28/8/21

## Pedestrian algorithm:

Training experience (E): data set of pedestrians with proper classifications
Task (T): Recognition of pedestrians crossing the road.
Performance measure (P): percentage of pedestrians classified accurately.

CNN: convolution, Relu, Pooling (max, avg, global), fully connected layer (entire matrix arg into single column)

## Data set:

Collection of many examples, where, an example is a collection of features.

## Machine Learning Types:

1) Supervised (Guidance Eg: actual O/P already fed as data set).
2) Unsupervised (Eg: clustering of data set based on features)
3) Reinforcement

## Supervised:

→ Classification, regression, brute force method → all features measured.
→ Machine learns examples along with class labels or targets
→ Test data → O/P for a given I/P, Training → O/P, I/P.
Classification → Which class it belongs to
Regression → Predicting a value in future.
→ O/P → real number, class label.

2

| ML | DL | ML | DL |
|---|---|---|---|
| data prep etc. (feature ext.) | Directly feature ext. | features → I/P required (already done) | → only I/P required Classifies on its own → more data required |

## Unsupervised:

→ Find patterns in data.
→ Grouping similar feature objects together
→ No class labels provided, Learn prob model of data.
→ CLUSTERING, decision making, future values.
→ No supervised O/P target, no reward from environment.
→ Build representation of I/P s.

## Reinforcement learning:

→ Goal oriented algorithm, knowledge to learn how to match a complex goal.
→ Automatically determine the ideal behavior within a specific context to maximise the computer's performance.

Eg: Dog, trainer
Action matches command → +ve reward
Action doesn't match " → −ve or 0 reward

Dog ⇐ actions, rewards, observation ⇒ Trainer

right match

→ Mapping b/w observation, action → policy, which has to be made maximum.
→ Reward signal provided to evaluate the goodness of a trial and guide learning process.

| Dog | Computer |
|---|---|
| → Training is inside the dog's brain | → Training is supervised by training algorithm |
| → Agent is dog | → Agent is computer |
| → Environment is surrounding the dog | → Environment is everything outside the vehicle including car dynamics, weather, nearby vehicles etc. |
| → Observations is what the dog observes | → Observations from car's sensors |
| → Action is how the dog reacts/responds | → Action is generation of steering, braking, acceleration. |
| → Reward is getting a treat from the trainer of the dog | → Reward is 1 correct parking with right orientation, (a signal to mark the same). |

→ ML, probability go hand in hand.

PROBABILISTIC MODEL: 30/8/2021

→ Quantitative description of a situation, a phenomenon or an experiment whose outcome is uncertain.

LEGITIMATE PROBABILITY:

→ That probability which has the highest value.

Sample space:

→ List of all possible outcomes.
→ List is mutually exclusive, collectively exhaustive, and at the right granularity (noise / uncertainty)

PROBABILITY AXIOMS:

Event: Subset of the sample space
- Probability is assigned to events.

AXIOMS FOR LEGITIMATE PROBABILITY:

Consequences

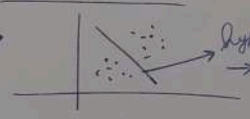
Non negativity: $P(A) \geq 0$ — $P(A) \leq 1$

Normalisation: $P(\Omega) = 1$ ($\Omega$: sample space) — $P(\phi) = 0$.

(Finite) Additivity: If $A \cap B = \phi$, $P(A \cup B) = P(A) + P(B)$
(Disjoint).

31/8/21

HYPERPLANE:

→ 

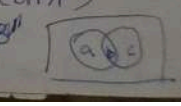
→ distinguishes 2 classes.

AXIOMS:

→ $P(A \cap A^c) = 0$
→ $P(A \cup A^c) = 1$
→ If $A \subset B$, $P(A) \leq P(B)$

→ $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$B = A \cup (B \cap A')$



$a = P(A \cap B')$

$b = P(A \cap B)$

$c = P(B \cap A')$

$a+b+c = P(A \cup B)$

$= P(A \cap B') + P(A \cap B) + P(B \cap A')$

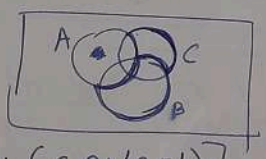
Now, $P(A) + P(B) - P(A \cap B)$

$= (a+b) + (b+c) - b$

$= a+b+c$ //

$\leftrightarrow P((A \cup C) \cap B) = P((A \cap B) \cup (C \cap B))$ if A, C are independent

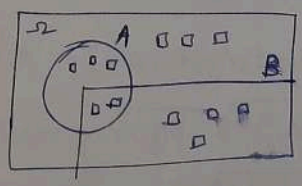
$\leftrightarrow P(A \cup B \cup C) = P(A) + P(A' \cap B) + P(A' \cap B' \cap C)$




$A \cup B \cup C = A \cup [(B \cap A') \cup (C \cap A' \cap B')]$

---

1/9/21 CONDITIONAL PROBABILITY




$P(A/B) = \frac{P(A \cap B)}{P(B) \neq 0}$

$P(A) = \frac{5}{12}$

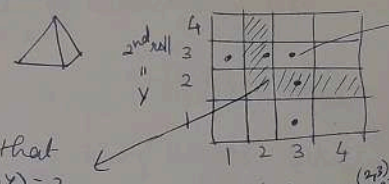
$P(B) = 6/12$

$P(A/B) = \frac{2}{2+4} = \frac{1}{3} \Rightarrow \frac{P(A \cap B)}{P(B)}$

$P(B/B) = 6/6 = 1$

$> 0$

a) Tetrahedral die rolled 2X. Let B event has occurred.




B → event that $\min(X, Y) = 2$

$P(M=3/B)^2$

$= \frac{P(M=3 \text{ and } B)}{P(B)}$

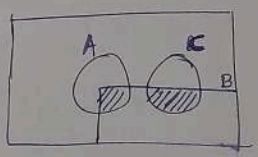
X = first roll $\begin{matrix}(2,3)\\(3,2)\end{matrix}$ $= \frac{2/16}{(\bullet)\ 5/16} = \frac{2}{5}$

---

AXIOM:

$P(A \cup C/B) = P(A/B) + P(C/B)$

Proof:




$P(A/B) + P(C/B) = \frac{P(A \cap B)}{P(B)} + \frac{P(C \cap B)}{P(B)}$

$= \frac{P(A \cap B) + P(C \cap B)}{P(B)}$ — (1)

$P(A \cup C/B) = \frac{P((A \cup C) \cap B)}{P(B)} = \frac{P((A \cap B) \cup (C \cap B))}{P(B)}$

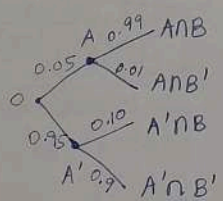
since $A \cap B$, $C \cap B$ are disjoint,

$= \frac{P(A \cap B) + P(C \cap B)}{P(B)}$ — (2)

(1) = (2) proved //.

9) Event A : Aeroplane is flying above.
Event B : Something registered on radar.

$P(A/B) = \frac{P(A \cap B)}{P(B)}$ ; $P(A) = 0.05$



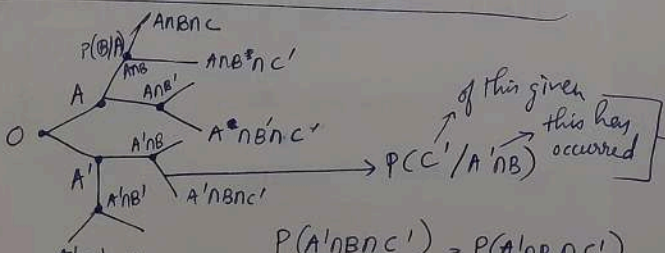

$P(A/B) = \frac{P(A \cap B)}{P(B)}$

$P(A \cap B) = P(A)\,P(B/A) = 0.05 \times 0.99$

$P(B) = 0.05 \times 0.99 + 0.95 \times 0.1 = 0.145$


$P(A/B) = \frac{P(A \cap B)}{P(B)} = \frac{0.05 \times 0.99}{0.145} = 0.34$

2/9/21 MULTIPLICATION RULE

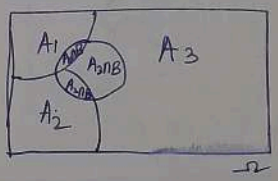



$P(A' \cap B \cap C') = P(A' \cap B \cap C')$
$= P(A' \cap B).P(C'/(A' \cap B))$
$= P(A').P(B'/A').P(C'/(A' \cap B))$

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

TOTAL PROBABILITY THEOREM

$P(B) = P(B/A')P(A') + P(B/A)P(A)$




Sample space is partitioned into $A_1, A_2, A_3$ $P(A_i)$ for every $i$

$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + P(A_3 \cap B)$
$= P(A_1)P(B/A_1) + P(A_2)P(B/A_2) + P(A_3)P(B/A_3)$

$P(B) = \sum_i P(A_i)P(B/A_i)$ $= P(B) = \sum_i P(A_i \cap B)$

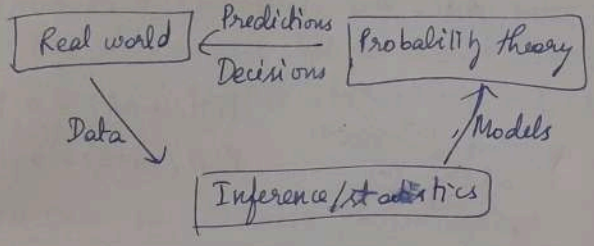
4/9/

4/9/21 PROBABILITY THEOREM

→ Framework to analyse phenomena with uncertain outcomes.

→ Rules for consistent reasoning

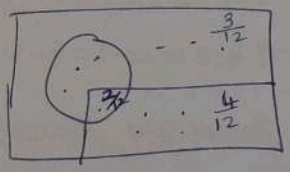
→ Predictions, decisions.




CONDITIONAL, BAYES THEOREM:

→ 3 important rules

→ Bayes' theorem → inference




$$P(A_i/B) = \frac{\{P(A_i)\,P(B/A_i)\}}{\left[\sum_j P(A_j)\,P(B/A_j)\right]}$$

$P(B \cap A_i)$

$P(B)$

INFERENCE!

→ Initial beliefs $P(A_i)$ on possible causes of observed B.

$B \xrightarrow{\text{inference}} A_i$ $P(A_i/B)$ (revised belief) already B has occurred, what is $P(A_i)$?

from basic partition $A_i$, model B ← $A_i$ $\xrightarrow{\text{model}}$ B $P(B/A_i)$ (given $A_i$), what is $P(B)$.

for every i, there will be $A_i$, $P(A_i)$ → initial belief

6/9/21 MODEL BASED ON CONDITIONAL PROB

Head, Tail.

$P(H) = p$

$P(T) = 1-p$.

$P(H)+P(T) = 1$

(~~independent~~, mutually exclusive, collectively exhaustive)


$P(1H) = p(1-p)^2 + p(1-p)^2 + p(1-p)^2$

$= 3p(1-p)^2$

$P(1^{st}H/1H) = \frac{P(1^{st}H \cap 1H)}{P(1H)}$

$= \frac{p(1-p)^2}{3p(1-p)^2} = \frac{1}{3}$

→ $H_2$ or $T_2$ is independent of $H_1$ or $T_1$,
(getting 2nd place H/T is ind. of 1st place H/T).

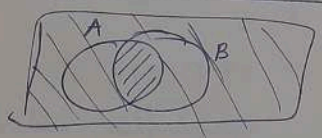
$\therefore P(H_2 \text{ or } T_2 / H_1) = P(H_2 \text{ or } T_2 / T_1)$

INDEPENDENT EVENTS:

$P(A/B) = P(A)$ if A, B independent.

$\Rightarrow P(A \cap B) = P(A)P(B)$

$\Leftrightarrow$ If A, B are independent, A, B' are independent:

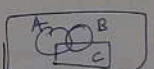



$P(A \cap B') = P(A) - P(A \cap B)$
$= P(A)[1 - P(B)]$
$\boxed{P(A \cap B') = P(A)P(B')}$ //.

$A = (A \cap B) \cup (A \cap B')$

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

CONDITIONAL INDEPENDENCE:



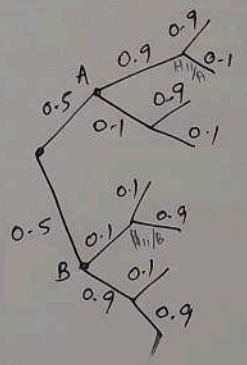

~~If A, B independent,~~ $\boxed{P(A \cap B / C) = P(A/C) \cdot P(B/C)}$

→ Here A, B are not independent.

→ A, B are conditionally independent.
that is, A, B are not independent, but (A/C), (B/C) are independent.

7/9/21 INDEPENDENCE VS. CONDITIONAL INDEPENDENCE

2 coins A, B. Are coin tosses independent?




$P(H/\text{coin A}) = 0.9$
$P(H/\text{coin B}) = 0.1$

$P(\text{toss 11} = H) = P(A)P(H_{11}/A) + P(B)P(H_{11}/B)$
$= 0.5 \times 0.9 + 0.5 \times 0.1 = 0.5$

$P(\text{toss 11} = H / \text{first 10 tosses are Heads})$
$= P(H_{11}/A) = 0.9$
$\therefore$ A has more prob of H.

## INDEPENDENCE OF COLLECTION OF EVENTS:

Eg: $A_1, A_2, \ldots$ are independent events

$P(A_3 \cap A_4') = P(A_3 \cap A_4' / A_1 \cup A_2 \cap A_5')$

$P(A_1 \cap A_2 \cap A_3) = P(A_1)P(A_2)P(A_3)$

### PAIRWISE INDEPENDENCE:

If there are 3 events $A_1, A_2, A_3$ then, they are pairwise independent if $P(A_1 \cap A_2) = P(A_1)P(A_2)$ ; $P(A_2 \cap A_3) = P(A_2)P(A_3)$; $P(A_1 \cap A_3) = P(A_1)P(A_3)$.

### INDEPENDENCE Vs. PAIRWISE INDEPENDENCE:

$H_1 \rightarrow$ first H

⋮ ⋮

| | |
|---|---|
| TT 1/4 | HH 1/4 |
| TH 1/4 | HT 1/4 |

$P(H_1) = P(H_2) = \frac{1}{2}$

$H_n, T_n$ are ~~pairwise~~ independent.

Event $C \rightarrow$ 2 tosses having same result. (TT or HH).

$P(H_1 \cap C) = P(H_1 \cap H_2) = 1/4$

$P(H_1)P(C) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$ } same

thus $H_1, C$ are independent.

$P(H_1 \cap H_2 \cap C) = P(HH) = \frac{1}{4}$

$P(H_1)P(H_2)P(C) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$ } different

$P(C/H_1) = P(H_2/H_1) = P(H_2) = \boxed{1/2 = P(C)}$

Thus $H_1, H_2, C$ are pairwise independent but not independent.

8/9/21

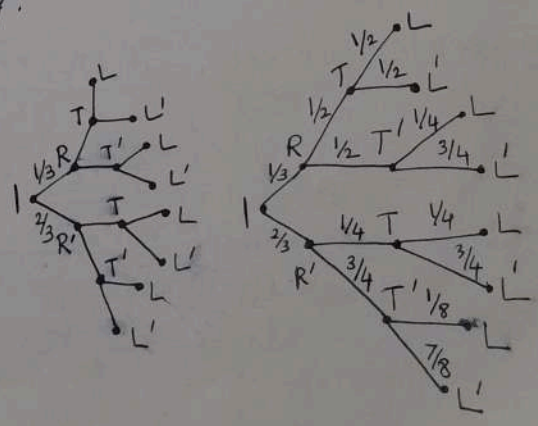
Q) It's raing 1/3 of the days. Given that it is rainy, there will be heavy traffic with prob. 1/2, given it is not rainy, there will be heavy traffic with prob 1/4. If it's rainy and there is heavy traffic, I arrive late with prob. 1/2 Prob. of late reduced to 1/8 if it is not rainy, no heavy traffic. In other wit, prob. of being late = 0.25. You pick a random day.

(a) what is prob that it's not rainy, there is heavy traffic, I am not late?

(b) What is prob I am late?

(c) Given I am late, what is prob that it rained that day?

Ans:



(a) $P(R'TL') = P(R' \cap T \cap L')$

$= \frac{2}{3} \times \frac{1}{4} \times \frac{3}{4}$

$= \frac{1}{8}$

(b) $P(L) = ?$

By total probability theorem,

$P(L) = P(R \cap T \cap L) + P(R \cap T' \cap L) + P(R' \cap T \cap L) + P(R' \cap T' \cap L)$

$= \left(\frac{1}{3} \times \frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{3} \times \frac{1}{2} \times \frac{1}{4}\right) + \left(\frac{2}{3} \times \frac{1}{4} \times \frac{1}{4}\right) + \left(\frac{2}{3} \times \frac{3}{4} \times \frac{1}{8}\right)$

$= \frac{1}{12} + \frac{1}{24} + \frac{1}{24} + \frac{1}{16}$

$= \frac{4+2+2+3}{48} = \frac{11}{48}$

(c) $P(R/L) = \frac{P(R \cap L)}{P(L)} = \frac{P(RTL) + P(RT'L)}{P(L)}$

$= \frac{\left(\frac{1}{12} + \frac{1}{24}\right)}{\frac{11}{48}} = \frac{\left(\frac{4+2}{48}\right)}{\frac{11}{48}}$

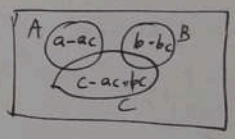
$= \frac{6}{11}$ //

13/9/24

Q) A, B are disjoint events.

