Machine Learning

Learning → change in behaviour

USML → coins sorting

training model w/o any labels, identify patterns on its own.

Machine learning is a subset of AI enables computers to learn & make decisions based on data, w/o being explicitly programmed.

Clustering → learning through observat. based on common features.

- Hierarchical
  - Agglomerative (Bottom Up) merges
  - Divisive (Top Down) divide
- Dendrograms

Types of Clustering

DBSCAN → cluster points that are closely packed tog. (point density)

Grid Based: grid structure env. performs clustering on grid.

Large Spatial data., weather forecasting

- Spectral: (multi-dimensional) reduce dimensions, social network analysis., eigenvalues of similarity matrix

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)}$$

$$P(X|C_i) = \prod P(\text{---}) \cdot P(C_i|X)$$

SML

input/output, training, error correction, Labelled

training model with data, for every input output is provided

→ Speech → Medical Data Processing

→ Image Recognition, Recognition, Diagnosis

→ Automation complex data.

Reinforcement Learning → agent learns from make decisions by performing actions and receiving feedback in form of rewards / penalty

Environment, Action, Rewards

Centroid Based Clustering

inter cluster similarity

Manpower, Accuracy, Data Req.

Regression: statistical technique to relate dependent var. (y) to one or more independent var. (x)

$y = \bar{w}\bar{x} + b$ Linear

Logistic Regression → probability input follows to certain category

Concept learning → Inference from label training data., patterns

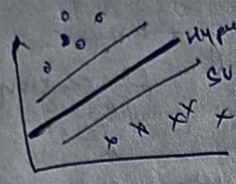
Bayes Classifier → highest posterior probab
- theoretical
- problem.

(spam / 'offer' 'promotion')

$$P(y/x_1 x_2 \cdots x_n) = \frac{P(y)\, P(x_1/y) P(x_2/y) \cdots P(x_m/y)}{P(x_1) \cdot P(x_2) \cdots P(x_n)}$$

• Bayes Belief Networks

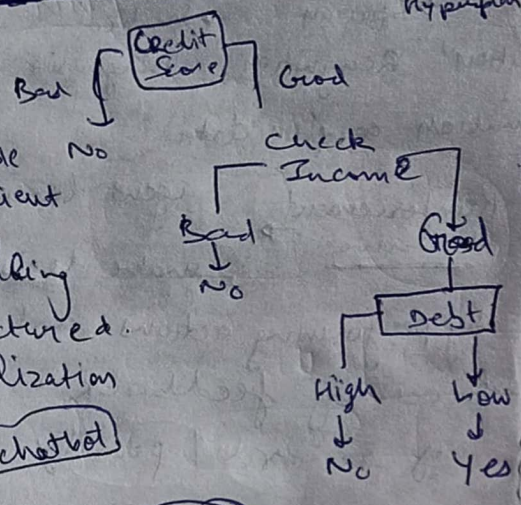
Tree Base multi factor



Support Vector Machines

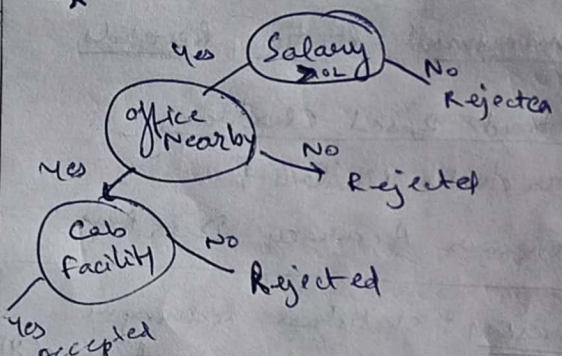
maximum Margin Hyperplan

Decision Trees



- Scalable
- Efficient
- Data Handling
- Structured Visualization

AI chatbot



Entropy → measure of uncertainty
→ amount of randomness in target variable
Split data

$E=0$ min uncerta | $E=1$ max uncertain | $0<E<1$ mix

IG → measure change in entropy due to factors.

Subset A, B, C; H(A), H(B)

$H(S) = P(A)H(A) + P(B)H(B)$

$(I.G.) = H - H(S)$

Highest IG is first selected.

Bayes

$P(y=c/x_i) \propto \prod_{i=1}^{n} P(x_i/y=c) \cdot P(y=c)$

ID3 → top down, greedy, High IG first, more overfit

C4.5 → extension of ID3

GainRatio $(1 - \Sigma(P_i)^2)$

prunes the tree, handles missing values & removes overfitting, categorical & Complem. Val

CART → Gini Impurity, Binary Splits, pruning

Random forest, Robust

Inductive Bias

assumptions made by a learn, algo to generalise for training to make predictions.

G.I. = $1 - \Sigma p_i$ (measure of impurity)

G.I=0, pure

KNN → read self

non parametric SML. Euclidean Distance.

lazy learning. (K) | imbalanced data |

ANN ⇒ Computing System to simulate human brain

data processing system, neuron, self learning capabili

Input Hidden Output

↓

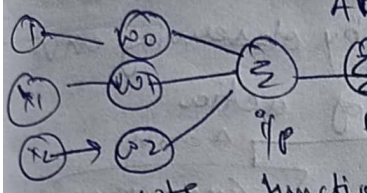
feature extraction

hidden layers (neurons

Recurrent Network | Perceptron

Artificial Neuron.

