PAC-learning

Ronald J. Williams
CSG220
Fall 2004

Containing many slides from the Andrew Moore tutorial of the same name.

Probably Approximately Correct (PAC) Learning

- Imagine we're doing classification with categorical inputs.
- All outputs are binary.
- Data is noiseless.
- There's a machine $f(x,h)$ which has H possible settings (a.k.a. hypotheses), called $h_1, h_2 .. h_H$.

Example of a machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm that contain only logical ands.
- Example hypotheses:
- X1 ^ X3 ^ X19
- X3 ^ X18
- X7
- X1 ^ X2 ^ X2 ^ x4 … ^ Xm
- Question: if there are 3 attributes, what is the complete set of hypotheses in f?

Example of a machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm that contain only logical ands.
- Example hypotheses:
- X1 ^ X3 ^ X19
- X3 ^ X18
- X7
- X1 ^ X2 ^ X2 ^ x4 … ^ Xm
- Question: if there are 3 attributes, what is the complete set of hypotheses in f? (H = 8)

True	X2	X3	X2 ^ X3
X1	X1 ^ X2	X1 ^ X3	X1 ^ X2 ^ X3

And-Positive-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm that contain only logical ands.
- Example hypotheses:
- X1 ^ X3 ^ X19
- X3 ^ X18
- X7
- X1 ^ X2 ^ X2 ^ x4 … ^ Xm
- Question: if there are m attributes, how many hypotheses in f?

And-Positive-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm that contain only logical ands.
- Example hypotheses:
- X1 ^ X3 ^ X19
- X3 ^ X18
- X7
- X1 ^ X2 ^ X2 ^ x4 … ^ Xm
- Question: if there are m attributes, how many hypotheses in f? ($H = 2^m$)

And-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm or their negations that contain only logical ands.
- Example hypotheses:
- X1 ^ ~X3 ^ X19
- X3 ^ ~X18
- ~X7
- X1 ^ X2 ^ ~X3 ^ … ^ Xm
- Question: if there are 2 attributes, what is the complete set of hypotheses in f?

And-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm or their negations that contain only logical ands.
- Example hypotheses:
- X1 ^ ~X3 ^ X19
- X3 ^ ~X18
- ~X7
- X1 ^ X2 ^ ~X3 ^ … ^ Xm
- Question: if there are 2 attributes, what is the complete set of hypotheses in f? (H = 9)

True		True
True		X2
True		~X2
X1		True
X1	^	X2
X1	^	~X2
~X1		True
~X1	^	X2
~X1	^	~X2

And-Literals Machine

- Equivalent to what we've called pure conjunctive concept descriptions when the attributes are Boolean
- E.g. X1 ^ ~X3 ^ X19 is equivalent to (X1 = true) ^ (X3 = false) ^ (X19 = true)

And-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm or their negations that contain only logical ands.
- Example hypotheses:
- X1 ^ ~X3 ^ X19
- X3 ^ ~X18
- ~X7
- X1 ^ X2 ^ ~X3 ^ … ^ Xm
- Question: if there are m attributes, what is the size of the complete set of hypotheses in f?

True		True
True		X2
True		~X2
X1		True
X1	^	X2
X1	^	~X2
~X1		True
~X1	^	X2
~X1	^	~X2

And-Literals Machine

- f(x,h) consists of all logical sentences about X1, X2 .. Xm or their negations that contain only logical ands.
- Example hypotheses:
- X1 ^ ~X3 ^ X19
- X3 ^ ~X18
- ~X7
- X1 ^ X2 ^ ~X3 ^ … ^ Xm
- Question: if there are m attributes, what is the size of the complete set of hypotheses in f? (H = 3^m)

True		True
True		X2
True		~X2
X1		True
X1	^	X2
X1	^	~X2
~X1		True
~X1	^	X2
~X1	^	~X2

Lookup Table Machine

- f(x,h) consists of all truth tables mapping combinations of input attributes to true and false
- Example hypothesis:
- Question: if there are m attributes, what is the size of the complete set of hypotheses in f?

X1	X2	X3	X4	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Lookup Table Machine

- f(x,h) consists of all truth tables mapping combinations of input attributes to true and false
- Example hypothesis: →
- Question: if there are m attributes, what is the size of the complete set of hypotheses in f?

$$H = 2^{2^m}$$

X1	X2	X3	X4	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

A Game

- We specify f, the machine
- Nature chooses hidden hypothesis h*
- Nature randomly generates R datapoints
  - How is a datapoint generated?
    1. Vector of inputs $\boldsymbol{x}_k = (x_{k1}, x_{k2}, x_{km})$ is drawn from a fixed unknown distrib: D
    2. The corresponding output $y_k = f(\boldsymbol{x}_k , h^*)$
- We learn an approximation of h* by choosing some h^{est} for which the training set error is 0

