IV PAC Learning.

1. Introduction
2. Learning Rays
3. Probably Approximately Correct (PAC) Learning
4. Learning Rays is PAC
5. Probably Exactly Correct Learning
6. Summary of Lecture

1 Introduction

This lecture is based on chapter 3 of Anthony and Biggs.

We introduce PAC learning, the main learning framework we will study.

PAC learning asks: how many examples do we need to learn a concept Probably Approximately Correctly?

- Approximately – learning a concept close to the target concept is good enough. We will set a desired accuracy level.
- Probably – we're not 100% certain how close to the target concept we are. We will set a desired confidence level.

E.g., I want to be 90% certain that my hypothesis is 95% correct. How many examples do I need to see?

Overview of Lecture

Part 2
We see an algorithm for learning rays, and we use it to illustrate:

- Probability distributions.
- Hypothesis error.
- How sample size contributes to error.
- How the distribution of examples contributes to error.

Part 3

- We define PAC learning, using probability distributions and hypothesis error.

Overview

Part 4

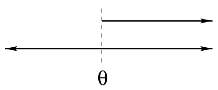
- We prove that learning rays has the PAC property.
- We obtain an explicit formula for the number of examples m needed to reach a given confidence and accuracy.

Part 5

- We define Probably Exactly Correct learning, a special case of PAC for finite example spaces.
- We prove the monomial algorithm is probably exactly correct on finite Boolean example spaces.

2 Learning Rays

A ray is a right half-line with end-point θ.



We can consider each ray a concept r_θ:

$$r_\theta(y) = \begin{cases} 1 & \text{if } y \geq \theta; \\ 0 & \text{otherwise.} \end{cases}$$

which gives us the continuous hypothesis space $H = \{r_\theta \mid \theta \in \mathbf{R}\}$.

Given a series of labelled positive and negative examples drawn from an unknown distribution we want to find a ray r_λ which is as close as possible to r_θ.

1. set $\lambda = \infty$
2. for each example in the training sample
3. if it's positive and its value x_i is $< \lambda$ then $\lambda = x_i$
4. return the hypothesis r_λ

If the training sample is:

$$\big((5.432, 1), (4.801, 1), (3.021, 0), (8.926, 1), (2.387, 1)\big)$$

the sequence of hypotheses will be:

$$r_\infty, r_{5.432}, r_{4.801}, r_{4.801}, r_{4.801}, r_{2.387}$$

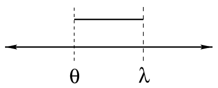
λ Approximates θ

Clearly at all times $\lambda \geq \theta$.

As the number of examples m increases, λ approaches θ.

Since H is continuous, we cannot expect $\lambda = \theta$.

But as m increases, the difference between λ and θ decreases.



How Good is λ?

How many errors will we make if we use λ to approximate θ?

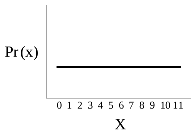
- We will mis–classify any positive example $x < \lambda$.
- The number of such examples depends on how close λ and θ are.
- But it also depends on the distribution of examples.

A *probability distribution* is a function $\mu : X \to [0, 1]$, where the sum of the probabilities of all examples sums to 1.

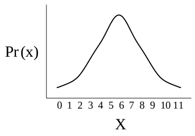
We denote the probability of x as $Pr(x)$.

Some Probability Distributions

Here all values of X have the same probability.



Here medium values of X have higher probability.

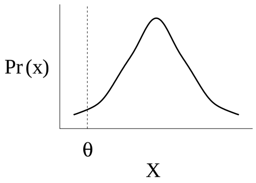


Distribution can make it Hard for λ to Approach θ

The more examples we see, the closer λ should be to θ.

If we've seen a lot of examples, and λ is not close to θ, it is *probably* because of the distribution of examples.

E.g., with this distribution, examples close to θ are rare:



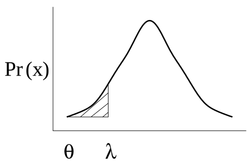
so it is hard for λ to get close to θ.

