Lesson 3

Probably Approximately Correct

3.1 Working PAC definition

Consider a concept class $C \subseteq P(X)$. C is PAC learnable iff there exists a learning algorithm L with the following property:

- for all $c \in C$,
- for all D over X,
- for all $0 < \epsilon, \delta$,
- there exists $m = f(\frac{1}{\epsilon}, \frac{1}{\delta})$ such that
- f is a polynomial function, and
- after drawing m samples from $EX(c, D)$, the probability that L outputs a hypothesis h with $\textit{error}_{c,D}(h) < \epsilon$ is at least $1 - \delta$.

(Later this definition is modified so that $m = f(\textit{size}(h), \frac{1}{\epsilon}, \frac{1}{\delta})$ where $\textit{size}(h)$ is a measure of the size of the representation of the concept h.)

3.2 PAC analysis of Axis-Aligned Rectangles

Here we prove that the axis-aligned rectangles is PAC learnable by the rectangle learning strategy discussed by Kearns and Vazirani (1994, chapter 1).

Recall that $\textit{error}(h) = \Pr_{x \in D}[c(x) \neq h(x)]$. We want to bound this error by ϵ with probability δ. How do we keep the error smaller than ϵ? The only area that contributes to the error is the region between the target R and the hypothesized rectangle R', which is $R \backslash R'$. So we want the probability associated with $R \backslash R'$ to be less than ϵ.

We divide $R \backslash R'$ into four overlapping strips and consider one strip T'. We want to bound the error associated with T' to be less than $\epsilon/4$ so we can ultimately be sure the whole area will have error less than ϵ.

Consider the strip T whose error under the distribution D equals $\epsilon/4$. If T' properly includes T then the error associated with T' exceeds $\epsilon/4$. If this happens, $\Pr[\textit{error}(h)]$ could be greater than ϵ. So we want to show that the error of T' is bounded by the error associated with T, which equals $\epsilon/4$.

Note that if any point in T appears in S then in fact T includes T'. This is because if a point in T occurs in S then T' only extends as deep as that point since R' includes all positive points. And if T includes T' then the error associated with T' is bounded by $\epsilon/4$.

What is the probability that a point in S is in T (from which it would ultimately follow that $\Pr[\textit{error}(h) < \epsilon]$)? The region T is defined to be the probability that a random draw from $EX(c, D)$ lies in T is $\epsilon/4$. Therefore the probability that a random draw from $EX(c, D)$ does *not* lie in T is $1 - \epsilon/4$. It follows that the probability that none of m random draws lie in T is $(1 - \epsilon/4)^m$. (From which it will ultimately follow that $\Pr[\textit{error}(h) > \epsilon]$ is bounded by that number.)

Since there are 4 overlapping strips like T, the probability that none of m random draws lies in any of the 4 strips is less than $4(1 - \epsilon/4)^m$. Formally, we have established

$$\Pr[\textit{error}(h) > \epsilon] < 4(1 - \epsilon/4)^m \ .$$

We want this probability to be less than δ:

