ML 3rd unit - ml unit 3 notes

R18 B.Tech. Cse (Computer Networks) Iii & Iv Year Jntu Hyderabad (Jawaharlal Nehru Technological University, Hyderabad)

UNIT-III

: BAYESIAN LEARNING :

It is based on the assumption that the quantities of interest are governed by Probability Distributions & the Optimal decisions can be made by reasoning about these probabilities together with observed data.

Bayesian Learning methods are similar to our Machine Learning for two different reasons :

1. These calculate explicit probabilities for hypothesis such as Naive Bayes Classifier.
2. Comparing the classifier with decision Tree and Neural Network Algorithms.

The Problem of learning to classify text documents such as electronic news articles., the Naive Bayes classifier is most effective algorithms.

2. They provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

⇒ We analyze the algorithms such as FIND-S and CANDIDATE ELIMINATION ALGORITHM to determine conditions under which they Output the most probable hypothesis given by Training Data.

Features Of Bayesian Learning Methods :

1. Each Observed training example can incrementally decrease or increase the estimated probability that a hypothesis is Correct.
   It also provides the complete elimination of hypothesis if it is inconsistent with any single example.
2. Prior Knowledge can be combined with observed data to determine the final probability of hypothesis.
3. Bayesian methods can accomadate hypotheses that make probabilistic predictions.
4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
5. Bayesian methods prove computationally intractable, they can provide a standard of Optimal decision making against which other practical methods can be measured.

11th may, Presents

50, 55, 57, 61, 62, 63, 64
71, 66, 72, 73, 75, 80, 82, 83, 81
87, 89, 92

10th may, presents

50, 55, 57, 61, 62, 63, 66, 70, 73
75, 82, 89, 92
LE-502, 504, 505, 507, 512, 513

BAYES THEOREM

In ML we are often interested in determining the best hypothesis from Space H, ow observed Training data D. To specify the best hypothesis, we demand the most Probable hypothesis.

Bayes Theorem provides a way to calculate the probability of hypothesis directly based on its prior probability.

Based on previous knowledge of System

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

⇒ P(A/B) is a hypothesis, finding the probability of A, where B is occured.

⇒ P(B/A) finding the probability of B, where A is occured
↳ Likelihood

⇒ P(A) is Prior ⇒ P(B) Marginal.

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \quad -(*)- \quad P(B/A) = \frac{P(B \cap A)}{P(A)}$$

⇒ $P(A/B) \cdot P(B) = P(A \cap B)$ — ①

$P(B/A) \cdot P(A) = P(B \cap A)$ — ②

From eq. ① & ② RHS are equal i.e., $A \cap B = B \cap A$

then $P(A/B) \cdot P(B) = P(B/A) \cdot P(A)$

i.e., $$\boxed{P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}}$$ Bayes Theorem

Terms : A is Hypotheses
B is Given Data

P(A/B) – finding Probability of Hypothesis, when prob of Training Examples given.

P(B/A) – finding Probability of given data provided with prob of hypotheses that is True.

P(A) – Prob of hypothesis before observing given data.

P(B) – Prob of hypothesis of given data is True.

Example: P(King/face)

Probability of finding King that it is a face Card.

Face Cards ⇒ J, Q, K i.e., 4 × 3 = 12 face cards in single set of pack.

$$= \frac{P(face/King) \cdot P(King)}{P(face)}$$

$\frac{4 \text{ kings}}{4 \text{ colors}} = \frac{4}{4} = 1$

$$= \frac{1 \cdot \frac{4}{52}}{\frac{12}{52}}$$

[4 kings among 52 cards
12 faces among 52 cards]

$$= \frac{1 \times \frac{1}{13}}{3/13} = 1/3 \Rightarrow P(K/f) = 1/3$$

1 face among 3 cards is probability.

Bayes:
It will give Probability of an event based on previous Conditions.

Concepts of Bayes Theorem :

1. Product Rule :

Probability $P(A \cap B)$ of a conjunction of two events A & B

$$P(A \cap B) = P(A/B)\,P(B) = P(B/A)\,P(A)$$

2. Sum Rule :

Probability of disjunction of two events A and B.

$$P(A \vee B) = P(A) + P(B) - P(A \cap B)$$

3. Bayes Theorem :

The Priority Probability $P(h/D)$ of h given D.

