EE656A

Artificial Intelligence, Machine Learning and Deep Learning

Dr. Nishchal Verma

Aman

AI, ML & DL (Lecture #0 Introductory Class)

- AI is branch of CS that aims to create intelligent machines capable of mimicing human intelligence while performing tasks.
- The ultimate goal of AI is to develop systems that can learn, reason and solve problems in a manner similar to humans.

striving for automation → robots →
can robot know do you need water? machine learns by experience
AI can never surpass humans they say, ∵ it is man-made.
ultimate aim → learn by experience/training. and then make decisions.

- AI applications are widespread and impact our day to day life.
- AI now-a-days assist various industries, including healthcare, education and more.
- AI technology advances, the field of AI continues to evolve, raising ethical considerations and possibilities for groundbreaking innovations.

Course Description

theoretical advancements in AI
ML
DL } real life applications
best suited for PG students of all depts.

course pre-req: - EE 658 Fuzzy sets, systems and applications (pref.)

course content:
- AI : history, intro
- Agents of AI
- Fuzzy systems (FS), ANN, EC, GA, SA, PSO, etc
- ML
- clustering Biclustering
- classifications
- curve fitting
- Performance Measurement
- DL
- CV, etc.

TA: · mohd. Aquib (aquib@)
· seetaram ( )

Schedule: Tues 12 to 13:15 } TB212
Wed 12 to 13:15
lab → Tues 14 to 17
venue: NA — do on own system
assignments – submit around 5PM.
↳ getting just before lab begins.

Evaluation:-

class performance (Attendance/surp. quiz/assignment)	10%
midsem	20%
course project (journal/research/term paper analysis, implemen. sim. results)	30%
endsem	40%

groups formed. (maybe) depends. individual also. after midsem

Books:- (class discussions suff.)
1. Russell, Norvig AI: modern approach (3rd ed.)
2. Pattern Classification Richard O Duda.
some journals as well, sota, etc . . .
coding background: python preferred.

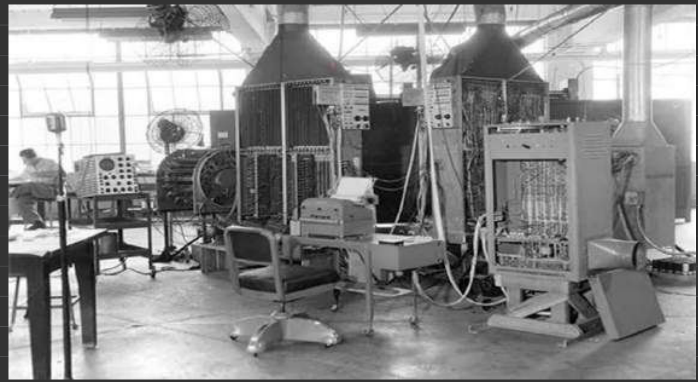
LECTURE #1 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Intelligence
- The capacity to learn for problem solving, decision making, etc
- Faster Learning ⟺ Better Intelligence
- Natural Learning in living beings (humans, animals, etc)
  The grade of learning differs. It comes from experience & brain.

Artificial Intelligence (Machine Intelligence
Artificially created capacity to make non-living beings / machines learn how to mimic the human intelligence.

The major advantages of AI:
1. Machines do not require sleep or breaks, and are able to function without stopping with same efficiency.
2. Machines can continuously perform the same task without getting bored or tired.
3. Machines are needed to carry out dangerous tasks where the human health and safety are at risk.