Simple neural network



$$\Sigma(w_i x_i + b) = \begin{cases} 1, & >0 \\ 0, & \leq 0 \end{cases}$$

And gate

$x_1 + x_2 - 1 = y$

$\Delta w_i = \alpha (t-o) \times x_i$

$\Delta b_i = \alpha (t-o)$ $\quad b = b + \Delta b$

$w \Rightarrow w + \Delta w$

Linearly Separable Data.

properly separated by straight line

adjust wt & bias to minimize error

New value = $x - f'(x)\alpha$

$$E = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$\Delta w = -\alpha \nabla E(w)$$

$$\nabla E(w) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right)$$

$\frac{\partial E}{\partial w_i}$

α - also imp

$\alpha \downarrow$ (slow)

$\alpha \uparrow$ (can fast jump)

Stochastic (random) | Mini Batch

Smaller Batches of

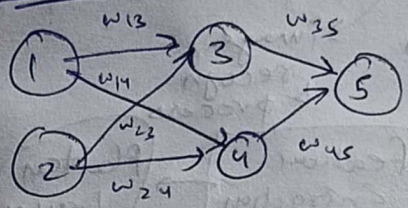
Backpropagation

Repeatedly adjust weights of connections to minimize difference b/w actual o/p & desired o/p

Question

Fwd pass

B/w Pass



$\Delta w_{35} = \eta \delta_5 H_3$

$\delta_5 = (y - O_5) O_5 \cdot (1 - O_5)$

$\delta_3 = \delta_5 \cdot w_3 \cdot H_3 (1 - H_3)$

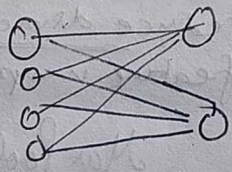
w or $\rightarrow \delta$

Competitive Learning Algo

Self Organizing Maps

to lower dimensions

$D = \sqrt{\sum (x_i - w_i)^2}$ euclide



min

$w_{new} = w_{old} + \alpha (x_i - w_{i0})$

update

stop when changes = 0.

Neural Network → Deep Learning

subset of ML that involves NN with many layers.

large amts of Data

Hierarchical clustering is connectivity based clustering model that groups data points together that are close to each other based on measure of similarity (distance

Deep Learning → CV, speech, NLP

Healthcare, finance (Fraud Detection)

- Feature Extraction
- High performance
- Versatile

CNN — DL model with grid-like topology

filter optimization → image recogn & process

Input → Feature Extraction → Pooling, Conv. + ReLU → Flatten Layer → Classification → Activation → O/p

(1) Feature Extraction (convolution ops)
→ filters / kernel → matrix
(dot product b/w filter)

(2) Padding → adding zeros to preserve dims

(3) Pooling → reduce dimensions of feature maps

Avg Pooling: avg value from each patch
Max Pooling: (2×2) window
Reduce Noise, Distortion

ReLU max(0, x)

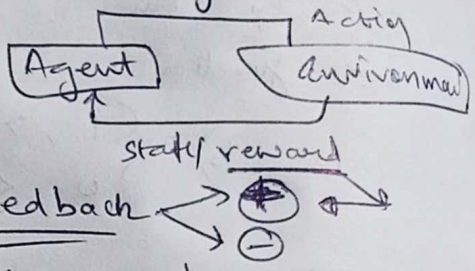
Policy (π)
↳ strategy / rule that defines action to be taken in each case.

Adam → adjusts speed & optimizer learning Rate

Gradient Descent

Reinforcement Learning

Feedback based ML.
Agent learns actions to perform by observing env. & results of action



Feedback → (+) (−)

State → situation

Semisupervised

Sequential approach, experience

(+) Reinforcement → encourage with (+) rew.

(−) ----

Maze Game, Grid game.

- Actions are interdependent
- Delayed feedback
- Time consuming

traffic light, industrial automation, driverless cars, recommendation system.

Markov Decision Process

formal model for decision making where outcomes are partly random partly under control of agent

Memoryless property
↳ future state depends on ↳ current state, Action
not on sequence of events

ROC → class distribution is balanced
fp ≈ fn

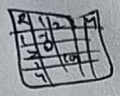
PR curve → imbalanced, Positive class predict
dataset

State, Actions, Transition (P)
Reward, Discount factor

Q-Learning → model-free reinforcement algo

optimal action-state value

Value based



greedy

DBSCAN

Density Based Spatial Clustering of Applications with Noise

$$Q_{new} = Old\ value + \alpha\,(Reward + Max\ factor\ Q_{next})$$

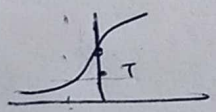
update table

Reward function $Q(s,a)$

assigns numerical value to an agent based on its current state & action. (process matter)

Value function (Recall mean eqn)

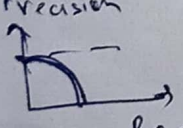
a function that estimates the expected long term reward an agent will receive for a given state / action



T ↑, Recall ↓

Precision ↑

FP ↓, FN ↑



Area under curve

Covariance matrix for each feature

$$S = \begin{bmatrix} x_1x_1 & x_1x_2 \\ x_2x_1 & x_2x_2 \end{bmatrix}$$

- $\frac{1}{N-1} \Sigma (x_{1k} - \bar{x}_1)(x_{2k} - \bar{x}_2)$

$$(S - \lambda I) = 0$$

② Solve determinant

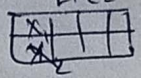
get $\lambda \rightsquigarrow \lambda_1, \lambda_2$

③ Eigen values

$$(S - \lambda I) \cdot U = 0 \qquad U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\frac{u_1}{a} = \frac{u_2}{b} = t$$

$$V = f(x) = [=]$$



④ unit eigen vector.

$$\hat{v} = \left[\frac{v}{\|v\|}\right]$$

$e_1 = f(\lambda_1) \qquad e_2 = f(\lambda_2)$

⑤ First Principal Component

$$e_1^T \cdot \begin{bmatrix} x_{1k} - \bar{x}_1 \\ x_{2k} - \bar{x}_2 \end{bmatrix}$$

for each example

$$P(y|x) = [h_\theta(x)]^y [1 - h_\theta(x)]^{1-y}$$

Logistic maximum likelihood estimati

$$\frac{\partial}{\partial \theta}(\log LL) = \frac{1}{m} \Sigma (y_i - h_\theta(x)) x_i$$

$y_i \log(h_\theta(x)) + (1-y_i) \log(1-h_\theta)$

= Binary Cross Entropy

CNN

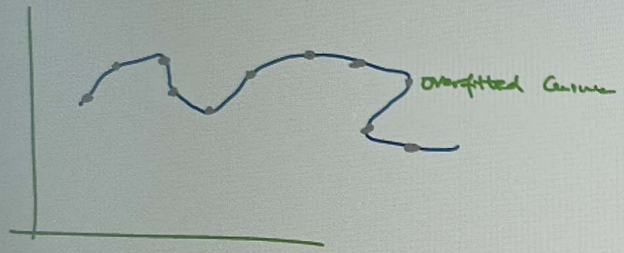
Convolutional Layer – Feature extraction

Pooling Layer → reduce size

Fully Connected Layer → classification

Overfitting:

Model performs very well on training data but doesnot perform good on test data.