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

4. Theorem of Total Probability:

If events $A_1 \ldots A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, $P(B) = \sum_{i=1}^{n} P(B/A_i)\,P(A_i)$

MAP Hypothesis [Maximum A Posterior]

Method: Learner Considers some set of hypothesis H and is interested in finding the most probable Hypothesis $h \in H$.

Given the Observed data D [Atleast one of the maximally probable if there are many]

Such maximally probable hypothesis is called a maximum A posterior [MAP]

Determine: The MAP hypothesis by using Bayes Theorem.

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

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

[$\because$ P(D) is Common for all hypothesis we ignore]

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

BRUTE FORCE MAP LEARNING :

The Simplifications does not alter the main Conclusions of this method. We can design a straight-forward Concept learning Algorithm to output the maximum Posteriori Hypothesis.

Brute-Force map Algorithm :

i. For each hypothesis h in H, calculate the Posterior Probability

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

ii) Output the hypothesis h_{map} with the highest posterior Probability.

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

This may require Significant Computation because it applies Bayes Theorem to each hypothesis in H to calculate P(h/D)

* It is impractical for large hypothesis Space
* Still the method is interested because it provides a standard against which we may judge the performance of other Concept learning algorithms.

31st May: Presenties
59, 60, 64, 74, 75, 76, 78
82, 83, 88, 89

⇒ Brute force MAP learning algorithm must specify values for $P(h)$ and $P(D/h)$

⇒ $P(h)$ and $P(D/h)$ can be chosen to be consistent with the assumptions.

i. Training Data D is Noise free.

ii. The Target Concept C is contained in hypothesis space H.

iii. We have no prior reason to believe that any hypothesis is more probable than any other.

Gaiven these assumptions, what values should we specify for $P(h)$.

→ Prior Probability can be given as :

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

→ what choice shall we make for $P(D/h)$

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

[$\because d_i$ is Target & $h(x_i)$ is Output]

The Probability of data D given hypothesis h is 1, if D is Consistent, with h otherwise 0.

Recalling Bayes Theorem :

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

where h is inconsistent with Training Data D.

i.e., $P(D/h)$ to be 0.

$$\therefore P(h/D) = \frac{0 \cdot P(h)}{P(D)} = 0$$ if h is inconsistent with D.

where h is Consistent with Training data D

i.e., $P(D/h)$ to be 1.

$$\therefore P(h/D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)}$$

$$= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}}$$ [VS - Version Space]

$$P(h/D) = \frac{1}{|VS_{H,D}|}$$ if h is Consistent with D.

where $VS_{H,D}$ is the subset of hypothesis from H that are Consistent with D.

$$P(D) = \sum_{h_i \in H} P(D/h_i) P(h_i)$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|}$$

[VS - Version space]

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

To Summarize, Bayes theorem implies that the posterior probability $P(h/D)$ under our assumed $P(h)$ and $P(D/h)$ is

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

Least Squared Error Hypothesis :

Bayesian analysis can sometimes be used to show that a particular learning algorithm outputs MAP hypotheses even though it may not explicitly use Bayes Rule.

⇒ This method considers the problem of learning a continuous valued Target function.

⇒ Under certain assumptions any learning algorithm that minimizes the Squared Error Hypothesis between the output hypothesis predictions and Training data will output a maximum likelihood hypotheses.

Finding maximum likelihood hypothesis in Bayesian learning

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

[Maximum Likelihood]

[Small p : Probability Density function]

Let us Take Training Instances $x_1, x_2 \ldots x_n$ and Target values

$$D = \{d_1, d_2 \ldots d_m\}$$

∴ we write $p(D/h)$ as product of $p(d_i/h)$ as various probabilities

$$\text{i.e., } h_{MAP} = \underset{h}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i/h)$$

[Probabilities must be multiplied]

By assuming Normal Distribution for probability distribution input $x \rightarrow$ variable.

$\mu \rightarrow$ mean & $\sigma \rightarrow$ Standard Deviation.