$$\Pr[\mathit{error}(h) > \epsilon] < 4(1-\epsilon/4)^m < \delta\ .$$

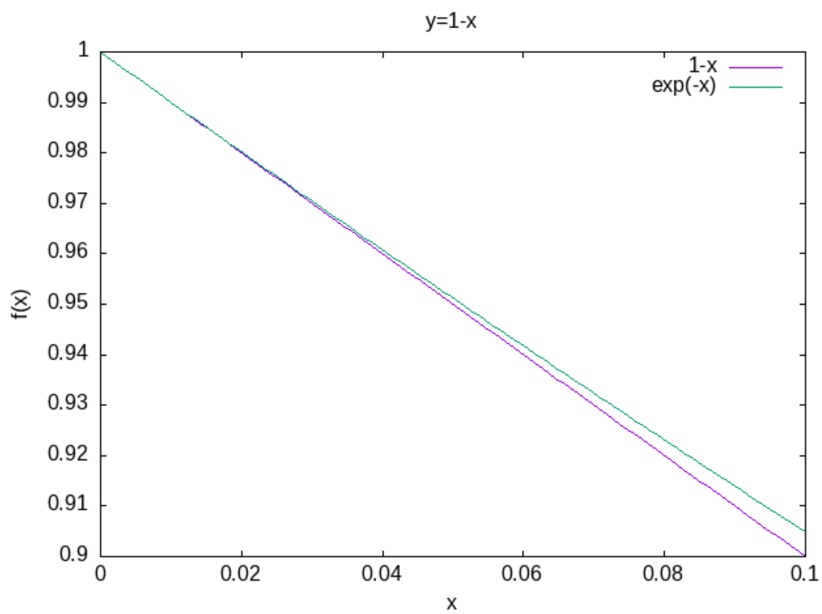
In the region [0,1], it is a fact that

$$1 - x \leq e^{-x}\ .$$

It follows that

$$(1-x)^m \leq e^{-mx}$$

in the same region.



Consequently we have shown the following.

$$\Pr[\mathit{error}(h) > \epsilon] < 4(1-\epsilon/4)^m \leq 4e^{-m\epsilon/4}$$

Thus any value of m which satisfies

$$4e^{-m\epsilon/4} < \delta$$

will also satisfy

$$\Pr[\mathit{error}(h) > \epsilon] < \delta\ ,$$

which is equivalent to

$$\Pr[\mathit{error}(h) < \epsilon] > 1 - \delta\ .$$

Here is a complete derivation for finding such m.

$$\begin{aligned}
4e^{-m\epsilon/4} &< \delta \\
e^{-m\epsilon/4} &< \delta/4 \\
-m\epsilon/4 &< \ln(\delta/4) \\
-m\epsilon/4 &< \ln(\delta) - \ln(4) \\
m\epsilon/4 &> \ln(4) - \ln(\delta) \\
m\epsilon/4 &> \ln(4/\delta) \\
m &> (4/\epsilon)\ln(4/\delta)
\end{aligned}$$

Kearns and Vazirani (1994, p. 6) write "In summary, provided our tightest-fit algorithm takes a sample of at least $(4/\epsilon)\ln(4/\delta)$ examples to form its hypothesis rectangle R', we can assert that with probability at least $1-\delta$, R' will misclassify a new point (drawn according to the same distribution from which the sample was chosen) with probability at most ϵ."

3.3 Monomials

Monomials is another term for the conjunctions of literals. The elimination algorithm Valiant (2013) discusses in Chapter 5 PAC-learns monomials and in this section we see mathematically why that is the case. But first let's review what the concept class is, and how the elimination algorithm proceeds.

3.3.1 Variables and Literals

A literal is either positive (x) or negative ($\bar{x}$, also sometimes written $\neg x$) variable. If there are n variables then there are $2n$ literals. The variables can refer to any property such "has eyes," "more than 100 pounds," or "contains a stressed syllable". The instance space X contains elements a which can be evaluated according to these variables. If a has property x then the positive literal x is true of a otherwise it is false. Conversely, if a does not have the property x then the positive literal x is false of a otherwise it is true.

Suppose we have n variables: $x_1, x_2, \ldots x_n$. The conjunction of literals then is a formula like the following:

$$x_1 \wedge \bar{x_2} \wedge x_4 \ .$$

This means "Elements in X satisfying the formula have property x_1, do not have property x_2 and have property x_4." So each possible formula defines a concept, and every finite set of variables defines a concept class by considering all possible conjunctions of positive and negative literals.

An example familiar to students from computational linguistics 2 would be the Strictly 2-Local languages. These are formal languages that can be described by forbidding finitely many substrings of size 2. For example, if $\Sigma = \{a, b\}$ then the "baba" language $\{ba, baba, bababa, \ldots\}$ is given by forbidding the substrings $\rtimes a$, bb, aa, $b\ltimes$, $\rtimes\ltimes$.

The variables are all the substrings of length 2 drawn from $\{\rtimes\}\Sigma^*\{\ltimes\}$. Each positive literal x is interpreted as "contains the substring x" and negative literal $\bar{x}$ is interpreted as "does not contain the substring x." Thus a monomial describing the "baba" language is shown below.

$$\overline{\rtimes a} \wedge \overline{bb} \wedge \overline{aa} \wedge \overline{b\ltimes} \wedge \overline{\rtimes\ltimes}\ .$$

3.3.2 The Elimination Algorithm

1. Set $h = x_1 \wedge \bar{x_1} \wedge x_2 \wedge \bar{x_2} \wedge \ldots \wedge x_n \wedge \bar{x_n}$
2. Receive $(a, c(a))$ from $EX(c, D)$. If a is a negative example (so $c(a) = 0$) repeat this step; otherwise move on.
3. For each $1 \leq i \leq n$: if x_i is true of a then remove $\bar{x_i}$ from h and if $\bar{x_i}$ is true of a then remove x_i from h. Return to step 2.