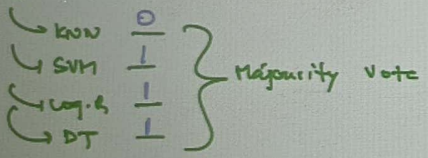


Solution:

→ k fold cross validation

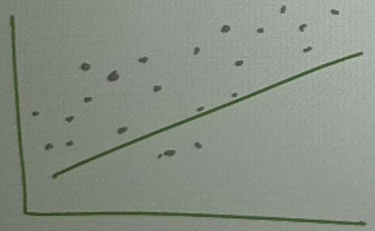
→ Sufficient data → Bootstrap

→ Ensembling technique.

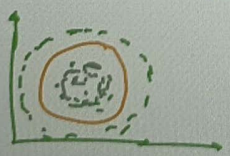


Underfitting:

Model is not going to learn patterns from training data.



Underfit model will give poor performance on train as well as test data.



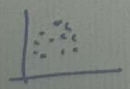
Solutions:

→ No. of features increase

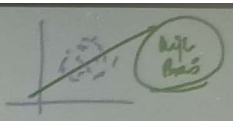
→ Models Complexity

→ Reduce noise

→ Increase the duration of training.



BIAS AND VARIANCE:



Bias: wrong assumptions about data — like assuming data is linear in reality it follows a complex fn.

→ It is the inability of the model bcz of that there is diff in predicted value & actual value

Low Bias: lower assumptions
make a model which closely matches the training dataset. (Simple model)

High Bias: more assumptions
model will not match training dataset closely. (Complex model)

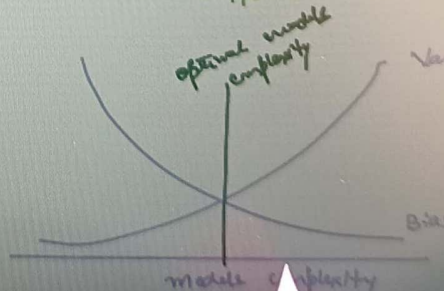
Variance:

→ measure of spread in data from its mean position

→ Sensitive to a subset which follows same distribution as your training dataset

Low variance: model is very less sensitive to changes

High variance: model is very sensitive to changes and can result in significant changes on a different subset.



ROC Curve
(Receiver Operator Characteristics Curve)

- An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds.

- This curve plots two parameters: TPR & FPR

- True Positive Rate (TPR)= $\frac{\#\text{ true positives}}{\#\text{ actual positives}} = \frac{\#\text{ true positives}}{\#\text{ true positives} + \#\text{ false negatives}}$

- False Positive Rate (FPR) = $\frac{\#\text{ false positives}}{\#\text{ actual negatives}} = \frac{\#\text{ false positives}}{\#\text{ ture negatives} + \#\text{ false positives}}$

Common Approaches for Reward Function Design

Sparse vs. Dense Rewards

- **Sparse rewards** provide feedback only when a significant event occurs (e.g., the agent only receives a reward when it reaches the goal). Sparse rewards can make learning challenging but encourage exploration.
- **Dense rewards** provide feedback at every step (e.g., the agent receives a small reward for each correct action that moves it closer to the goal). Dense rewards facilitate faster learning but may lead to premature convergence on suboptimal strategies.

Shaping Rewards

Semi-supervised learning is a machine learning paradigm where the dataset contains both labeled and unlabeled data. This approach is particularly useful when labeled data is scarce or expensive to obtain, but there is an abundance of unlabeled data available.

Example of Semi-supervised Learning

Imagine you're working on a sentiment analysis task where you want to determine the sentiment (positive, negative, or neutral) of customer reviews for a product.

However, you only have a small dataset of labeled reviews, but there's a vast amount of unlabeled reviews available online.

$\lambda_1 > \lambda_2$

So, $\lambda_1 \rightarrow$ is our principal component.

$-\frac{1}{2} \log \frac{1}{2}$

$-\log(0.5)$ - (1)

① Policy: Defines agents behaviour for a given state / time.

State → (mapping) → Action

② Reward fn: It provides a numerical score based on state of environment

③ Value fn: value of state is total amount of reward an agent can expect to gain over the future starting from that particular state.