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

According to Bayes x with d_i & μ with $h(x_i)$

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

Use in function : when we apply log value, product Converts to Summation i.e., $\pi \Rightarrow \Sigma$

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

Maximizing this negative quantity is equivalent to minimizing the Corresponding positive quality.

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

we can discard Constants, which are common to all data sets i.e., $\frac{1}{\sqrt{2\pi\sigma^2}}$

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

↓ Least

↓ Error minimised & Squared.

13th May

→ The equation shows that the maximum likelihood hypothesis h_{ML}, which minimizes the sum of squared errors between observed training values & hypothesis prediction.

→ It determines the most probable hypothesis.

→ This approach is used in many neural networks like curve fitting.

→ The analysis considers noise only in the target value of training example.

Maximum Likelihood hypothesis for Predicting Probabilities

Collect observed frequencies of 1's & 0's for each possible value of x, train neural network to update target frequency. which criterion should we optimize in order to find a maximum likelihood hypothesis for f'. (f-dash)

* first obtain an expression for $p(D/h)$
* Assume the training data D is of the form $D = \{(x_1, d_1), (x_2, d_2) \ldots (x_m, d_m)\}$, where d_i is observed 0 or 1 for value for $f(x_i)$.
* both x_i & d_i as random variables and assuming that each training examples is drawn independently, we can write $p(D/h)$ as

$$p(D/h) = \prod_{i=1}^{m} p(x_i, d_i/h) \quad \text{— (1)}$$

Applying Product Rule

$$p(D/h) = \prod_{i=1}^{m} p(d_i/h, x_i)\, p(x_i) \quad \text{— (2)}$$

The Probability $p(d_i/h, x_i)$

$$p(d_i/h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \quad -(3)$$

Re-express it in a more mathematically manipulate form as:

$$p(d_i/h, x_i) = h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \quad -(4)$$

Substitute eq-(4) in eq (1) to obtain eq-(5)

$$p(D/h) = \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} p(x_i) \quad -(5)$$

$\therefore$ d_i, h are posterior examples.

we write an expression for maximum likelihood hypothesis.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \cdot p(x_i)$$

The last term is a constant independent of h, so it can be dropped.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} \quad -(6)$$

It is easier to work with log of likelihood yielding

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln (1 - h(x_i)) \quad -(7)$$

[$\because$ ln is length]

eq-(7) describes the quantity that must be maximised in order to obtain the maximum likelihood hypothesis in our current problem setting.

Minimum Description Length Principle:

Assumption:

Representing a Concept in minimum possible way, then it is said to be good Concept.

Mathematically:

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

Applying logarithm:

$\underset{h \in H}{\operatorname{argmax}}\ \log P(D/h) + \log P(h)$

[$\because \log ab = \log a + \log b$]

$\therefore$ Convert it into min to get shortest

$\underset{h \in H}{\operatorname{argmin}} - \log P(D/h) - \log(P(h))$

[minimum length / short hypothesis is preferred]

Example:

* 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} msg $= P_i$
* while tramitting, we need the code that minimize the expected number of bits.

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

we represent length of message i we write to see as $l_c(i)$ [i is message]

$\therefore h_{MAP} = \underset{h}{\operatorname{argmin}}\ L_{C_H}(h) = L_{C_{D/h}}(D/h)$

C_H is optimal encoding for H

$C_{D/h}$ " " D given h

$\therefore h_{ML} = h_{MAP}$

$\therefore h_{MDL} = \underset{h \in H}{\text{argmin}}\ L_{C_1}(h) + L_{C_2}(D/h)$

The above analysis shows that if we choose C_1 to be the optimal encoding of hypotheses C_H, and if we choose C_2 to be optimal encoding $C_{D/h}$ then

$$\boxed{h_{MDL} = h_{MAP}}$$

: 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

$$x = \{x_1, x_2, x_3 \ldots x_n\} \quad \{y\} \rightarrow \text{Yes / No} \quad \text{+ve} \quad \text{-ve}$$

$$p(y/x_1, x_2 \ldots x_n) = \frac{[p(x_1/y) \cdot p(x_2/y) \ldots p(x_n/y)]\,p(y)}{p(x_1) \cdot p(x_2) \ldots p(x_n)}$$

$$= \frac{p(y) \prod_{i=1}^{n} p(x_i/y)}{p(x_1) \cdot p(x_2) \ldots p(x_n)}$$

$$= p(y) \prod_{i=1}^{n} p(x_i/y)$$ [Denominator is Omitted it doesn't make change of probability]

CSE - III - II 6th Jun

Absentees: 50, 55, 57, 58, 60, 61, 63, 64, 66, 72, 74, 75, 82, 83, 87.

