mood-book

UNIT - III

* BAYES THEOREM :-

Bayes theorem gives probability of an event based on prior knowledge of conditions (Previous knowledge)

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$P(A|B)$ → hypothesis (posterior prob.)

$P(B|A)$ → likelihood

$P(B)$ → marginal

$P(A)$ → prior

Proof: Let us take 2 events : A & B

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$P(A|B)$ – Conditional prob.

↓

Means prob. of A at given prob. of B

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

$$P(A|B) \cdot P(B) = P(A \cap B) \quad — (1)$$

$$P(B|A) \cdot P(A) = P(B \cap A) \quad — (2)$$

$P(A \cap B)$ → Common in A & B.

(1) & (2) are equal, since $P(A \cap B) = P(B \cap A)$

$$\therefore P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

$$\boxed{P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}}$$

A - Hypothesis , B - Given data

$P(A|B)$ – Finding prob of hypothesis when prob. of training examples given. da

$P(B|A)$ – Find prob. of given data provided with prob. of hypothesis that is true.

P(A) = Prob. of hypothesis before Considering the given data.

P(B) = Prob. of given data.

Example: 1) P(King | face) → Prob. of King given that it is a face card.

↓

J, K, Q

Heart, ACE, SPADE, DIAM

3 3 3 3

4 × 3 = 12 face Cards.

$$P(\text{King} \mid \text{face}) = \frac{P(\text{face} \mid \text{king}) \cdot P(\text{King})}{P(\text{face})}$$

P(face/king) = 4/4 = 1

face kings = 4

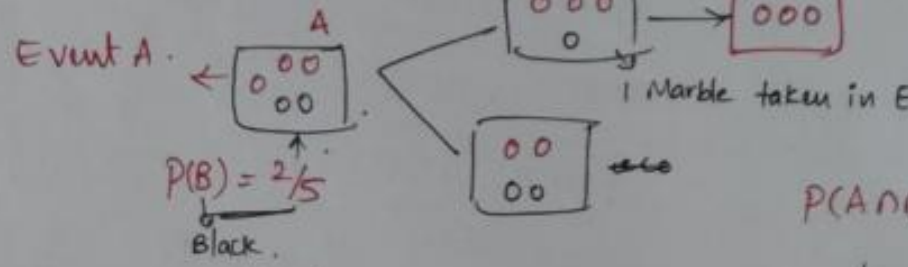
4 ← 4

$$\frac{1 \times \frac{4}{52}}{\frac{12}{52}} = \frac{1 \times \frac{4}{52}}{\frac{3}{13}} = 1/3.$$

2)

Prob. of black marble at given event A.

P(B|A) 1/4



P(B) = 2/5

Black.

P(B|A) = 1/4

P(A ∩ B) = 2/5 × 1/4

↓ = 1/10

Combined prob. of events A & B.

* Bayes theorem AND Concept theorem Learning:

→ what is the relation between Bayes theorem and Concept learning?

→ Bayes theorem calculates the probability of each possible hypothesis and outputs the most probable one.

↓ Highest probability.

* Bruteforce Bayes Concept learning:

1) For each hypothesis in H calculate posterior probability

$$P(h/D) = \frac{P(D|h)\,P(h)}{P(D)} \quad \text{—— (1)}$$

h - hypothesis
D - Set of training examples

2) output the hypothesis

hmap (hML) - maximum likelihood with highest probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\ P(h/D)$$

H - hypothesis space

To calculate, we need values of P(h) and P(D|h)

Some assumptions,

1) Training data D is noise free (No irrelavant data)
2) Target Concept c is present in hypothesis Space H
3) we have no prior reason to believe that any hypothesis is more probable than any other.

$$P(h) = \frac{1}{|H|} \quad \text{for all } h \text{ in } H$$

$$P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

from ① $P(h/D) = \frac{P(D/h)\,P(h)}{P(D)}$

Case 1): Hypothesis h is inconsistent i.e $P(D/h) = 0$.

$\therefore P(h/D) = \frac{0 \times P(h)}{P(D)} = 0$ $[\because d_i \neq h(x_i)$ So $P(D/h) = 0]$

Case 2): Hypothesis is Consistent i.e $P(D/h) = 1$ Consistent means

$P(h/D) = \frac{1 \times P(h)}{P(D)} = \frac{\frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}}$ $[\because d_i = h(x_i)$ So $P(D/h) = 1]$

$= \frac{1}{|VS_{H,D}|}$ $[\because P(D) = \frac{|VS_{H,D}|}{|H|}$

Picking version spaces from overall hypothesis H]

$\therefore P(h/D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is Consistent with } D \\ 0 & \text{otherwise.} \end{cases}$

V.S. (H)

* Maximum Likelihood And Least Squared Error Hypothesis:

To find the maximum likelihood hypothesis in bayesian learning,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\ p(h/D)$$

P – Prob. density func.

→ Let us take training instances $(x_1, x_2 \dots x_n)$ and consider target values $D = (d_1, d_2, \dots d_m)$ we write $P(D/h)$ as product of $\underbrace{P(d_i/h)}$ Prob. of each instance at given h.

$$h_{MAP} = \underset{h}{\operatorname{argmax}} \prod_{i=1}^{m} P(d_i/h)$$

By assuming normal distribution,

$$f(x/\mu) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

x – instance
μ – Mean
σ – Standard deviation

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2}$$

$$= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{1}{2\sigma^2}(d_i-h(x_i))^2}$$

[$\because f(x/\mu) = \dots$]

→ use ln function (log. fun) → Π becomes Σ.

$$= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i-h(x_i))^2$$

↓ no 'i' term so const

$$= \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} (1) - \frac{1}{2\sigma^2}(d_i-h(x_i))^2$$

↓ becoz of -ve Symbol argmax becomes argmin.

$$= \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

$$= \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2 \quad [\because \tfrac{1}{2\sigma^2} \text{ is const}].$$

$$\therefore h_{MAP} \text{ or } h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

∴ The maximum likelihood hypothesis is the one which has minimum squared error between the hypothesis. (Least squared error)

* Minimum Description Length Principle :-

Assumption:

Representing a concept in minimum possible way
- Then it is said to be good one.

Mathematically,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \; P(D|h)\, P(h)$$

Applying logarithm,

$$\underset{h \in H}{\operatorname{argmax}} \log P(D|h) + \log P(h) \quad [\because \log(ab) = \log a + \log b]$$

$$\underset{h \in H}{\operatorname{argmin}} [-\log P(D|h) - \log P(h)].$$

(minimum length / short hypothesis is preferred)

Example:

1) Let us Consider a probability of designing a Code to transmit messages drawn at random from a set D where probability of drawing an i^{th} message = P_i

2) While transmitting, we want a Code that minimizes the expected no. of bits.