Distribution Affects how many Errors are Made

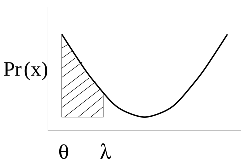
Given:

- The same set of examples for which errors will be made: $[\theta, \lambda)$.
- But different distributions.

Different numbers of errors will be made:



Few Errors



Many Errors

Training and Testing Distributions

On the last slide we saw the distribution affects the number of errors made.

If we train the learner on a distribution where positive examples close to θ are rare, and test it on one where they are common, the error rate will be higher than it would otherwise be.

But if the algorithm trains and tests using the same distribution, this effect does not occur.

Consequently, in PAC learning we will assume the distribution is the same.

Summary of Ray Learning

λ should approximate θ.

- but we expect some error in the approximation.

How much error?

- The size of the training sample influences error.
- The distribution of examples influences error.
- The learner doesn't know the distribution, but if we train and test using the same distribution, error should be small.

3 PAC Learning

What else can we say about the ray learning algorithm? In order to compare it with other algorithms, it would be good to know how efficient it is.

- Efficiency can be measured by the size of the training sample needed to learn the concept.

But we don't expect it to learn the concept exactly.

- So we only ask that it learn a hypothesis which is accurate enough, i.e., with a low enough error. (We want it to be approximately correct.)

And we can't be certain how good the approximation is.

- So we set some confidence level.

Error Sets and Errors

The *error set* of a hypothesis h with respect to a concept t is the set of examples which they classify differently:

$$\{x \in X \mid h(x) \neq t(x)\}$$

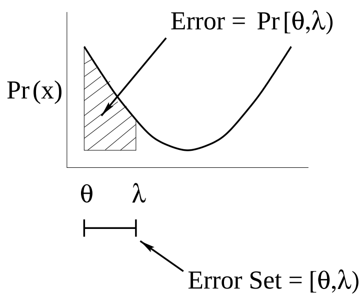
The *error* of a hypothesis h with respect to a concept t is the error set weighted according to the probability distribution μ:

$$\text{error} = er_\mu(h, t) = \mu\{x \in X \mid h(x) \neq t(x)\}$$

Notes:

- To simplify notation we normally write "error" instead of $er_\mu(h, t)$.

Illustration of Error and Error Set



How Accurate should the Hypothesis be?

We use the *accuracy parameter* ϵ $(0 < \epsilon < 1)$ to say how accurate the final hypothesis should be. We want:

$$\text{error} < \epsilon$$

- Smaller ϵ means the hypothesis must be *more* accurate.
- Bigger ϵ means the hypothesis can be *less* accurate.

How Confident are we that the Hypothesis is Accurate?

What is the probability that error $< \epsilon$?

We use the *confidence parameter* δ $(0 < \delta < 1)$ to say how confident we want to be that error $< \epsilon$.

Specifically, we want:

$$\Pr(\text{error} < \epsilon) \ > \ 1 - \delta$$

- Smaller δ means we are *more* confident.
- Bigger δ means we are *less* confident.

PAC Learning

We say an algorithm L is PAC for H if, given:

- a confidence parameter δ $(0 < \delta < 1)$,
- an accuracy parameter ϵ $(0 < \epsilon < 1)$,

there is a positive integer $m_0 = m(\delta, \epsilon)$ such that

- for any target concept $t \in H$, and
- for any probability distribution μ on X

whenever $m \geq m_0$,

$$\Pr(\text{error} < \epsilon) \ > \ 1 - \delta \tag{1}$$

Notation

Anthony and Biggs uses more complex notation to define PAC learnability.

- The error is written: $er_\mu(h, t)$ or just er_μ.
- The hypothesis output by the learning algorithm is written $L(\mathbf{s})$ and its error is written $er_\mu(L(\mathbf{s}))$.
- Just as X has a probability distribution μ, X^m has a probability distribution μ^m.
- The training set of samples of length m for a given target concept t is denoted $S(m, t)$.