LE: 504, 505, 506

Ex: Enjoy Sport with 14 days info

We took only Two attributes among all.

P(Y) = Probability of Yes

P(N) " " No.

* Outlook

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

* Play

Yes	9	9/14
No	5	5/14
Total	14	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%

Total (Sunny, Hot)

$$P(\text{Yes}/\text{sunny}, \text{hot}) = P(\text{Sunny}/\text{Yes}) \times P(\text{hot}/\text{Yes}) \times P(\text{Yes})$$

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

$$P(\text{No}/\text{Sunny}, \text{hot}) = P(\text{Sunny}/\text{No}) \times P(\text{hot}/\text{No}) \times P(\text{No})$$

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

$$\text{Total} = 0.031 + 0.08571$$

$$= 0.27$$

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

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

Probability is No is more

∴ player will not enjoy sport.

: GIBS ALGORITHM :

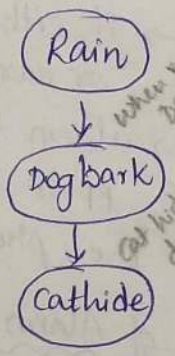
* Chooses one hypothesis at Random, according to $P(h/D)$
* Use this Classify new instance.
* ~~Uses~~ Error frequency of Gibbs is two times less than Bayesian Classifier.
* Simply applies a hypothesis drawn at random according to current posterior probability Distribution.

: BAYESIAN BELIEF NETWORKS :

Before going to learn about Bayesian Belief Networks we come across about :

* Directed Acyclic Graph. [DAG]
* Conditional Probability Table. [CPT]

DAG :



A directed graph with no cycles.

⇒ As Rain falls, Dog barks if dogbarks Cat hides.

Two probabilities for above example for all nodes i.e., may or maynot.

Rain is a node with two hypothesis +ve or -ve ⇒ (2)

Total we consider (48) events

The Conditional Probability Table :- (Ex: 1)

	R	~R
B	9/48	18/48
~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

R - Rains & ~R - Not Rains
B - Barks & ~B - Not Barks

⇒ Rain is Parent of Dogbark
Dogbark is parent of Cathide

⇒ Calculating the probability, can be done by parent i.e., Using parent node not child. So, we haven't use the Concept of Cat-hide.

(81)
9th: 50,51,54,61,70
64 72,73,74,75,77,78
66 82 83 84 86 89
58 502,504,505,507,510

(8)
49,60,61,64
71,82,87,89
91,502,505
506

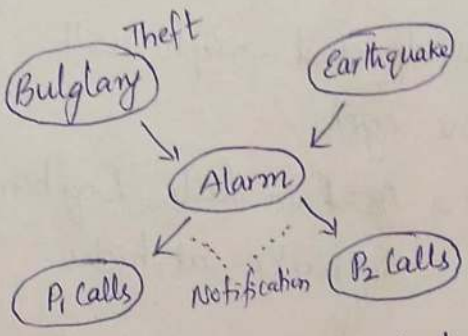
(3rd)
7th June :- 49,50,55,60,61,63
64,70,71,72,75,78,82,83,87
89,91, LE- 502,504,505,506
510,512

Bayesian Belief Networks :

→ It is a probabilistic Graphical model (PGM) that represents Conditional dependencies between random variables through DAG.

→ It is also suitable for representing probabilistic relation between multiple events. [more than Two]

Ex:-2



→ if either theft occurs or earthquake

→ alarm rings either P1 or P2 i.e., phone 1 or phone 2

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:

Bulglary	Earthquake	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

Alarm (A)	P(P1=T)	P(P1=F)
T	0.90	0.10
F	0.05	0.95

Probability of P2

A	P2=T	P2=F
T	0.70	0.30
F	0.01	0.99

Ex:- Find the Probability of P_1 is T, P_2 is T & A is T, B is F & E is F

i.e., $P(P_1, P_2, A, \sim B, \sim E)$

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

$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$

$= 0.00062$

No Burglary, No Earthquake then ringing of alarm is very little possibility.