To do this, we should assign shorter Codes to the most probable.

we represent the length of message i with respect to 'C' as $L_C(i)$ (Length) i - msg. , $L_C(1)$ - first msg. $L_C(2)$ - second msg.

$$\therefore h_{MAP} = \arg\min L_{C_H}(h) + L_{C_{D/h}}(D/h)$$

$C_H \rightarrow$ optimal encoding for H

$C_{D/h} \rightarrow$ Optimal encoding for D given h.

$$\therefore h_{MDL} = h_{MAP}.$$

* BAYES OPTIMAL CLASSIFIER :

Bayes optimal classifier is a probabilistic model that makes the most probable prediction for a new example.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

for a dataset, $\{y\}$ → yes/No.

$$x = \{x_1, x_2, x_3 \text{ ---- } x_n\}$$

$$P(y|x_1 x_2 \text{ --- } x_n) = [P(x_1|y) \cdot P(x_2|y) \text{ --- } P(x_n|y)] \times P(y)$$

for a dataset,

$$x = \{x_1, x_2, x_3, \ldots x_n\} \{y\}$$

$$P(y \mid x_1, x_2, \ldots x_n) = \frac{[P(x_1|y) \cdot P(x_2|y) \ldots P(x_n|y)] \times P(y)}{P(x_1) \cdot P(x_2) \ldots P(x_n)}$$

$$= \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1) P(x_2) \ldots P(x_n)}$$

$$\Rightarrow P(y) \prod_{i=1}^{n} P(x_i|y)$$

Suppose 10 samples.
$x_1 \to 2$, $x_2 \to 4$
$P(x_1) \to \frac{2}{10}$ $P(x_2) \to \frac{4}{10}$
we are getting optimal sol but the value, so can eliminate constant term

Example :

* outlook * temperature.

	Yes	NO	P(Y)	P(N)
Sunny	2	3	2/9	3/5
Outcast	4	0	4/9	0/5
rain	3	2	3/9	2/5
Total	9	5	100%	100%

* temperature.

	Yes	NO	P(Y)	P(N)
Hot	2	2	2/9	2/5
mild	4	2	4/9	2/5
Cold	3	1	3/9	1/5
Total.	9	5	100%	100%

* play

Yes	9	9/14
NO	5	5/14
Total	14	100%

III (5)

Total (Sunny, hot)

$$P(\text{Yes} \mid \text{sunny, hot}) = \frac{P(\text{Sunny} \mid \text{yes}) \times P(\text{Hot} \mid \text{Yes})}{\text{Total (Sunny, hot)}} \times P(\text{Yes})$$

$$= \frac{2}{9} \times \frac{2}{9} \times \frac{9}{14} = 0.031$$

$$P(\text{NO} \mid \text{Sunny, hot}) = P(\text{Sunny} \mid \text{NO}) \times P(\text{hot} \mid \text{NO}) \times P(\text{NO})$$

$$= \frac{3}{5} \times \frac{2}{5} \times \frac{5}{14} = 0.08571$$

$$\text{Total} = 0.031 + 0.08571 = 0.27 \text{ //.}$$

$$P(\text{Yes}) = \frac{0.031}{0.27} = 0.114$$

$$P(\text{No}) = \frac{0.08571}{0.27} = 0.317$$

Probability of NO is more, therefore player will not enjoysport.

GIBS Algorithm :-

1) Chooses one hypothesis at random, according to $P(h|D)$
2) Use this to classify new instance.

$$E[\text{error}_{gibs}] \leq 2E[\text{error}_{Bayesoptimal}]$$

→ Why GIBS :-

Bayesian optimal classifier will give best results, but need more hypothesis so more expensive.
So we go for GIBS algorithm with the same proce and also the error we get in GIBS algorithm will be ≤ 2 (error in Bayesian optimal classifier).

* NAIVE BAYES CLASSIFIER :-

- classification technique based on Bayes theorem with an assumption of independence among features.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

Example :- Prb: fruit = {yellow, sweet, long}

Fruit	Yellow	sweet	long	total	
orange	350	450	0	650	{∵ repeated fruit}
Banana	400	300	350	400	
others	50	100	50	150	
total	800	850	400	1200	

→ We need to find/pick more Yellow, more Sweet & more long/lengthy one from the table.

Sol:

$$P(\text{Yellow}/\text{orange}) = \frac{P(\text{orange}/\text{Yellow})\,P(\text{Yellow})}{P(\text{orange})}$$

$$= \frac{\frac{350}{800} \times \frac{800}{1200}}{\frac{650}{1200}} = 0.53 \text{ //}$$

$$P(\text{Sweet}/\text{orange}) = \frac{P(\text{orange}/\text{sweet})\,P(\text{Sweet})}{P(\text{orange})}$$

$$= \frac{\frac{450}{850} \times \frac{850}{1200}}{\frac{650}{1200}} = 0.69 \text{ //}$$

$$P(\text{Long}/\text{orange}) = \frac{P(\text{orange}/\text{long})\,P(\text{long})}{P(\text{orange})}$$

$$= \frac{0 \times 400/1200}{650/1200} = 0 \text{ //}$$

$$P(\text{Fruit}/\text{orange}) = P(\text{Yellow}/\text{orange}) \times P(\text{Sweet}/\text{orange}) \times P(\text{long}/\text{orange})$$

$$= 0.53 \times 0.69 \times 0 = 0 \text{ //}$$

$$P(\text{Fruit}/\text{Banana}) = P(\text{Yellow}/\text{Banana}) \times P(\text{Sweet}/\text{Banana}) \times P(\text{long}/\text{Banana})$$

$$= 1 \times 0.75 \times 0.89 = 0.65 \text{ //} \rightarrow \text{More value.}$$

$$P(\text{Fruit}/\text{others}) = P(\text{Yellow}/\text{others}) \times P(\text{Sweet}/\text{others}) \times P(\text{long}/\text{others})$$

$$= 0.33 \times 0.66 \times 0.33 = 0.072 /$$

∴ Banana is having more probability, so we can pick Banana which is satisfying the given problem conditions

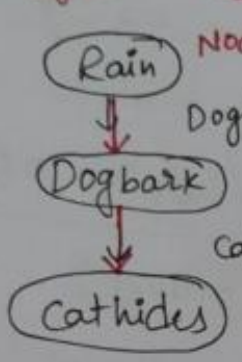
* BAYSIAN BELIEF NETWORKS :-

- 2 important concepts.

