PAC Model of Learning:

There is a distribution D on $\{0, 1\}^n$ $(\mathbb{R}^n)$
with function class C = {decision trees of size s}

A Learner (runs in polynomial time) receives:

Fix $c \in \mathcal{C}$, c is the unknown decision tree that we want to learn.

script

Receives (x^1, y^1), $x \sim D$, and $y = c(x)$.

drawn from probability Distribution D

(x^2, y^2)

$\vdots$

(x^n, y^n) and $y^i = c(x^i)$

classifier/hypothesis.

Goal: Learner output $h \in C$

$$\Pr_{x \sim D}\left[h(x) \neq c(x)\right] \leq \epsilon$$

something small.

Learner should be efficient: (n, s) runtime.

Goal of PAC

with probability "Drawn over distribution D" at least $1-\delta$, the learner should output a hypothesis h s.t.

$$\Pr_{\substack{x \sim D}}[\underset{\uparrow \text{ our model}}{h(x)} \neq \underbrace{c(x)}_{\text{theoretical best model drawn from } C}] \leq \epsilon$$

Runtime = polynomial $\left(\frac{1}{\epsilon}, \frac{1}{\delta}, n, s\right)$.

Why need $\overbrace{1-\delta}^{\text{p of success}}$? Imagine learner requests $x^1, \ldots, x^m$, but receives the same training example every time, so no way the error will be small. So we have to allow for some probability of failure. The probability of seeing the same example over and over again is very small.

PAC = Probably Approximately Correct, L. Valiant [1984].

δ = failure probability. If you want an algorithm that has smaller + smaller probability of inaccuracy, ϵ, you get to run in more time + use samples.

This works for any binary classification (boolean).
↑ PAC Learning.

Probably: confidence $1-\delta$

Approximately Correct: $1-\epsilon$

When can we PAC learn a function class?
What function classes can we PAC learn?
Give the learner an algorithm A.

A : training sets $\xrightarrow{\text{map}}$ decision trees.

A(S) output a tree T that is consistent with S.
Size of T is going to be at most s.
# of nodes. little s

A always outputs a consistent hypothesis from $\mathcal{C}$ given any training set (assuming there is one).
script C

<u>Question</u>: Given algorithm A: how can we PAC learn C?
- Draw sufficiently many training points.
- Use A to find $c \in \mathcal{C}$ consistent with S.
- output c. little c

function class / script C.

Q: How large should S (training set) be?

Example: Jar 1: all blue marbles. | Jar 2: 90% red 10% blue.

Goal: Figure out if given Jar 1 or 2.

Case 1: red marble → Jar 2.
Case 2: blue marble → probably Jar 1.
draw 100 total.

$\Pr[\text{failure}] = (0.1)^{100} =$ "δ - parameter."

Let's Return to PAC learning:

- Draw many samples S
- Run A (algorithm)
- output classifier c that is consistent with S given from A.

What is the probability this procedure fails? $\leq \delta$

Bad Event: we output $c \in C$ that is consistent with S but the true error is greater than ϵ.

P[Bad Event]? Let's imagine that we have enumerated all functions in $C = \{c_1, c_2, \cdots, c_n\}$

Fix c_1. Assume c_1 has true error $> \epsilon$.

What is $\Pr_S[c_1 \text{ is consistent with } S]$?

Answer: $\leq \underbrace{(1-\epsilon)}_{\text{success}}{}^{|S|}$ ← size of S (same as marble example: draw the wrong example)

Fix c_2. " c_2 has " $> \epsilon$.

" $\Pr_S[c_2$ " $]$? Also $\leq (1-\epsilon)^{|S|}$.

For every c_i (with error $> \epsilon$)

$\Pr_S[c_i \text{ is consistent on } S] \leq (1-\epsilon)^{|S|}$.

Q: Randomly form S, what is the prob there exists a function $c \in \mathcal{C}$ whose error $> \epsilon$ and is Consistent with S?

Use the union bound: $P(A \cup B) \leq P(A) + P(B)$. want the bound to be less than δ.

$$Pr[\underbrace{\text{Bad Event}}_{\text{There is } c \in \mathcal{C} \text{ that is consistent with } S \text{ but has true error} > \epsilon}] \leq \underbrace{|C|}_{\text{at most script C functions to consider}} \cdot (1-\epsilon)^{|S|} \leq \delta$$

There is $c \in \mathcal{C}$ that is consistent with S but has true error $> \epsilon$.

at most script C functions to consider. So we can just add up the failure probabilities. This is worse case scenario.