EM-ALGORITHM

Expectation - Maximization.

→ Used to find Latent variable.

→ This is Basic for many unsupervised clustering Algorithm.

The variables that directly cannot derived or observed are called as Latent Variables. These are observed with old or past derived variables.

STEPS:

1. Initially, set of values initially considered. A set of incomplete data is given to system.
2. Estimation or Expecting value from incomplete set of data. [fill the incomplete data]
3. Maximization or M-step.
   Here, we use the complete generated data in preceding e-step to update the value.
4. We check if values are converging or not i.e., we are getting expected value.
   if not repeat the steps ③ & ④ i.e., E & M.

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 and M-step are easy to implement.

Disadvantages:

1. Slow Convergence.
2. makes Convergence to Local optimal only but Cannot be done by global optimal.

(CHAP-7)

COMPUTATIONAL LEARNING THEORY:

The Theoritical characterization of difficulties of Several types of machine Learning problems and the Capabilities of Several ML Algorithms.

Answers the questions like:

⇉ "Under what Conditions is a particular Algorithm is possible & Impossible"

⇉ "Under what Conditions a particular Algorithm is assured of learning Successfully"

Two Specific frameworks for analyzing learning algorithms are Considered by :-

⇒ PAC : Probably Approximately Correct.

Depending on attributes of learning problem such as :

→ the size or complexity of hypothesis Space Considered by learner.

→ the accuracy to which the target Concept must be approximated.

→ the probability that the learner will output a succesful hypothesis.

9th June :

57

50, 55, 58, 60, 63, 64, 69, 70, 72, 73, 74, 75, 76, 77, 78, 84, 86, 87, 88, 89

505, 506, 507, 510

Our Goal is to answer such as :

Sample Complexity : How many training examples are needed for a learner to converge for a successfull hypothesis.

Computational Complexity : How much computational effort is needed for a learner for a succesfull hypothesis.

Mistake Bound : How many training examples will the learner misclassify before converging a succesful Hypothesis.

Probably Learning An Approximately Correct Hypothesis

Consider a particular setting for learning problem called PAC [Probably Approximately Correct] model.

→ Specify the Problem Setting that define PAC.
→ Consider how many training examples.
→ How much Computation is required.

1. Problem Setting :-

→ Let X is set of all possible instances over which target function may defined.
→ Let C be set of target Concepts that our Learner might be called upon to learn.
→ Each target Concept c in C corresponds to Some Subset of X

i.e., $c : X \rightarrow \{0, 1\}$

if X is positive example then $c(x) = 1$.

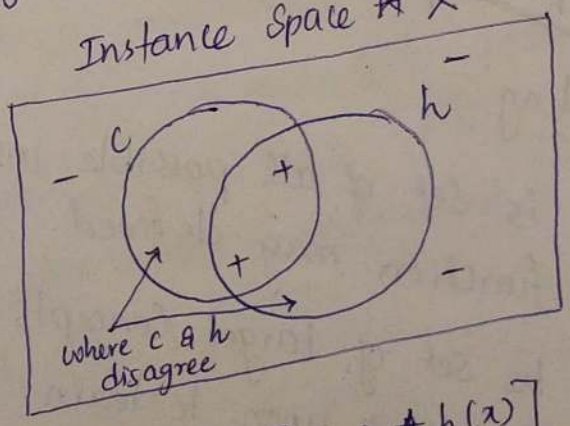
if X is negative example then $c(x) = 0$.

The instances we assume are generated at random from X according to some probability distribution D., it may be any distribution, generally not known to learner. Training examples are generated by drawing an instance x., along with its target value c(x).

This setting is interested in characterizing the performance of various learners L using various hypothesis spaces H.

Error Of Hypothesis: We define true error of a hypothesis h with respect to target concept c and instance distribution D.

True Error: denoted by $error_D(h)$, the error which will misclassify an instance drawn at random according to D.



$$error_D(h) = \Pr_{x \in D} [c(x) \neq h(x)]$$

∴ Pr indicates probability taken over instance distribution D.

