V Consistent Algorithms and Learnability.

1. Introduction
2. Potentially Learnable Hypothesis Spaces
3. Finite Hypothesis Spaces are Learnable
4. Summary of Lecture

1 Introduction

In lecture IV we saw proofs that the algorithm for learning rays is PAC, and that the standard monomial learning algorithm is Probably Exactly Correct.

It would be a lot of work to prove that each algorithm is PAC for each possible H.

Instead, in this lecture we give one proof that many algorithms are PAC for many hypothesis spaces.

Note that we will not cover sections 4.3 and 4.4 on decision lists.

Overview

Part 2

- We define "potentially learnable" hypothesis spaces.
- We show that if H is potentially learnable, *any* consistent algorithm is PAC for H.
- If we can show a space H has this property, then we don't need a specific proof for a specific consistent algorithm.

Part 3

- We show that any finite H is potentially learnable.
- This means if H is finite, any consistent algorithm is PAC for H.
- We obtain an explicit formula for the number of examples needed for a certain level of confidence and accuracy.

2 Potentially Learnable Hypothesis Spaces

We begin with a review:

- An algorithm L is *consistent* if its output hypothesis $L(\mathbf{s})$ agrees with any training sample it is given.
- The *version space* $H[\mathbf{s}]$ is the subset of the hypothesis space H which is consistent with the training sample $\mathbf{s}$:

$$H[\mathbf{s}] = \{h \in H \mid h(x_i) = t(x_i)\ (1 \leq i \leq m)\}$$

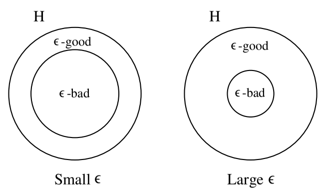
So L is consistent if and only if $L(\mathbf{s}) \in H[\mathbf{s}]$ for all training samples $\mathbf{s}$.

ϵ-bad hypotheses

The set of ϵ-bad hypotheses B_ϵ for a given target concept t consists of those with error at least ϵ.

$$B_\epsilon = \{h \in H \mid \mathrm{error}(h) \geq \epsilon\}$$

The set of ϵ-good hypotheses consists of those which are *not* ϵ-bad.



Consistent Hypotheses can be ϵ-bad

- A consistent hypothesis $h \in H[\mathbf{s}]$ is consistent with the training sample.
- This means it doesn't make any errors on examples in the training sample.
- But when it is tested, it *can* make errors on examples that were not in the training sample. If it makes enough errors it is ϵ-bad.

Potentially Learnable Hypotheses

A hypothesis space H is *potentially learnable* if it is possible to see enough data for the probability that no consistent hypotheses are ϵ-bad to be $> 1 - \delta$.

In other words, H is potentially learnable if it is possible to see enough data so that:

$$\Pr(H[\mathbf{s}] \cap B_\epsilon = \emptyset) \quad > \quad 1 - \delta \tag{1}$$

Anthong and Biggs's Notation

A hypothesis space H is *potentially learnable* if, given $(0 \leq \delta \leq 1)$ and $(0 \leq \epsilon \leq 1)$, there is a positive integer $m_0 = m_0(\delta, \epsilon)$ such that, whenever $m \geq m_0$:

$$\mu^m\{\mathbf{s} \in S(m,t) \mid H[\mathbf{s}] \cap B_\epsilon = \emptyset\} \quad > \quad 1 - \delta$$

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

Note: Recall that $S(m,t)$ is the set of training samples of length m for target concept t.

Theorem 5.1 If H is potentially learnable, and L is a consistent learning algorithm for H, then L is PAC.

Proof

- If L is consistent then $L(\mathbf{s}) \in H[\mathbf{s}]$.
- If H is potentially learnable then $H[\mathbf{s}] \cap B_\epsilon = \emptyset$.
- Therefore, $L(\mathbf{s}) \notin B_\epsilon$ and so $\text{error}(L(\mathbf{s})) < \epsilon$, meaning that L is PAC. □

Same Proof In English

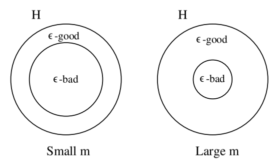
- If L is consistent then its output is in the version space.
- If H is potentially learnable then no hypotheses in its version space are ϵ-bad, i.e., they all have error $< \epsilon$.
- Therefore the hypothesis output by L has error $< \epsilon$, meaning L is PAC. □

3 Finite Hypothesis Spaces are Learnable

Theorem 5.2 Any finite hypothesis space is potentially learnable.

Proof Suppose H is finite and δ, ϵ, t, and μ are given.

To show potential learnability we must satisfy (1). We will show that by making m big enough (seeing enough samples), the probability of some consistent hypothesis having error $\geq \epsilon$ is $< \delta$, in which case the probability of no consistent hypothesis having error $\geq \epsilon$ is $> 1 - \delta$, which shows potential learnability.