Test Error Rate

- We specify f, the machine
- Nature chooses hidden hypothesis h*
- Nature randomly generates R datapoints
  - How is a datapoint generated?
    1. Vector of inputs $\boldsymbol{x}_k = (x_{k1}, x_{k2}, x_{km})$ is drawn from a fixed unknown distrib: D
    2. The corresponding output $y_k = f(\boldsymbol{x}_k, h^*)$
- We learn an approximation of h* by choosing some h^{est} for which the training set error is 0
- For each hypothesis *h* ,
- Say *h* is consistent if *h* has zero training set error: TRAINERR(*h*) = 0
- Define TESTERR(*h*)
  - = Fraction of test points that *h* will classify incorrectly
  - = P(*h* classifies a random test point incorrectly)
- Say *h* is bad if TESTERR(*h*) > ε
- Otherwise, say *h* is ***approximately correct***

Test Error Rate

- We specify f, the machine
- Nature chooses hidden hypothesis h*
- Nature randomly generates R datapoints
  - How is a datapoint generated?
    1. Vector of inputs $\boldsymbol{x}_k = (x_{k1}, x_{k2}, x_{km})$ is drawn from a fixed unknown distrib: D
    2. The corresponding output $y_k = f(\boldsymbol{x}_k, h^*)$
- We learn an approximation of h* by choosing some h^{est} for which the training set error is 0
- For each hypothesis *h* ,
- Say *h* is consistent if *h* has zero training set error: TRAINERR(*h*) = 0
- Define TESTERR(*h*)
  - = Fraction of test points that *h* will classify incorrectly
  - = P(*h* classifies a random test point incorrectly)
- Say *h* is bad if TESTERR(*h*) > ε
- Otherwise, say *h* is ***approximately correct***

Let's consider a worst-case scenario: Among all consistent hypotheses, if any one is bad, then there's a danger that that's somehow the one we end up learning.

How probable is it that there is even one such consistent yet bad hypothesis?

$$P(\text{we learn a bad } h)$$

$$\leq P\left(\exists h \mid h \text{ is consistent} \wedge h \text{ is bad}\right)$$

$$= P\begin{pmatrix} (h_1 \text{ is consistent} \wedge h_1 \text{ is bad}) \vee \\ (h_2 \text{ is consistent} \wedge h_2 \text{ is bad}) \vee \\ \vdots \\ (h_H \text{ is consistent} \wedge h_H \text{ is bad}) \end{pmatrix}$$

$$\leq \sum_{i=1}^{H} P\left(h_i \text{ is consistent} \wedge h_i \text{ is bad}\right)$$

$$\leq \sum_{i=1}^{H} P\left(h_i \text{ is consistent} \mid h_i \text{ is bad}\right)$$

Bounding the probability of learning a bad hypothesis

- What is $P(h_i \text{ is consistent} \mid h_i \text{ is bad})$?
- Note that if h_i is a bad hypothesis, then the probability it classifies any single training example correctly is ≤ 1-ε.
- Then, using the i.i.d. assumption, the probability it classifies all *R* training examples correctly is ≤ $(1-\varepsilon)^R$.
- Therefore we have shown that

$$P(h_i \text{ is consistent} \mid h_i \text{ is bad}) \le (1-\varepsilon)^R$$

for any *i* .

Bounding the prob. of a bad hypothesis

- Thus

$$\begin{aligned} P(\text{we learn a bad } h) &\le \sum_{i=1}^{H} P(h_i \text{ is consistent} \mid h_i \text{ is bad}) \\ &\le \sum_{i=1}^{H} (1-\varepsilon)^R \\ &= H(1-\varepsilon)^R \end{aligned}$$

- We can combine this with the fact that 1-ε ≤ $e^{-\varepsilon}$ to conclude

$$P(\text{we learn a bad } h) \le H(1-\varepsilon)^R \le He^{-\varepsilon R}$$

Probably Approximately Correct

- Suppose we want the probability to be at least 1-δ that the *h* we learn is not bad.
- A sufficient condition is that

$$\delta \geq He^{-\varepsilon R}$$

- If H, R, δ, and ε satisfy this relationship, then with probability ≥ 1-δ we are assured that the test error rate of the h we learn is ≤ ε.
- The h we learn is probably (with probability ≥ 1-δ) approximately (with error rate ≤ ε) correct.

PAC Learning

Two ways to use a sufficient condition like

$$\delta \geq He^{-\varepsilon R}$$

1. Given that we've found a consistent hypothesis h^{est} for a training set of size *R*, how confident are we that its test error rate is no worse than some given ε? Like confidence intervals in statistical parameter estimation theory.