The randomly drawn instance will fall into the region where h and c dis-agree on its classification. The + & - indicates positive and negative examples. we use Training Error to refer to fraction of training examples misclassified by h.

PAC Learnability:

PAC: Probably Approximately Correct.

It characterize the classes of Target Concepts that can be reliably learned from randomly drawn training examples and reasonable amount of computation.

→ characterizing number of Training Examples needed to learn a hypothesis h for which $error_D(h) = 0$.

Definition: Consider a Concept class C defined over a set of instances X of length n and a learner L using hypothesis space H.

C is PAC-learnable by L using H, if for all $c \in C$, distributions D over X, such that $0 < \epsilon < 1/2$ and δ such that $0 < \delta < 1/2$, Learner L with probability atleast $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_D(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon, 1/\delta$.

14th Presenties: (30)
(50), 54, 55, 58, 60, (63), 64, 66, 69
(70), (71), 72, 73, 74, 75, (76), 77, 78, 82, 81
83, 84, 87, (88), 89,

Sample Complexity For Finite Hypothesis Spaces

PAC-learnability is largerly determined by the number of training examples required by the learner.

→ The increase in number of required number of training examples with problem size, called Sample Complexity.

→ The success of learner is limited availability of Training examples.

→ A genral bound on sample Complexity for a very broad class of learners called Consistent learners.

→ A learner is Consistent if output hypothesis perfectly fits the training data.

→ Derive a bound on Training Examples required by any Consistent learner.

→ To accomplish we recall Version Space $VS_{H,D}$

i.e., $VS_{H,D} = \{h \in H (\forall \langle x, c(x) \rangle \in D)(h(x) = c(x))\}$

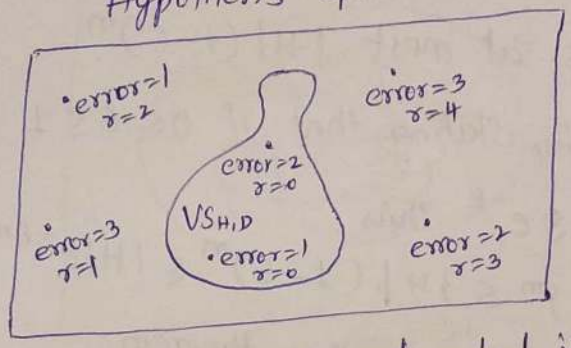
The Significance of version space is that every Consistent learner outputs a hypothesis belonging to version Space.

Instance Space X, hypothesis H, Training data D.

Def: Consider a hypothesis Space H, target Concept c, instance distribution D, training examples D of c. The Version Space is said to be ϵ-exhausted with respect to c & D $VS_{H,D}$ has an error lesthan ϵ with respect to c and D.

$$(\forall h \in VS_{H,D}) \text{error}_D(h) < \epsilon$$

Hypothesis Space H



$error_D(h) = Pr_{x \in D}[c(x) \neq h(x)]$

The Version Space is ϵ-exhausted just in the case that all the hypothesis Consistent with the observed Training examples.

Zero training error $r = 0$

Theorem : ϵ-exhausting the version Space.

If the hypothesis space H is finite and is D is Sequence of $m >= 1$ Independent Randomly drawn examples of Some target Concept C., for any $0 \leq \epsilon \leq 1$.

$$K \leq |H| e^{-\epsilon m}$$

Proof : Let $h_1, h_2, h_3 \ldots h_K$ all hypothesis in H. we have error greaterthan ϵ with respect to c.

⇒ we fail to ϵ-exhaust the Version Space if and only if atleast one of these k-hypothses happens to be ~~Considered~~ Consistent with m independent training examples.

⇒ the probability any single hypothesis having true error greaterthan ϵ is Consistent with $(1-\epsilon)$.

$\therefore (1-\epsilon)^m$ probability of drawing m independent examples.

$K(1-\epsilon)^m$

$\because K \leq |H|$ is at most $|H|(1-\epsilon)^m$

General Inequality stating that if $0 \leq \epsilon \leq 1$ then

$(1-\epsilon) \leq e^{-\epsilon}$ Thus

$K(1-\epsilon)^m \leq |H|(1-\epsilon)^m \leq |H|e^{-\epsilon m}$.

proves the theorem.