Solve for $|S|$: use property $(1+x) \approx e^{x}$, $(1-x) \approx e^{-x}$

$$P[\text{Bad Event}] = |C| \cdot (1-\epsilon)^{|S|} \leq \delta$$

$$|C| \cdot e^{-\epsilon|S|} \leq \delta$$

$$e^{-\epsilon|S|} \leq \frac{\delta}{|C|}$$

$$-\epsilon|S| \leq \ln\left(\frac{\delta}{|C|}\right)$$

$$\boxed{|S| \geq \frac{\ln\left(\frac{|C|}{\delta}\right)}{\epsilon}}$$

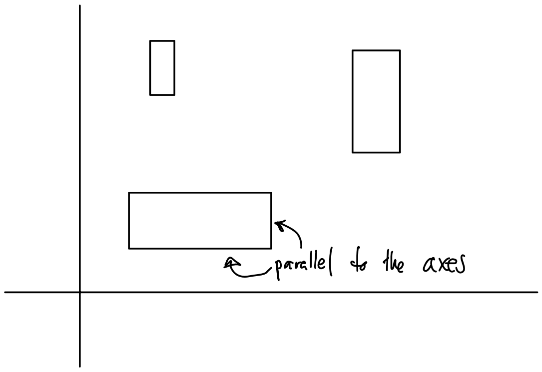
$\therefore$ If you choose # training points larger than $\dfrac{\ln\left(\frac{|C|}{S}\right)}{\epsilon}$, then with probability $\geq 1-\delta$, function output c is at least $(1-\epsilon)$ accurate.

$\Rightarrow$ suggest polynomial size bounds.

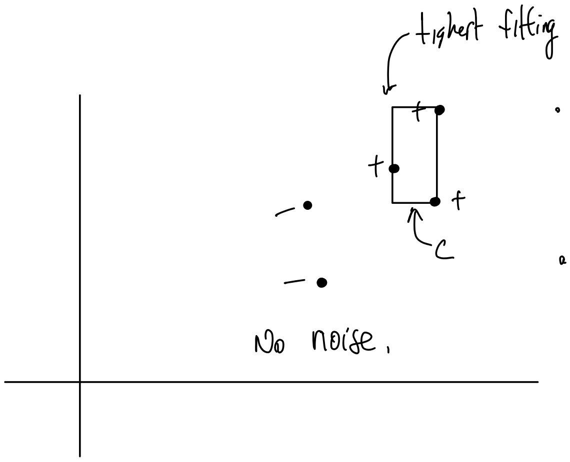
$\Rightarrow$ Suggest a "consistent hypothesis" approach to learning.

Infinite Function

PAC - learning. Axis - parallel rectangles.



Infinite many axis-parallel rectangles.



- labelled + if the point is inside C, unknown axis-parallel rectangle.
- labelled − if the point is outside C.

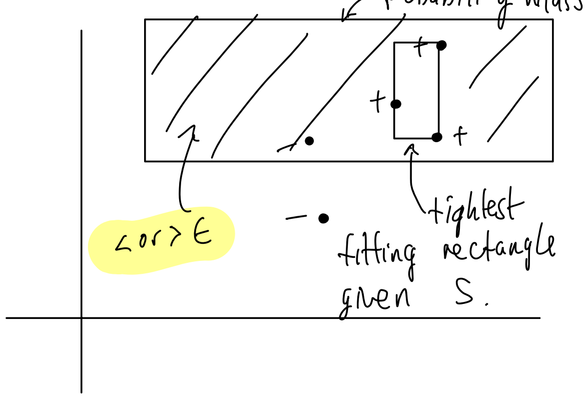
Goal: Given ϵ, δ, output h that is ϵ-accurate with prob $\geq (1-\delta)$.

Claim: the tightest fitting rectangle works. Tightest fitting rectangle containing all the positive points in the set S.

Question: How large to choose $|S|$?

training set

probability mass of true rectangle (correct theoretical function).