$P(B \cup C) = 3/4$ $\quad P(A \cup C) = \frac{2}{3}$

$P(A \cup B \cup C) = 11/12$ $\quad P(C) = ?$




$P(A \cup C) = \frac{2}{3} = a + c - ac$ — (1)

$P(B \cup C) = \frac{3}{4} = b + c - bc$ — (2)

$P(A \cup B \cup C) = \frac{11}{12} = a + b + c - ac - bc$ — (3)

(1) + (2) $\Rightarrow$ $a + b + 2c - ac - bc = \frac{17}{12}$ — (4)

(4) − (3) $\Rightarrow$ $c = \frac{17 - 11}{12} = \frac{6}{12} = \frac{1}{2}$

$P(C) = \frac{1}{2}$ //

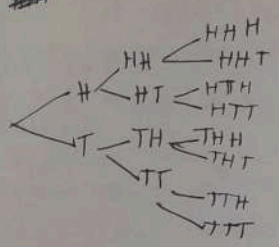
Q) Coin tossed 3 times.

(a) Probability of 3 heads = ?

(b) Prob that you get exactly one head

(c) Given atleast one head, what is prob that you get atleast 2 heads?

Ans:




(a) $P(HHH) = \frac{1}{8} = P(H)P(H)P(H)$

(b) $P(\text{exactly one head}) = P(HTT) + P(THT) + P(TTH)$

$= \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}$

(c) $P(\geq 2H / \geq 1H) = \frac{P(\geq 2H \cap \geq 1H)}{P(\geq 1H)}$

$= \frac{P(HHT) + P(HTH) + P(THH) + P(HHH)}{1 - P(TTT)}$

$= \frac{\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}}{1 - \frac{1}{8}} = \frac{4}{7}$

14/9/21 COVARIANCE, VARIANCE

COVARIANCE:

→ Gives information about how X, Y variables are statistically related.

$cov(X,Y) = E[(X - E(X))(Y - E(Y))]$

$= E[XY - XE(Y) - YE(X) + E(X)E(Y)]$

$= E[XY] - E(X)E(Y) - E(X)E(Y) + E(X)E(Y)$

$= E(XY) - E(X)E(Y)$

If X, Y are directly proportional, cov(X,Y) is +ve

If X, Y are indirectly proportional, cov(X,Y) -ve.

PROPERTIES:

1) $cov(X,X) = var(X) = E(X^2) - E^2(X)$
2) X, Y are independent, $cov(X,Y) = 0$. $\because E(XY) = E(X)E(Y)$
3) $cov(X,Y) = cov(Y,X)$
4) $cov(aX, Y) = a\,cov(X,Y) = aE(XY) - a(E(X)E(Y))$
5) $cov(X+c, Y) = cov(X,Y)$
6) $cov(X+Y, Z) = cov(X,Z) + cov(Y,Z)$

$= E((X+Y)Z) - E(X+Y)E(Z)$

$= E(XZ + YZ) - E(X+Y)E(Z)$

$= E(XZ) + E(YZ) - E(X)E(Z) - E(Y)E(Z)$

$= E(XZ)E(Z) + E(Y)E(Z) - E(X)E(Z) - E(Y)E(Z)$

$= E(XZ) + E(YZ) - E(X)E(Z) - E(Y)E(Z)$

$= cov(X,Z) + cov(Y,Z)$.

7) Covariance of a sum: $Z = X+Y$

$Var(Z) = cov(Z,Z) = cov(X+Y, X+Y)$

$= cov(X,X) + cov(X,Y) + cov(Y,X) + cov(Y,Y)$

$= var(X) + 2\,cov(X,Y) + var(Y)$

8) $Var(aX+bY) = a^2 var(X) + 2ab\,cov(X,Y) + b^2 var(Y)$

CORRELATION COEFFICIENT: $\rho$

$\rho(X,Y) = cov(U,V)$

$U = \frac{X-E(X)}{\sigma_X}$ ; $V = \frac{Y-E(Y)}{\sigma_Y}$

$cov(X+a, Y) = cov(X,Y)$

$\Rightarrow \rho(X,Y) = cov\left(\frac{X}{\sigma_X}, \frac{Y}{\sigma_Y}\right)$

$\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$ → constants.

PROPERTIES:

1) $-1 \le \rho(X,Y) \le 1$

2) $\rho(X,Y) = 1$, $Y = aX+b$, $a>0$

3) $\rho(X,Y) = -1$, $Y = aX+b$, $a<0$

4) $\rho(aX+b, cY+d) = \rho(X,Y)$ for $a,c>0$

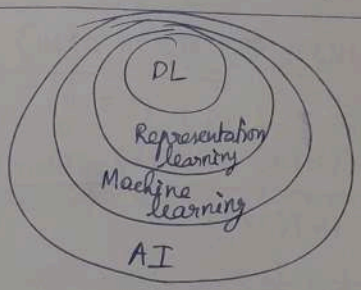
5) $\rho(X,Y) = 0 \Rightarrow X, Y$ not correlated.

6) $\rho(X,Y)$ +ve or −ve $\Rightarrow X, Y$ +ve or −ve correlated.

7) If $X, Y$ are random var, $X, Y$ are independent then $X, Y$ are not correlated.

If $X, Y$ are not correlated, then $X, Y$ may or may not be independent rand. var.

15/9/21 DEEP LEARNING




DL:

→ To extract useful patterns from data.
→ Neural N/W + optimisation.
→ Python + Tensorflow + friends
→ Good questions + good data
→ Data, hardware, community, tools, investment
→

DL → ML with multilayer NN
RL → ML extracting features of datasets
ML → Machines improve with experience.
AI → Computers mimic human intelligence.

## Alternative Data:

→ Bank, Checking, Employment, Income, Insurance, Tenant, Utilities.
→ Cash flow underwriting.
→ App usage, Email receipt, Browsing history, Geolocation, Social Media Data.

## Natural Language Processing:

→ Computer I/p, interpretation, O/p of Human Lang.
→ Audio, image, text, video including spoken, written, gestured.
→ Content generation, speech recognition, translation.
→ Chatbots, conversational interfaces, voice assistants.

## BIG DATA:

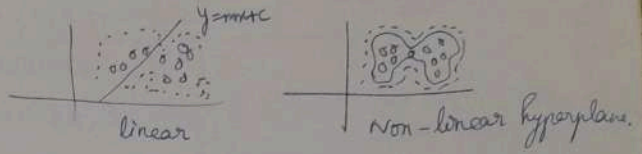
→ Describes storage, analysis of large / complicated data sets using techniques like AI.

## ML:

→ Method of designing sequence of actions to solve problem (algorithms) that optimise automatically through experience.

## ACTIVATION FN:

→ Introduce non-linearities into N/w.




## ANN:

I/p, Hidden layer, O/p
↓
bias, weights, activ. fn

---

11/9/21 VARIANCE

→ Defined as the measure of probability mass function spread.
→ Distance from mean (avg) = $E[X-\mu] = E[X]-\mu = 0$.

$$\boxed{Var(X) = E[(X-\mu)^2]} \geq 0.$$

$$E[g(X)] = \sum_x g(x) P[X(x)]$$

$$var(X) = E[g(X)] = \sum_x (x-\mu)^2 P[X(x)]$$

$$S.D = \sigma_x = \sqrt{var(X)}$$

## PROPERTIES OF VAR(X):

1) $var(aX+b) = a^2 Var(X)$

2) $var(X+b) = E[(Y-E(Y))^2] = E[(X+b-(E(X)+b))^2]$
$(Y = X+b)$
$= E[(X-E(X))^2] = E[(X-\mu)^2]$
$= var(X)$

3) $\text{var}(aX) = E[(X-\mu)^2] = E[X^2 - 2\mu X + \mu^2]$

$= E[X^2] - 2\mu E(X) + \mu^2$

$= E[X^2] - 2E(X)E(X) + [E(X)]^2$

$= E[X^2] - E[X]^2$

$= E[X^2] - (E[X])^2$

## CONDITIONAL PROBABILITY MASS FUNCTION:

$P_X(x) = P(X = x)$

$P(A) > 0.$

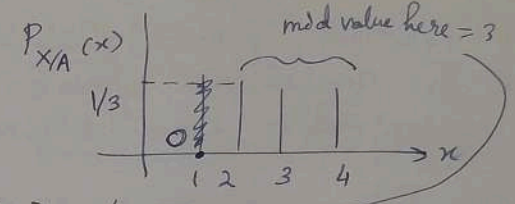
$P_{X/A}(x) = P(X = x/A)$

$\sum_x P_{X/A}(x) = 1 \; ; \; E[X/A] = \sum_x x P_X$

$\sum_x P_{X/A}(x) = P(X = x/A)$

$\sum_x P_{X/A}(x) = 1 \; ; \; \boxed{E[X/A] = \sum_x x P_{X/A}(x)}$

Let $A = X \geq 2$

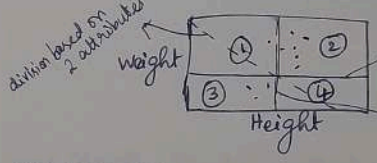



$E[X/A] = 3$

$\text{var}[X/A] = \frac{1}{3}[(4-3)^2] + \frac{1}{3}(3-3)^2 + \frac{1}{3}(2-3)^2 + 0$

$= \frac{1}{3} + 0 + \frac{1}{3} = \frac{2}{3}$

16/9/21 HEIGHT, WEIGHT




(clustering)

## OBJECTIVE FUNCTION:

→ A threshold set to cluster the individuals.

## FEATURES:

Eg:

| Snake names | egg | layey | scales | poison | Cold blooded | legs | ... |
|---|---|---|---|---|---|---|---|
| 1 | True | False | | | | | |
| 2 | | | True | False | True | False | True |
| 3 | | | | | | | |
| : | | | | | | | |

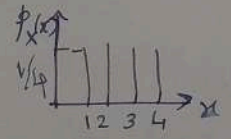
Euclidean distance:

$Z = \sqrt{x^2 + y^2}$


Manhattan distance:

$Z = x + y.$

9)




$E(X) = 2.5 = \frac{10}{4}$

$\text{var}(X) = \frac{1}{12}(b-a)(b-a+2)$

$= \frac{1}{12}(4-1)(4-1+2)$

$= \frac{1}{12} \times 5 \times 3 = \frac{5}{4}$

Eg: alligator = Animal ('alligator', [1,1,0,1,1])
animals.append(alligator)
compareAnimals(animals, 3)
frog = (1,0,1,0,1)

dist b/w frog, alli ⇒ 1 1 0 1 1 } 5 features
1 0 1 0 1

$\underbrace{\sqrt{0^2+1^2+1^2+1^2+0^2}}_{\text{difference squares}} = \sqrt{3} = 1.732$ is dist b/w alligator, frog.

→ when dist of features b/w 2 elements is less, they are closely related.

18/9/21 GRADIENT DESCENT OPTIMISATION

EPOCH → one turn of Complete training of model.

cost fn
↑
$J(w) = \frac{1}{2} \sum (\text{target}_i - \text{expected output}_i)^2$
(Sum of squared error).

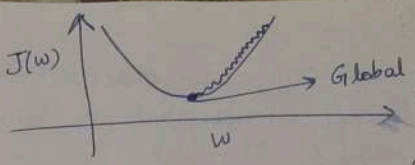
$\Delta w_j = -\eta \frac{dJ}{dw_j}$

↓ learning rate

$w = w + \Delta w$ ; $\Delta w = -\eta \sum_i (\text{target}_i - o/p_i)(-x_j^i)$

training data

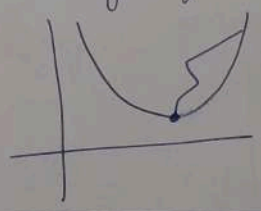
$\Delta w = \eta \sum_i (\text{target}_i^i - o/p_i^i)\, x_j^i$




weight coefficient → hiker ; leg length of hiker → learning rate (diff in step lengths).
J(w) → aim of climbing down.

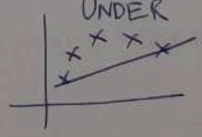
DISADVANTAGE:
→ Only after each epoch, weight updated.
→ Large data set takes much time to reach global min (least cost).

STOCHASTIC GRADIENT DESCENT:
→ Weights are updated only after completion of training dataset, in normal gradient descent.
→ Here, weight of 1st data updated, then it goes on.

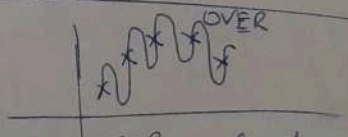


→ Path to global min need not be direct.

21/9/21 UNDER, OVER FITTING




low complexity
high bias
low variance




high complexity
low bias
high variance

## CROSS VALIDATION:

→ Over, underfitting have very clear ~~data~~ difference.
→ Dividing entire training set into parts for training, testing, so as to validate the model.
→ This is maximum likelihood estimation.

## UNDERFITTING:

→ If all the weights have the same value, then there is no learning happening in the neural network.

---

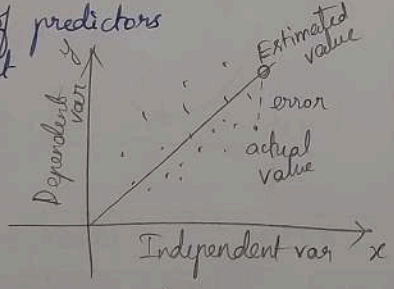
22/9/21 LINEAR REGRESSION

---

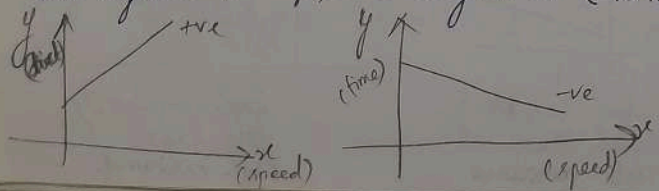
→ For evaluating trends, sales estimates.
→ Analysing impact of price changes.
→ Risk in financial services.

## 3 main uses:

→ Finding strength of predictors
→ Forecasting an effect
→ Trend forecasting.




+ve regression (+ve slope) — -ve regression (-ve slope)




$y = mx + c$ → Line of linear regression.

## BEST FIT:

→ Best fitted line has least error.
→ Aim is to have least possible error.

## FINDING REGRESSION EQUATION:

| | | | | | | Mean |
|---|---|---|---|---|---|---|
| x | 1 | 2 | 3 | 4 | 5 | 3 |
| y | 3 | 4 | 2 | 4 | 5 | 3.6 |

$y = mx + c$ ← actual

$\bar{x}$ = mean of $x = \frac{\Sigma x}{5} = 3$

$\bar{y}$ = mean of $y = \frac{\Sigma y}{5} = 3.6$

slope $m = \frac{\Sigma (x-\bar{x})(y-\bar{y})}{\Sigma (x-\bar{x})^2} = \frac{4}{10} = \boxed{0.4 = m}$

$\bar{y} = m\bar{x} + c \Rightarrow 3.6 = 0.4(3) + c$

$\Rightarrow c = 2.4$

$$\boxed{y = 0.4x + 2.4}$$

~~Actual~~ Regressed value:

| x | y |
|---|---|
| 1 | 2.8 |
| 2 | 3.2 |
| 3 | 3.6 |
| 4 | 4 |
| 5 | 4.4 |

} predicted

GOODNESS OF FIT: (Least square estimation)

$R^2$ → represents how close the data are being fixed.

$$R^2 = \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2} = \frac{1.6}{5.2} \simeq 0.3$$

($y_p$: predicted, $\bar{y}$: mean, $y$: actual)

$R^2 \propto$ goodness of fit.

23/9/21 GRADIENT DESCENT OPTIMISATION

Q) Find minimum value of the function $f(x,y) = x^2+y^2+2x+4$

$\bar{x}^0 = \begin{Bmatrix} x \\ y \end{Bmatrix} = \{2,1\}$ ⇒ Initial estimate of the optimal solution.

Ans
Gradient Objective function: $f(x,y) = y^2+x^2+2x+4$

To find $f_{min}(x,y)$

Iteration 1: To calculate the gradient,

$\frac{\partial f}{\partial x} = 2x+2$ $\quad$ $\frac{\partial f}{\partial y} = 2y$

$x_{min} = -2$ ↑ $\quad$ $y_{min} = -1$ ↑

⇒ $2(-2)+2$ ⇒ 2

$= -6$

$\nabla f = 6i + 2j$

$\bar{x}^{(i+1)} = x^i + h\nabla f$

$\bar{x}^{(i+1)} = \begin{Bmatrix} 2 \\ 1 \end{Bmatrix} + h\begin{Bmatrix} 6 \\ 2 \end{Bmatrix} = \begin{Bmatrix} 2+6h \\ 1+2h \end{Bmatrix}$

$f(x,y) = x^2+y^2+2x+4$

$= (2+6h)^2 + (1+2h)^2 + 2(2+6h) + 4 = g(h)$

gradient of h

Golden section search $\quad$ Newtons method.