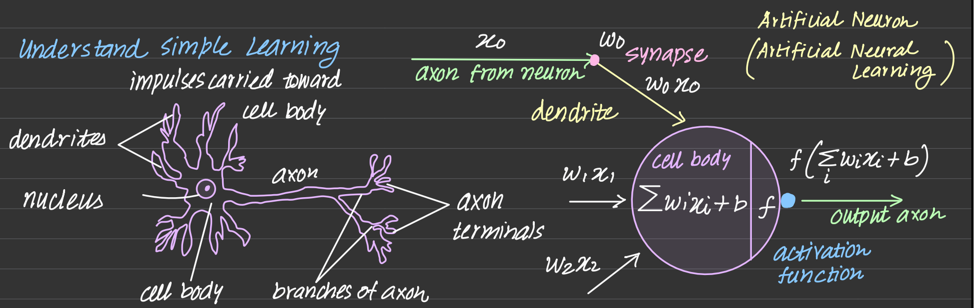
Alan Turing's Machine in 1937 (Universal Computing Machine)



Some learning capacity was introduced in machines through some logic.
↓
By Alan Turing

Philosophically AI was described way back many centuries ago.
But in engineering sense this computing machine was landmark.

Understand simple Learning



- An artificial neuron contains a non-linear activation function and has several incoming and outgoing weighted connections.
- 1942/43 — Warren McCulloch & Walter Pitts created a computational model for neural networks.

Artificial Intelligence

- John McCarthy coined the word Artificial Intelligence (1955)
- Lisp functional language is the first practical and still widely used AI programming language developed by John McCarthy in late 1950s.
  (Lisp and Prolog) ∼ AI

AI in History

1936-37	Allen's universal Turing machine was proposed
1942/43	Warren McCulloch & Warren Pitts created a computational model for neural networks called threshold logic.
1950	Turing Test was proposed
1955	John McCarthy has coined the term Artificial Intelligence
1957	Perceptron model was introduced
1960s	Genetic Algorithm
1965	Fuzzy Logic / Deep Learning* ∼ coined by Prof. Rina Dechter
1970s	Evolutionary Computing
1980s	Neural Computing, Swarm Intelligence

1990s Hybrid models; Neuro Fuzzy Systems; Neuro Fuzzy Genetic, etc.

Beyond 1990s Research Areas / Domains (statistical learning)
- Adaptive systems (AS)
- Evolutionary Computing (EC)
- Data Mining (DM)
- Simulated Annealing (SA)
- Particle Swarm Optimization (PSO)
- Deep Neural Networks (DNN)
- Deep Fuzzy Networks (DFN)

Discussion

- Experience → Data → brain is trained with data.
- Now with that data even with new scenarios you can make right decisions.
- Intelligence comes from learning. Learning comes from experience.
- Our goal is to go one step ahead ⇝ looking at the past data robots are getting trained.
- Siri, Alexa ⇝ they are also taking data and learning by past data. Remind you when you don't do usual tasks, etc.
- How to train and make any machine learn something? "Correspondence" i.e. mapping inputs/things with outcomes.
- How you trusted people and made friends? You throw some data/stimulus and saw the response of it. If it is in acceptable range then you are good to go and with time develop trust in people.

Input ⟶ Output
(Stimulus) (Response)
CORRESPONDENCE

More input & output (events)
better decision making

Artificial (↗ manmade) Intelligence (↗ you know)

- We create a abstract model.
- Develop this model in machine learning.
- Basis of model can be anything (neural network, fuzzy network, etc) ⟶ It can be anything that is learnable.
- This intelligence developed is limited and does limited job.
- Case Based system is not intelligence.
- Non-match based is intelligence. "Reflexes"

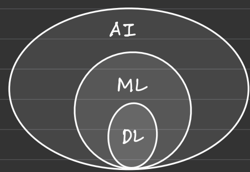
Supervised learning — you have correspondence

Unsupervised learning — you don't have correspondence. No outcomes known

Semi-supervised learning — partially correspondence some outcomes known, some not known

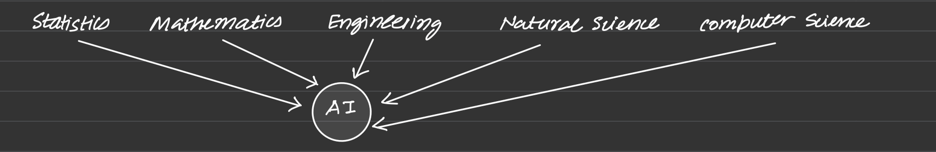
Computational Intelligence

- CI is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which mathematical or traditional modelling can be useless for a few reasons: the processes might be too complex for mathematical reasoning, it might contain some uncertainties during the process, or process simply be stochastic in nature.
- Major constituents of CI are fuzzy systems, neural networks, evolutionary algorithms and hybrid intelligence systems.
- AI contains CI (CI ⊆ AI)