Sample Complexity for Infinite Hypothesis Space

As the sample Complexity for PAC learning grows as the algorithm of the size of hypothesis space.

There are two drawbacks for finite hypothesis space

1. It can lead to quite weak bounds
2. Cannot apply the equation for infinite bounds.

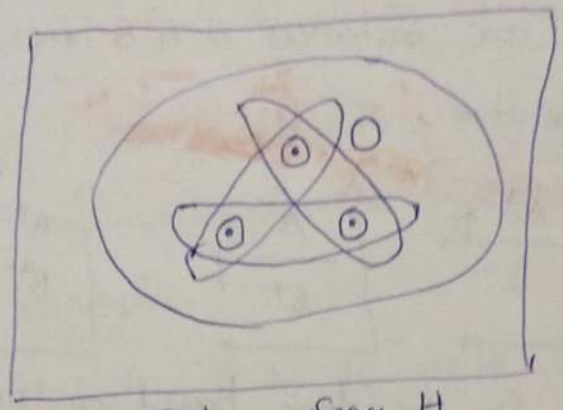
Shattering a Set of Instances:

We measure the complexity of H Called VC-Vapnik-chervonenkins dimension of H

i.e., VC(H).

VC dimension measures the complexity of hypothesis space H, not by number of distinct hypothesis $|H|$, but instead by number of distinct instances from X.

Def: A set of instances S is shattered by Hypothesis space H, if and only if for every dichotomy of S there exists some hypothesis in H Consistent with this dichotomy.



Instance Space H

A set of three Instances Shattered by eight hypothesis, for every possible dichotomy of the instances, there exists a Corresponding hypothesis.

NOTE: If a set of instances is not shattered by a hypothesis space, then there must be some Concept that Can be defined over the instances, but Cannot be represented by hypothesis space.

VAPNIK CHERVONENKIS DIMENSION:

The ability to shatter a set of instances is closely related to inductive bias of a hypothesis space.

Def: The VC(H), of hypothesis space H defined over instance space X is the size of largest finite subset of X shattered by H.

If large finite sets of X Can be shattered by H, then

$$VC(H) \equiv \infty.$$

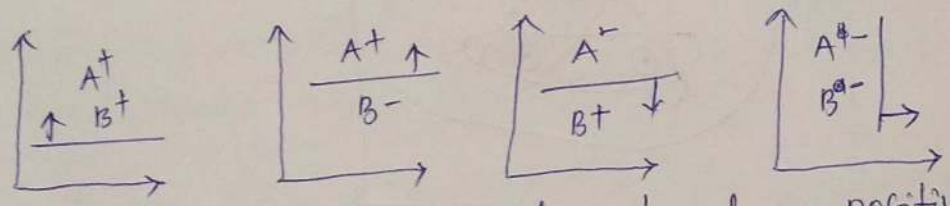
Note that for any finite H, VC(H) Consider few example hypothesis.

$$VC(H) \leq \log_2 |H|$$

i.e., $VC(H) = d$, then H will require 2^d distinct hypothesis to shatter d instances. $2^d \leq |H|$ &

$$d = VC(H) \leq \log_2 |H|$$

Ex:- If we Consider two instances A & B i.e., 2^2 distinct hypothesis Can be done i.e.,



Shattering Can be done based on positivity & negative i.e., probability of an instance.

Ex:- If three instances then 2^3 distinct hypothesis are taken i.e., 8 hypothesis and shattered according to their probability.

MISTAKE BOUND MODEL FOR LEARNING :

→ We Consider the number of mistakes the learner make before it learns the Correct hypothesis.

→ Learner is evaluated by the number of mistakes it makes.

→ Set of training examples are provided to the learner. The learner must predict the target value c(x) before it is shown the Correct value by the trainer.

→ we check how many mistakes it makes in its predictions, before it learns the target Concept.

→ Learning the target Concept exactly means Converging to a hypothesis.

15th Preparties
50, 55, 58, 61 63, 64, 66, 68, (70), 71
72, 73, 75, (76) 77, 78, 81, 82, 83
84, 87, 88, 89, LE: 502, 504, 505
506, 507,

Mistake Bound for FIND-S Algorithm

Recall Find-S algorithm, which incrementally computes the maximally specific hypothesis consistent with the training examples.