Proof Continued

Since the ϵ-bad hypotheses B_ϵ have error $\geq \epsilon$, the probability that an ϵ-bad hypothesis h is correct for an example x is $\leq 1 - \epsilon$. That is,

$$\Pr\big(h(x) = t(x)\big) \;=\; 1 - \text{error}(h) \;\leq\; 1 - \epsilon$$

and the probability h is correct for all m examples in a training set $\mathbf{s}$ is $\leq (1-\epsilon)^m$, i.e.:

$$\Pr\big(h(x_i) = t(x_i) \text{ for all } x \in \mathbf{s}\big) \;\leq\; (1-\epsilon)^m$$

Proof Continued

In other words, the probability that any given ϵ-bad hypothesis is consistent with the training sample (i.e., is in $H[\mathbf{s}]$) is $\leq (1-\epsilon)^m$.

The probability that one or more ϵ-bad hypotheses is in $H[\mathbf{s}]$ is $\leq |H|(1-\epsilon)^m$, where $|H|$ is the number of hypotheses in H. That is:

$$\Pr(H[\mathbf{s}] \cap B_\epsilon \neq \emptyset) \;<\; |H|(1-\epsilon)^m \tag{2}$$

Now we want to choose m so that $|H|(1-\epsilon)^m < \delta$, so we make

$$m \geq m_0 = \left\lceil \frac{1}{\epsilon} \ln \frac{|H|}{\delta} \right\rceil \tag{3}$$

Proof Continued

because in that case:

$$|H|(1-\epsilon)^m \;\leq\; |H|(1-\epsilon)^{m_0} \;<\; |H| \exp(-\epsilon m_0) \;\leq\; |H| \exp\left(\ln(\delta/|H|)\right) \;=\; \delta$$

Substituting δ for $|H|(1-\epsilon)^m$ in (2) yields:

$$\Pr(H[\mathbf{s}] \cap B_\epsilon \neq \emptyset) \;<\; \delta$$

for which the complementary event is:

$$\Pr(H[\mathbf{s}] \cap B_\epsilon = \emptyset) \;>\; 1 - \delta$$

which is (1), showing potential learnability. □

Consequences

We have proved two theorems:

Theorem If H is potentially learnable, and L is a consistent learning algorithm for H, then L is PAC for H.

Theorem Any finite hypothesis space is potentially learnable.

From which we obtain:

Corollary If H is finite and L is consistent for H, then L is PAC for H.

Thus we have reached the goal given at the start of the lecture that we would obtain a general method for showing algorithms are PAC.

Notes

- All Boolean example spaces $X \subseteq \{0,1\}^n$ with fixed n are finite, so for them any consistent algorithm is PAC.
- For example, the algorithms in lecture 2 for learning monomials and disjunctions of small monomials are PAC.
- The algorithm for learning by enumeration from lecture 1 is consistent and applies to any finite hypothesis space. Given theorem 5.2, it follows there is a PAC algorithm for any finite hypothesis space.

A Formula for the Number of Examples Needed

The proof that any finite hypothesis space is potentially learnable gives us an explicit formula (3) for the number of examples needed for prescribed levels of confidence and accuracy.

For example, suppose $H = M_n$. Given that $|M_n| = 3^n$, the number of examples needed to ensure that, with probability greater than $1 - \delta$, a consistent algorithm outputs a hypothesis with error less than ϵ is:

$$m_0 = \left\lceil \frac{1}{\epsilon} \ln \frac{|M_n|}{\delta} \right\rceil = \left\lceil \frac{1}{\epsilon}(n \ln 3 + \ln(1/\delta)) \right\rceil$$

Efficiency

We've just seen that, in principle, learning by enumeration suffices for any finite hypothesis space, but consider how many examples are needed to learn in B_n, the space of all Boolean functions of length n. Since $|B_n| = 2^{2^n}$, the number of samples needed is:

$$m_0 = \left\lceil \frac{1}{\epsilon} \ln \frac{|B_n|}{\delta} \right\rceil = \left\lceil \frac{2^n}{\epsilon} \ln \frac{2}{\delta} \right\rceil$$

Unfortunately, the 2^n means that m_0 grows very quickly. So far we have ignored the issue of efficiency, which obviously needs to be dealt with; whether an algorithm is PAC or not is irrelevant if it takes millions of years to run. Efficiency is the subject of the next lecture.

4 Summary

- We introduced ϵ-bad hypotheses, whose error $> \epsilon$.
- We defined a potentially learnable hypothesis space as one for which some number of samples makes it probable that no consistent hypothesis is ϵ-bad.
- We proved that if hypothesis space H is potentially learnable, and algorithm L is consistent for H, then L is PAC for H.
- We proved that any finite hypothesis space is potentially learnable, and showed that any finite space has a PAC algorithm.
- We saw that we still need to look at the efficiency of algorithms to see when learning is practical.

Interpretation

We proved that any finite hypothesis space is potentially learnable, and showed that any finite space has a PAC algorithm.

- This is a very general result, which applies to any finite H and any consistent learner.
- However, it is also a very weak result:
  - It only says: if H is finite, we only need a finite amount of data (and time) to learn.
  - This does not tell us if it is realistic (or efficient) to learn in H.

Interpretation Continued

Note that the proof is for consistent algorithms.

- Not all algorithms are consistent (e.g. neural nets may forget).
  - However, it can be shown that the proof holds for algorithms which are 'nearly consistent'. (See Anthony and Biggs, p. 36.)
- We have ignored the role of induction, and focused only on memorising the training data.
  - So the formula for the number of examples needed (3) must be a loose approximation of real-world algorithms.