- Deep learning ~ use only when needed. Otherwise it will give adverse effect.
- Rule based systems input ⟶ ☐ ⟶ output
- Classified machine learning

AI: Fields of study



- CV ↓ Facial Recognition
- NLP ↓ ChatBOT
- Information Retrieval ↓ Search Engine
- Information Filtering ↓ Shopping
- Predictive Analytics ↓ Credit Score
- Decision Analysis ↓ Industry
- Robotics ↓ Robot

Areas of AI :→

- Fuzzy Systems
- Artificial Neural Systems
- Evolutionary
- Probabilistic Systems
  - > Bayesian Networks
  - > Gaussian Mixture Models
- Chaos Theory
- Simulated Annealing
- Rough Set Theory
- Support Vector Machines

- John McCarthy coined the word Artificial Intelligence (1955)

Some Applications of AI

- Condition Based Maintenance and remaining useful life prediction of Machines like military ground vehicles
- CV: Object Recognition, identification, counting, Tracking and survelliance
- Future Image Generation
- Systems Model Development for Prediction or forecasting
- Network Enabled Manufacturing
- Border Patrolling
- Bomb Disposal
- Rescue Operations
- Cyber security
- Natural Language Processing (NLP)
- Speech Processing
- Supply chain management
- Medics: maintaining electronic medical records for identification of critical health problems

Condition Based Monitoring (IITK & BOEING COMPANY)

1. Sensitive Position → Data Acquisition → Preprocessing → Feature Extraction → Classification
   - * Pattern Analysis framework with Graphical Indices for Condition based monitoring.
   - * Intelligent Cond. Based Monitor. for turbines, compressors and other rotating machines.

   AI used to predict the condition based monitoring of machines. Know and rectify the problems beforehand.
2. Condition-based maintenance and remaining useful life prediction of Machines
3. CV Yolo for object detection and recognition
4. CV outdoor environment recognition

- Use AI only where needed, not unneccesarily using it.
  ↳ Not told anywhere!

Challenges

- Data to support the training of the machines.
- High performance computing for real-time response.

Better training → Better learning → Better results.

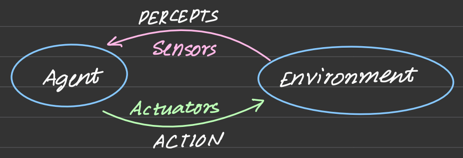
LECTURE # 2 AGENTS OF ARTIFICIAL INTELLIGENCE / INTELLIGENT AGENTS

Russell Book on AI (3rd ed.)
Ch 2 Intelligent Agents

Artificial Intelligent Agents

1. Simple Reflex agents
2. Model-based Reflex agents
3. Goal-based agents
4. Utility-based agents
5. Learning agents

AI Agent



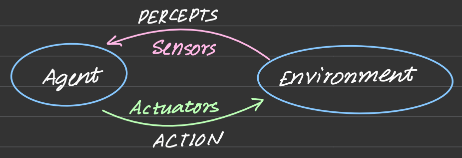
- Input - sensing / perceiving
- Output - action

actuators also called effectors

- Sensors — convert physical phenomenon into usable electrical signals.
- Transducers — convert one physical phenomenon into another (often electrical signal)
- Actuators — opposite — convert electrical signal into physical phenomenon.

Agent

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- A robot agent might have cameras, infrared range, temperature, etc for sensors and various motors for actuators.
- A Software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

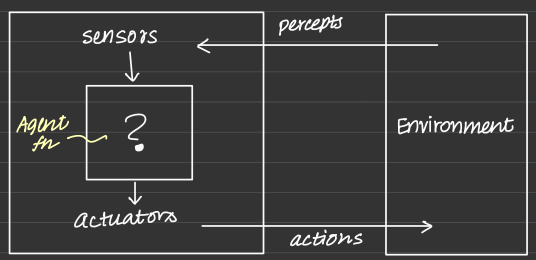


Huski Robot — EE IITK students working on it.