$g(h) = 40h^2+40h+13$

$g_{min} \Rightarrow \frac{dg}{dh} = 0 \Rightarrow 80h+40 = 0$

$h = -1/2 = -0.5 = h$

$\bar{x}^{(i+1)} = \begin{Bmatrix} 2+6(-0.5) \\ 1+2(-0.5) \end{Bmatrix} = \begin{Bmatrix} -1 \\ 0 \end{Bmatrix}$

$\bar{x}^{(0)} \Rightarrow \{2,1\} \Rightarrow (4^2)+(1)+2(2)+4 = 13 = f$

$\bar{x}^{(i+1)} \Rightarrow \{-1,0\} \Rightarrow 1+0+(-2)+4 = 3 = f$ } obj function

To find if objective function value is global minimum or not,

Iteration 2:

Initial points: $(x^{i+1}, y^{i+1}) = (-1, 0)$

$\frac{\partial f}{\partial x} = 2x + 2$

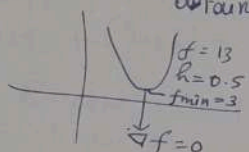
$= 2(-1) + 2$

$= 0$

$\frac{\partial f}{\partial y} = 2y \Rightarrow (2(0)) = 0$

$\nabla f = 0i + 0j$ (Gradient is 0, so global min obtained)

$f_{min}$ at $(-1, 0) = 3$.




27/9/21 ML

→Learn from data
→Establishes relationship b/w multiple features.
→Extract statistical patterns
→Reasoning under uncertain conditions.

SUPERVISED LEARNING ALGORITHMS:

→Data set collection → Brute force method (all relevant information is collected).
→There might be noisy data, missing of important data
→Algorithm selection based on prediction accuracy.
→Cross validation present.

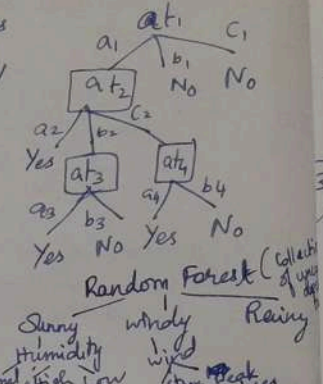
1) DECISION TREE SUPERVISED LEARNING ALGORITHM:

→Classify instances by sorting them based on its features in training set.

Training set

| $at_1$ | $at_2$ | $at_3$ | $at_4$ | class |
|---|---|---|---|---|
| $a_1$ | $a_2$ | $a_3$ | $a_4$ | Y |
| $a_1$ | $a_2$ | $a_3$ | $b_4$ | Y |
| $a_1$ | $b_2$ | $a_3$ | $a_4$ | Y |
| $a_1$ | $b_2$ | $b_3$ | $b_4$ | N |
| $a_1$ | $c_2$ | $a_3$ | $a_4$ | Y |
| $a_1$ | $c_2$ | $b_3$ | $b_4$ | N |
| $b_1$ | $b_2$ | $b_3$ | $b_4$ | N |
| $c_1$ | $b_2$ | $b_3$ | $b_4$ | N |

Tree:



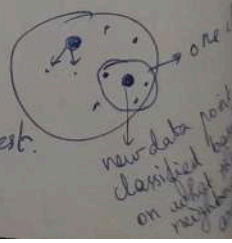

→Depending on the data, there decisions are made.
→Vertical → features.
→Decision tree with least no. of branches is good, others are pruned (Eg: Here, $b_1, c_1 \Rightarrow$ No, no need of other branches)

2) KNN ALGORITHM (K nearest neighbours):

→It is a lazy algorithm.
→Takes a lot of time for computation.
→Calculates distance b/w k neighbours, find which 2 are nearest to this point.

Euclidean dist. $D(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

↓ chooses which k point is nearest.
lot of computation

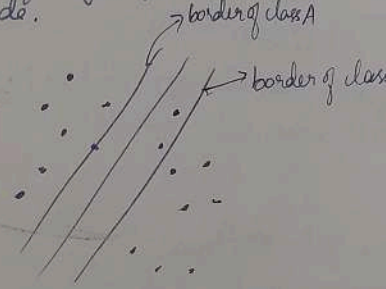


Minkowsky dist $D(x,y) = \left[\sum_{i=1}^{n} (x_i - y_i)^r\right]^{1/r}$

Manhattan dist: $D(x,y) = \sum_{i=1}^{n} |x_i - y_i|$

3) SUPPORT VECTOR MACHINE: (Used for both classification, regression).

→ Uses hyperplane (a line that separates two classes with max distance possible) (best fit)
→ Optimisation
→ Maximises separation b/w classes, maximises distance of hyperplane to the support vector on either side.




→ Quadratic polynomial hyperplane.
→ Radial basis function.
→ Gaussian plane kernel.

---

25/9/21 MAXIMUM LIKELIHOOD ESTIMATION

Random variable: $X_1, X_2, \ldots X_n$

Likelihood fn: $L(\theta) = P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$

Aim: To find the value of $\theta$, for which max likelihood fn.

Example of MLE:

$X_1, X_2, \ldots, X_n \Rightarrow$ Random samples.
$X_i = 0 \Rightarrow$ Any selected student doesn't own a car.
$X_i = 1 \Rightarrow$ Any selected student owns a car.
$X_i \rightarrow$ independent Bernoulli R.V with unknown parameter $p$.
(estimator of $p$ to be found)

Ans: Probability mass function of each $X_i$ is

$$f(x_i, p) = p^{x_i}(1-p)^{(1-x_i)}$$ such that- $p_i \in [0,1]$, $x_i \in \{0,1\}$

1) $L(p)$ (likelihood) $= \prod_{i=1}^{n} f(x_i, p) = p^{x_1}(1-p)^{x_1} p^{x_2}(1-p)^{x_2} \ldots p^{x_n}(1-p)^{x_n}$

(unknown parameter)

2) $0 < p < 1$.

$$L(p) = p^{\sum x_i}(1-p)^{n-\sum x_i}$$ PMF

3) $\ln \Rightarrow$ $\ln(L(p)) = \left[\sum x_i \ln(p)\right] + \left[(n - \sum x_i)\ln(1-p)\right]$

4) diff w.r.t $p \Rightarrow$ $\frac{1}{L(p)} \frac{dL(p)}{dp} = \left[\frac{1}{p}\sum x_i\right] + \frac{(n-\sum x_i)}{1-p}(-1) = 0$

$$\frac{p(1-p)}{L(p)}\frac{dL(p)}{dp} = (1-p)\Sigma x_i - p(n-\Sigma x_i) = 0$$

$$\Sigma x_i - p\Sigma x_i - np + p\Sigma x_i = 0$$

$$\boxed{\Sigma x_i = np} \qquad \boxed{p = \frac{\Sigma x_i}{n}}$$

max. likelihood estimator

## LIKELIHOOD FUNCTION:

$x_1, x_2, \ldots x_n$ are random variables that depend on one or more unknown parameters $\theta_1, \theta_2, \ldots \theta_m$.

PMF $f(x_i; \theta_1, \theta_2, \ldots \theta_m)$

$\theta_1, \theta_2, \ldots \theta_m$ are restricted to a parameter space of $\Omega$.

Joint PMF of $x_1, \ldots x_n$

$$L(\theta_1, \theta_2, \ldots \theta_m) = \prod_{i=1}^{n} f(x_i; \theta_1, \theta_2, \ldots \theta_m)$$

$(\theta_1, \theta_2, \ldots \theta_m)$ in $\Omega$ is called as likelihood fn.

→ Let $u_1(x_1, x_2, \ldots x_n)$, $u_2(x_1, \ldots x_n)$ .... $u_m(x_1, \ldots x_n)$ be a $m$-tuple that maximises likelihood fn.

→ The estimator (max) $\theta_i = u_i(x_1, \ldots x_n)$ for $i = 1 \ldots m$

28/9/21 BAYESIAN LEARNING/INFERENCE

→ Bayes rule is transformed into naive Bayes classifier.

→ ML models are based on probability of observed data

## CLASSIFICATION PROBLEM:

Training info: Desired (target) o/p

(data) I/p → [Supervised Learning] → o/p (hypotheses)

Error = [Target o/p − Actual o/p]

## Training data:

→ Of the form $(d, h(d))$

→ $d$ → data objects to classify (I/ps)

→ $h(d)$ → correct class info for $d$

## GOAL:

→ Give $d_{new}$, get $h(d)_{new}$ or $h(d_{new})$

## BAYESIAN FRAMEWORK:

→ Combining observed data, prior knowledge

→ Provides practical learning algorithm.

→ Generative (model based) approach, offers useful conceptual framework.

→ Any kind of objects (time series, trees etc.) can be classified based on probabilistic model specs. (Bayesian Framework)

BAYES RULE:

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

- $d \rightarrow$ data
- $h \rightarrow$ hypothesis

$P(h) \rightarrow$ Prior belief (prob. of hypothesis before seeing data)

$P(d/h) \rightarrow$ likelihood (prob of data is hypothesis is true)

$P(d) = \sum_h P(d/h)\,P(h) \rightarrow$ Data evidence (marginal probability of data)

Joint probability: $P(A,B) = P(A \cap B) = P(A/B)\,P(B)$

$$P(X) = \sum_i P(X,Y) = \sum P(X/Y)\,P(Y) = 1$$

→ Conditional probabilities sum to 1 provided that their conditions are the same.

---

q) Patient takes lab test, result is positive. Test returns correct positive result 98% of the cases, correct negative in 97% cases. 0.008 of entire population has this disease.

(i) What is the probability that this person has cancer?

(ii) What is the probability that he doesn't have cancer?

(iii) What is the diagnosis?

Ans

(i) $P(\text{cancer}/+) = \frac{P(+/\text{cancer})\,P(\text{cancer})}{P(+)}$

$= \frac{P(\text{cancer} \cap +)}{P(+)}$

$P(+/\text{cancer}) = 0.98$

$P(\text{cancer}) = 0.008$

$P(+) = P(+/\text{cancer})\,P(\text{cancer}) + P(+/-\text{cancer})\,P(-\text{cancer})$

$P(\text{cancer}/+) = \frac{0.98 \times 0.008}{0}$

$P(+/-\text{cancer}) = 0.03 \Rightarrow (1-0.97)$

$P(+) = (0.98 \times 0.008) + (0.03)(0.992)$

$= 0.0376$

$P(\text{cancer}/+) = \frac{0.98 \times 0.008}{0.0376} = 0.2085$

(ii) $P(-\text{cancer}/+) = \frac{P(-\text{cancer} \cap +)}{P(+)}$

$= \frac{P(+/-\text{cancer})\,P(-\text{cancer})}{P(+)}$

$= \frac{0.03 \times 0.992}{0.0376} = 0.79148 = (1-0.2085)$

(iii) The diagnosis is probably false since (i) < (ii), the person doesn't have cancer.

## CHOOSING HYPOTHESES:

Maximum likelihood hypothesis : $h_{ML} = \underset{h \in H}{\operatorname{argmax}} P(d/h)$

Generally, most probable hypothesis given training data, which is maximum a posteriori hypothesis. : $h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h/d)$

Useful observation : It doesn't depend on denominator $P(d)$

## NAIVE BAYES CLASSIFIER:

→ Assumptions:

Attributes that describe data instances are conditionally independent given the classification hypothesis.

$$P(d/h) = P(a_1, \ldots a_n / h) = \prod_t P(a_t / h)$$

$a_i$ → attributes

→ It is a simplifying assumption, it is violated in reality.

→ Bayes classifier that uses Naive Bayes assumption, computes MAP hypothesis is Naive Bayes classifier.

→ It is a practical learning method.

Applications:

- Medical diagnosis
- Text classification

## NAIVE BAYES SOLUTION:

→ Classify any new datum instance $x = (a_1, \ldots a_t)$ as:

$$h_{\text{Naive Bayes}} = \underset{h}{\operatorname{arg\,max}} P(h) P(x/h)$$
$$= \underset{h}{\operatorname{argmax}} P(h) \prod_t P(a_t / h)$$

→ To do this based on training examples, the parameters have to be estimated from training examples.

→ For each target value (hypothesis) $h$

$\hat{P}(h)$ = estimate $P(h)$

$\hat{P}(a_t / h)$ = estimate $P(a_t / h)$

→ For each value at each of the datum (i.e. data row)

Eg: $x = (\text{outlook} = \text{sunny}, \text{temp} = \text{cool}, \text{hum} = \text{High})$

$$h_{NB} = \underset{h}{\operatorname{argmax}} P(h) P(x/h) = \underset{h}{\operatorname{argmax}} P(h) \prod_t P(a_t / h)$$

$$\underset{h}{\operatorname{argmax}} P(h) P(\text{out} = \text{sunny}/h) P(\text{temp} = \text{cool}/h) P(\text{high}/h)$$

Here,

$P(\text{playtennis} = \text{yes}) = 9/14 = 0.64$ → no. of rows with yes

$P(\text{playtennis} = \text{no}) = 5/14 = 0.36$

$P(\text{wind} = \text{strong} / \text{Playtennis} = \text{yes}) = \frac{3}{9} = 0.33$

$P(\text{wind} = \text{strong} / \text{playtennis} = \text{no}) = \frac{3}{5} = 0.6$

⋮

$P(\text{yes})\, P(\text{sunny}/\text{yes})\, P(\text{cool}/\text{yes})\, P(\text{high}/\text{yes}) = 0.0053$

$P(\text{no})\, P(\text{sunny}/\text{no})\, P(\text{cool}/\text{no})\, P(\text{high}/\text{no}) = 0.0206$

answer: ~~playTennis~~ playTennis $(x) =$ no ($\because$ argmax with highest prob)

→ Bayes rule can be turned into a classifier.

→ Maximum A posteriori (MAP) hypothesis estimation uses prior knowledge, max likelihood doesn't.

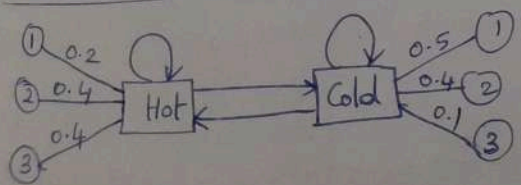
→ Naive Bayes classifier assumes that attributes are independent given the class.

---

11/12/21 REVISION

---

Q) Hidden Markov model:




Transition

| | H | C |
|---|---|---|
| H | 0.7 | 0.3 |
| C | 0.4 | 0.6 |

Emission

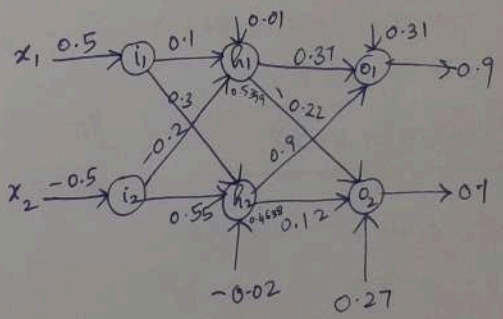
| | 1 | 2 | 3 |
|---|---|---|---|
| H | 0.2 | 0.4 | 0.4 |
| C | 0.5 | 0.4 | 0.1 |

$T_1 = [0.8 \;\; 0.2]$ initial (H C)

---

18/12/21 BACKWARD PROPAGATION

| Pattern index | $x_1^k$ | $x_2^k$ | $d_1^k$ | $d_2^k$ |
|---|---|---|---|---|
| 1 | 0.5 | −0.5 | 0.9 | 0.1 |
| 2 | −0.5 | 0.5 | 0.1 | 0.9 |




$h_1 = (0.5)(0.1) + 0.01 + (-0.5 \times -0.2)$

$= 0.05 + 0.01 + 0.1$

$= 0.16$

$h_1' = \frac{1}{1+e^{-h_1}} = 0.5399$

$h_2 = -0.5 \times 0.55 + (-0.02) + (0.5 \times 0.3)$

$= -0.275 - 0.02 + 0.15$

$= -0.145.$

$h_2' = \frac{1}{1+e^{-h_2}} = \frac{1}{1+e^{0.145}} = 0.46381$

$o_1 = (0.5399 \times 0.37) + 0.31 + (0.9 \times 0.46381)$

$= 0.19976 + 0.31 + 0.4174.$

$= 0.927163$

$o_1' = 0.7164$

$o_2 = (-0.22 \times 0.5399) - (0.12 \times 0.46381) + 0.27$

$= -0.11877 - 0.05565 + 0.27$

$= 0.09557$

$o_2' = 0.5238$

$y_1 = 0.9$

$y_2 = 0.1$

$e_1 = 0.9 - 0.7164 = 0.1836$

$e_2 = 0.1 - 0.5238 = -0.4238$

Backward pass:

O/p to hidden layer

$\delta'' = e_1 * \delta(o_1')$

$= 0.1836 \times o_1' (1 - o_1')$

$= 0.1836 * 0.7164 (1 - 0.7164) = 0.0373$

$o_2'' = e_2 \ \delta(o_2')$

$= -0.4238 \ (o_2')(1 - o_2')$

$= -0.4238 \ (0.5238)(1 - 0.5238)$

$= -0.1057$

$\Delta W_{new} = \Delta W_{old} - \eta \frac{d\varepsilon}{dw_{ij}}$

$h_1'' = [(0.0373 \times 0.37) + (-0.1057 \times -0.22)] \times$

$(0.5399)(1 - 0.5399)$

$\delta'(h_1') = h_1'(1 - h_1')$

$= 0.00912$

$h_2'' = [(0.0373 \times 0.9) + (-0.1057 \times -0.12)] (0.46$

$= [0.03357 + 0.012684] (1 - 0.46$

$= 0.0115$

$\Delta w_1 = 0.31 - 0.5(0.0373 \times 1) = 0.29135$ (bias)

$\Delta w_2 = 0.37 - 0.5(0.0373 \times 0.5399) = 0.3599$ ($o_1''$, $o_1''$)

$\Delta w_3 = 0.9 - 0.5(0.438 \times 0.0373) = 0.8918$ ($h_1'$, $h_2'$, $o_1''$)

$\Delta w_4 = 0.27 - 0.5(-0.1057) = 0.32285$

$\Delta w_5 = -0.12 - 0.5(-0.1057 \times 0.4638) = -0.0954$

$\Delta W_6 = -0.22 - 0.5(0.5399 \times -0.1057)$

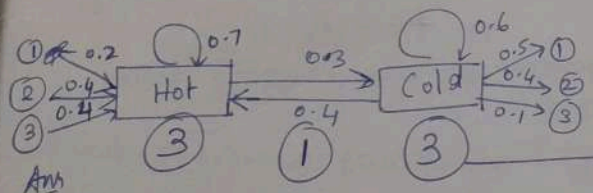
$= -0.19146$

update all weights, then do forward pass. Check $o_1'$, $o_2'$, till it becomes equal to $y_1, y_2$.