(1) Directed Acyclic Graph (DAG)

(2) Conditional probability table (CPT)

* Directed acyclic Graph.



→ when it rains, the dog will bark, when dog barks, the cat hides (May / May not happen)

→ Acyclic (no loop)

* Conditional probability table :-

	Rain (R)	Not Rain (¬R)
Bark (B)	9/48	18/48
Not Bark (¬B)	3/48	18/48

(B=T & R=T) = 9/48 = 0.19.
(B=T & R=F) = 18/48 = 0.375
(B=F & R=T) = 3/48 = 0.06
(B=F & R=F) = 18/48 = 0.375

→ when we calculate the conditional probability, we need to calculate w.r.t the parent node (Rain) But not with child node, that's why we are not considering cat hide (child node) here.

Rain → Bark → cat hide.

Parent to Bark Parent to C.H.

* Bayesian Belief N/ws :-

→ Bayesian belief N/w is a probabilistic graphical model (PGM) that represents conditional dependencies between random variables through DAG. (Direct Acyclic Graph)

→ Also suitable for representing probabilistic relation between multiple events (more than 2 events)

Ex :- 2)

Alarm detection System

Given probabilities are,

$P(B = T) = 0.001$

$P(B = F) = 0.999$

$P(E = T) = 0.002$

$P(E = F) = 0.998$

Probability of Alarm. (Parents of Alarm → B & E)

Burglary (B)	Earthquake (E)	P(A = T)	P(A = F)
T	T	0.95	0.05
T	F	0.99	0.06
F	T	0.29	0.71
F	F	0.001	0.999.

Probability of P1 (For P_1 & P_2) parent → Alarm.

Alarm (A)	$P(P_1 = T)$	$P(P_1 = F)$
T	0.90	0.10
F	0.05	0.95.

Probability of P_2.

A	$P(P_2 = T)$	$P(P_2 = F)$
T	0.70	0.30
F	0.01	0.99.

→ Find the probability of P_1 is T, P_2 is T, A is T, B is F and E is F.

i.e $P(P_1, P_2, A, \neg B, \neg E)$ (root nodes / no parents so not taking Condi. Prob.)

$= P(P_1/A)\ P(P_2/A)\ P(A/\neg B, \neg E) \cdot P(\neg B)\ P(\neg E)$

$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998. = 0.00062$ //.

* EM Algorithm (Expectation - Maximisation)

→ Used to find latent variable (Not directly observed variable)

→ Basic for many unsupervised clustering Algorithm.

* Steps involved in EM algorithm.

1. Initially, a set of initial values are considered.
   A set of incomplete data is given to system.
2. Next Step - expectation Step → E Step.
   Here, we use observed data to estimate (By previous data.) or guess the values. (incomplete data)
   of missing / incomplete data.
3. Maximisation Step or M-Step.
   Here, we use the complete data generated in preceding e-Step to update the values.
4. We check if values are converging / not.
   If Converging - Stop.
   Otherwise, repeat Step 2 & 3 (E & M) till the Convergence occurs

* Usage.

1) Used to fill missing data.
2) Used for unsupervised clustering.
3) Used to discover values of latent variables

* Advantages :

1) With each iteration, likelihood increases.
2) E-Step & M-Step are easy to implement.

* Disadvantages :

1) Slow Convergence.
2) Make Convergence to local optimal only.

* INSTANCE BASED LEARNING.

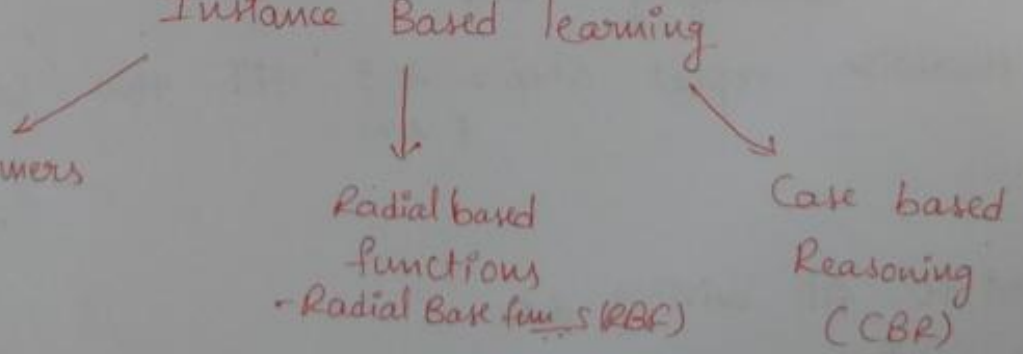
Memorise and then apply.

- Instead of performing explicit generalisation, it compares new problems with instances in training, which are stored in memory.

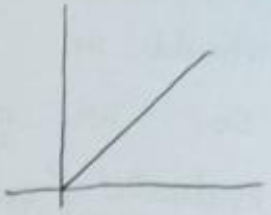
Example : spam mails.

→ Also called as memory based learning / lazy learning.

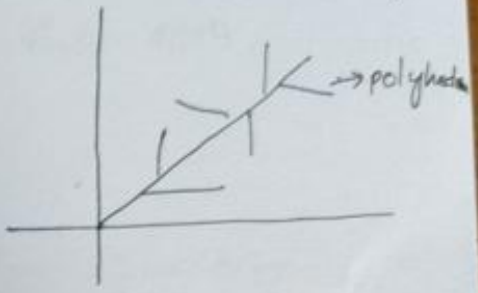
→ - done with 3 different approaches.



Ex: Initially, we have $f(y) = 2x+5$.



If we have $f(y) = 2x^2 + 3x^{\infty}$ and, divides into pockets/Segments



* K-NEREST NEIGHBOUR ALGORITHM (KNN):

- example for lazy learning.

Ex: Given data Query ⇒ x = (Maths = 6, Comp.sc = 8)
and k = 3. - nearest neighbours.
classification - pass/fail.

	Maths	CS	Result
1)	4	3	F
2)	6	7	P ✓
3)	7	8	P ✓
4)	5	5	F.
5)	8	8	P ✓

Euclidean distance (d)

$$d = \sqrt{|x_{O_1} - x_{A_1}|^2 + |x_{O_2} - x_{A_2}|^2}$$

O - Observed value
A - Actual Value.

1 - Maths
2 - CS.