Notation (continued)

The PAC condition:

$$\Pr(\text{error} < \epsilon) \ > \ 1 - \delta$$

is written in Anthony and Biggs like this:

$$\mu^m\{s \in S(m,t) \mid er_\mu(L(\mathbf{s})) < \epsilon\} \ > \ 1 - \delta$$

4 Learning Rays is PAC

Theorem The ray-learning algorithm is PAC.

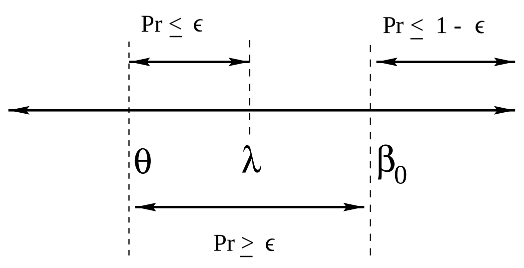
Proof

Suppose $\delta, \epsilon, r_\theta$, and μ are given. Let $\mathbf{s}$ be a training sample of length m for r_θ and let the final hypothesis $= r_\lambda$. Clearly, the error set is the interval $[\theta, \lambda)$. For a given value of ϵ, and the given μ, define:

$$\beta_0 \ = \ \beta_0(\epsilon, \mu) \ = \ \sup\{\beta \mid \mu[\theta, \beta) < \epsilon\}$$

I.e., β is the largest value for which the error $\mu[\theta, \beta)$ is less than ϵ.

Illustration of $\lambda \leq \beta_0$



It follows that $\mu[\theta, \beta_0) \leq \epsilon$ and $\mu[\theta, \beta_0] \geq \epsilon$. So if $\lambda \leq \beta_0$ then

$$\text{error} = \mu[\theta, \lambda) \ \leq \ \mu[\theta, \beta_0) \ \leq \ \epsilon \tag{2}$$

I.e., the error of the learnt hypothesis is less than ϵ.

Note: This just says that:

$$\mu[\theta, \beta_0) \ \leq \ \epsilon$$

and

$$\mu[\theta, \lambda) \ \leq \ \mu[\theta, \beta_0)$$

so

$$\mu[\theta, \lambda) \ \leq \ \epsilon$$

I.e., $B \leq A$ and $C \leq B$ so $C \leq A$.

$\mu[\theta, \beta_0] \geq \epsilon$ is the probability that a positive example occurs in the interval $[\theta, \beta_0]$.

Since all positive examples are greater than θ, the probability that an example is not in this interval is at most $1 - \epsilon$.

The probability that none of m examples occur in this interval is at most $(1-\epsilon)^m$.

The probability that at least 1 example *is* within this interval (i.e., $\lambda \leq \beta_0$) is the complement of the previous event, i.e., $1 - (1-\epsilon)^m$.

Since (as noted in equation (2)) $\lambda \leq \beta_0$ implies error $\leq \epsilon$,

$$\Pr(\text{error} \leq \epsilon) \quad \geq \quad 1 - (1-\epsilon)^m \tag{3}$$

I.e., the probability that the error is less than or equal to ϵ is greater than or equal to $1 - (1-\epsilon)^m$.

Now, according to (1), to show the algorithm is PAC we must choose the number of examples m so that the right hand side of (3) is greater than $1 - \delta$. To do so, let

$$m \geq m_0 = \left\lceil \frac{1}{\epsilon} \ln \frac{1}{\delta} \right\rceil \tag{4}$$

since then

$$(1-\epsilon)^m \quad \leq \quad (1-\epsilon)^{m_0} \quad < \quad \exp(-\epsilon m_0) \quad < \quad \exp(\ln \delta) \quad = \quad \delta$$

Substituting δ for $(1-\epsilon)^m$ in (3) we obtain (1), showing that the algorithm is PAC. □

How Many Examples do we Need?