---

Q) Hidden Marco Model.

$T_1 = [\overset{H}{0.8}, \overset{C}{0.2}]$




→ states to be considered in this sum

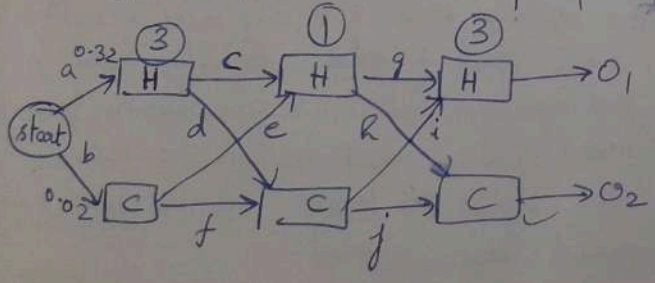
Ans

Transition matrix:

| | H | C |
|---|---|---|
| H | 0.7 | 0.3 |
| C | 0.4 | 0.6 |

Emission matrix:

| | 1 | 2 | 3 |
|---|---|---|---|
| H | 0.2 | 0.4 | 0.4 |
| C | 0.5 | 0.4 | 0.1 |




$a = P(3,H) = P(3/H)\ P(H)$

$= 0.4 \times 0.8 = 0.32$

(emission)

$b = P(3,C) = P(3/C)\ P(C)$

$= 0.1 \times 0.2 = 0.02$

$c = P(1,H) = P(1/H)\ P(H)$ → Transition matrix

$= 0.2 \times 0.7 = 0.16$

$d = P(1,C) = P(1/C)\ P(C/H)$ → ∵ initially Hot, then cold

$= 0.5 \times 0.3 = 0.15$

$e = P(1,H) = P(1/H)\ P(H/C)$

$= 0.2 \times 0.4 = 0.8$

$f = P(1,C) = P(1/C)\ P(C)$

$= 0.5 \times 0.2$

$= 0.1$

$c = P(1/H)\ P(H/H)$

$d = P(1/C)\ P(C/H)$

# UNIT I PROB. SUMS

Q) A, B play a game. A, B toss coins, If both H, A gets 200, if both T B gets 100. One H, One T, no money given. A → biased coin with 60% prob, B fair coin. Find E(A), E(B), variances.

Ans:

$P(H/A) = 0.6$, $P(T/A) = 0.4$

$P(H/B) = P(T/B) = 0.5$

$P(H/A)P(H/B) = 0.6 \times 0.5 = 0.3$ (both head) A gets 200

$P(T/A)P(T/B) = 0.4 \times 0.5 = 0.2$. (both tail) B gets 100

$P(H,T) = P(H/A)P(T/B) + P(H/B)P(T/A) = 0.6(0.5) + 0.5 \times 0.4$

$= 0.3 + 0.2$

$P(H,T \text{ combo in } A,B) = 0.5$.

$E(A) = \sum A P(A) = (200 \times 0.3) + (0 \times 0.5) + (-100 \times 0.2)$

$= 60 + 0 - 20 = 40. = E(A) = \mu.$

$Var(A) = E\{(A - E(A))^2\} = E[A^2] - (E[A])^2$

$= 200^2(0.3) + (100^2 \times 0.2) - 40^2 = \sum A^2 P(A) - (40)^2$

$14000 - 40^2 = 12400$

$\downarrow Var(A)$, $S.D = \sqrt{Var(A)}$

$Var(B) = E[(B - E(B))^2] = E(B^2) - [E(B)]^2$

$E(B) = (-200 \times 0.3) + (0 \times 0.5) + (100 \times 0.2)$

$= -60 + 20 = -40$

$E(B^2) = 40000 \times 0.3 + 100^2 \times 0.2 = 14000$

$E(B)^2 = 40^2$, $E(B^2) - [E(B)]^2 = 14000 - 40^2 = 12400.$

Q) Decide b/w England, India to host ~~next~~ matches. 60 matches. No. of people attending matches, prob.

Ans:

| X England | India | Probability |
|---|---|---|
| 10000 | 30000 | 0.2 |
| 20000 | 40000 | 0.5 |
| 30000 | 60000 | 0.3 |

Each ticket ⇒ £30 in England, Rs.600. Find Expected value, SD, compare.

Ans

$E(X) = \sum X P(X) = (10000)(0.2) + 20000(0.5) + (30000)(0.3)$

$= 2000 + 10000 + 9000 = 11000 + 10000 = 21000$

England:

$E(X^2) = \sum X^2 P(X) = (10000)^2(0.2) + (20000)^2(0.5) + (30000)^2(0.3)$

$= 490000000$

$Var(X) = E(X^2) - [E(X)]^2 = 490000000 - (21000)^2$

$= 49000000$

Similarly India

Q) Investor buys stock of 2 companies spending 10000 in each. Stock of each company goes up by 50% with a prob. of 0.6 or goes down by 40% with prob. of 0.4? Let random variable X present value after one month. 1) Find prob of 2) Find Expected value. 3) Find S.D

Ans: $P(X) = \begin{cases} (10000+5000)+(10000+5000) = 30000, & P(X) = (0.6 \times 0.\_) \\ (0.4 \times 0.4), & X = (10000-4000)+(10000-40000) \\ & = 6000+6000 = 12000 \\ (0.6 \times 0.4), & X = (10000-4000)+(10000+5000) \\ & = 21000 \end{cases}$

value the investor has after a month

2) $E(X) = \Sigma X P(X) = 22800$

3) S.D = $\sqrt{var}$ = 6235.4

## CONFUSION MATRIX:

| Pred \ Actual | Positive | Negative |
|---|---|---|
| + | | FP |
| − | FN | |

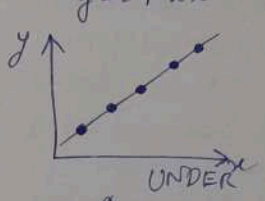
TP.
TN
FP − P→P, A→N
FN − P→N, A→P

## UNDER, OVER FITTING

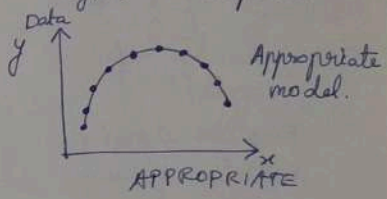
**Under Fitting:**
→ Not able to learn from the data very well.
→ Low variance, high bias.

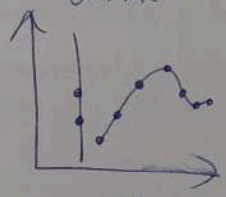
**Over Fitting:**
→ High Variance, low bias.

$y = b + wx$




$(y, x)$ → data points




**OVERFITTING:** (More than the requirement)
→ Training error ↓, Testing error ↑.
→ System performs well during training, not during testing.

**UNDERFITTING:** (Less than requirement)
→ High training error, low testing error.

**OVERCOMING:** (Appropriate capacity) Data exactly on curve
→ Capacity has to be ~~increased~~ appropriate, it needs to solve the present task.

**OVERCOMING:**
→ Reduce the number of features (feature selection)
→ Regularisation (Keep all features but reduce magnitude of weights).
→ Early stopping (Stop in those iterations where the accuray is nearly 100%, and stop with appropriate ↑ in accuracy)

## REGULARISATION : ($L_1$, $L_2$)

L1 : LASSO : Minimise the absolute value of weights] (Some values of weights made 0.

L2 : RIDGE : (Minimise the squared magnitude of weights) Ignore none. Sync all values.

## HYPERPARAMETERS:

→ High level parameters that describe structural information of a model.

Eg: Learning rate ($\alpha$)
Weights
Regularisation ($\lambda$)
Activation Functions

→ These control ML model's behavior.
→ Maximum model capacity resulting in overfitting.

## VALIDATION:

→ ~~Testing the models with random data~~
→ Testing the model with datasets that are already not used while training.

## TESTING:

→ Part of the dataset that were already used while training the model, are used but in random manner.

## CROSS VALIDATION:

→ Split the data into 5 equal parts.
→ Allows to tune the hyperparameters, with the training set.
→ Same as testing.

## MACHINE LEARNING PROCESS:

→ Data collection ~~(Small amount of data)~~
→ Data Preprocessing (To enhance the usable quality of data).
(Data augmentation)
→ Split the data (Training, Cross validation)
→ Train the model
→ Evaluate the mode (Unseen data)
→ Deploying the model (Implement as an app or interface).

## ERRORS IN MACHINE LEARNING:

- Reducible
  - Bias error ⟷ (inversely proportional) Variance Error
    - Bias error: (Model can't capture important features of the data)
    - Variance Error: (When model performs only when already trained data is used
- Irreducible

## SUPERVISED LEARNING ALGORITHMS

→ Labelled data are used.

1) CLASSIFICATION: (SVM, Decision Trees, Kneared Neighbors, Random forest)

→ To classify the data into different groups
→ Just discriminating the features.

2) REGRESSION: (Linear reg, logistic, polynomial)

→ Based on past data, predicting the future data.
→ Determines the strength of and character of relationship between one dependent variable and other independent variables.

## UNSUPERVISED LEARNING ALGORITHM:

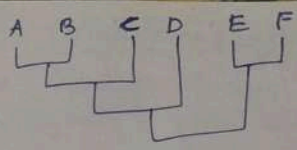
Raw data
↓
Clustering
T1 T2

1) HIERARCHIAL CLUSTERING: (HCA ← analysis)

→ Agglomerative or divisive.
→ Bottleneck approach
→ Different clusters combined together based on similarities.

Ward's linkage:

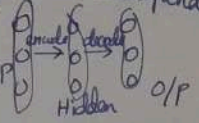
→ Distance b/w 2 clusters defined by increased in sum of squared after the clusters are merged.




Dendrogram

## DIMENSIONALITY REDUCTION:

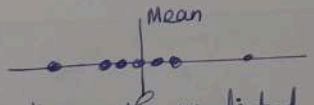
→ Principle component analysis (Based on Eigen values, Eigen vectors)
→ Similar value decomposition (Independent component analysis)
→ Auto encoders.




With compressed data, new representation of dataset, reducing feature set

## 2) MAXIMUM LIKELIHOOD ESTIMATION:

→ To find the optimal way to fit a distribution to the data.

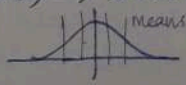


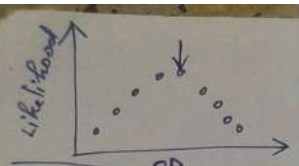

Eg: Here the predicted values are closer to mean, so normal distribution.

Prob. v.s. likelihood:

P(data / distribution) → probability.
(mean value)

P(distribution / data) → likelihood (fitting a dist by finding mean for given data set)
(mean)






Bayesian statistics:
Frequentist statistics
Eg: Taking a fair coin,

BAYESIAN STATISTICS:

Difference → 0.5 × (No. of tosses) − No. of Heads.

0.5 × (No. of tosses) → prob. of getting head in each toss.

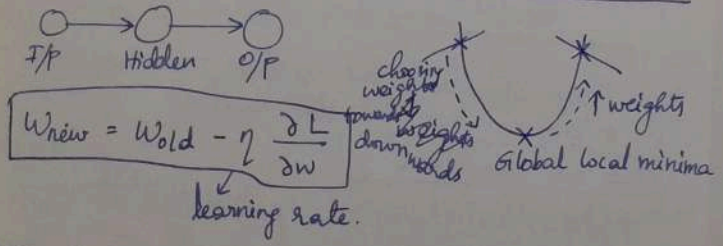
No. of Heads → Actual no. of heads for all the tosses.

Dependence of the result → No. of times exp. is repeated. (∵ for each value of no. of trials, prob. changes).

p-Value:

→ Sample size fixed
→ Stopping intension – different.
→ CI (Confidence).

STOCHASTIC GRADIENT DESCENT

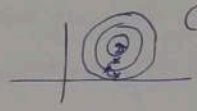



$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial w}$$

learning rate.

1) Gradient descent → When n data points considered (total population)
2) Stochastic gradient descent → When only 1 point is considered.

3) Minibatch stochastic gradient descent → when only a sample (k no. of points) are taken for consideration, then it is Minibatch stochastic G.D.



(It takes time to jump to the centre).

LINEAR REGRESSION:

$y = mx + c$ (one dep, ind var.)

$\hat{y} = b_0 + b_1 x$

y, x values → given as table

$$b_1 = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\Sigma(x-\bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

Standard error = $\sqrt{\frac{\Sigma(\hat{y}-y)^2}{n-2}}$

(Difference b/w actual, predicted) n → no. of samples

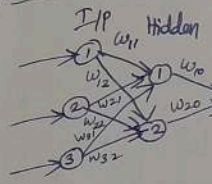
Multiple linear regression → $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots$

Polynomial linear regression → $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \dots$

UNIT 2

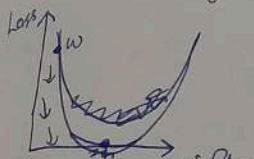
## OPTIMIZERS :

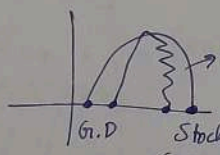
### 1) GRADIENT DESCENT OPTIMIZER:




$$\hat{y} = \frac{1}{2}\sum_{i=1}^{3}(y_i - \hat{y}_i)^2$$

Epoch → No. of times of visiting the algorithm.
Iteration → No. of times of processing it (no. of dataset passed once).




$$w_{new} = w_{old} - \eta \times \frac{\partial L}{\partial w_{old}} \rightarrow G.D$$

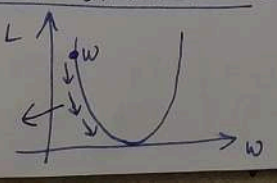
$t \Rightarrow$ batch size

$$w_t = w_{t-1} - \eta \times V_{dw}$$ → Exponential weighted avg.

↳ fixed

### 2) ADAPTIVE GRADIENT DESCENT OPTIMISER: (ADA)

$\eta \rightarrow$ ~~fixed~~ dynamic learning rate.




$$\eta'_t = \frac{\eta}{\sqrt{\alpha_t + E}}$$ → initial learning rate

$$\alpha_t = \sum_{i=1}^{t}\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

### 3) RMS Prop (Root Mean Squared Propagation) (Ada delta)

$$\eta' = \frac{\eta}{\sqrt{S_{dw} + \varepsilon}}$$ ↓ weighted avg.

$$S_{dw} = \beta \times S_{dw} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$ ↓ weighted avg.

(Both weight, Learning rate change dynamically)

$$w_t = w_{t-1} - \eta'_t \frac{\partial L}{\partial w_{old}} \rightarrow V_{dw}$$

$$b_t = b_{t-1} - \eta'_t\left(\frac{\partial L}{\partial b}\right) \rightarrow V_{db}$$

### 4) ADAM OPTIMISER: (Better than others)

$$w_0 = w_{t-1} - \eta \times V_{dw}$$ → Exponential weighted avg.

(Dynamic LR) $\eta'_t \leftarrow \eta$ / $\sqrt{S_{dw} + \varepsilon}$ → Small +ve no.

→ For both output function and learning rate, if the weighted avg. is used, so the adam optimiser is the best.

# LINEAR DISCRIMINANT ANALYSIS (Dimensionality reduction)

→ Dimensionality reduction technique, for supervised learning.
→ Preprocessing step for pattern classification and ML applications.
→ Finds axes that ~~minimise~~ maximise separation between multiple classes.

GOAL: To project a feature space (N-dimensional data) onto smaller subspace K ($K \leq n-1$) while maintaining class discrimination.

Eg: 2D data set.

