PAC Learning

The following document is a brief overview of PAC learning to provide supplementary detail for Michael's lectures. This document will also provide further motivation by discussing examples and heuristics for some of the more technical parts of the lectures.

In his lecture on computational learning theory, Michael covers PAC learning. The goal with PAC learning is to determine which classes of target concepts can be learned from a reasonable number of randomly drawn training examples with a reasonable amount of computation.[1] In other words, we are asking questions like, "Is this a difficult concept to learn or not?", or "Will my learner be able to learn this in a reasonable amount of time?".

Before we can go any further, we should refresh some definitions:

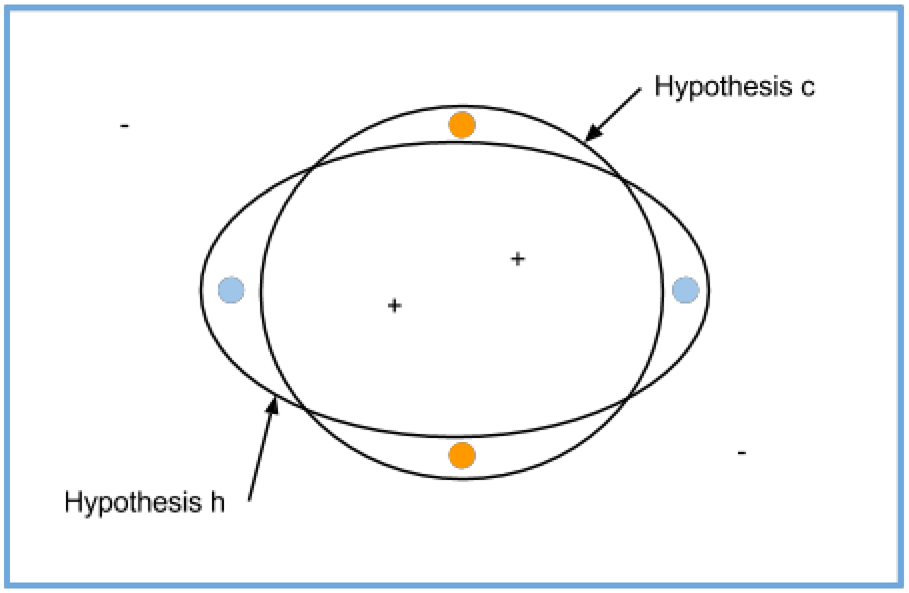
<u>Definition:</u> The *true error* of a hypothesis *h* with respect to a target concept *c* and an instance distribution *D* is the probability that *h* will misclassify a an instance drawn at random according to *D*. We write this as:

$$error_D(h) = Pr_{x \in D}[c(x) \neq h(x)]$$

Note that this is the actual, real-world error that our hypothesis h would have, if we could somehow know the probability distribution of the data that we might encounter.

[1] Mitchell, Tom M. "Machine learning. 1997." *Burr Ridge, IL: McGraw Hill* 45 (1997).

The Instance Space X



The image above illustrates the true error defined above. If *D* is a probability distribution that assigns the same probability to every instance in *X*, then the error for *h* will be the fraction of the total instance space where *c* and *h* disagree. The plusses and minuses indicate training examples, and the areas where c and h disagree are marked with blue and orange dots. Note that *h* has non-zero true error, even though *c* and *h* agree on all the training data.

Ok. Now that we know about error, we are almost ready to define what it means for a concept class to be PAC-learnable. First, a little more notation. Let's let *C* denote a class of target concepts under consideration, X denote the space of possible instances, *L* denote a learner (i.e., a learning algorithm or model), and *H* is the hypothesis space of *L* (i.e., the set of hypotheses that *L* can produce).

Note that the definition below differs slightly from the definition in Michael's lecture. This is (roughly) the more detailed definition given in Mitchell's book, so we are running with it.

Definition: Assume $0 \leq \varepsilon, \delta \leq \frac{1}{2}$. Then *C* is *PAC-learnable* by *L* if and only if for all distributions D, $c \in C$, ε and δ as above, *L* will, with probability $(1-\delta)$, output a hypothesis $h \in H$ such that $error_D(h) \leq \varepsilon$, in time that is polynomial in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, and $|H|$.

So what does this mean? Well, let's break it into pieces. Let's start by fixing an instance distribution D. Given D, we can't always expect a learner to produce hypotheses with 0 error, so we have to agree that our learner *L* does a pretty good job if it outputs an "approximately correct" hypothesis *h*, i.e. one that has very low error, say, like, $error_D(h) \leq \varepsilon$ for some small ε. On the other hand, we can't even expect a learner to always produce hypotheses with such low error. Why? Well, what happens if the training examples that are randomly chosen are misleading? Then the resulting hypothesis will have pretty high error. So instead, we might be pretty happy if our learner produced a low-error-hypothesis most of the time. In other words, we would be pretty happy if our learner was Probably (with probability $(1-\delta)$) going to produce an Approximately Correct ($error_D(h) \leq \varepsilon$) learner (hence the PAC of the name). This is exactly what the PAC definition stipulates, along with some other things about the resources the learner is able to use to do this.

Ok, well, the definition of PAC learning is largely theoretical. In practice, people are less interested in the "time" part of the definition above, and more interested in the number of computations *L* might take to learn. In the real world, time $\approx$ number of computations. So can we say anything else about PAC learning, in terms of number of computations?

It turns out that for a specific class of learners we can actually say a lot more. This class of learners is the class of *consistent learners*.

Definition: A learner *L* is a *consistent learner* if *L* outputs hypotheses that perfectly fit the training data.