(1) Calculate $d_1 = \sqrt{(6-4)^2+(8-3)^2} = 5.38$ //

✓(2) $d_2 = \sqrt{(6-6)^2+(8-7)^2} = 1$ //

✓(3) $d_3 = \sqrt{(7-6)^2+(8-8)^2} = 1$ //

(4) $d_4 = \sqrt{(6-5)^2+(8-5)^2} = 3.16$ //

(5) $d_5 = \sqrt{(6-8)^2+(8-8)^2} = 2$ //.

→ We have to choose 3 neighbours, the distance should be as min as possible.

→ So, we get 3-pass, so given query is evaluated / calculated on pass category based on the KNN algorithm.

Here, in 3 neighbours 3P & 0F.

↓

Majority.

* Regression:-

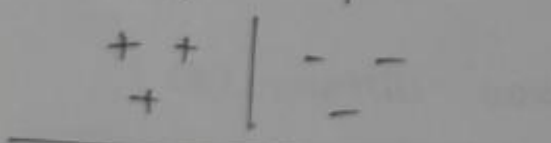
Satistical tool used to understand and quantify the relation between 2 or more variables.

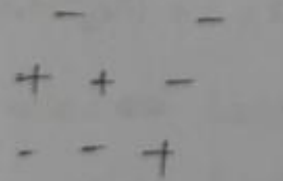
* linear Regression:

$$y = \beta_0 + \beta_1 x + \epsilon$$

y - dependent variable
x - Independendent variable
β_0 - Constant / Intercept
β_1 - x - slope / co-efficient
ϵ - Error.

→ best suited for linearly seperable data only.





Non-linearly seperable.

→ for non linearly separable data we use locally weighted regression.

* Locally weighted Regression

— To overcome the problem of non linearly Separable data.

— LWR algorithm assigns weights to data to overcome the prob.

— Computationally more expensive.

Finding weights - ? By Kernel Smoothing.

$$D = a\, e^{\frac{-\|x - x_0\|}{2c^2}}$$

x → each training i/p

x_0 → Value we are predicting

→ If the i/p is more closure to the predicting value then the weight at that feature. (data item)

c - Constant

— we construct a weight matrix (w), for each training i/p (x) and for the value we are trying to predict (x_0) [If 10-data set is there, 11 - weight matrices - 1 for x_0, 10 for x]

weight matrix - diagonal matrix

$$\beta = (x^T w x)^{-1} x^{-1} w y$$

β - model parameter

then, Prediction can be defined as

$y = \beta x_0$, ∴ $y \Rightarrow$ Prediction

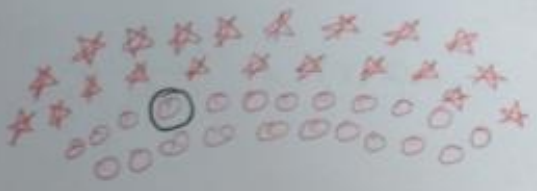
* Drawbacks :-

1) Need to evaluate whole dataset everytime.
2) Computation cost is more
3) Memory requirement is more.

* RADIAL BASIS FUNCTIONS :

- used in ANN
- has only one hidden nodes

Example:



→ 2 classes (stars & circle)

→ data is not linearly separable.

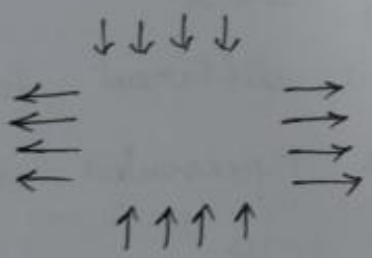
- 2 steps.

1. Increase the dimensionality (2D - 3D)

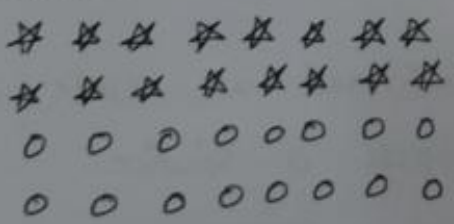
(But this step is not mandatory, only based on req) - requirement

2. Expand the direction (Horizontal)

Compress the direction (Vertical)



Resultant dataset is



How RBF works?

- Consider one center randomly.
- draw Concentric Circles (Same Center)

To expand / compress, we use ③ functions

1) Multiquadric:

$$\phi(r) = (r^2 + c^2)^{1/2}$$

$c > 0 \Rightarrow$ Constant

2) Inverse multiquadric:

$$\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}}$$

**

3) Gaussian function:

$$\phi(r) = \exp\left[\frac{-r^2}{2\sigma^2}\right]$$

σ - const

* Case Based Reasoning :-

All instance based learners have 3 properties

* 1) They are lazy learners
* 2) classification is different for each instance.
3) Instances are represented with n dimensional Euclidean space.

In CBR,

Everything is considered as case and based on previous cases - we propose a solution.

- Instances are represented as symbols (not values)

CBR has 3 components:

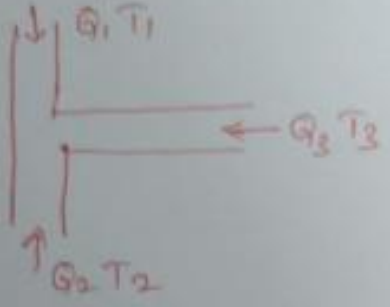
1. Similarity functions or distance measure
2. Approximation / Adjustment of instances
3. Symbolic representation of instances.

For modelling CBR, we use CADET System (Case based design tool)

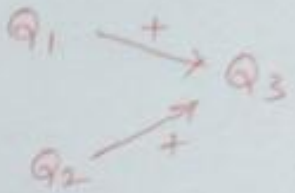
has 75 predefined libraries.

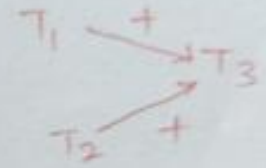
Example: Modern water Taps.

T- Junction Pipe:



Functions.





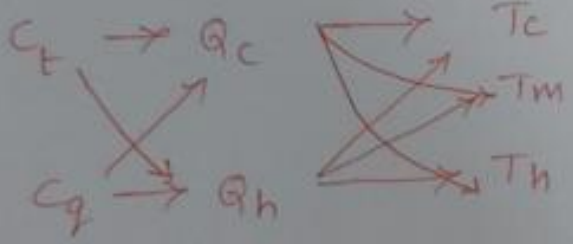
Q- water flow

T - temp

Another tap: Control temp and waterflow.

C_t → Control of temp

C_Q → Control water flow



→ Based on previous predefined cases, giving sol to the new system.