$C_1 \Rightarrow X_1 = (x_1, x_2) = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$
$C_2 \Rightarrow X_2 = (x_1, x_2) = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$

Step 1: Compute within class scatter matrix.

total covariance matrix ← $S_W = S_1 + S_2$

$S_1 \rightarrow$ Covariance matrix for $C_1$
$S_2 \rightarrow$ Covariance matrix for $C_2$

Step 2: Compute covariance matrix of each class

$$S_i = \left[\sum_{x \in C_i} (x-\mu_i)(x-\mu_i)^T\right] / n$$

($\mu_i$ = mean of class $C_i$)

$$\mu_1 = \left\{\frac{4+2+2+3+4}{5}, \frac{1+4+3+6+4}{5}\right\} = [3.00, 3.6]$$

$$\mu_2 = \left\{\frac{9+6+9+8+10}{5}, \frac{10+8+5+7+8}{5}\right\} = [8.4, 7.6]$$

$$(x-\mu_1) = \begin{bmatrix} 4-3 & 2-3 & 2-3 & 3-3 & 4-3 \\ 1-3.6 & 4-3.6 & 3-3.6 & 6-3.6 & 4-3.6 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & -1 & 0 & 1 \\ -2.6 & 0.4 & -0.6 & 2.4 & 0.4 \end{bmatrix}$$

$\left[\sum (x-\mu_1)(x-\mu_1)^T\right] / n = S_1$

$$\begin{bmatrix} 1 \\ -2.6 \end{bmatrix}\begin{bmatrix} 1 & -2.6 \end{bmatrix} = \begin{bmatrix} 1 & -2.6 \\ -2.6 & 6.76 \end{bmatrix} - (1)$$

$$\begin{bmatrix} -1 \\ 0.4 \end{bmatrix}\begin{bmatrix} -1 & 0.4 \end{bmatrix} = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 0.16 \end{bmatrix} - (2)$$

$$\begin{bmatrix} -1 \\ -0.6 \end{bmatrix}\begin{bmatrix} -1 & -0.6 \end{bmatrix} = \begin{bmatrix} 1 & +0.6 \\ +0.6 & 0.36 \end{bmatrix} - (3)$$

$$\begin{bmatrix} 0 \\ 2.4 \end{bmatrix}\begin{bmatrix} 0 & 2.4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 5.76 \end{bmatrix} - (4)$$

$$\begin{bmatrix} 1 \\ 0.4 \end{bmatrix}\begin{bmatrix} 1 & 0.4 \end{bmatrix} = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 0.16 \end{bmatrix} - (5)$$

$$\frac{(1)+(2)+(3)+(4)+(5)}{5} \Rightarrow S_1 = \begin{bmatrix} \frac{4}{5} & -\frac{2}{5} \\ -\frac{2}{5} & 2.6 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} 0.8 & -0.4 \\ -0.4 & 2.6 \end{bmatrix}$$

(similarly) $S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix}$

within class scatter matrix

$$S_W = S_1 + S_2 = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

Step 3: Between class scatter matrix:

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$= \begin{bmatrix} -5.4 \\ -4 \end{bmatrix} \begin{bmatrix} -5.4 & -4 \end{bmatrix} = \begin{bmatrix} 29.16 & 21.6 \\ 21.6 & 16.00 \end{bmatrix}$$

Step 4: Find best LDA projection vector.

$$\boxed{> V = S_W^{-1} S_B V}$$

projection vector

Find Eigen vector having largest Eigen value.

$$\begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = S_W^{-1} (\mu_1 - \mu_2)$$

$$\begin{bmatrix} V_1, V_2 \end{bmatrix} = \begin{bmatrix} 0.1921 & -0.032 \\ -0.032 & 0.38 \end{bmatrix} \begin{bmatrix} -5.4 \\ -4 \end{bmatrix} = \begin{bmatrix} -0.91 & -0.39 \end{bmatrix}$$

Step 5: Dimension reduction

$y = W^T x \rightarrow$ I/p data samples.

↓ projection vector

$$= \begin{bmatrix} -0.91 \\ -0.39 \end{bmatrix} [$$

PCA (PRINCIPLE COMPONENT ANALYSIS):

→ High dimension data is complex to process due to inconsistencies in features that increase computation time, make data processing complex.

→ This is a dimensionality reduction technique that identifies correlations, patterns in data set, so that it can be transformed into dataset of significantly lower dimension without loss of important info.

STEP 1: Scaling of data

$$Z = \frac{\text{variable value} - \text{Mean}}{S.D}$$

STEP 2: Find covariance matrix (expresses correlation b/w different variables in dataset)

"Identify heavily dependent variables, as they contain biases", "Redundant info reduces overall performance of model".

NXN matrix
N dimensions

COVARIANCE VALUE:

$$\begin{bmatrix} cov(X,X) & cov(X,Y) \\ cov(Y,X) & cov(Y,Y) \end{bmatrix}$$

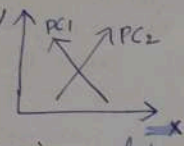
STEP 3: Find Eigen values, Eigen vectors.

PCA → new set of data derived from original data.

→ Compress, possess most of the useful info that was scattered among initial variables.

→ Eigen vectors don't change direction (linear transformation).

→ Eigen values just denote scalars of the respective Eigen vectors.




(To identify mere variance).

Eg: Movie review dataset.

Q) Given following data, use PCA to reduce dimension from 2 to 1.

| Feature | E-1 | E-2 | E-3 | E-4 |
|---|---|---|---|---|
| x | 4 | 8 | 13 | 7 |
| y | 11 | 4 | 5 | 14 |

No. of features $n = 2$

No. of samples $N = 4$

Mean: $\bar{x} = 8$ , $\bar{y} = 8.5$

Covariance matrix:

$$cov(x,y) = \left(\frac{1}{N-1}\right) \sum_{k=1}^{N} (x_{ik} - \bar{x}_i)^2$$

$$= \left(\frac{1}{4-1}\right)\left[(4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right]$$

$$= \frac{1}{3}[16 + 0 + 25 + 1] = 14$$

$$cov(x,y) = \frac{1}{4-1}\left[(4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)\right]$$

$$= -11$$

$$cov(x,y) = \frac{1}{N-1} \sum_i (x_i - \bar{x})(y_i - \bar{y}) = cov(y,x)$$

$$cov(y,y) = \frac{1}{N-1} \sum_i (y_i - \bar{y})^2 = 23$$

$$S = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix} = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

Eigen value:

$$\det[S - \lambda I] = 0.$$

$$\left| \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix} \right| = 0 \Rightarrow \lambda^2 - 37\lambda + 201 = 0$$

$$\lambda = 30.3849, 6.615$$

$\lambda_1 > \lambda_2$

Hence, $\lambda_1 = 30.3849 \rightarrow PC_1$

Eigen vector of $\lambda_1$:

$$(S - \lambda_1 I) U_1 = 0$$

$$\begin{bmatrix} 14-\lambda_1 & -11 \\ -11 & 23-\lambda_1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(14-\lambda_1)U_1 - 11U_2 = 0$$

$$-11U_1 + (23-\lambda_1)U_2 = 0$$

$\frac{U_1}{11} = \frac{U_2}{14-\lambda_1} = t$

When $t = 1$,

$U_1 = 11$ ; $U_2 = 14 - \lambda_1$

Eigen vector $U_1$ of $\lambda_1$ $= \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$

$= \begin{bmatrix} 11 \\ 14-30.3849 \end{bmatrix} = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix}$

Normalised Eigen Vector:

$e_1 = \begin{bmatrix} 11 \\ -16.3849 \end{bmatrix} \Rightarrow \begin{bmatrix} 11/\sqrt{11^2+16.3849^2} \\ -16.3849/\sqrt{11^2+16.3849^2} \end{bmatrix}$ ($U_1$ over $\sqrt{U_1^2+U_2^2}$; $U_2$ over $\sqrt{U_1^2+U_2^2}$)

$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$

For $\lambda_2$: $e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$

NEW dataset:

| $E_1$ | $E_2$ | $E_3$ | $E_4$ |
|---|---|---|---|
| $P_{11}$ | $P_{12}$ | $P_{13}$ | $P_{14}$ |

$P_{11} = e_1^T \begin{bmatrix} x_i - \bar{x}_i \\ y_i - \bar{y}_i \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} 4-8 \\ 11-8.5 \end{bmatrix}$

$= -4.3052$

$P_{12} = 3.7361$

$P_{13} = 5.6928$

$P_{14} = -5.1238$

## FISHER'S LINEAR DISCRIMINANT FUNCTION: (FLD)

→ First introduced to discriminate 2 plant species ~~iris~~ setosa, iris versicolor

d → dimensional data points $x_1 \ldots x_2 \ldots x_n$

$C_1$, $C_2$

↓ class 1, ↓ class 2

$N_1$, $N_2$ → No. of samples in each class.

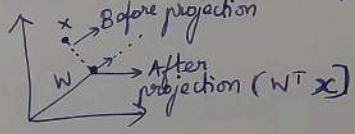
w → unit vector to project data points.

$N = N_1 + N_2$

If $x[n]$ are samples of feature space, then $\boxed{w^T x[n]}$ → Data points after projection

$m_i$ → mean of classes before projection

$M_i$ → " " " after " : $M_i = w^T m_i$



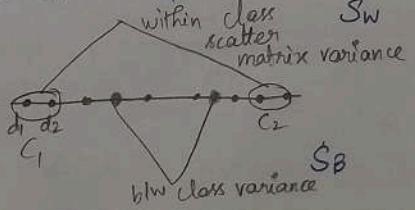

Scatter matrix (m x m)

↳ +ve semi definite matrix

Sample variance × No. of samples.

→ Within class scatter matrix.

→ Between " " ".



$$\arg\max j(w) = \frac{(M_1 - M_2)^2}{S_1^2 + S_2^2}$$ → Between class scatter; → Within class scatter

**Numerator:**

$$(M_1 - M_2)^2 = (w^T m_1 - w^T m_2)(w^T m_1 - w^T m_2)^T$$
$$= w^T (m_1 - m_2)(m_1 - m_2)^T w$$
$$= w^T S_b w$$

$S_b$ → b/w class scatter.

**Denominator:**

$S_1^2 + S_2^2$

Before projection $S_i = \sum_{x[n]} (x[n] - m_i)(m(n) - m_i)^T$

After proj $S_i^2 = \sum (w^T x[n] - M_i)(w^T x[n] - M_i)^T$

$$\boxed{M_i = w^T \times m_i}$$

$$S_i^2 = w^T \left( \sum (x[n] - m_i)(x[n] - m_i)^T \right) w$$
$$= w^T S_i w$$

$$S_1^2 + S_2^2 = w^T (S_1 + S_2) w = w^T S_w w$$

$S_w$ → within class scatter

$$J(w) = \frac{(M_1 - M_2)^2}{S_1^2 + S_2^2} \Rightarrow \frac{w^T S_b w}{w^T S_w w}$$ → differentiate, equate to 0,

V → Eigen value
W → Eigen vector

$S_b w = v \times S_w w$

$S_w$ → full rank matrix $\Rightarrow S_w^{-1} S_b w = v w$

**LDA for multiple classes:**

$S_k = \sum (x[n] - m_i)(x[n] - m_i)^T$

$S_w$ → within class transfer

$C$ → No. of distinct classes

$S_w = \sum_{k=1}^{c} S_k \Rightarrow m_i = \frac{1}{N_i} \sum x[n]$

$S_b = \sum_{i=1}^{C} N_i (m_i - m)(m_i - m)^T$

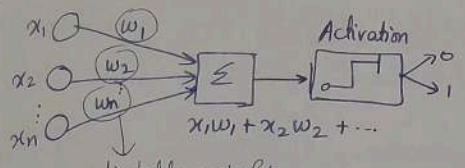
$m = \frac{1}{N} \sum_{i=1}^{n} x_i$ ; $W = eig(S_w^{-1} S_b)$

$S_b$ ∠ = C - 1

LDA → reducing dimension, computing time
↳ Multiple class classification

| LDA | FLD |
|---|---|
| | |

## PERCEPTRON LEARNING ALGORITHM: (Supervised algorithm)




$$y = \sum x_n w_n + b$$

$$w_{new} = w_{old} + \alpha \Delta w$$

($\alpha$: learning rate; $\Delta w$: error)

(actual - predicted) → error

$$b_{new} = b_{old} + \alpha t$$

($t$: target)

updating weights and bias:

$$w_{new} = w_{old} + \alpha t x_i$$

Q) Implementing AND (bipolar I/P, targets) using perceptron.

| $x_1$ | $x_2$ | $t$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | -1 |

Ans Initialisation ⇒ $w_1 = w_2 = 0$, $b = 0$

$\alpha = 1$ (default)

$$y = x_1w_1 + x_2w_2 + b$$

(i) $x_1 = x_2 = 1$ ; $y_{in} = 0+0+0 = 0$

sigmoid $f(y_{in}) = \begin{cases} 0 & y_{in} \le 0 \\ 1 & y > 0 \end{cases}$

updation of weights:

$w_{new} = w_{old} + \alpha \Delta w = w_{old} + \alpha t x_i$

$w_{new_1} = 0 + (1)(1)(1) = 1$

$w_{new_2} = 0 + (1)(1)(1) = 1$

$b_{new} = b_{old} + \alpha t$

$= 0 + (1)(1) = 1$

(ii) $x_1 = 1$ , $x_2 = -1$

$y = 1\times1 + (-1)\times1 + 1 = 1 - 1 + 1 = 1$, $f(y_{in}) = 1$

$w_1 = 1 + (1)(-1)(1) = 0$

$w_2 = 1 + (1)(-1)(-1) = 2$

$b = 1 + (1)(-1) = 0$

⋮

$$\text{Sigmoid} = \frac{1}{1+e^{-x}}$$

## PROBABILISTIC GENERATIVE MODELS:

| Generative approach | Discriminative approach |
|---|---|
| → Eg: To differentiate or find a language from all the sets, learn everything then discriminate. | → To find the language just know the difference, and how it sounds, without learning the lang. |
| → Focus on how data was generated | → Draw boundaries, then separate one class from another |
| → Outlier drawback | → Outlier advantage. |

$$P(Y/X) = \frac{P(Y) \times P(X/Y)}{P(X)}$$

Posterior ← Prior ← Likelihood (generative model)

$P(X)$ ← evidence (conditional gen. model)

$P(X/Y) = \frac{P(Y/X)\,P(X)}{P(Y)}$ (discriminative model)

$P(Y)$ → prior over labels

→ Discriminative models separate classes instead of modeling conditional prob.
→ separate one class from another.

Outliers → Something irrelevant in the I/p set

⁂ Discriminative work better with outliers
⁂ Generative ~~do~~ have disadvantage with outliers.

| Generative | Discriminative |
|---|---|
| → More biased (have stronger assumptions). | → Less biased, and have lesser assumptions. |
| → Can tolerate with missing data, and still produce good results | → Can't produce efficient results with missing data. |
| → Generative less accurate | → Assign labels to data |
| → For fewer data sets. | → More accurate |
| → ~~Assign labels to data~~, detect outliers | → Classifying data with some conditions, ~~eg~~ can be used for classifying, not for regression. |
| → Learns $p(x)$ | → Learns $p(y/x)$ |

PROBABILISTIC GENERATIVE MODELS:

CONDITIONAL GENERATIVE MODEL → Assign labels (unsupervised learning while rejecting outliers)

Eg: Rain prediction (if less data)
Stock market prediction (if ~~if~~ (for regression).

## Maximum Likelihood Estimation (MLE)

→ Probability distribution function
→ Likelihood function.
→ MLE
→ Parameter estimation of Bernoulli density functi

$f(x/\theta)$ (parameter)
↓ data

$x \rightarrow$ Data (Source)
$y \rightarrow$ Label (Target)

→ To estimate parameters of prob. distribution, to test whether given dataset follows some par distribution.
→ Method to estimate parameters of prob. dist, given the observations.
→ MLE attempts to find parameter values that maximise likelihood function.
→ Resulting estimate is maximum likelihood functi
→ Random sample $X = \{x_1, x_2, \ldots x_n\}$ taken from prob. distribution having prob. density fn $p(x/\theta)$

$x \rightarrow$ Value of random variable
$\theta \rightarrow$ Set of parameters that appear in functi

→ Likelihood of sample $x$ is a fn. of parameter $\theta$, defined as the chance that the parameter $\theta$ would generate the observed data.

$$l(\theta) = P(x/\theta) = P(x_1/\theta)P(x_2/\theta)\ldots P(x_n/\theta)$$

$\log(L(\theta)) = \log[P(x_1/\theta)\,P(x_2/\theta)\cdots P(x_n/\theta)]$

$L(\theta) = \log(P(x_1/\theta)) + \log P(x_2/\theta) + \cdots + \log P(x_n/\theta)$

## BERNOULLI DENSITY FUNCTION:

→ Success $X=1$ $(p)$

→ Failure $X=0$ $(1-p)$

$f(x/p) = p^x (1-p)^{1-x}$ $\quad x = 1, 0.$

Estimation of parameter $p$ of Bernoulli density function using MLE

$f(x/p) = p^x (1-p)^{1-x}$

$L(\theta) = \log(f(x_1/\theta) + \log f(x_2/\theta) + \cdots \log f(x_n/\theta)$

$p = \log[p^{x_1}(1-p)^{1-x_1}] + \log[p^{x_2}(1-p)^{1-x_2}]$

$+ \cdots \log[p^{x_n}(1-p)^{1-x_n}]$

$= x_1 \log p + (1-x_1)\log(1-p) + \cdots + x_n \log p + (1-x_n)\log(1-p)$

$= (x_1+x_2+\cdots+x_n)\log p + [(1-x_1)+(1-x_2)+\cdots(1-x_n)]\log(1-p)$

let $x_1+x_2+\cdots+x_n = Y$

$L(p) = Y\log p + (n-Y)\log(1-p)$

To find $p$ for max value of $L(p)$, diff $\Rightarrow$

$\frac{dL}{dp} = 0, \quad \frac{d(Y\log p + (n-Y)\log(1-p))}{dp} = 0$

$Y\left(\frac{1}{p}\right) + (n-Y)\left(\frac{1}{1-p}\right)(-1) = 0$

$\boxed{\frac{Y}{p} = \frac{n-Y}{1-p}}$ $\quad \boxed{\frac{1-p}{p} = \frac{n-Y}{Y}}$

$\frac{1}{p} - 1 = \frac{n}{Y} - 1$

$\frac{1}{p} = \frac{n}{Y}$ ; $\boxed{p = \frac{Y}{n}}$

$p = \frac{1}{n}(x_1+x_2+\cdots+x_n)$

$(\theta)$

$f(x/\theta) \rightarrow$ Prob. dist fn.