For a consistent learner *L,* if *H* is finite we can actually bound the number of training examples needed for a concept to be PAC-Learnable by *L.* More on this to follow, but first we need to recall another couple of definitions:

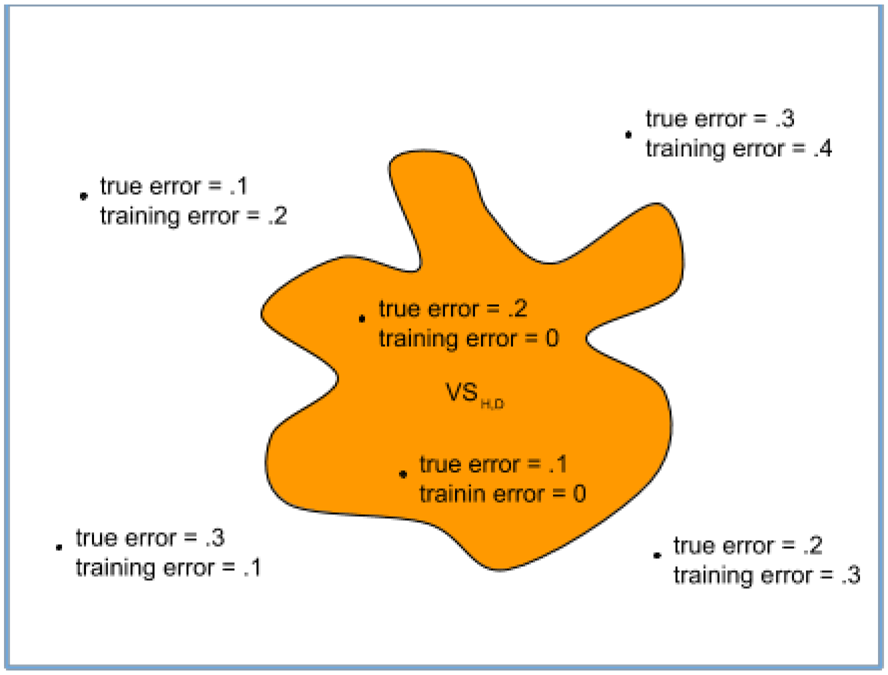
Definition: Let *S* be a set of training examples. The *version space VS(S)* is the set of hypotheses in *H* that perfectly fit the training data:

$$VS(S) = \{h \in H \mid h(x) = c(x) \ \textit{for all}\ x \in S\}$$

Note that for a consistent learner, the version space and hypothesis space are the same.

Definition: Given a hypothesis *H*, target concept *c*, instance distribution *D* and set of training examples *S*, we say that the version space *VS(S)* is $\varepsilon - exhausted$ if every hypothesis in *VS(S)* has true error less than ε:

$$error_D(h) \leq \varepsilon \text{ for every } h \in VS(S)$$



In brief, the definition above just says that *VS(S)* is $\varepsilon - exhausted$ if the hypotheses that perfectly fit the training data (those in the version space) actually have low true error as well. This is a good thing, and we are happy if our well trained hypotheses also do well in the real world. Now we will take a slight detour and consider the definition above. How many examples would it take to (probably) $\varepsilon -$exhaust the version space?

The answer is given by Haussler's Theorem:

Theorem: If H is finite, and m is a sequence of independent randomly drawn training points, then for any $0 \leq \varepsilon \leq 1$, the probability that the version space is not $\varepsilon -$exhausted is bounded above by

$$|H|e^{-\varepsilon m}.$$

<u>Proof:</u> Let $h_1, h_2, \dots, h_k$ denote the hypotheses in H with true error greater than or equal to ε with respect to the target concept c. Our version space will not be $\varepsilon-$exhausted if any one of these hypotheses is included in the version space, so we are actually going to compute the probability that one of these hypotheses happens to be included in the version space.

The probability that one of these hypotheses will be consistent with the target concept (i.e. $h = c$) for a randomly drawn training example is at most $(1-\varepsilon)$. Since the training samples are independently drawn, then the probability that this hypothesis will be consistent with the target concept for m training examples is at most $(1-\varepsilon)^m$. Hence, the probability that any one of the k hypotheses is included in the version space is $k(1-\varepsilon)^m$.

Since we don't really know much in general about k except the fact that $k \leq |H|$, we will generously use this bound. Thus, so far we have

$$P(The\ version\ space\ is\ not\ \varepsilon-exhausted) \leq |H|(1-\varepsilon)^m.$$

Now, using MacLaurin series we see that:

$$log(1-\varepsilon) = -\sum_{k=1}^{\infty}\frac{\varepsilon^k}{k} = -\varepsilon-\frac{\varepsilon^2}{2}-\dots \leq -\varepsilon.$$

Thus, $(1-\varepsilon) \leq e^{-\varepsilon}$. The rest of the proof follows by substitution of this inequality.

End of proof.

So, with Haussler's Theorem, we can now come full circle to the class of consistent learners and PAC learning. Suppose we bound the probability in Haussler's Theorem by some desired level δ:

$$|H|e^{-\varepsilon m} \leq \delta$$

Then with probability $(1-\delta)$ the version space (and hence the hypothesis space for a consistent learner) is $\varepsilon-$exhausted. In other words, with probability $(1-\delta)$ a consistent learner will produce a hypothesis with error $error_D(h) \leq \varepsilon$. This is the very definition of PAC learning, except for the last part of the PAC learning definition about time/number of computations. Rearranging the terms, we have:

$$m \geq \frac{1}{\varepsilon}(\ln|H| + \ln(\frac{1}{\delta}))$$

Note that the bound is polynomial in $\frac{1}{\varepsilon}, |H| \; and \; \frac{1}{\delta}$. So for a fixed ε, δ and $|H|$, the above equation gives us a lower bound for the number of training samples needed for a consistent learner to PAC-learn a concept.