Remarks on Lazy and Eager Algorithms:-

* Lazy learning:

1. Simply stores training data and waits until it gets a test tuple
2. Less training time, more prediction time.
3. Ex: All instance based learning algorithms

* Eager learning:

1. When we give a training set, it constructs a model for classification before getting new example.
2. More training time, less prediction time.
3. Ex: Decision tree, Naive Bayes, ANN etc.

Computational learning theory

1 PAC Learning

We want to develop a theory to relate the probability of successful learning, the number of training examples, the complexity of the hypothesis space, the accuracy to which the target concept is approximated, and the manner in which training examples are presented.

1.1 Prototypical Concept Learning

Consider *instance space X*, the set of examples, and *concept space C*, the set of target functions that could have generated the examples such that there exists a $f \in C$ that is the hidden target function. For example, C could be all n-conjunctions, all n-dimensional linear functions, etc.

The *hypothesis space* is the set of all possible hypotheses that our learning algorithm can choose from, where H is not necessarily equal to C. We consider our *training instances* to be $S \times \{0, 1\}$ – including both positive and negative examples of the target concept – such that training instances are generated by a fixed unknown probability distribution D over X. Each training instance can be thought of as a (data, label) tuple, as below

$$S = [(x_1, f(x_1)), (x_2, f(x_2)) \ldots (x_n, f(x_n))]$$

In this setting, our goal is to determine a hypothesis $h \in H$ that estimates f, evaluated by its performance on subsequent instances $x \in X$ drawn according to D.

Note the assumption that both training and testing instances are drawn from from the same distribution D, as it is important in the analysis that follows.

1.2 Intuition

Consider Figure 1, showing the space predicted by target function f and hypothesis function h, where points inside the circle are positive, points outside are negative, and the functions are given by

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

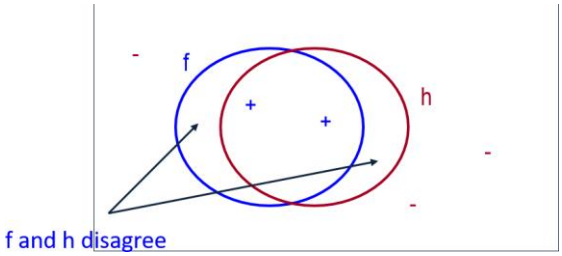


Figure 1: f not equal h

In this example, we have seen x_1 in all positive training instances (even though it is not active in f). Therefore, it is very likely that it will be active in future positive examples. If not, it is active in only a small percentage of examples, so the error should be small.

We can therefore consider the error as the probability of an example having different labels according to the hypothesis and the target function, given by

$$Error_D = Pr_{x \in D}[f(x) \; /= h(x)]$$

2 Conjunctions

Consider z to be a literal in a conjunction. Let $p(z)$ be the probability that a D-sampled example is positive and z is false in it. In the example above, $z = x_1$.

2.1 Error Bounds

Claim: $Error(h) \leq \sum_{z \in h} p(z)$

Proof
Consider that $p(z)$ is the probability that a randomly chosen example is positive and z is deleted from h[1].

If z is in the target concept, then $p(z) = 0$; f is a conjunction and thus z can never be false in a positive example if $z \in f$.

h will make mistakes only on positive examples. A mistake is made only if z that is in h but not in f . In such cases, when $z = 0$, h will predict a negative example while f indicates a positive example.

[1]Recall that h is a conjunction, and z is a literal, such that if – during training – we see a positive example where $z = 0$, z is removed from h

Therefore $p(z)$ is also the probability that z causes h to make a mistake on a randomly drawn example from D. In a single example there could be multiple literals that are incorrect, but since in the worst case they are seen one-by-one, the sum of z bounds the error of h.

We can also consider literal $z \notin h$ to be *bad* when $p(z) > \frac{\epsilon}{n}$. A *bad literal*, therefore, is a literal that is not in the target concept but has a significant probability to appear false in a positive example.

Claim: It there are no bad literals, then $Error(h) < \epsilon$

Proof
We have already stated that $Error(h) \leq \sum_{z \in h} p(z)$. Given that $|h| \leq n$ – the hypothesis has at most the same number of literals as there are features – then we know that the error cannot exceed $n \frac{\epsilon}{n} = \epsilon$.

Therefore $Error(h) < \epsilon$.

2.2 Example Bounds

Let z be a bad literal. We want to determine the probability that z has not been eliminated from h after seeing a given number of examples.

$$\begin{aligned} P(z \text{ survives one example}) &= 1 - P(z \text{ eliminated by one example}) \\ &\leq 1 - p(z) \\ &< 1 - \frac{\epsilon}{n} \end{aligned} \tag{1}$$

We can intuit that as we see more examples, the probability that bad literal z survives decreases exponentially, given by

$$p(z \text{ survives } m \text{ independent examples}) = (1 - p(z))^m < (1 - \frac{\epsilon}{n})^m$$

This is for one literal z. There are at most n bad literals, thus the probability that some bad literal survives m examples is bounded by $n(1 - \frac{\epsilon}{n})^m$

We want this probability to be bounded by δ, and thus we must choose m to be sufficiently large. Consider that we want

$$n(1 - \frac{\epsilon}{n})^m < \delta$$

Using $1 - x < e^{-x}$, it is sufficient to require

$$ne^{\frac{-m\epsilon}{n}} < \delta$$

Therefore, we need

$$m > \frac{n}{\epsilon}\{\ln(n) + \ln(\frac{1}{\delta})\}$$

to guarantee the probability of failure ($Error > \epsilon$) is less than δ

With probability $> 1 - \delta$, there are no bad literals, i.e., $Error(h) < \epsilon$

3 Formulating Prediction Theory

Given

- Instance space X
- Output space $Y = \{-1, +1\}$
- Distribution D, which is unknown over $X \times Y$
- Training examples S, where each is drawn independently from D ($|S| = m$

we can define the following

- True Error: $Error_D = Pr_{(x,y)\in D}[h(x) \neq y]$
- Empirical Error: $Error_S = Pr_{(x,y)\in S}[h(x) \neq y] = \sum_{1,m} [h(x_i) \neq y_i]$
- Function space C, or set of possible target concepts, where $f \in C : X \rightarrow Y$
- Set of possible hypotheses H

We cannot expect a learner to learn a concept exactly. There may be many concepts consistent with the available data, and unseen examples may have any label. Thus we must agree to misclassify uncommon examples that were not seen during training.

Further, we cannot always expect a learner to learn a close approximation to the target concept, since sometimes the training set does not represent unseen examples.