FIND-S: ⇒ Initialize h to the most specific hypothesis
$l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \cdots \wedge l_n$
[literals (+ve & -ve)]

⇒ For each positive Training instance x
- Remove from h any literal that is not satisfied by x.

⇒ Output hypothesis h.

FIND-S converges in the limit to a hypothesis that makes no errors $c \subseteq H$ & provide noise-free Training Data.

⇒ For the hypothesis that makes no errors, this generalization step consists of deleting unsatisfied Literals.

⇒ Never classifies a negative example as positive.

⇒ Current hypothesis h is always at least as specific as target concept c.

⇒ Learn how many mistakes occur before FIND-S learns c
& Consider first positive example encountered by FIND-S

→ Learner makes a mistake in classifying this example, because its intial hypothesis labels every instance negative.

→ The number of mistakes will be 1 more than n hypothesis i.e., 1+n or n+1, This number of mistakes will require in worst case.

Weighted-Majority Algorithm:

This algorithm make predictions by taking a weighted vote among a pool of prediction algorithms, and learns altering the weights.

It begins by assigning a weight of 1 to each prediction algorithm, then Considers training examples.

Algorithm: a_i denotes the i^{th} prediction algorithm in a pool of A of algorithms. w_i denotes the weight associated with a_i

- For all i intialize $w_i \leftarrow 1$
- For each training example $\langle x, c(x) \rangle$
  - Intialize q_0 and q_1 to 0
  - for each prediction algorithm a_i
    - if $a_i(x) = 0$ then $q_0 \leftarrow q_0 + w_i$
    - if $a_i(x) = 1$ then $q_1 \leftarrow q_1 + w_i$
- If $q_1 > q_0$ then predict $c(x) = 1$
  If $q_0 > q_1$ then predict $c(x) = 0$
  If $q_0 = q_1$ then predict 0 or 1 at random for $c(x)$.
- For each prediction algorithm a_i in A do
  If $a_i(x) \neq c(x)$ then $w_i \leftarrow \beta w_i$

Instance Based Learning

In general explicit description of the target function when training examples are provided.

⇒ Instance Based Learning methods simply stores the Training Examples.

⇒ It uses Nearest Neighbour & Locally weighted regression methods.

⇒ These are sometimes called "Lazy" learning methods because they delay processing until a new instance must be classified.

⇒ The methods used by this learning are straight forward for approximating real-valued or discrete-valued.

⇒ These use more complex, symbolic representations for instances.

⇒ "neighbouring" instances is elaborated accordingly.

K-Nearest Neighbour Learning:

The most basic instance-based method is K-NEAREST NEIGHBOUR algorithm.

The nearest neighbour of an instance are defined in terms of Examples.

It is an Lazy Learning, because it will not learn function from Training Data., it just Memorize.

Ex:-

Example: Given Data Query

x = [maths = 6, CS = 8] and $k = 3$ (constant gives inf. about nn)

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

we use Euclidean's Approach for distance

$$d = \sqrt{(x_{o_1} - x_{a_1})^2 - (x_{o_2} - x_{a_2})^2}$$

x_o is observed value

x_a is actual value

[6 is observed, 4 is Actual] Maths; [8 is observed, 3 is Actual] CS

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

$$= \sqrt{2^2 + 5^2} = \sqrt{29} = 5.38$$

$$d_2 = \sqrt{(6-6)^2 + (8-7)^2} = \sqrt{0+1} = 1$$

$$d_3 = \sqrt{(6-7)^2 + (8-8)^2} = \sqrt{1+0} = 1$$

$$d_4 = \sqrt{(6-5)^2 + (8-5)^2} = \sqrt{1+9} = \sqrt{10} = 3.16$$

$$d_5 = \sqrt{(6-8)^2 + (8-8)^2} = \sqrt{4+0} = \sqrt{4} = 2$$

∴ Among all distances we have to find nearest three neighbours i.e., d_2, d_3, d_5 are nearest with their values 1, 1, 2

Check the Result of d_2, d_3, d_5 i.e., 2, 3, 5 all of them are pass. so, what we selected are nearest neighbours