PAC Learning

Two ways to use a sufficient condition like

$$\delta \geq H e^{-\varepsilon R}$$

1. Given that we've found a consistent hypothesis h^{est} for a training set of size R, how confident are we that its test error rate is no worse than some given ε? Like confidence intervals in statistical parameter estimation theory.
2. Sample complexity: Given δ and ε, how large must R be to guarantee that, with probability at least 1- δ, h^{est} has a test error rate no worse than ε? Get an answer by solving for R:

$$R \geq \frac{1}{\varepsilon}\left(\ln H + \ln \frac{1}{\delta}\right)$$

PAC in action

<table>
<tr><th>Machine</th><th>Example Hypothesis</th><th>H</th><th>R sufficient to PAC-learn</th></tr>
<tr><td>And-positive-literals</td><td>X3 ^ X7 ^ X8</td><td>2^m</td><td>$\frac{1}{\varepsilon}\left(m \ln 2 + \ln \frac{1}{\delta}\right)$</td></tr>
<tr><td>And-literals</td><td>X3 ^ ~X7</td><td>3^m</td><td>$\frac{1}{\varepsilon}\left(m \ln 3 + \ln \frac{1}{\delta}\right)$</td></tr>
<tr><td>Lookup Table</td><td><table>
<tr><th>X1</th><th>X2</th><th>X3</th><th>X4</th><th>Y</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table></td><td>2^{2^m}</td><td>$\frac{1}{\varepsilon}\left(2^m \ln 2 + \ln \frac{1}{\delta}\right)$</td></tr>
<tr><td>And-lits or And-lits</td><td>(X1 ^ X5) v
(X2 ^ ~X7 ^ X8)</td><td>$(3^m)^2 = 3^{2m}$</td><td>$\frac{1}{\varepsilon}\left(2m \ln 3 + \ln \frac{1}{\delta}\right)$</td></tr>
</table>

Extensions to PAC Analysis

- What if our learner does not produce a hypothesis with TRAINERR(h) = 0 (perhaps because of noisy data or limited representational power)? More generally, say h is a bad hypothesis if TESTERR(h) > TRAINERR(h) + ε.
- In this case it turns out that the corresponding probability of learning a bad hypothesis is bounded by

$$He^{-2\varepsilon^2 R}$$

- Thus to guarantee with probability at least 1-δ that TESTERR(h) ≤ TRAINERR(h) + ε, it is sufficient to have a training set of size

$$R \geq \frac{1}{2\varepsilon^2}\left(\ln H + \ln\frac{1}{\delta}\right)$$

Extensions to PAC Analysis

- What if our hypothesis space is infinite?
- E.g.
  - perceptrons
  - multilayer neural networks
  - support vector machines
- In this case the bounds we've given are useless.
- Can we still bound the probability that TESTERR(h) ≤ TRAINERR(h) + ε for given ε?

Extensions to PAC Analysis

- What if our hypothesis space is infinite?
- E.g.
  - perceptrons
  - multilayer neural networks
  - support vector machines
- In this case the bounds we've given are useless.
- Can we still bound the probability that TESTERR(*h*) ≤ TRAINERR(*h*) + ε for given ε?
- Perhaps surprisingly, the answer is YES, at least in many situations
- Magic words: VC (Vapnik-Chervonenkis) dimension

Remarks

- This form of analysis makes no assumption about the underlying distribution of examples – just assumes same one used for both training and testing. Therefore valid for *any* distribution.
  - Distribution free.
- The lower bounds we've computed on the sample complexity are sufficient but not necessary for PAC-learning. But there are corresponding results providing lower bounds on the number of training examples necessary for PAC-learning with certain distributions.

Remarks

- The underlying randomness in this theory is based on the randomness in the training sample
- The bounds derived from this theory are very conservative, for several reasons:
  - designed to handle any distribution of examples, including worst-case
  - derivation in PAC case, for example, based on bounding the prob. that there is ***any*** *h* that is both consistent and bad – when we select one, it could easily be better than this worst-case one

Questions to test your understanding of our PAC analysis:

1. What can be said about the *best-case* consistent hypothesis?
2. Can you see how to easily make a very, very slight improvement in the bound we derived on the probability of learning a bad *h*?

What you should know

- Be able to understand every step in the math that gets you to

$$P(\text{we learn a bad } h) \leq H(1-\varepsilon)^R \leq He^{-\varepsilon R}$$

- Understand that you thus need this many records to PAC-learn a machine with *H* hypotheses

$$R \geq \frac{1}{\varepsilon}\left(\ln H + \ln\frac{1}{\delta}\right)$$

- Understand examples of deducing *H* for various machines

What you should know

- Understand the generalization to nonzero training error, where having this many records is sufficient to guarantee with high probability that TESTERR(*h*) is not much worse than TRAINERR(*h*) when learning a machine with *H* hypotheses:

$$R \geq \frac{1}{2\varepsilon^2}\left(\ln H + \ln \frac{1}{\delta}\right)$$