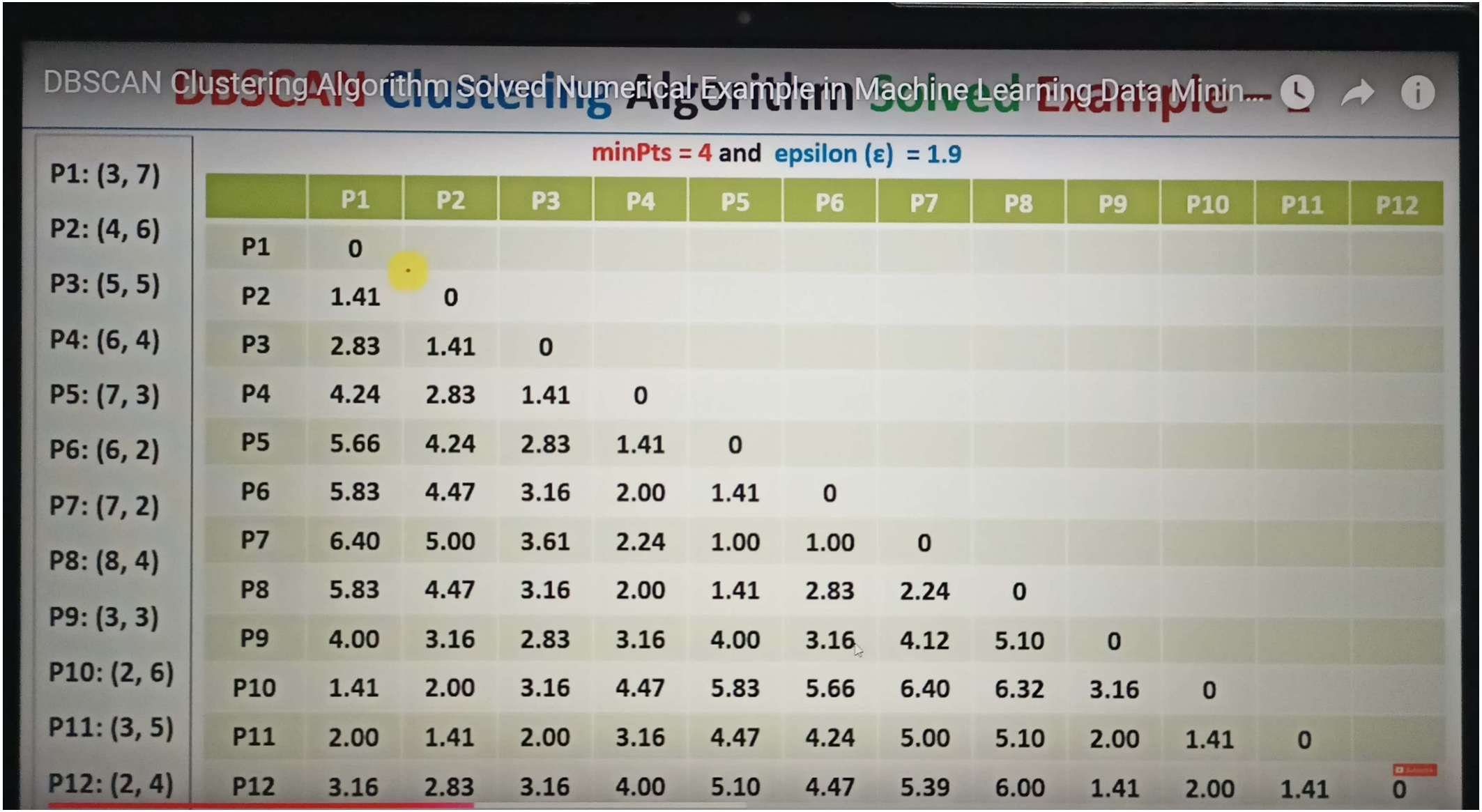
① Initialize Q table

↳ It's a simple lookup table where we calculate max. expected future rewards for action at each state.

Actions: ↑ → ↓ ←

S	0	0	0	0
B	0	0	0	0

Rows :- no. of states



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P1	0											
P2	1.41	0										
P3	2.83	1.41	0									
P4	4.24	2.83	1.41	0								
P5	5.66	4.24	2.83	1.41	0							
P6	5.83	4.47	3.16	2.00	1.41	0						
P7	6.40	5.00	3.61	2.24	1.00	1.00	0					
P8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
P9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
P10	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0		
P11	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
P12	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

- Ridge Regression

Regularization

Lasso Regression

RR = Loss + $\alpha \|w\|^2$

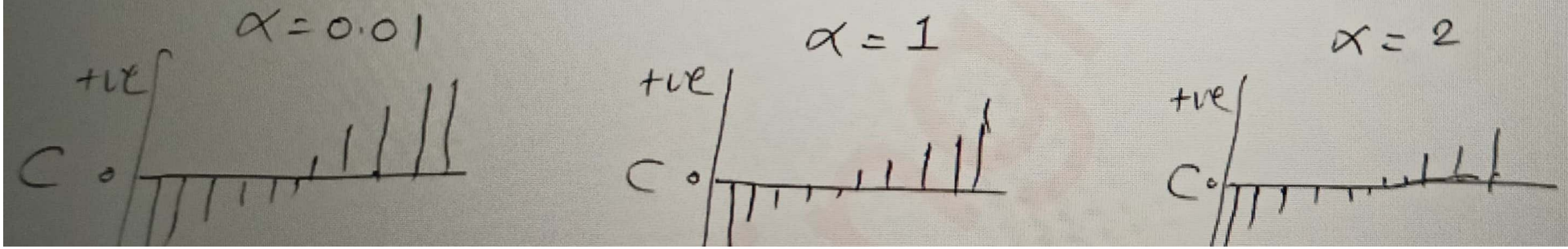
(penalty)