Therefore, the only realistic expectation of a good learner is that it will learn a close approximation to the target concept with high probability.

3.1 Probably Approximately Correct Learning

In Probably Approximately Correct (PAC) learning, one requires that given small parameters ϵ and δ – with probability at least $(1–\delta)$ – a learner produces a hypothesis with error at most ϵ.

This notion relies on the Consistent Distribution Assumption: there is one probability distribution D that governs both training and testing examples.

3.2 PAC Learnability

Consider a concept class C defined over an instance space X, and a learner L using a hypothesis space H.

C is PAC learnable by L using H
if $\forall f \in C$, $\forall D$ *over* X, and fixed $0 < \epsilon, \delta < 1$, L – given a collection of m examples sampled independently according to D – produces with probability at least $1 - \delta$ a hypothesis $h \in H$ with error at most ϵ where m is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n and $|H|$.

C is efficiently PAC learnable
if L can produce the hypothesis in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n and *size*(H).

Two Limitations

- Polynomial sample complexity, which is also called information theoretic constraint, governs if there is enough information in the sample to distinguish a hypothesis h that approximate f.
- Polynomial time complexity, also called computational complexity, which tells if there is an efficient algorithm that can process the sample and produce a good hypothesis h.

To be PAC Learnable, there must be a hypothesis $h \in H$ with arbitrary small error for every $f \in C$. We generally assume H is a super set of C.

The worst definition is that the algorithm must meet its accuracy for every distribution and every target function $f \in C$.

3.3 Occam's Razor

We want to prove the general claim that smaller hypothesis spaces are better.

Claim: The probability that there exists a hypothesis $h \in H$ that is consistent with m examples and satisfies *Error*(h) > ϵ is less then $|H|(1 - \epsilon)^m$.

Proof
Let h be a bad hypothesis. The probability that h is consistent with on examples is less than $1-\epsilon$. Since the m examples are independently drawn, the probability that h is consistent with m examples is less than $(1 - \epsilon)^m$.

The probability that any one of the hypothesis in H is consistent with m examples is less than $|H|(1 - \epsilon)^m$.

Given this fact, we now want this probability to be smaller than δ, that is

$$|H|(1 - \epsilon)^m < \delta$$

$$\ln(|H|) + m\ln(1-\epsilon) < \ln(\delta)$$

With the fact that $e^{-x} > 1 - x$, we have

$$m > \frac{1}{\epsilon}\{\ln(|H|) + \ln(\frac{1}{\delta})\}$$

This is called Occam's razor, because it indicates a preference towards small hypothesis space, i.e., if you have small hypothesis space, you do not have to see too many examples.

There is also a trade-off of the hypothesis space. If the space is small, then it generalizes well, but it many not be expressive enough.

4 Consistent Learners

Using the results from the previous section, we can get this general scheme for PAC learning:

Given a sample of m examples, find some $h \in H$ that is consistent with all m examples. If m if large enough, a consistent hypothesis will be sufficiently close to f . We can then check that m scales polynomially in the relevant parameters (i.e. m is not too large). "Closeness" guarantees

$$m > \frac{1}{\epsilon}(\ln|H| + \ln\frac{1}{\delta})$$

In the case of conjunctions, we used the elimination algorithm, which results in a hypothesis h that is consistent with the training set, and we showed directly that if we have sufficiently many examples (polynomial in relevant parameters), then h is close to the target function.

4.1 Examples

Conjunctions
For conjunctions, the size of the hypothesis space is 3^n, since there are 3 possible values for each of the n features (appear negative, positive, or not at all. Therefore, the number of examples we need according to the PAC learning framework, m, is given by

$$m > \frac{1}{\epsilon}\{\ln(3^n) + \ln(\frac{1}{\delta})\} = \frac{1}{\epsilon}\{n\ln 3 + \ln(\frac{1}{\delta})\}$$

Thus, if we want to guarantee a 95% chance of learning a hypothesis $(1-\delta)$ of at least 90% accuracy $(1-\epsilon)$, with $n = 10$ boolean variables, $m > 140$.

If we change to $n = 100$, $m > 1130$, which shows m is linear with n.

However, changing the confidence $(1-\delta)$ to 99% makes $m > 1145$, shows m is logarithmic with δ.

k-CNF

Consider Conjunctive Normal Form functions (CNFs), which can express any boolean function. Recall that CNFs are conjuctions of disjunctions. A subset of this class is that of k-CNF, where each disjunction contains k terms, as in

$$f = \wedge_{i=1}^{m}(l_{i_1} \vee l_{i_2} \vee ... \vee l_{i_k})$$

To determine if we can learn such a class of functions, we must know the size of this hypothesis space. In this case, the hypothesis space is given by $2^{(2n)^k}$, corresponding to the number of ways to choose subsets from among the k literals, including negations. Thus, the sample complexity is given by

$$\ln(|k-CNF|) = O(n^k)$$

Since k is fixed, we have an order polynomial in the number of examples and thus h is guaranteed to be PAC learnable. Next step is to learn a consistent hypothesis.

Now we must consider how to learn such a hypothesis. Using what we know now, we cannot learn this directly. We can learn k-CNFs, however, if we move to a new space.

Consider the example in which $n = 4$ and $k = 2$ for a monotone k-CNF. Here, there are six disjunctions for which we can create a new mapping from the orig-

$$y_1 = x_1 \vee x_2 \quad y_2 = x_1 \vee x_3$$
$$y_3 = x_1 \vee x_4 \quad y_4 = x_2 \vee x_3$$
$$y_5 = x_2 \vee x_4 \quad y_6 = x_3 \vee x_4$$

inal space to a new space with six features: (0000, 1) $\rightarrow$ (000000, 1)
(1010, 1) $\rightarrow$ (111101, 1)
(1110, 1) $\rightarrow$ (111111, 1)
(1111, 1) $\rightarrow$ (111111, 1)
Now we can apply a standard algorithm for learning monotone conjunctions.

Unbiased Learning

Consider the hypothesis space of all boolean function on n features. There are 2^{2^n} different functions and the bound $(\ln(|H|)$ is therefore exponential in 2^n, which means that in general the set of all boolean functions are not PAC learnable.

k-Clause CNF

Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge ...C_k; C_i = l_1 \vee l_2... \vee l_m$$

The size of the hypothesis space $\ln(|H|) = O(kn)$ is linear in n and thus PAC learnable.

k-DNF
Disjunctions of any number of terms where each conjunctive term has at most k literals.

$$f = T_1 \vee T_2 ... \vee T_m; T_i = l_1 \wedge l_2 \wedge ... \wedge l_m$$

4.2 k-term DNF Computational Complexity

Consider the class of k-term DNFs, or disjunctions of at most k conjunctive terms. From the sample complexity perspective, we should be able to learn in the same way as with k-CNFs, but computational complexity is challenging.