$\boxed{\theta = \frac{1}{n}(x_1+x_2+\cdots+x_n)}$

## PARAMETER ESTIMATION USING MLE METHOD:

Poisson dist : $f(x,\lambda) = \frac{e^{-\lambda}\lambda^x}{x!}$

Geometric dist : $f(x/p) = p(1-p)^{x-1}$

POISSON: $f(x,\lambda) = \frac{e^{-\lambda}\lambda^x}{x!}$

$L(\lambda) = \log f(x_1/\lambda) + \log f(x_2/\lambda) + \cdots + \log f(x_n/\lambda)$

$= \log\left[\frac{e^{-\lambda}\lambda^{x_1}}{x_1!}\right] + \log\left[\frac{e^{-\lambda}\lambda^{x_2}}{x_2!}\right] + \cdots + \log\left[\frac{e^{-\lambda}\lambda^{x_n}}{x_n!}\right]$

$= \log e^{-\lambda} + \log\lambda^{x_1} - \log(x_1!) + \cdots + \log e^{-\lambda} + \log\lambda^{x_n} - \log x_n!$

$= -n\lambda + \sum_{i=1}^{n} x_i \log\lambda - \sum_{i=1}^{n} \log x_i!$

$\frac{dL(\lambda)}{d\lambda} = 0 \Rightarrow -n + \sum_{i=1}^{n} x_i\left(\frac{1}{\lambda}\right) - 0 = 0$

$\frac{\sum_{i=1}^{n} x_i}{\lambda} = n$ ; $\boxed{\lambda = \sum_{i=1}^{n} x_i / n}$

## GEOMETRIC DISTRIBUTION:

$$f(x,p) = p(1-p)^{x-1}$$

$$L(p) = \log f(x_1/p) + \log f(x_2/p) + \dots + \log f(x_n/p)$$

$$= \log\left[p(1-p)^{x_1-1}\right] + \dots + \log\left[p(1-p)^{x_n-1}\right]$$

$$L(p) = n\log p + \log(1-p)\left[\sum_{i=1}^{n} x_i - n\right]$$

$$\frac{dL(p)}{dp} = 0 \Rightarrow n\left(\frac{1}{p}\right) + \left[\sum_{i=1}^{n} x_i - n\right]\left(\frac{1}{1-p}\right)(-1) = 0$$

$$\frac{n}{p} = \frac{\sum_{i=1}^{n} x_i - n}{1-p} \quad ; \quad \frac{1-p}{p} = \frac{\sum_{i=1}^{n} x_i - n}{n}$$

$$\frac{1}{p} - 1 = \frac{\sum_{i=1}^{n} x_i}{n} - 1 \Rightarrow p = \frac{n}{\sum_{i=1}^{n} x_i}$$

## LOGISTIC REGRESSION:

**Drawbacks of linear regression:**

→ The best fit line gets shifted due to outliers.

→ The decision gets changed due to outliers.

→ To handle the outliers, and in cases where the values are unbounded (predicted values), logistic regression used. (Why LOGISTIC?)

(Binary classification)

~~Mathematics Intuition:~~

Geometric Intuition:

(i)

(ii) Mathematics Intuition

$y = w^T x + b$ (weights)

$y = w^T x$

$y = +ve$ } different point $w^T x > 0$ (Above plane)

$y = -ve$ } $w^T x < 0$ (Below plane)

dist from plane

+ → point, +ve, -ve → plane

X → +ve, 0 → -

(0,0)

CASE 1: $y = +1$ $w^T x > 0$

$y \times w^T x > 0$

Case 2: $y = -1$ $w^T x < 0$

$-1 \times w^T x \geq 0$

Case 3: $y = +1$ $w^T x < 0$

$y \times w^T x \leq 0$

case 4: $y = -1$ $w^T x > 0$

$y \times w^T x < 0$

result, how the classification work and how points are divided on the plane

$$\max \sum_{i=1}^{n} y_i w_i^T x_i$$

### SIGMOID FUNCTION:

$$S = \frac{1}{1+e^{-x}}$$

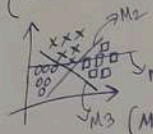
Brings any value (outlier) in the range (0-1).

+1, -1

Dist. b/w data points, line should be maximum.

## LOGISTIC REGRESSION FOR MULTICLASS CLASSIFICATION:

(One vs. rest) or (one vs. all) model.




The Hyperplanes should separate one type of class from the rest.

$(M_1, M_2, M_3) \rightarrow$ models

## 3 APPROACHES FOR CLASSIFICATION

### 1) DISCRIMINANT FUNCTIONS:

Prob. Generative Model:

→ Fit class conditional densities, class priors separately.

$$P(Y|x) = \frac{\text{Likelihood} \times \text{prior}}{\text{Evidence}} = \frac{P(X/Y)\,P(Y)}{P(X)}$$

posterior probability

→ Apply Bayes' theorem to find posterior class probabilities.

→ Posterior prob. of a class written as:

- Logistic sigmoid acting on a linear function of $x$ (2 classes). (Binary)
- Softmax transformation of a linear function of $x$ (Multiclass). (Multiclass)

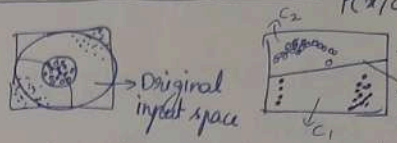
→ Parameters of densities and class priors found using Maximum Likelihood.

Prob. Discriminative Model:

→ Use functional form of generalized linear model explicitely.

→ Find the parameters directly using Maximum likelihood.

### 2) FIXED BASIS FUNCTIONS:




→ Transform inputs using a vector of basis functions

→ Resulting decision boundaries are linear in feature space.

→ To convert non-linearly separable function to linearly separable function.

Iterative reweighted least squares:

→ In linear regression models, ML solution on the assumption of Gaussian noise leads to close-form solution, as a result of quadratic dependence of log likelihood on parameter $w$.

→ Not a closed form solution, but error function is concave, has unique minimum, efficient iterative technique can be used, Newton-Raphson update to minimise a function $E(w)$

$$w^{new} = w^{old} - H^{-1}\nabla E(w)$$

Hessian matrix, ← second derivatives of $E(w)$.

Sum of squares Error function: $E(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2$

$$\nabla E(w) = \phi^T\phi w - \phi^T t$$

$$H = \nabla\nabla E(w) = \phi^T\phi$$

Newton Raphson update: $w^{new} = (\phi^T\phi)^{-1}\phi^T t$

Cross entropy error function: $E(w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$

$$\nabla E(w) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = \phi^T(y-t)$$

$$H = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T = \phi^T R \phi$$

Newton Raphson update: $w^{(new)} = (\phi^T R \phi)^{-1}\phi^T R z$

$z = \phi w^{(old)} - R^{-1}(y-t)$

$R_{nn} = y_n(1-y_n)$

### 3) MULTICLASS LOGISTIC REGRESSION:

$$p(C_k/\phi) = y_k(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}} \; ; \; a_k = w_k^T\phi$$

Likelihood function using 1-of-K coding scheme.

$$p(T/w_1,\dots w_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k/\phi_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

Cross entropy error function for multiclass classification

$$E(w_1,\dots w_k) = -\ln p(T/w_1,\dots w_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

$$\nabla_{w_j} E(w_1,\dots w_k) = \sum_{n=1}^{N}(y_{nj}-t_{nj})\phi_n$$

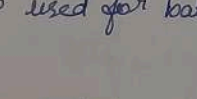
Same form, product of error times the basis function.

Hessian matrix: $(\phi^T\phi)^{-1}$

$$\nabla_{w_k}\nabla_{w_j} E(w_1,\dots w_k) = -\sum_{n=1}^{N} y_{nk}(I_{kj}-y_{nj})\phi_n\phi_n^T$$

IRLS algorithm is also used for batch processing.

### 4) PROBIT REGRESSION:

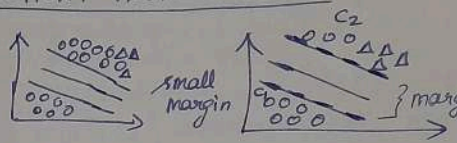



yes or no

→ for broad range of class conditional distributions, resulting posterior class prob are given by logistic or softmax transformation acting on a linear function of the feature variable

→ Not the case for all choices of class-conditional density
→ Not worth exploring other types of discriminative probabilistic model.

### SUPPORT VECTOR MACHINE:




(difference b/w logistic reg, SVM)

$\geq 0.5$ label 1 $\{0,1\}$
$< 0.5$ label 0

→ Drawback of logistic Reg:
→ If o/p > 1 (beyond range) it won't classify

Why SVM?:
→ Even if o/p > 1 it will classify. (outliers will also be classified).
→ Hyper plane, maximum margin, support vectors, linear separable.

HINGE LOSS:

Helps maximise margin

$$C(x,y,f(x)) = \begin{cases} 0 & \text{if } y \times f(x) \geq 1 \\ 1 - y \times f(x) & \text{else} \end{cases}$$

→ The cost is 0 if both the predicted, actual value are same
→ If not same, calculate loss function.

$$\min_w \lambda\|W\|^2 + \sum_{i=1}^{n}(1-Y_i(x_i;w))$$ → Loss fn for SVM

$\Delta E(w)$

$\frac{\partial}{\partial w_k} \gamma \|w\|^2 = 2\gamma w_k$

$$\frac{\partial}{\partial w_k}(1 - y_i(x_i, w))_+ = \begin{cases} 0 & \text{if } y_i(x_i, w) \geq 1 \\ -y_i x_{ik} & \text{else} \end{cases}$$

→ gradient fn.

$w = w - \alpha\ (2\gamma w)$ → Gradient update

When there is misclassification

$w = w + \alpha\ (y_i x_i - 2\gamma w)$ → gradient update when there is misclass.

~~SUPPORT VECTOR REGRESSION~~:

SVM-Kernel → Combined with SVM to make the segregation of non-separable data.

→ Takes low dimensional I/P, transforms into higher dimensional

→ Non-separable to separable problem.

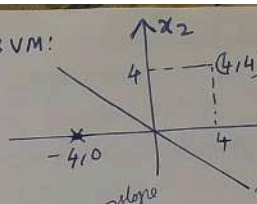
PROS OF SVM:

→ Clear margin of separation
→ High dimensional spaces.
→ No. of dimensions > No. of samples.
→ Memory efficient as it uses a subset of training pts.

CONS OF SVM:

→ Large dataset
→ More noise
→

Q) SVM:




SVM

$y = mx + c$

$y = w^T x + b$

$y = w^T x + 0$ (intercept)

$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4\ \ 0] = 4$ (+ve)

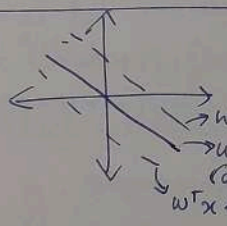
(slope, intercept)

so any point below the plane,

$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4\ \ 4] = -4$ (-ve)

so any point above plane -ve.

Q)




SVM

$w^T x_1 + b = -1$ — (1)

$(-) w^T x_2 + b = +1$ — (2)

$w^T (x_1 - x_2) = -2$

$w^T (x_2 - x_1) = 2$

$\frac{w^T}{\|w\|} (x_2 - x_1) = \frac{2}{\|w\|}$ → Optimisation

$\|w\|$ → norms

$(w^*, b^*)$ max $\left(\frac{2}{\|w\|}\right)$ such that

$$y_i = \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$$

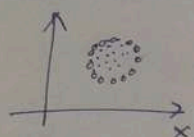
↓ Maximum margin

$\Downarrow$ $y_i \times w^T x_i + b_i \geq 1$

$(w^*, b^*) = \min \frac{\|w\|}{2} + C_i \sum_{i=1}^{n} \xi_i$ → value of the error

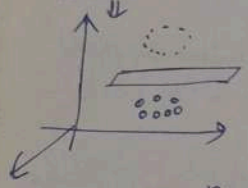
no. of errors (Regulariser)

## SVM KERNEL:




Linearly inseparable

Kernel → lower dim to higher dim



Linearly separable now (2D to 3D)

n dimen → m dim

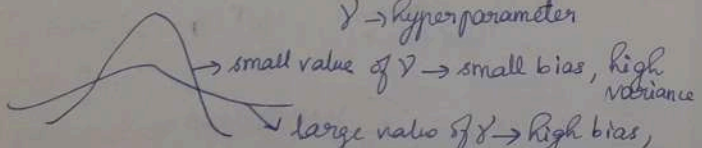
$m > n$

→ Polynomial kernel $k(x, x') = (1 + x * x')^k$

→ RBF Kernel $k(x, x') = e(-\gamma \|x - x'\|^2)$

→ Sigmoid Kernel

$\gamma$ → Hyperparameter



→ small value of $\gamma$ → small bias, high variance

→ large value of $\gamma$ → high bias, low variance

Eg: $x = (x_1, x_2, x_3)$ $y = (y_1, y_2, y_3)$

$f(x) = (x_1 x_1, x_1 x_2, x_1 x_3)$ ; $K(x, y) = \langle f(x), f(y) \rangle$

~~$(x_2 x_1, x_2 x_2, x_2 x_3, x_3 x_1, x_3 x_2, x_3 x_3)$~~ $\langle x, y \rangle$ → dot product.

$K(x, y) = (\langle x, y \rangle)^2$

$x = (1, 2, 3)$ $y = (4, 5, 6)$

$f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$ → multiplying x with x itself.

$f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$ → multiplying y with itself.

$\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$

dot product

$K(x, y) = (4 + 10 + 18)^2 = 32^2 = 1024$

$K(x, x') = (1 + x + x')^2$

$f(x) = (1, 2, 3)$ multiplying

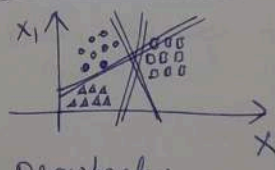
$f(y) = (4, 5, 6)$

## SVM for multiclass classification:

Techniques:

1. One v.s. One (OVO)
2. One v.s. all (OVA)
3. Directed Acyclic Graph (DAG)

### 1) ONE VS. ONE:




only 2 groups considered at a time and those two are separated by the hyperplane.

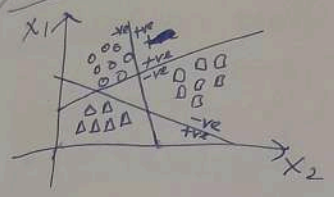
Drawback:

→ More no. of SVMs need to be trained.

→ Too much computation required.

→ Unbalancing (difference in the no. of points) occurs in the SVM.

2) ONE VS. ALL:




One class separated from the rest.

→ No unbalancing of data point arises as no data is left out.

Drawback:

→ Computation problem ∵ n no of classifiers required.

3) DIRECTED ACYCLIC GRAPH:

→ After logical grouping only this should be applied.
→ Requires only few no. of SVMs, compared to OVA.

SUPPORT VECTOR REGRESSION (SVR):

→ Use same principle as SVM.
→ Instead of classification, regression is used.
→ This is the only difference.

OVERLAPPING CLASS DISTRIBUTION:

→ Training data may not be linearly separable, which leads to misclassification of data points.
→ Resulting SVM gives exact separation in input space, although decision boundary will be non-linear.
→ Exact separation leads to poor generalisation, so SVM should be allowed to misclassify some training points.

Modifying SVM:

→ Implicit error function of SVM:

$$\sum_{n=1}^{N} E_\infty(y(x_n)t_n - 1) + \lambda\|w\|^2$$

→ Gives infinite error if point is misclassified.
→ Gives zero error if it was classified correctly.
→ Parameters are optimised to maximise margin.
→ Points are allowed to lie on the wrong side but ensuring that penalty is proportional to distance from boundary.

Slack variables: $\xi_n$

$$\xi_n \geq 0, \quad n = 1, \ldots N$$

→ One slack variable for each training point.
→ Defined by $\xi_n = 0$ for data points that are on or inside correct margin boundary.
→ $\xi_n = |t_n - y(x_n)|$ for other points.
→ Data point that is on decision boundary, $y(x_n) = 0$ has $\xi_n = 1$.
→ Points with $\xi_n \geq 1$ misclassified.

## ↔ Soft Margin Classification:

→ Allows some data points to be misclassified.
→ While slack variables allow overlapping class distributions, framework is sensitive to outliers.