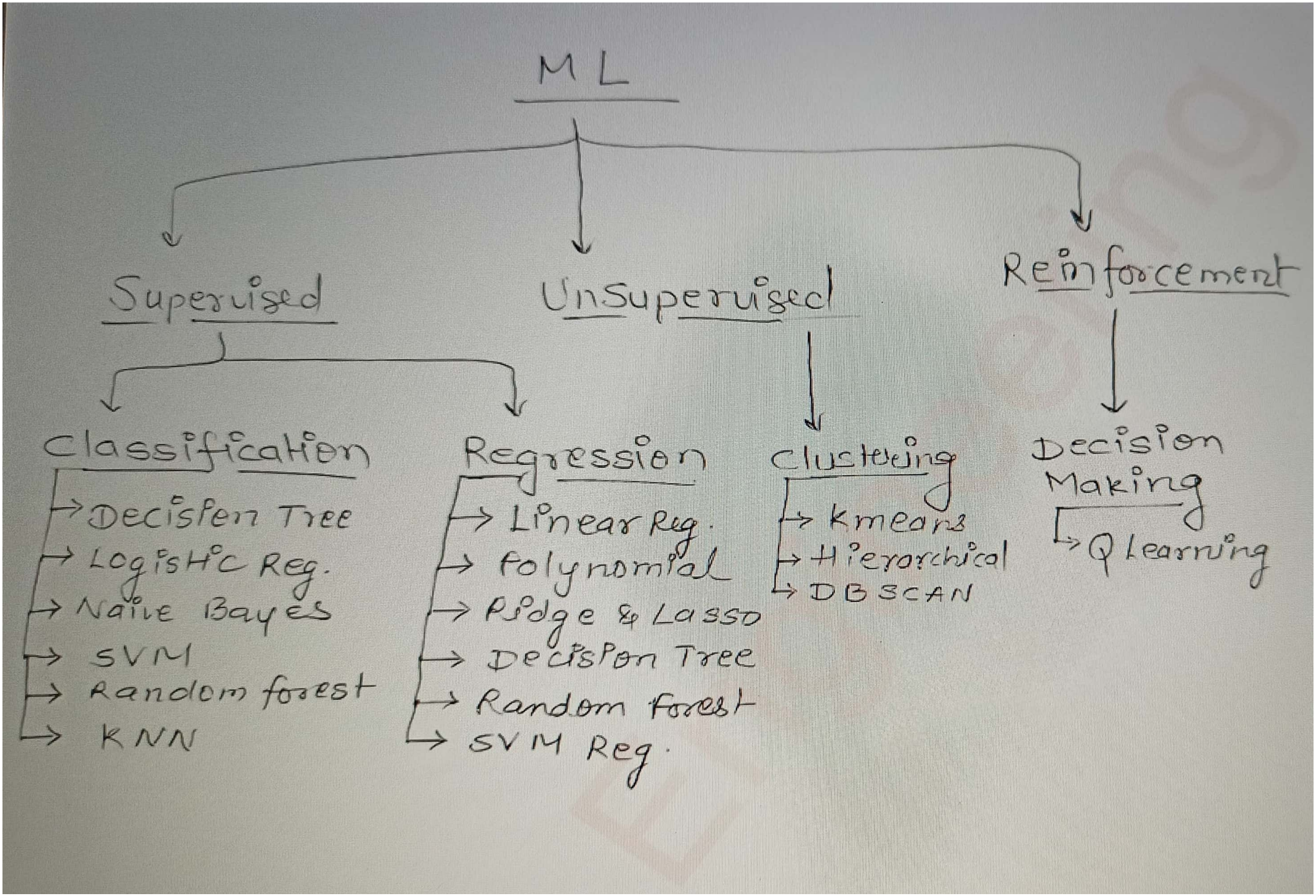
LR = Loss + $\alpha \|w\|$

(can be zero)

↑ absolute value.

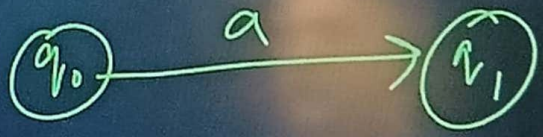
$\|w\|^2 = w_1^2 + w_2^2 + w_3^2 + w_4^2 + \cdots w_n^2$




ML
Supervised
UnSupervised
Reinforcement
Classification
Decision Tree
Logistic Reg.
Naive Bayes
SVM
Random forest
KNN
Regression
Linear Reg.
Polynomial
Ridge & Lasso
Decision Tree
Random Forest
SVM Reg.
Clustering
kmeans
Hierarchical
DBSCAN
Decision Making
Q learning

Markov Decision Process (MDP)

- A Markov Decision Process (MDP) is a mathematical framework used to describe an environment in reinforcement learning. It provides a formal model for decision-making in situations where outcomes are partly random and partly under the control of a decision-maker (agent).

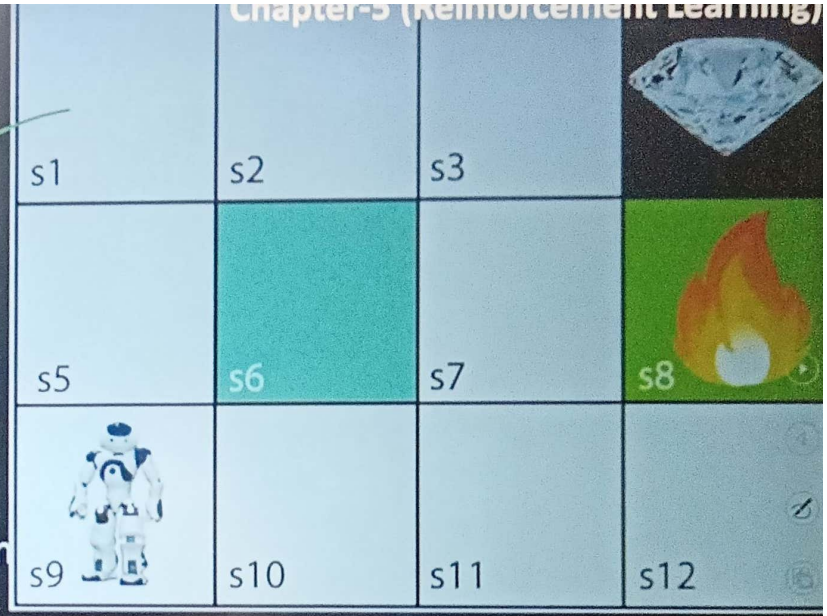


- **Characteristics of MDP**
- **Markov Property**:
  - The future state depends only on the current state and action, not on the sequence of events that preceded it.
  - This is known as the "memoryless" property.
- **Policy (π)**:
  - A strategy or rule that defines the action to be taken in each state.
  - Example: A policy that always chooses the action that leads to the highest reward.