The proof provides an explicit formula for the number of labelled examples needed by the learning algorithm in order to obtain the specified accuracy and confidence.

Suppose $\delta = 0.001$ and $\epsilon = 0.01$. Then from (4) we have:

$$m_0 \geq \lceil 100 \ln 1000 \rceil = 691$$

In other words, if the training sample contains at least 691 examples, we can be $(1-\delta) = 99.9\%$ confident that at most $\epsilon = 1\%$ of future examples will be misclassified, provided we use the same distribution μ for training and testing.

5 Probably Exactly Correct Learning

In continuous spaces we assume we will never see all positive examples, so we learn a hypothesis which is probably *approximately* correct.

But when the example space X is finite, we can do away with the approximation needed in continuous spaces, and calculate the probability that the output hypothesis is *exactly* correct.

First, redefine X to remove all examples with 0 probability, i.e., let $Y = \{x \in X \mid \mu(x) > 0\}$ and then $Y \to X$.

Let ϵ_μ be the probability of the least likely example:

$$\epsilon_\mu = \min_{x \in X} \mu(x)$$

Now take the definition of a probably approximately correct algorithm (1):

$$\Pr(\text{error} < \epsilon) \ > \ 1 - \delta$$

Setting $\epsilon \leq \epsilon_\mu$ implies the error set is likely to be empty, because there are no examples which have probability less than $\epsilon \leq \epsilon_\mu$.

That is, if we make ϵ small enough, we are likely to see *all* positive examples, in which case a consistent algorithm will output the target concept.

So in finite example spaces, algorithms can be probably *exactly* correct.

One problem is that:

- We choose ϵ based on ϵ_μ.
- So m_0 depends on δ and μ.

This conflicts with the aim of making performance guarantees which are independent of the distribution μ. (See the definition of of PAC algorithm.)

Example: Probably Exactly Correct Learning

Let's establish the 'probably exactly correct' property of the standard monomial learning algorithm on the boolean example space $\{0,1\}^n$, for fixed n, by an explicit argument.

Let $\epsilon_\mu = \min_{x \in X} \mu(x)$ for the set $X \subseteq \{0,1\}^n$ of examples with non-zero probability.

The probability of any given example is at least ϵ_μ.

The probability of m examples not including a given example is at most $(1-\epsilon_\mu)^m$.

The probability of none of a set of p examples appearing in m examples is at most $p(1-\epsilon_\mu)^m$.

If X^+ is the on-set (the set of positive examples) for the target concept t, then the probability that an element of X^+ does not appear during m samples is at most $|X^+|(1-\epsilon_\mu)^m$, where $|X^+|$ denotes the cardinality (number of elements) of X^+.

We need enough examples m so that

$$|X^+|(1-\epsilon_\mu)^m \ < \ \delta$$

Noting that $|X^+| \ \leq \ |X| \ \leq \ 2^n$, and $1 - \epsilon_\mu \ < \ \exp(-\epsilon_\mu)$, we can choose:

$$m \geq \left\lceil \frac{n}{\epsilon_\mu} \ln 2 + \frac{1}{\epsilon_\mu} \ln \frac{1}{\delta} \right\rceil$$

which establishes that the algorithm is probably exactly correct. □

6 Lecture Summary

- In continuous spaces we can only find a hypothesis which is probably approximately correct.
- We assume the learner does not know the distribution from which examples are drawn.
  - The distribution influences the error, and can potentially make learning very difficult.
  - But if we train and test on the same unknown distribution, the effect of the distribution should be minimised.

Summary (Continued)

- If, for some accuracy ϵ and confidence δ, we can show that for any target concept and any probability distribution μ on X that some number of labelled examples suffices to make

$$\Pr(\text{error} < \epsilon) \; > \; 1 - \delta$$

then we have shown the algorithm is PAC.

- If we can show the same in a finite example space, the algorithm is probably *exactly* correct.
- We saw a formula (4) for the number of examples m needed for a given δ and ϵ.