Note this process never ends!

Here is an example using the "baba" language. Below is a list of all possible literals of length 2 from $\{\rtimes\}\Sigma^*\{\ltimes\}$.

$$\begin{array}{ccccccccc} \rtimes a & \rtimes b & a\ltimes & b\ltimes & aa & ab & ba & bb & \rtimes\ltimes \\ \overline{\rtimes a} & \overline{\rtimes b} & \overline{a\ltimes} & \overline{b\ltimes} & \overline{aa} & \overline{ab} & \overline{ba} & \overline{bb} & \overline{\rtimes\ltimes} \end{array}$$

If the first positive example returned by $EX(c, D)$ is "ba" (so with word boundaries $\rtimes ba\ltimes$), then the following literals would be removed from h:

$$\rtimes a, \overline{\rtimes b}, \overline{a\ltimes}, b\ltimes, aa, ab, \overline{ba}, bb, \rtimes\ltimes$$

3.3.3 PAC analysis

Like the axis-aligned rectangles, the elimination algorithm only ever considers hypotheses h that cover all the observed positive examples. Furthermore, the literals in h always include the literals in the target c because all the literals are in h at the beginning and they are only removed when they contradict c. Because h includes the literals in c, any negative example will always be classified as negative by h. In other words, h never errs on negative examples.

So a literal (positive or negative) z in h only causes h to err on positive examples a where z is not true of a. The total probability of z not being true of a positive example a is denoted by $p(z)$ defined below.

$$p(z) = \Pr_{a \in D}[c(a) = 1, z \text{ is not true of } a]$$

It follows that

$$\textit{error}(h) \leq \sum_{z \in h} p(z)\ .$$

We would like to bound *error*(h) by ϵ. Since there are at most $2n$ literals z in h, we can achieve this if, for all z, $p(z) \leq \epsilon/2n$.

Call z a bad literal if $p(z) \geq \epsilon/2n$. Consider some bad literal z. The probability that z is removed from h after m calls to $EX(c, D)$ is $(1 - p(z))^m$. Since $p(z)$ is at least $\epsilon/2n$ then this probability is at most $(1 - \epsilon/2n)^m$. Since there are at most $2n$ bad literals, the probability that some bad literal is not removed from h is at most $2n(1 - \epsilon/2n)^m$.

In other words, $\Pr[\textit{error}(h) > \epsilon]$ is bounded by $2n(1-\epsilon/2n)^m$. So if we can find values of m that bound this latter value by δ, then we can conclude that $\Pr[\textit{error}(h) < \epsilon] > 1-\delta$.

Here is a complete derivation for finding such m.

$$\begin{aligned}
2ne^{-m\epsilon/2n} &< \delta \\
e^{-m\epsilon/2n} &< \delta/2n \\
-m\epsilon/2n &< \ln(\delta/2n) \\
-m\epsilon/2n &< \ln(\delta) - \ln(2n) \\
m\epsilon/2n &> \ln(2n) - \ln(\delta) \\
m\epsilon/2n &> \ln(2n/\delta) \\
m &> (2n/\epsilon)\ln(2n/\delta)
\end{aligned}$$

Since m is a polynomial function in terms of n, δ, and ϵ, we see that the elimination algorithm PAC-learns the concepts expressible as the conjunction of positive and negative literals.

In the case of our example, $n = 9$. So if we want to be 99% confident that the probability of the error is less than 1% then

$$m = 18/.01\ln(18/.01) = 1800 \times 7.496 \approx 13492$$

examples suffice for any concept $c \in C$, and for any distribution D over X.

Exercise 1. *What if* $\Sigma = \{a, b, c\}$*?*

Bibliography

Kearns, Michael, and Umesh Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

Valiant, Leslie. 2013. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books.