1. **States (S):**
   - The set of all possible states in which an agent can exist.
   - Example: Different rooms in a building.
2. **Actions (A):**
   - The set of all possible actions an agent can take.
   - Example: Moving from one room to another.
3. **Transition Function (P):**
   - A probability function $P(s'|s, a)$ that defines the probability of transitioning to state s' from state s when action a is taken.
   - Example: The probability of successfully moving from one room to another.
4. **Reward Function (R):**
   - A function $R(s, a, s')$ that provides the immediate reward received after transitioning from state s to state s' due to action a.
   - Example: Receiving a reward for successfully reaching a target room.
5. **Discount Factor (γ):**

Challenges of Reinforcement Learning

- **Reward Design**:
  - Designing appropriate rewards and determining their value can be complex. Example: In many games, setting suitable rewards is a significant challenge.
- **Absence of a Model**:
  - Some environments lack a fixed structure or rules, requiring simulations to gather experience. Example: Unlike chess, many games need simulations because they have no underlying model.
- **Partial Observability of States**:
  - Not all states are fully observable, leading to uncertainty. Example: In weather forecasting, complete information about the state is often unavailable.
- **Complexity**:
  - Large board configurations and numerous possible actions increase complexity. Example: Games like Go have vast possibilities, making labeled data unavailable and increasing algorithmic complexity.
- **Time-Consuming Operations**:
  - The need to explore extensive state spaces and possible actions increases the time required for learning. Example: Complex scenarios result in longer computational times, making the training process slow.



Criteria	Supervised Learning	Reinforcement Learning	Unsupervised Learning
Mapping	Present	Present	Not present
Feedback	Instantaneous feedback once the model is created	Constant feedback from environment	No feedback from environment
Supervisor	Presence of supervisor and labeled data	No supervisor and labeled dataset is not available	No supervisor
Decisions	Independent and based on training input	Dependent and made sequentially	Independent
Examples	Classifiers (e.g., image classification)	Chess, Go Games, Robotics	Clustering (e.g., customer segmentation)

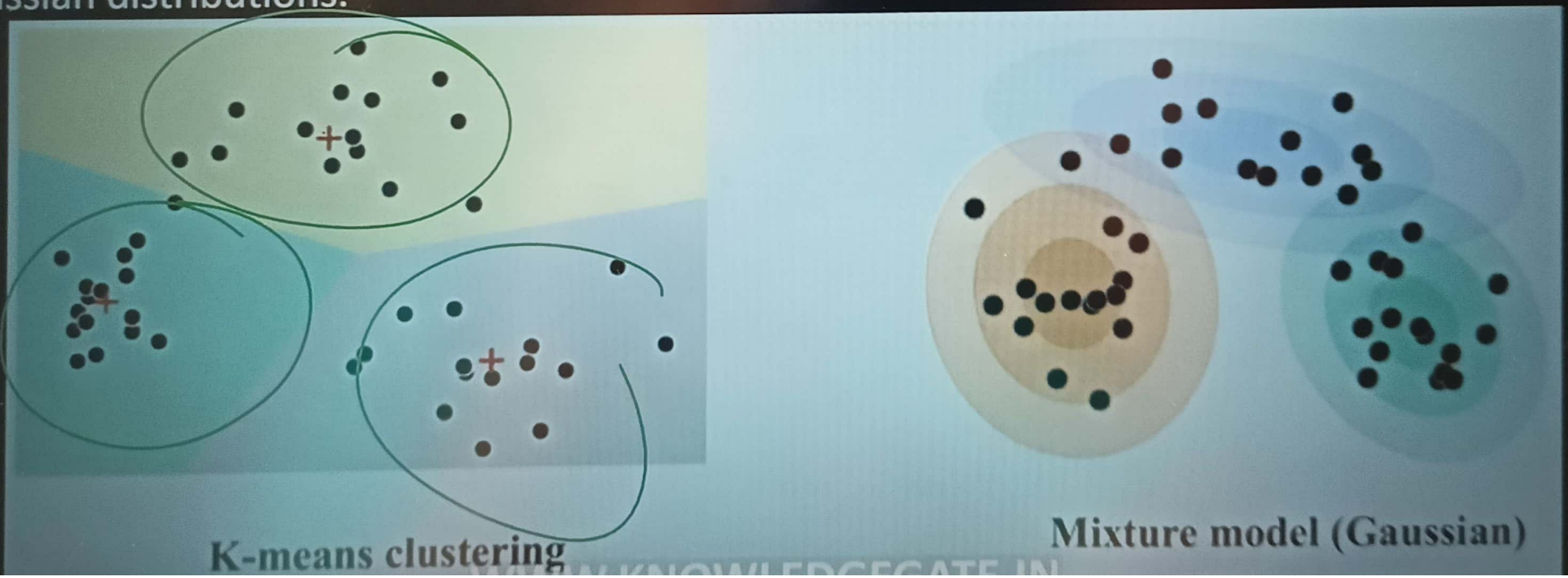
Linear SVM Vs Non-Liner SVM

Conclusion Point	Linear SVM	Non-Linear SVM
Data Suitability	Best for linearly separable data	Suitable for complex, non-linearly separable data
Decision Boundary	Straight line or hyperplane	Non-linear boundary using kernel functions
Training Complexity	Simpler and faster	More computationally intensive
Kernel Usage	Not required	Requires kernel functions (e.g., RBF, polynomial)
Flexibility and Applications	Less flexible, used for simpler problems	Highly flexible, used for complex data patterns

Aspect	Linear Regression	Logistic Regression
Type of Model	Supervised regression model	Supervised classification model
Prediction Outcome	Predicts continuous values	Predicts binary outcomes (0 or 1)
Mathematical Model	Uses linear functions	Uses logistic functions with an activation function
Purpose	Estimates values of a dependent variable	Estimates probability of an event
Example	Predicting house prices based on size	Predicting whether a patient has a disease or not

Partitional Clustering:

- **Centroid-based Clustering (e.g., K-means)**: Partitions data into clusters, each represented by a centroid. Clusters minimize distance between data points and centroid, optimizing intra-cluster similarity and inter-cluster dissimilarity. For example, retail customers can be clustered by buying patterns, with each cluster's centroid reflecting average behavior.
- **Model-based Clustering**: Uses a statistical model for each cluster, finding the best data fit. For instance, Gaussian mixture models assume data points in each cluster are Gaussian distributed. This method is used in image processing to model different textures as coming from different Gaussian distributions.