Consider a 2-term DNF consistent with a set of training data is NP-hard. Thus, even though 2-term DNFs are PAC learnable, they are not efficiently PAC learnable.

We can address this by enlarging the hypothesis space. If the hypothesis we wish to learn can be represented in a larger hypothesis space that we know is learnable, we can learn our desired hypothesis. In this case, we can represent k-term DNFs as k-CNFs, since k-CNF is a superset of k-term DNF. Consider a 3-term DNF (left) and its equivalent 3-CNF (right).

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{x \in T_1, y \in T_2, z \in T_3} \{x \vee y \vee z\}$$

Representation is important. Concepts that cannot be leaned using one representation may be learned using another more expressive representation.

However, this leaves us with two problems:

> How can we learn when data is not completely consistent with training data?
>
> How can we learn in an infinite hypothesis space?

5 Agnostic Learning

Assume we are trying to learn concept f using hypothesis space H, but $f \notin H$. We therefore cannot learn a a completely consistent hypothesis, and thus our goal is to find a hypothesis $h \in H$ that has as small training error as possible.

$$Err_{TR} = \frac{1}{m} \sum_{i}^{m} f(x^i) \neq h(x^i)$$

where x_i is the i^{th} training example.

We want to guarantee that a hypothesis with a small training error will have good accuracy on unseen examples, and one way to do so is with Hoeffding bounds. This characterizes the deviation between the true probability of some event and its observed frequency over m independent trails.

$$Pr[p > \hat{p} + \epsilon] < e^{-2m\epsilon^2}$$

To understand the intuition, consider tossing a biased coin. The more tosses, the more likely the observed result will correspond with the expected result. Similarly, the probability that an element in H will have training error which is off by more than ϵ can be bounded as follows:

$$Pr[Err_D(h) > Err_{TR}(h) + \epsilon] < e^{-2m\epsilon^2}$$

If we consider $\delta = |H|e^{-2m\epsilon^2}$, we can get a generalization bound, or how much will the true error E_D deviate from the observed (training) error E_{TR}.

For any distribution D, generating training and test instances with probability at least $1 - \delta$ over the choice of the training set of size m, (drawn i.i.d.), for all $h \in H$

$$Error_D(h) < Error_{TR}(h) + \sqrt{\frac{\log|H| + \log(\frac{1}{\delta})}{2m}}$$

An agnostic learner which makes no commitment to whether f is in H returns the hypothesis with least training error over at least the following number of examples m can guarantee with probability at least $1-\delta$ that its training error is not off by more than ϵ from the true error. We therefore require a number of examples given by

$$m > \frac{1}{2\epsilon^2}\{\ln(|H|) + \ln(\frac{1}{\delta})\}$$

Learnability depends on the log of the size of the hypothesis space.

6 VC Dimension

For both consistent and agnostic learners, we assumed finite hypothesis spaces. We know consider an infinite hypothesis space.

6.1 Learning Rectangles

Consider a target concept as an axis parallel rectangle (positive points inside, negative outside, as given by Figure 2.

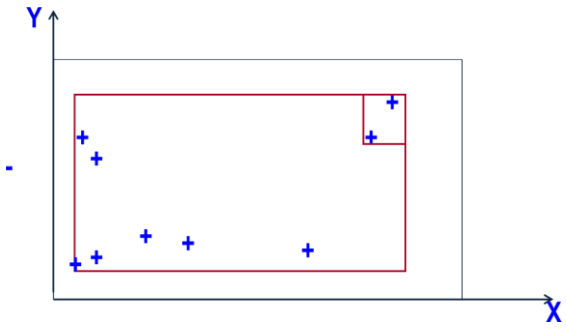


Figure 2: Learning Rectangles

We can simply choose the maximum x and y values as well as the minimum x and y values of the positive examples as the boundary for the rectangle. This is generally a good algorithm because it learns efficiently, but we cannot use the theorem from before to derive a bound because the hypothesis space is infinitely large.

Therefore, we need to find out how to derive a bound given an infinitely large hypothesis space.

6.2 Infinite Hypothesis Space

Just as before, where we discussed small hypothesis spaces (conjunctions) and large hypothesis spaces (DNFs), some infinite hypothesis spaces are larger (more expressive) than others; rectangles and general convex polygons have different levels of expressiveness.

We therefore need to measure the expressiveness of an infinite hypothesis space. The Vapnik-Chervonenkis dimension – or VC dimension – provides such a measure. Analogous to $|H|$, there are bounds for sample complexity using $VC(H)$.

6.3 Shattering

The key notion behind VC-dimension is that of *shattering*. Assume a set of points. We want to use a function to separate all possible labelings of the set of points. In Figure 3, a linear function (green) can separate the two points, regardless of how they're labeled.

For two points on a plane, there are two different ways of labeling the two points. A line can separate those two points no matter how they are labeled.

Figure 3: A linear function can shatter 2 points

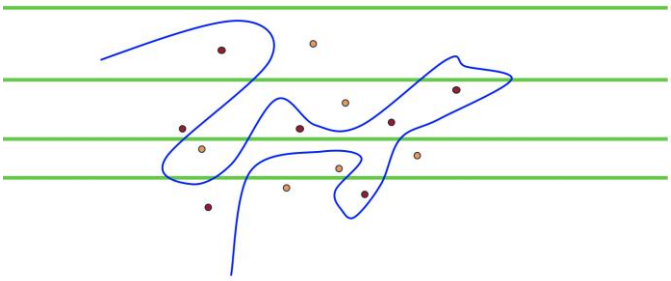


Figure 4: A set of multiple points

However, if there are many points in the set on a plane, as in Figure 4, no straight line can separate any labeling of these points. Linear functions are not expressive enough to shatter 13 points, but more expressive functions are.

We say that a set S of examples is shattered by a set of functions H if – for every partition of the examples in S into positive and negative examples – there is a function in H that gives exactly there labels to the examples. The intuition is that a rich set of functions shatters a large set of points.

(a) A function example (b) function fails to shatter

Consider the function in which left bounded intervals on the real axis for some number is positive ([0, a), for some $a > 0$).