### Optimisation with slack variables:

→ To maximise the margin while softly penalising points that lie on the wrong side of margin boundary.

→ Minimise $C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|w\|^2$

→ Parameter $C > 0$ controls trade-off b/w slack variable penalty and margin.

Subject to constraints:

$t_n y(x_n) \geq 1 - \xi_n, \quad n = 1, \ldots, N$

### Lagrangian with slack:

$$L(w, b, a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n [t_n y(x_n) - 1 + \xi_n] - \sum_{n=1}^{N} \mu_n \xi_n$$

$\{a_n \geq 0\}, \{\mu_n \geq 0\}$ → Lagrange multipliers.

### KKT conditions:

$a_n \geq 0$
$t_n y(x_n) - 1 + \xi_n \geq 0$
$a_n \{t_n y(x_n) - 1 + \xi\} = 0$
$\mu_n \geq 0$
$\xi_n \geq 0$
$\mu_n \xi_n = 0$

## BAYESIAN LOGISTIC REGRESSION:

→ Discriminative probabilistic linear classifier.
$p(C_1|\phi) = \sigma(w^T\phi)$

→ Exact Bayesian inference for Logistic Regression
$P(C_1|\phi) = \int \sigma(w^T\phi)\, p(w)\, dw$ is not easy to control, because:

1) Evaluation of posterior distribution $p(w|t)$
Needs normalisation of prior $p(w) = N(w|m_0, S_0)$ times likelihood (product of sigmoids)
Solution: Use Laplace approx.
2) Evaluation of predictive distribution
Convolution of sigmoid and Gaussian.

## LAPLACE APPROXIMATION:

→ Need mode $w_0$ of posterior distribution $p(w|t)$ (Done by numerical optimisation algorithm).

→ Fit a Gaussian centered at mode

$$q(w) = \frac{1}{Z} f(w) = \frac{|A|^{1/2}}{(2\pi)^{M/2}} e^{\{-\frac{1}{2}(w-w_0)^T A (w-w_0)\}}$$

$$= N(w|w_0, A^{-1})$$

→ Needs second derivatives of log posterior.
Equivalent to finding Hessian matrix.

$$S_N = -\nabla\nabla \ln p(w|t) = S_0^{-1} + \sum_{j=1}^{n} y_n(1-y_n)\phi_n \phi_n^T$$


→ Hessian

## EVALUATION OF POSTERIOR DISTRIBUTION:

→Gaussian prior

$p(w) = N(w/m_0, S_0)$

where $m_0$, $S_0$ are hyper-parameters.

→Posterior distribution

$p(w/t) \propto p(w)\, p(t/w)$

$t = (t_1, \cdots t_N)^T$

Substituting $p(t/w) = \prod_{n=1}^{N} y_n^{t_n} \{1-y_n\}^{1-t_n}$

$\ln p(w/t) = \frac{-1}{2}(w-m_0)^T S_0^{-1}(w-m_0)$

$+ \sum_{i=1}^{n} (t_n \ln y_n + (1-t_n)\ln(1-y_n) + \text{const}$

Where,

$y_n = \sigma(w^T \phi_n)$

## GAUSSIAN APPROXIMATION OF POSTERIOR:

↔Maximise posterior $p(w/t)$ to give

- MAP solution $w_{map}$ (Done by numerical optimisation)
- Defines mean of Gaussian.

↔Covariance given by

- Inverse of matrix of 2nd derivatives of negative log-likelihood $S_N = -\nabla\nabla \ln p(w/t) = S_0^{-1} + \sum_{j=1}^{n} y_n(1-y_n)\phi_n \phi_n^T$

↔Gaussian approximation to posterior

$q(w) = N(w/w_{map}, S_N)$

↔Need to marginalise w.r.t this distribution to make predictions.

## Predictive Distribution:

→Predictive dist for class $C_1$, given new feature vector $\phi(x)$.

→Obtained by marginalising w.r.t posterior $p(w/t)$.

$p(C_1/\phi, t) = \int p(C_1, w/\phi, t)\, dw$ (sum rule)

$= \int p(C_1/\phi, t, w)\, p(w/t)\, dw$ (Product rule)

$= \int p(C_1/\phi, w)\, p(w/t)\, dw$ (Given $\phi$, w, $C_1$ is independant of t)

$\simeq \int \sigma(w^T\phi)\, q(w)\, dw$ Approximate $p(w/t)$ by Gaussian $q(w)$

Corresponding probability for class $C_2$

$p(C_2/\phi, t) = 1 - p(C_1/\phi, t)$

## Predictive Dist. is a Convolution:

$p(C_1/\phi, t) \simeq \int \sigma(w^T\phi)\, q(w)\, dw$

- Function $\sigma(w^T\phi)$ depends on w only through its projection onto $\phi$.

→ $a = w^T\phi$, $\sigma(w^T\phi) \simeq \int \delta(a - w^T\phi)\, \sigma(a)\, da$.

(where $\delta$ is Dirac delta fn)

→ Thus $\int \sigma(w^T\phi)\, q(w)\, dw = \int \sigma(a)\, p(a)\, da$

$p(a) = \int \delta(a - w^T\phi)\, q(w)\, dw$

→ Can evaluate $p(a)$ because:

delta function imposes linear constraint on w, since $q(w)$ is Gaussian, its marginal is also Gaussian

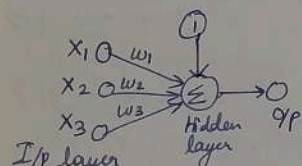
Evaluate its mean, covariance:

$$\mu_a = E(a) = \int p(a)\,da = \int q(w)\, w^T \phi\, dw = w_{mp}^T \phi$$

$$\sigma_a^2 = var[a] = \int p(a)\{a^2 - E[a]^2\}\,da$$

$$= \int q(w)\{(w^T\phi)^2 - (m_N^T\phi)^2\}\,dw = \phi^T S_N \phi$$

## BACKPROPAGATION ALGORITHM:

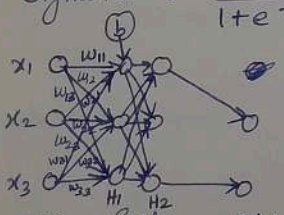



(random weights)

$$H_1 = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$z = f(H_1) = \text{activation}(H_1)$$

Sigmoid : $\frac{1}{1+e^{-x}} = \frac{1}{1+e^{-H_1}}$




forward pass (feedforward) →

← Backpropagation

Multilayer perceptron

**w=3:**

| Input (x) | Desired O/P (y) | Predicted O/P (model) (wx x) | \|y − wx\| Error |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 2 | 3 | 1 |
| 2 | 4 | 6 | 2 |

**w = 4:**

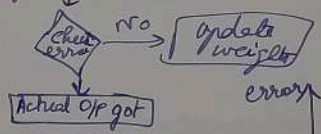
| I/P (x) | desired O/P (y) | Predicted O/P (wx x) | Error \|y − wx\| |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 2 | 4 | 2 |
| 2 | 4 | 8 | 4 |

**w = 2:**

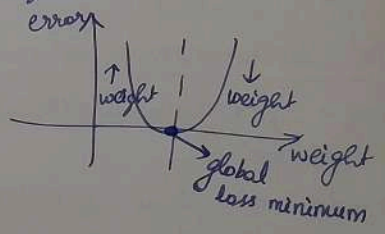
| I/P | y | wx | \|y − wx\| |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 2 | 2 | 0 |
| 2 | 4 | 4 | 0 |

0 error so weights are proper.

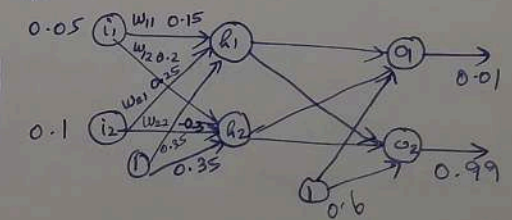
To calculate the error, adjust the weights




Updating weights, ← Back prop.




Eg:




Net $h_1$

$= w_1 i_1 + w_2 i_2$

$= 0.15 \times 0.05$

$+ 0.1 \times 0.25$

$+ 0.35 \times 1$

O/p of $H_1$ : $\frac{1}{1+e^{-H_1}} = \frac{1}{1+e^{(\text{---})}} = 0.5932$

$H_2 = w_{12} i_1 + w_{22} i_2 + b_1 \times 1 = 0.5968$

O/p of $H_2 = 0.5968$

Net $O_1 = w_{31} h_1 + w_{32} h_2 + b_2 \times 1 = 1.1059$

O/p of $O_1$ = Sigmoid of $O_1 = \frac{1}{1+e^{-9}} = 0.7513$

$O_2 = w_{41} h_1 + w_{42} h_2 + b_2 \times 1 = 0.7729$

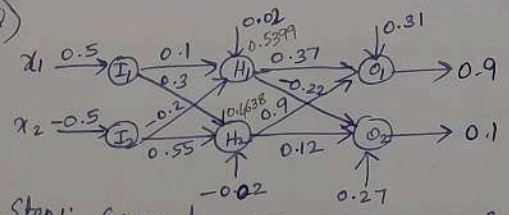
Error $O_1$ : $EO_1 = \sum \frac{1}{2} (\text{target} - \text{O/p})^2$

$= \frac{1}{2} (0.01 - 0.7513) = 0.274 \quad E_1$

$EO_2 = \frac{1}{2}(0.99 - 0.7729) = 0.0235 \quad E_2$

Total error = $E_{O1} + E_{O2} = 0.274 + 0.0235 = 0.2983$

Q)




$\eta = 1.2$

Step 1: forward pass, $Z = \sum x_i^k w_{ik}$

$Z_1' = x_1' w_{11} + x_2' w_{21} + b \times 0.01$

$= 0.5 \times 0.1 + (-0.2 \times -0.5) + 0.02 \times 1$

$= 0.16$

$Z_2' = (0.5 \times 0.55) + (0.3 \times 0.5) + (-0.02 \times 1)$

$= -0.145$

STEP 2: O/p from hidden layers Activation fn: Sigmoid

$\delta(Z_1') = \frac{1}{1+e^{-Z_1'}} = \frac{1}{1+e^{-0.16}} = 0.5399$

$\delta(Z_2') = \frac{1}{1+e^{-Z_2'}} = \frac{1}{1+e^{-0.145}} = 0.4638$

STEP 3: Hidden to O/p layer

$y_1' = 0.5399 \times 0.37 + 0.4638 \times 0.9 + 0.31 \times 1$

$= 0.9272$

$y_2' = 0.4638 \times -0.12 + 0.5399 \times -0.22 + 0.27 \times 1$

$= 0.0955$

STEP 4:

$\delta(y_1') = \frac{1}{1+e^{-y_1'}} = \frac{1}{1+e^{-0.9272}} = 0.7164$

$\delta(y_2') = \frac{1}{1+e^{-0.0955}} = 0.5238.$

Backward pass:

$d_1' = 0.9$ , $d_2' = 0.1$ Error:

$e_1' = d_1' - \delta(y_1') = 0.9 - 0.7164 = 0.1836$

$e_2' = d_2' - \delta(y_2') = 0.1 - 0.5238 = -0.4238$

O/p to hidden layer:

$\delta_1' = e_1 \times \delta'(y_1)$ ; $\delta'(y_1) = \delta(y_1')(1 - \delta(y_1'))$

$= 0.1836 \times (0.7164)(1 - 0.7164) = 0.0373 = \delta_1'$

$\delta_2' = e_2 \times \delta(y_2')(1 - \delta(y_2'))$

$= -0.4238 \times (0.5238)(1 - 0.5238)$

$\delta_2' = -0.1057$

## Hidden to I/p layer:

$\delta_1'(z_1)$ = $(\delta_1' w + \delta_2' w)\ \delta' z_1$

$= \left[(0.0373 \times 0.37) + (-0.1057 \times -0.22)\right] (\delta(z_1')(1-\delta(z_1')))$

$= \left[(0.0373 \times 0.37) + (-0.1057 \times -0.22)\right] \times 0.5399\,(1-0.5399)$

$= 0.0092$

$\delta_2'(z_2) = \delta'(z_2)\ (\delta_1 w + \delta_2 w)$

$= \left[(0.0373 \times 0.9) + (0.1057 \times 0.12)\right] \times (0.4638)(1-0.4638)$

$\delta_2'(z_2) = 0.0115$

## weight change for pass 1:

$w_{new} = w_{old} - \eta \times \frac{\delta E}{\delta w_{ij}}$ → bias $\qquad \eta = 1.2$

$\Delta_{n01}' = 1.2 \times 0.0373 \times 1 = 0.0447 + 0.31 = 0.3547$ (0.31 +)

$\Delta_{n11}' = 1.2 \times 0.0373 \times 0.5399 = 0.0241 + 0.37 = 0.3941$ ($\delta(z_1')$, 0.37 +)

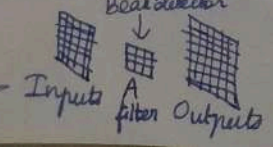
$\Delta n_{21}' = (0.9) + (1.2 \times 0.0373 \times 0.4638) = 0.9207$

$\Delta n_{a2}' =$

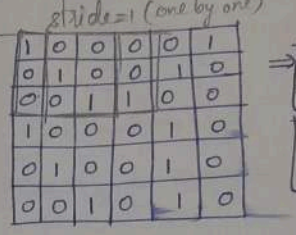
## CONVOLUTIONAL LAYER

→ CNN is a neural network with some convolutional layers, and some other layers.

→ Convolutional layer has no. of filters that does convolutional operation.




## CONVOLUTION OPERATION:


RGB → (3 channels).

## CONVOLUTION vs. FULLY CONNECTED N/W:

→ In convolution, the values are finally taken as a single row and the dimension reduces.

→ Dropout is present (those values of less importance are not connected to the O/p.

## Convolution PROCESS:

Convolution → Max pooling (dimension reduction) → Convolution

Fully connected Feed Forward Neural Network ← ... ←

## COMPRESSION IN CNN:

→ Reducing no. of connections

→ Shared weights on the edges

→ Max pooling reduces complexity.

## RNN:

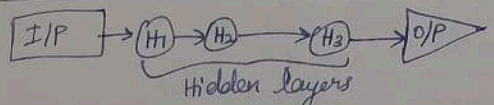
O/P of previous step are fed as I/P to current step

I/P → Hidden layer → O/P

Hidden state → Remembers some info about a sequence

| CNN | RNN |
|---|---|
| → Image based model. | → Language based model |
| → Image processing | → NLP |
| → All neurons in I/P connected to hidden layers. | → Hidden state (has some memory), uses same parameters. |

### RNN WORKING:




$H_1$ $(w_1, b_1)$ $H_2$ $(w_2, b_2)$ $H_3$ $(w_3, b_3)$ → (Each layer is indep of each other)

→ 1st layer O/P → I/P of 2nd.

$h_t = f(h_{t-1}, X_t)$ → I/P of present state

($h_t$ ↓ current state) ($h_{t-1}$ ↓ prev. hidden layer state)

Formula for applying activation function (tanh)

$h_t = \tanh(W_{nh} h_{t-1} + W_{nx} X_t)$

($W_{nh}$ ↓ recurrent neuron) ($W_{nx}$ ↓ I/P neuron)

O/P: $y_t = W_{hy} * h_t$ → $W_{hy}$ → weight at O/P layer.

$y_t$ → O/P.

PROS: RNN → time series problems (remembering is demanded)

Eg: Weather prediction

RNN → LSTM (Long Short Term Memory).

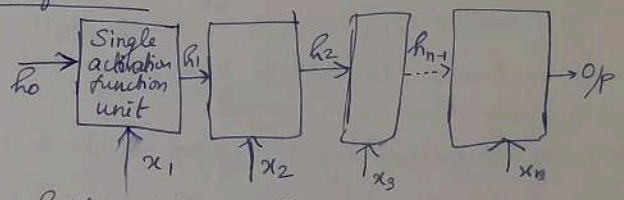
CONS:

→ Gradient vanishing and exploding.

→ RNN training → difficult

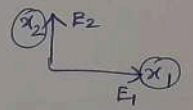
→ Can't process very long sequences.

### Flow of RNN:




$h_0$ → hidden state

$x_1$ → Current state




$x_1, x_2, x_3, \ldots x_n$ → Given at each time instant

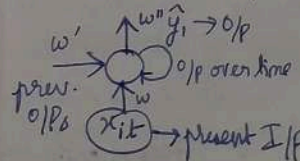
→ That's why RNN is time series problem.

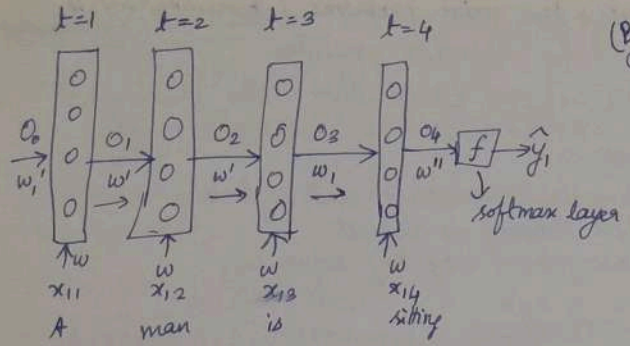
→ O/P generated at each time instant.