Example 3·75. *A coin is tossed $(m + n)$ times, $(m > n)$. Show that the probability of at least m consecutive heads is $\frac{n+2}{2^{m+1}}$.*

Solution. Since $m > n$, only one sequence of m consecutive heads is possible. This sequence may start either with the first toss or second toss or third toss, and so on, the last one will be starting with $(n + 1)th$ toss.

Let E_i denote the event that the sequence of m consecutive heads starts with ith toss. Then the required probability is : $P(E_1) + P(E_2) + \ldots + P(E_{n+1})$. ... (*)

$P(E_1) = P$ [Consecutive heads in first m tosses and head or tail in the rest] $= \left(\frac{1}{2}\right)^m$

$P(E_2) = P$ [Tail in the first toss, followed by m consecutive heads and head or tail in the next] $= \frac{1}{2}\left(\frac{1}{2}\right)^m = \frac{1}{2^{m+1}}$

In general,

$P(E_r) = P$ [tail in the $(r-1)$th trial followed by m consecutive heads and head or tail in the next]

$= \frac{1}{2}\left(\frac{1}{2}\right)^m = \frac{1}{2^{m+1}}, \forall\, r = 2, 3, \ldots, n+1.$

Substituting in (*), required probability $= \frac{1}{2^m} + \frac{n}{2^{m+1}} = \frac{2+n}{2^{m+1}}$.

Central Limit Theorem

Theorem :- If $\bar{X}$ is the mean of a random sample of size n taken from a population with mean μ and variance σ^2, then the sampling distribution of the sample mean tends to be normal distribution with mean μ and variance $\frac{\sigma^2}{n}$ as the sample size tends to be large ($n \geq 30$), regardless the form of the parent population,

i.e $\bar{X} \sim N(\mu, \frac{\sigma^2}{n})$.

Note : ① In term of ~~standar~~ standard normal variate, we can write it as

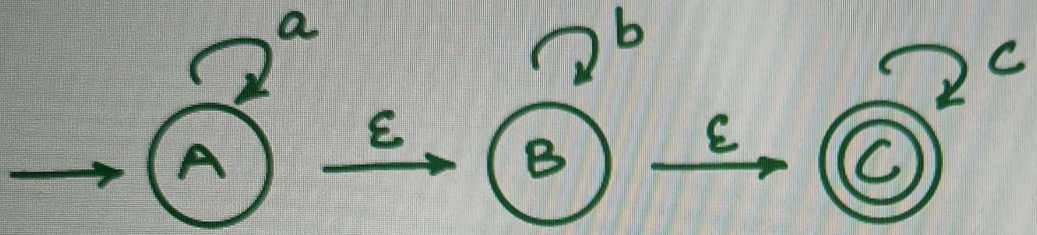
$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \sim N(0,1).$$

② In case of $n < 30$, the approximation is good only if the population is not too different from a normal distribution.

③ If the population is to be normal, then the sampling distribution of $\bar{X}$ is normal for any size of n.

④ CLT is really useful because it characterizes large samples from any distribution

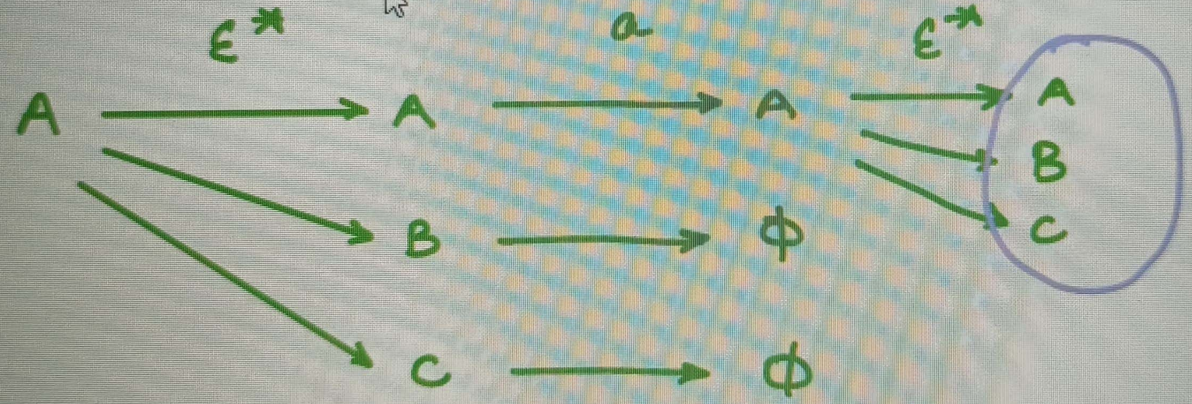
eg:



$\Sigma = \{a, b, c\}$

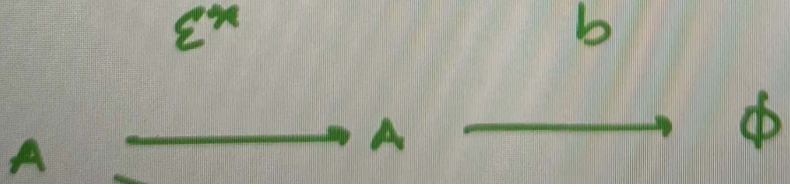
ε-closure(A) = AB
ε-closure(B) = B
ε-closure(C) = C

A, a



A, b

ε*



ε*

B

Machine learning — gradient descent — optimization algo used to train model. by minimizing error b/w predicted & actual value.

one hot encoding → representation of categorical values as numerical values.

Regression Tree → feature prediction from another feature.

→ **Precision** & **Recall** (phone)

	actual	
predicted	TP	FP
	FN	TN

$F_1 = \frac{2}{1/p + 1/r}$

Precision → all predicted +ve, how many TP

Recall → all actual +ve, how many TP.