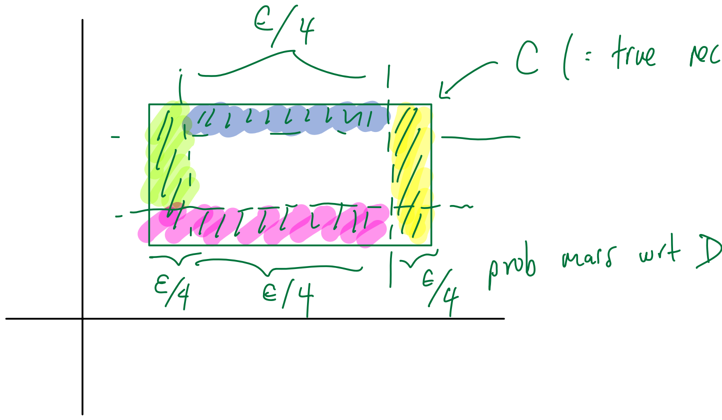


Bad Event: tightest fitting rectangle is small, i.e. lots of probability mass of the true rectangle exists outside of h, the tightest fitting rectangle.

p(Bad Event) = shaded region. If p(Bad Event) is $\geq \epsilon$, then trouble.

How do we bound the probability this happens?

analyze h (tightest fitting rectangle) and say something about h vs all rectangles that are large and contain h.



Intuition: taking more examples/training data decreases the probability for Bad Event.

Bad Events = $B_1 + B_2 + B_3 + B_4$.

B_1 is the event we see no points in the right strip.
B_2 is event we see no point in the bottom strip.
B_3 " left "
B_4 " top "

jar example.

Claim: If neither B_1, B_2, B_3, B_4 occur, then h, which is the tighest fitting rectangle, is ϵ - accurate.

What does it mean?

True error of the tighest fitting rectangle is going to be at most ϵ as long as none of the bad events occur.

Claim: If choose m random samples

$$\Pr[B_1] \leq \left(1 - \frac{\epsilon}{4}\right)^m \quad \text{(marble jar example)}.$$

$\uparrow$ no point in right strip. $\uparrow$ each draw independent.

$$\Pr[B_1 \cup B_2 \cup B_3 \cup B_4] \leq 4 \cdot \left(1 - \frac{\epsilon}{4}\right)^m$$

want it $\leq \delta$: $4\left(1 - \frac{\epsilon}{4}\right)^m \leq \underbrace{\delta}_{1-\delta \text{ accurate.}}$

$$\left(1 - \frac{\epsilon}{4}\right)^m \leq \frac{\delta}{4} \qquad 1 + x \approx e^x \quad 1 - x \approx e^{-x}$$

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

$$-\frac{\epsilon m}{4} \leq \ln\left[\frac{4}{\delta}\right]$$

$$\Rightarrow \boxed{m \geq \frac{4 \ln\left[\frac{4}{\delta}\right]}{\epsilon}}$$

h, the tighest triangle, will be ϵ - accurate with probability $\geq (1 - \delta)$.

C = Half Spaces

← function class

- $C = \{\text{half spaces}\}$ = dividing n-dimension Euclidean spaces into half-spaces.

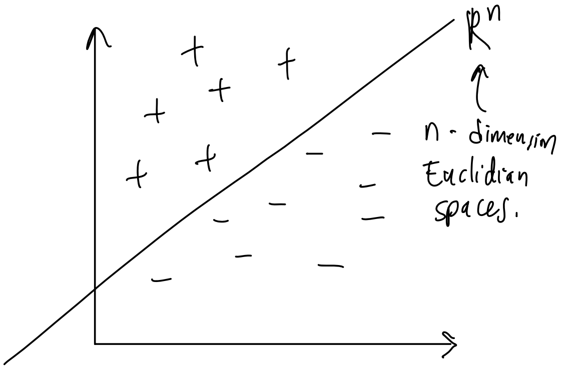
↑ infinite.

$$f = \text{sign}(w \cdot x - \theta)$$

θ: R. Unknown. ↑ scalar.

$w \in R^n$ unknown, $x \in R^n$ ← vector

f is boolean. 0 if output is negative, 1 if output is positive.



Want: Come up with PAC learning algorithms for half spaces.

One Approach for Learning Half Spaces:

$w \in R^n$ is unknown

$\theta \in R^n$ is unknown

$$f = \text{sign}\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

Given draws from $\mathcal{D}$ $(x, f(x))$

x

$(0\,1\,0\,1\,0,\ \text{pos}) \rightarrow w_2 + w_4 > \theta$

$(0\,1\,1\,0,\ \text{neg}) \rightarrow w_2 + w_3 \leq \theta$

Assume w_i is integer in some bounded range.

Each labeled example $\rightarrow$ linear inequalities.
System of linear inequalities.
Can we find a consistent hypothesis?

Use: General-purpose tool called linear programming.

Perceptron: algorithm for learning half-spaces.

Read: Ch 3 of textbook.