in fig 9.4).

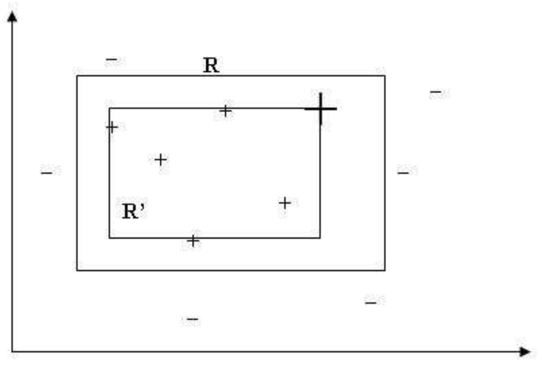

Figure 9.4: The tightest-fit rectangle R' over the given examples.

Note that R' is always a subset of R as is evident from fig 9.4. Hence the error region i.e. the symmetric difference between R and R' is limited to the union of four rectangular strips outside R' but inside R. Also note that if we can guarantee that weight under $\mathcal{D}$ of each strip (i.e. the probability over $\mathcal{D}$ of falling under such strip) is not more than $\epsilon/4$, them we can be sure that the total error of R' is at most ϵ. This is because of the union bound i.e.

$$Pr[AuB] \leq Pr[A] + Pr[B] \quad (2)$$

So, consider what can be a bad event for us in this setting. Please agree it would be a bad event if the probabilistic weight (or the probability), over $\mathcal{D}$, of the top strip (that is a part of the error region) is more than $\epsilon/4$ i.e. at least $\epsilon/4$ (see figure 9.5). In other words, it would be a bad event if the probabilistic weight of the rest of the region i.e. non-error region is at most $(1 - \epsilon/4)$. And thus,

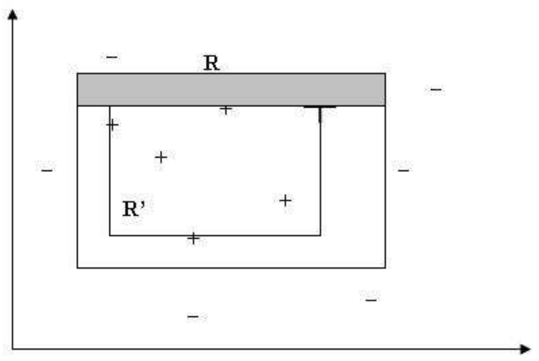

Figure 9.5: The top strip (shown shaded) is one of the four error strips in our analysis.

a bad event over m draws would be when the probability that we draw all our m examples from the non-error region is at most $(1-\epsilon/4)^m$. *Here we made use of the fact that all the draws/examples are independent of each other, a fact that we have discussed earlier. The same analysis holds for the other three strips as well. So, to calculate the total probabilistic weight of the bad event, we take the union bound of all four to give* $4(1-\epsilon/4)^m$. *And this probability must be bound* δ. *This is because, as discussed in the formal definition of PAC learning, our confidence measure in the good event must be at least* $1-\delta$ *i.e. the probability of the bad event must be at most* δ. *So we get*

$$4(1-\epsilon/4)^m \leq \delta$$

which can be solved using the inequality $(1-x) \leq e^{-x}$, *we get*

$$m \geq (4/\epsilon)\ln(4/\delta)$$

. *So, if our algorithm takes at least 'm' examples, then with a probability at least* $1-\delta$, *the resultant hypothesis h will have an error at most* ϵ *with respect to* f *and over* $\mathcal{D}$. *Also, since the processing of each example would require at most four comparisons, the running time is linearly bounded by* m. *Consequently, it is bounded by a polynomial in* $n, 1/\epsilon$, *and* $1/\delta$ *as required. Hence, PAC learnable.* ■

9.2.2 Learning 1-Disjunctions

Lastly, lets re-visit the learning of 1-Disjunctions(*i.e.* where each term has only one literal) that we also did for the MB model. Here, we intend to prove the following :

Theorem 2 *The concept class of 1-Disjunctions is efficiently PAC learnable.*

Proof:

Like previous illustration, the strategy here too would be the same as the conventional PAC strategy i.e. draw a sufficiently large number of example from the distribution and come up with a hypothesis consistent with them. In formulating a consistent h, we would use the same algorithm as we did for the MB approach i.e. on every negative example, get rid of all the literals that evaluate to true. Now, the only questions that is left to be answered is how many examples need to be drawn. For this, consider the following analysis.

Surely, our hypothesis is erroneous whenever there is any literal which is there in our hypothesis h but not in the target function f. We shall call such a literal, a bad literal. As per the PAC definition, we must reduce the chances of having a bad literal to less than ϵ. Again note here that since we can have as many as 2n literals, if we could guarantee that the probability of each literal being bad is less than $\epsilon/2n$ then we can be sure that the total probability of any literal being bad would be less than ϵ (from the union bound). So, we say that it would be a bad event if the probability of a literal being bad is more than $\epsilon/2n$ i.e. at least $\epsilon/2n$. Rephrasing this, we can claim that it would be a bad event if the probability that a literal is not bad is just at the most $1-\epsilon/2n$. So, the probability of this bad event after 'm' examples would be at most $(1-\epsilon/2n)^m$. And performing the union bound over all the 2n literals, we can say that the total probability of this bad event would be no more than $(1-\epsilon/2n)^m$. And to confirm this probability-estimate to PAC requirements, we must bound it by δ where $1-\delta$ is the desired confidence (see Defn. 2). Thus,

$$2n(1-\epsilon/2n)^m \leq \delta$$

Using the inequality $(1-x) <= e^{-x}$, we get

$$m \geq O((n/\epsilon)\ln(n/\delta)).$$

Thus, if our learning algorithm takes at least this much number of examples, then with a probability at least $1-\delta$, the resultant hypothesis h will have an error at most ϵ with respect to f and over $\mathcal{D}$. Also, since the algorithm takes linear time to process each example, the running time is bounded by O(mn), and hence is polynomially bounded in $n, 1/\epsilon and 1/\delta$.

■

References

[1] L.G. Valiant, *A Theory of the Learnable*, Communications of the ACM 27(11):1134–1142 (1984).

[2] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*, MIT Press, 1994.