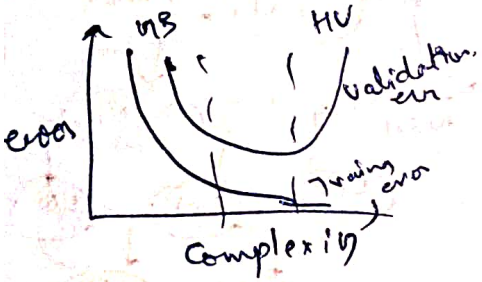
Covariance

$$C(x,y) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

+ve, -ve, 0

direction of linear relations b/w two variables

High bias underfit



Bias — measure of how close the predictions match actual values

Variance → error due small noise fluctuations. How much R_V is diff from E_{DOP} pop generalized

Bias ↑ model complex

Variance ↓ model complex ↑ training data

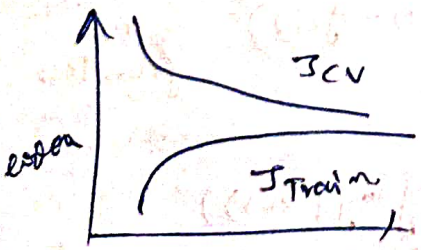
(Regularization)

Bias-Variance Trade off

$Bias \propto \frac{1}{Variance}$

Compute J_{cv} for all and choose min J_{CV} one.

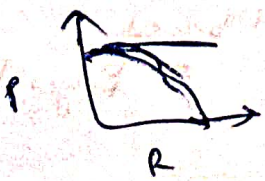
60 2020



$$Recall = \frac{\#TP}{\#\text{ actual +ve}}$$

$$Precision = \frac{\#Top.}{\#\text{ predicted +ve.}}$$

High P, R, minimize (FP, FP)



TPR = Recall

FPR = ~~Precision~~ (X)

Entropy → degree of purity. $H(X) = -\sum_{i=1} P(x_i) \log(x_i)$ randomness

IG → information gain → measures changes in entropy due to other feature

$IG(X) = H(X) - H(X|$ feature A

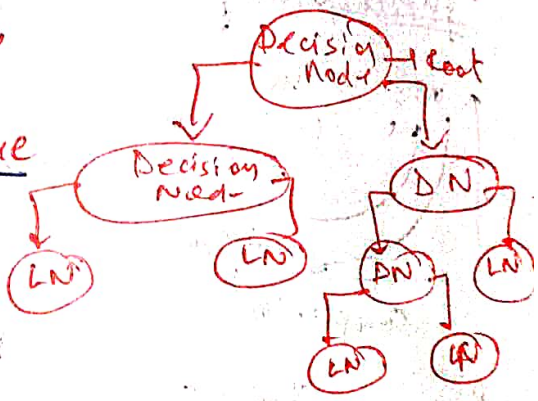
Mutual information $MI = H(X) + H(Y) - H(X,Y)$

information shared b/w X & Y.

Decision Trees

① SLT. for classification. Tree structured classifier, internal nodes represents features, branch represents decision rules & leaf nodes represent the outcome.

Decision & LeafNode.



CART algo → Classification & Reg. Tree Alg.

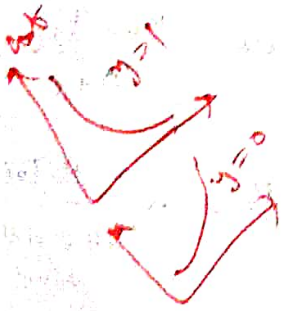
Attribution Selection Measure

Information Gain

→ Tree structure splitting,

② pruning → remove nodes which add less value.

Gini index → probability of misclassification in CART (impurity)



convex fun ← $J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}(i) - y(i))^2$ cost function

convergence

Cost for logistic reg → $-y \log(h(x)) - (1-y) \log(1-h(x))$

Naive Bayes → based on Bayes Theorem.

classification

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

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

K-nearest neighbour

① find K nearest points and find its class/ category but eucledian distance

Identify category of KNN.

for Regression: find avg.

Limitations

→ not for huge dataset
→ outliers sensitive
→ sensitive to missing value.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$F_1 = (1+\beta^2)\frac{P \cdot R}{\beta^2(P+R)}$$

DECISION TREE

non parametric

→ $FP \gg\gg FN$, $\beta = 1/2$

→ $FN \gg FP$, $\beta = 2$

→ FP, FN $\beta = 1$

Information Gain = $E(\text{Total}) - \sum \frac{n}{N} Ent(E_i)$

① Find entry & information
② gain: Pick maximum or
③ Do function, make Tree.

impurity pure split → only one either Y/N.

IG → $\sum p_i \log p_i$ GI → $1 - \sum p_i^2$ ← [0, 0.5]

$n(S) \to (0,1)$

No missing data, may have overfitting.

Naive Bayes

→ all independent
→ are equally imp.

email & spam → text meaning → class pred.

Pruning

remove nodes

PrePruning → Early stopping → faster

Post Pruning → pruning after full growth → Better Generalized → no overfitting

PrePruning
→ maximum tree depth
→ minimum threshold.
→ node becomes pure

① ID3 used entropy & IG

stopping criteria

Tree Pruning

naive bayes
probabilistic classifier based on Bayes Thm.

Linear Discriminant Analysis (LDA)

→ dimensionality reduction technique

→ high dimensional space to lower Dim. one. projection of feat

Assumptions

→ Gaussian Distribution → Covariance are equal of Matrices

→ linearly Separable

- Maximize Distance b/w means
- Minimize Variation ~~b/w~~ class

Methods to avoid Overfitting in Decision Trees:

1. Pruning Techniques → remove parts not necessary. Simplify.
2. Limit tree depth → reduce complexity
3. Cross validation → better generalization.
4. Feature selection → remove noisy ones, select approp

Find a line for best fit for set of points (Least Square Techniques)

$a^n . b^m . c^n \mid n, m \geq 1$ alphabet, stack top

$S \rightarrow aRc \mid aSc$ . . . remove