### NATURAL LANGUAGE PROCESSING:

→ I/P (text data)

→ BOW, TF-IDF, WORD2VEC ⇒ algorithms






→ Same hidden layers are repeated, so only all are carrying their own weights.

→ O/p of 1st hidden layer is given as I/p to 2nd.

$O_1 = f(x_1 w + O_0 \times w')$ $\quad O_3 = f(x_3 w + O_2 w')$

$O_2 = f(x_2 w + O_1 \times w')$ $\quad O_4 = f(x_4 w + O_3 w')$

O/p = prev + present state

Loss = $(\hat{y} - y)$

↓ reduce the loss to get desired o/p.

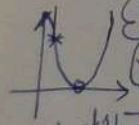
$$\frac{\partial L}{\partial \hat{y}_1} \times \frac{\partial \hat{y}_1}{\partial w} = \frac{\partial L}{\partial y_1} \times \frac{\partial \hat{y}_1}{\partial O_4} \times \frac{\partial O_4}{\partial w} \Rightarrow \frac{\partial L}{\partial w}$$

$$w = w + \frac{\partial L}{\partial w}$$

$$w_{new} = w_{old} + \frac{\partial L}{\partial w}$$

Vanishing, exploding gradients

DRAWBACK OF RNN:

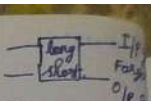


Drawbacks

Vanishing gradient { negligible weight updation, so waste of processing)

Exploding gradient { (the weight keeps changing randomly, doesn't reach global optimum)

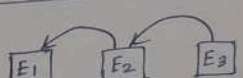
(Better accuracy) **LSTM (Long short term memory)**




→ Vanishing, exploding gradient overcome.

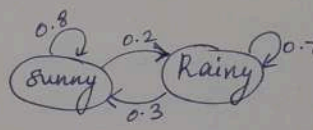
→ Since RNN can retain info only for short time.

## HIDDEN MARKOV MODEL:




$E_n$ (Event n) depends on $E_{n-1}$ (previous one)

Q)




MARKOV MODEL

States: Sunny, Rainy

Initial probability: {0.4, 0.6}

Transmission matrix:

prob. of this given that

| given this | Sunny | Rainy |
|---|---|---|
| Sunny | 0.8 | 0.2 |
| Rainy | 0.3 | 0.7 |

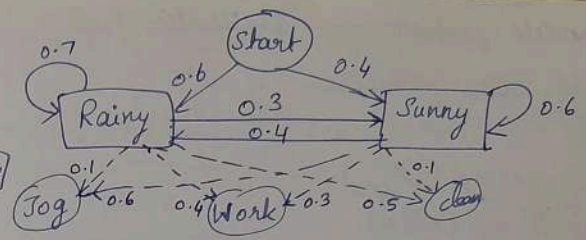
Q = {Sunny, Rainy, Rainy}
Today Tom day after tom

$P(Q) = P(S) * P(R/S) * P(R/R)$

$= 0.4 \times (0.2) \times (0.7)$

$= 0.08 \times 0.7 = 0.056$

| MARKOV MODEL | HIDDEN MARKOV MODEL |
|---|---|
| → Observables are visible (explicit) | → Observables are not visible (implicit) |
| → Eg: If it will be sunny or rainy is visible. (explicit) | → Eg: If it will be sunny or rainy is not visible (implicit) |
| | → Whether the weather is sunny or rainy can be implicitly drawn from {cotton, woollen} clothes. |




Hidden Markov model

~~DATA AUGMENTATION~~: PARAMETER TRYING, SHARING IN CNN

## HIDDEN MARKOV MODEL:

→ One of the statistical models used in ML algorithms to go for classification problems.

Eg: {cotton, woollen} ⇒ {sunny, rainy}

a) States : {Rainy, Sunny}

observations : {Jogging, office, cleaning}

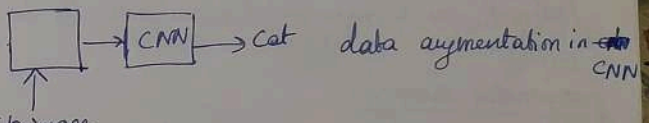
Starting probability : { Rainy = 0.6 Sunny = 0.4 }

Transition probability | Actual R / S | Probabilities of 2 states: {Rainy = 0.7, Sunny = 0.3} {Rainy = 0.4, Sunny = 0.6}

Emission probability = {

Rainy : { Jog : 0.1, work = 0.4, clean = 0.5 }

Sunny : { Jog: 0.6, work = 0.3, clean = 0.1 }




I/P image

data augmentation in CNN

→ Due to data augmentation, sample images are increased, so better model, better results, more useful features.

### Parameter tying, sharing:

→ To reduce complexity.

→ To reduce computation time

Types:

(i) Other methods for prior knowledge of parameters

(ii) Parameter tying

(iii) Parameter sharing

(iv) " " in CNNs.

2 models perform some classification task:

model 1 with parameters of $w^A$
model 2 " " " $w^B$

$\hat{y}^{(A)} = f(w^{(A)}, x)$
$\hat{y}^{(B)} = f(w^{(B)}, x)$
$w_i^{(A)} \simeq w_i^{(B)}$

parameter norm penalty:

$\Omega(w^{(A)}, w^{(B)}) = \|w^{(A)} - w^{(B)}\|_2^2$ (similar features tied)

→ Used for regularising parameters of one model trained as a supervised classifier to be closer to parameters of another model.

parameter sharing:

→ Single type of parameter used for all the tasks.

Eg: Parameters $c_1, c_2$

Tasks $T_1, T_2$

Either $c_1$ used for both $T_1, T_2$ ~~and~~ (or) $c_2$ used for both $T_1, T_2$.

| | | | |
|---|---|---|---|
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |
| m | n | o | p |

Input

⇒

| | |
|---|---|
| w | x |
| y | z |

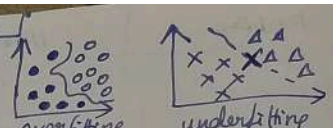
Kernel filter

→ $\begin{pmatrix} aw+bx \\ +ey+fz \end{pmatrix} \begin{pmatrix} bw+cx \\ +fy+gz \end{pmatrix}$

→ Done because transferring the ~~image~~ pixels or scaling them or doing some little changes doesn't make a big effect on the overall image.

REGULARISATION:

EARLY STOPPING:




Training, testing phase

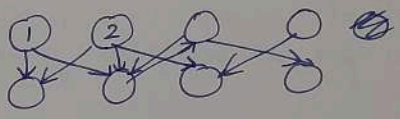
~~95%~~ 90%.

(2% error)

→ Here the error has nearly saturated, further epoch don't change any of these parameters, hence, early stopping is necessary to save time.

DROPOUT:

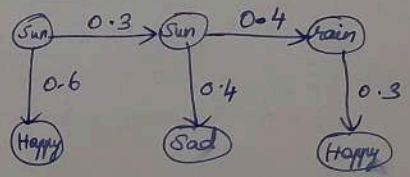
→ Helps reducing interdependent learning among the neurons.




→ When more than one neuron convey the same info, redundant (non-contributing) neurons are removed to save time, reduce complexity

FORWARD ALGORITHM FOR HMM:

Q)




Current time step to derive the probability of the next time step.

Given a sequence of visible state $V^T$, what will be the probability that the hidden Markov model will be in a particular hidden state $S$ at a particular time step $t$?

DERIVATION:

$$\alpha_j(t) = P(v(1) \dots v(t), s(t) = j)$$

Solution using probabilities:

When $t = 1$:

$$\alpha_j(1) = P(V_k(1), s(1) = j)$$
$$= P(V_k(1) \mid s(1) = j)\ P(s(1) = j)$$
$$= \pi_j \cdot \underbrace{P(V_k(1) \mid s(1) = j)}_{b_{jk}}$$

$$\boxed{\alpha_j(1) = \pi_j \times b_{jk}}$$

When $t = 2$:

$$\alpha_j(2) = P(V_k(1), V_k(2), s(2) = j)$$
$$= \sum_{i=1}^{M} P(V_k(1), V_k(2), s(1) = i, s(2) = j)$$
$$= \sum_{i=1}^{M} P(V_k(2), s(2) = j, V_k(1), s(1) = i) \times P(V_k(1), s(2), s(1) = i)$$
$$= \sum_{i=1}^{M} P(V_k(2) \mid s(2) = j, V_k(1), s(1) = i) \times P(s(2) \mid V_k(1), s(1) = i) \times P(V_k(1), s(1) = i)$$
$$= \sum_{i=1}^{M} P(V_k(2) \mid s(2) = j)\ P(s(2) \mid s(1) = i)\ P(V_k(1), s(1) = i)$$
$$= \underbrace{P(V_k(2) \mid s(2) = j)}_{b_{jk_2}} \sum_{i=1}^{M} \underbrace{P(s(2) \mid s(1) = i)}_{a_{i2}} \underbrace{P(V_k(1), s(1) = i)}_{\alpha_j(1)}$$

$$\boxed{= b_{jk_2} \sum_{i=1}^{M} a_{i2} \cdot \alpha_i(1)}$$

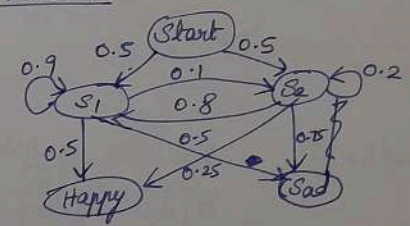
Any time step $t+1$:

$$\alpha_j(t+1) = P(V_k(1) \dots V_k(t+1), s(t+1) = j)$$
$$= \sum_{i=1}^{M} P(V_k(1) \dots V_k(t+1), s(t) = i, s(t+1) = j)$$
$$\vdots$$

$$\boxed{= b_{jk_v}(t+1) \sum_{i=1}^{M} a_{ij} \alpha_j(t)}$$

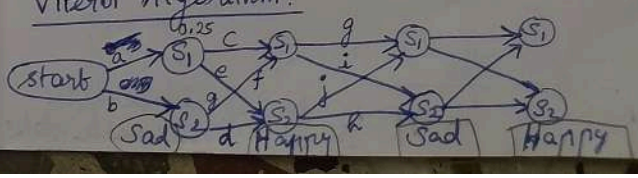
TRELLIS DIAGRAM:

9)




Pattern → Sad - Happy - sad - Happy

Viterbi Algorithm:



| Sad | Happy | Sad | Happy |
|---|---|---|---|

a ⇒ $0.5 \times 0.5 = 0.25$
(start → $S_1$, $S_1$ → Sad)

c ⇒ $0.25 \times 0.9$ (state value) ($S_1$ → $S_1$) $\times 0.5 = 0.1125$
($S_1$ - Happy)

b ⇒ $0.5 \times 0.75 = 0.375$

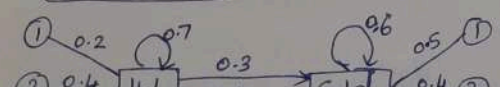
f ⇒ $0.375 \times 0.8 \times 0.75 = 0.225$

d ⇒ $0.25 \times 0.2 =$

## DECODING WITH VITERBI ALGORITHM: (HMM)

$\Pi = [0.8, 0.2]$
H C

1, 2, 3 → Prob. of taking that many icecreams.

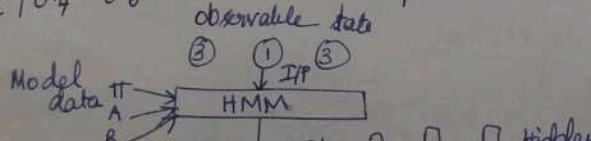



Transition matrix (A)

| | H | C |
|---|---|---|
| H | 0.7 | 0.3 |
| C | 0.4 | 0.6 |

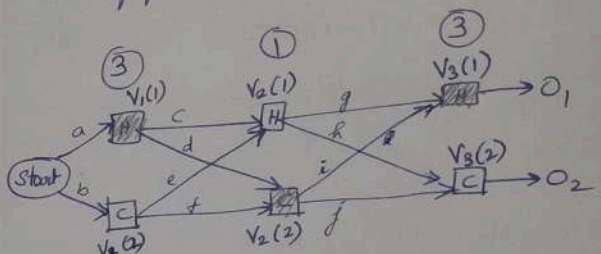
Emission matrix (B)

| | 1 | 2 | 3 |
|---|---|---|---|
| H | 0.2 | 0.4 | 0.4 |
| C | 0.5 | 0.4 | 0.1 |




N = 2, T = 3 → No. of observables (1, 2, 3)
↓
no. of states

No. of possible outcomes = $N^T = 2^3 = 8$




a ⇒ $P(3, H) = P(3|H)\,P(H)$
$= 0.4 \times 0.8 = 0.32$
↳ initial state

b ⇒ $P(3, C) = P(3/C)\,P(C)$
$= 0.1 \times 0.2 = 0.02$

c ⇒ $= [P(1/H) \times P(H/H)] = P(1, H) = 0.2 \times 0.7 = 0.14$
H → H

d ⇒ $[P(1/C) \times P(C/H)] = 0.5 \times 0.3 = 0.15$

e ⇒ $[P(1/H) \times P(H/C)] = 0.2 \times 0.4 = 0.08$

f ⇒ $[P(1/C) \times P(C/C)] = 0.5 \times 0.6 = 0.3$

$V_1(1) = 0.32$, $V_1(2) = 0.02$

$V_2(1) = 0.32 \times c$ (or) $0.02 \times e$

$V_2(2) = 0.32 \times d$ (or) $0.02 \times f$

Max value taken for Viterbi algorithm among the two

$V_3(1) \Rightarrow P(3,H) = P(3/H)\,P(H/H) \times V_2(1) = \boxed{0.4 \times 0.7 \times 0.32c}$

(or) $P(3/H)\,P(H/C) \times V_2(2) = \boxed{0.4 \times 0.4 \times 0.32d}$

$V_3(2) \Rightarrow P(3,C) = P(3/C)\,P(C/H) \times V_2(1) = \boxed{0.1 \times 0.3 \times V_2(1)}$

(or) $P(3/C)\,P(C/C) \times V_2(2) = \boxed{0.1 \times 0.6 \times V_2(2)}$

Now $V_1(1)$ is maximum among $V_1(1)$, $V_1(2)$

$V_2(2)$ is max

$V_3(1)$ is max

NOTE: For forward algorithm, sum of 2 values taken for a node

For Viterbi algorithm, max of 2 values taken

## BACKWARD ALGORITHM FOR HMM:

DERIVATION:

$$B_i(t) = P(V_k(t+1) \ldots V_k(t) / (s(t)) = i)$$

$$= \sum_{j=0}^{M} P(V_k(t+1) \ldots V_k(t), s(t+1) = j \mid s(t) = i)$$

$$= \sum_{j=0}^{M} P(V_k(t+2) \ldots V_k(i), s(t+1) = j \mid s(t) = i)$$

$$= \sum_{j=0}^{M} P(V_k(t+1) s(t+1) = j, s(t) = i)$$

$$P(s(t+1) = j \mid s(t) = i)$$

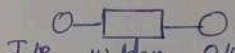
$\sum_{j=0}^{M} \beta_j(t+1)\, b_{jkv}(t+1)\, a_{ij}$

- $a_{ij}$ → transition probability
- $b_{jkv}$ → Emission probability
- $\beta_j$ → Backward probability.

## TYPES OF RNN:

### 1) ONE-ONE RNN:

→ (Vanilla's NN)



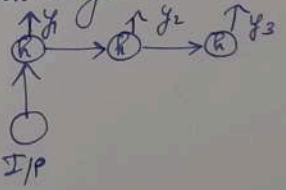

Applications:
- → weather prediction
- → Stock prediction

Single I/P given, only one O/P.

### 2) ONE - MANY RNN:

→ Only one I/P fed, many O/Ps obtained at each stage.

Eg: Image captioning

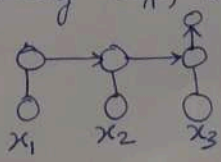



### 3) MANY - ONE RNN:

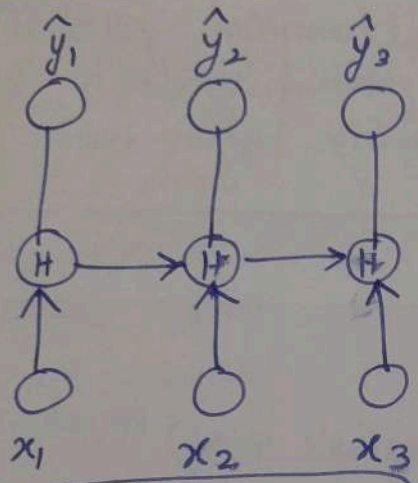
→ Many I/P, one O/P.

Eg: Sequential analysis

→ Movie tre on -ve rating based on comments.



4) MANY - MANY RNN's

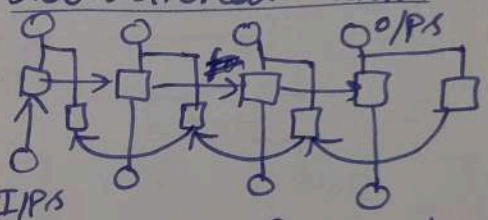



Eg: Google translate

DEEP RNN's:

→ More than 1 hidden layers

Bidirectional RNN:




B

GRU (Gated Recurrent Units):

→ 2 gates only (update, reset)

update → adds new info

reset → removes old info

→ Computation is less, efficient processing, but less accuracy.