It is trivial for this function to shatter a single point. However, in any set of two points on the line, the left can be labeled negative and the right positive, which

this *H* cannot label correctly. Thus, left bounded intervals cannot shatter two points.

Similarly, if we consider the class of functions for which real numbers $b > a$ and points within $[a, b]$ is positive, all sets of one or two points are shatterable, but no set of three points can be shattered.

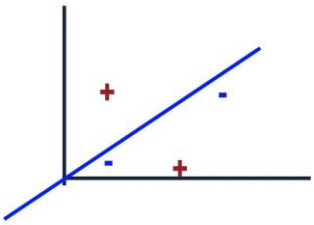


Figure 6: Half space

As a final example, consider half-spaces on the plane. We can trivially shatter all sets of one point and two points. We can shatter some sets of three points, but cannot shatter any set of four points. If the 4 points form a convex polygon, then by labeling each point different from its neighbors, the four points cannot be shattered. If, on the other hand, they do not form a convex polygon, then one point is inside the convex hull defined by the other three, and if that point is negative, there is no way to shatter them.

6.4 Definition

An unbiased hypothesis space *H* shatters the entire instance space *X* if it is able to induce every possible partition on the set of all possible instances.

The larger the subset *X* that can be shattered, the more expressive a hypothesis space is (i.e. less biased). The VC dimension of hypothesis space *H* over instance space *X* is the size of the largest finite subset *X* (even if there is only one subset) that is shattered by *H*.

If there exists a subset of size *d* that can be shattered, then $VC(H) \geq d$.

If no subset of size *d* can be shattered, then $VC(H) < d$.

- $VC(\textit{Half intervals}) = 1$; No subset of size 2 can be shattered
- $VC(\textit{Intervals}) = 2$; No subset of size 3 can be shattered
- $VC(\textit{Half-spaces in the plane}) = 3$; No subset of size 4 can be shattered

6.5 Sample complexity and VC dimension

VC dimension serves the same role as the size of the hypothesis space. Using VC dimension as a measure of expressiveness, we can give an Occam algorithm for infinite hypothesis spaces.

Given a sample D of m examples we will find $h \in H$ that is consistent with all m examples, if

$$m > \frac{1}{\epsilon}\{8VC(H)\log\frac{13}{\epsilon} + 4\log(\frac{2}{\delta})\}$$

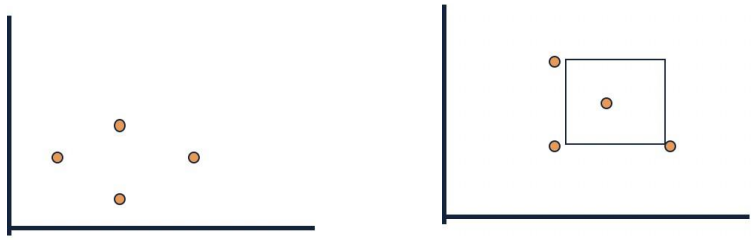
then with probability at least $1-\delta$, h has error less then ϵ.

We consider that the hypothesis space has to be infinite if we want to use this bound. If we want to shatter m points, then H has to be at least 2^m in order to shatter any configurations of those m examples.

Thus $|H| > 2^m$, $\log(|H|) \geq VC(H)$.

6.6 Axis-Parallel Rectangles, Continued

Consider again the problem of learning axis-parallel rectangles. To determine if axis-parallel rectangles are PAC-learnable, we must determine sample complexity. Here, we show that $VC(H) \geq 4$. In Figure 7a, it is trivial to find an



(a) A set that can be shattered (b) a set that cannot be shattered

axis-parallel rectangle to shatter the points, regardless of their labeling. Though Figure 7b illustrates a set of four that is not shatterable, it is sufficient to find any set of four to prove the VC dimension is greater or equal to 4.

Next we need to argue that no set of five points can be shattered. For any layout of five points, we just need to show one kind of labeling that cannot be shattered. In this case, we can say that – of the five points – there must be a minimum and maximum x and y. There must by definition be a point within those minimum and maximum bounds, and thus by labeling the extreme four points as positive but the fifth, internal point as negative, axis-parallel rectangles cannot shatter five points.

Thus, from sample complexity perspective, axis-parallel rectangles are PAC learnable.

To determine if this hypothesis class is efficiently PAC learnable, we need an efficient algorithm to find the rectangle. Here, we can find the smallest example rectangle that contains all positive examples. This is likely not the best rectangle – it cannot generalize to new positive examples – so in the ideal case we would like a margin, that is, a rectangle slightly larger than the minimum one we've seen during training.

Given such an algorithm, we have shown that axis-parallel rectangles are efficiently PAC learnable.

6.7 Sample Complexity Lower Bound

We've discussed upper bounds on the number of examples; that is, if we have seen m examples, we can be reasonably sure our algorithm will perform well on new examples. There is also a general lower bound on the minimum number of examples necessary for PAC learning.

Consider any concept class C such that $VC(C) > 2$. For any learner L and small enough ϵ, δ, there exists a distribution D and a target function in C such that if L observes less than

$$m = max[\frac{1}{\epsilon}\log(\frac{1}{\delta}), \frac{VC(C) - 1}{32\epsilon}]$$

examples, then with probability at least δ, L outputs a hypothesis having $error(h) > \epsilon$.

This is the inverse of the bound algorithm we have seen before.

Ignoring constant factors, the lower bound is the same as the upper bound, except for the extra $\log(\frac{1}{\epsilon})$ factor in the upper bound.

7 Conclusion

The PAC framework provides a reasonable model for theoretically analyzing the effectiveness of learning algorithms.

We discussed that the sample complexity for any consistent learner using the hypothesis space, H, can be determined from a measure of Hs expressiveness ($|H|$, $VC(H)$).

We discussed consistent and agnostic learners, showing that the log of the size of a finite hypothesis space is most important, and then extended this notion to the infinite hypothesis space.

We also discussed sample and computational complexity, showing that if sample complexity is tractable, the computational complexity of finding a consistent hypothesis governs the complexity of the learning problem.

Many additional models have been studied as extensions of the basic one: learning with noisy data, learning under specific distributions, learning probabilistic representations, etc..

An important extension is PAC-Bayesians theory, where the idea is that – rather than simply assume that training and test are governed by the same distribution – assumptions are also made about the prior distribution over the hypothesis space.

It's important to note that though the bounds we compute are loose, they can still guide model selection. A lot of recent work is on data dependent bounds.

The impact COLT has had on practical learning system in the last few years has been very significant: SVMs, Winnow (Sparsity), Boosting, and Regularization, to name a few.