- We use the term percept (P) to refer to the agent's perceptual inputs at any given instant.
- An agent's percept sequence is the complete history of everything the agent has ever proposed.
- In general, an agent's choice of action (A) at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.
- Mathematically speaking, we say that an agent's behaviour is described by the agent function (f) that maps any given percept sequence to an action.

$$f : P \rightarrow A$$

percepts ↗ ↖ action



Architecture — hardware (sensor + actuator)

Agent Program — $f : P \rightarrow A$

Agent = Architecture + Agent Program

- Architecture is the machinery that the agent executes on. It is a device with sensors and actuators. for eg: a robotic car, camera, PC.
- Agent Program is the implementation of an agent function.
- An Agent function is a map from the percept sequence (history of all that an agent has perceived till date) to an action.

$$f : P \rightarrow A$$

percepts ↗ ↖ action

Rational Agent

- A rational agent is one that does the right thing i.e. conceptually speaking every job has been carried out correctly. obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?
- Rationality at any given time depends on four things :-
  1. The performance measure that defines the criterian of success
  2. The agent's prior knowledge of the environment.
  3. The actions that the agent can perform
  4. The agent's percept sequence to date.
- For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Agents & their performance measures, Environment, Actuators & Sensors

Agent	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, traffic, maximize profits	Roads, other pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
English tutor (Interactive)	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Full Observable vs Partially Observable Agents

- If an agent's sensors give it access to the complete state of the environment at each point in time then we say that the task environment is fully observable.
- A task environment is effectively fully observable if the sensors detect all the aspects that are relevant to the choice of action; relevance in turn depends on the performance measure.
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data — for eg a vacuum agent with only a local dirt sensor can't tell whether there is dirt in other squares, and an automated taxi can't see what other drivers are thinking.
- If the agent has no sensors at all then environment is unobservable.
- One might think that in such cases the agent's plight is hopeless, but as we discuss the agent's goal may still be achievable, sometimes with certainty.

- The job of AI is to design an agent program that implements the agent function i.e. the mapping from percepts (P) to actions (A).

$$\underbrace{f: P}_{\text{percepts}} \longrightarrow \underbrace{A}_{\text{actions}}$$

- We assume this program will run on some sort of computing device with physical sensors and actuators — we call this the architecture.

$$\text{agent} = \underbrace{\text{architecture}}_{\text{hardware}} + \underbrace{\text{program}}_{\text{software}}$$

There are four basic kinds of agent programs that embody the principles underlying almost all Artificial Intelligent systems:—

1. Simple Reflex agents
2. Model-based Reflex agents
3. Goal-based agents
4. Utility-based agents

- Each kind of the above program combines particular components in particular ways to generate actions.

- In general terms all these agents can be converted into Learning Agents that can improve the performance of their components so as to generate better actions.

LECTURE #3 AGENTS OF ARTIFICIAL INTELLIGENCE) INTELLIGENT AGENTS

Simple Reflex Agent

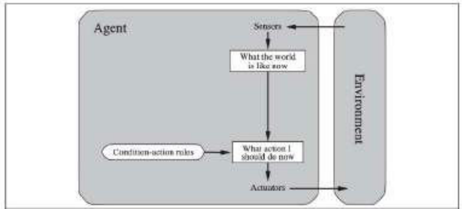


Figure 2.9 Schematic diagram of a simple reflex agent.

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

The advantages of Simple reflex agents are :
1.Very easy to implement
2.Computational complexity is minimal

Problems with Simple reflex agents are :
1.Very limited intelligence.
2.No knowledge of non-perceptual parts of the state.
3.Usually too big to generate and store.
4.If there occurs any change in the environment, then the collection of rules need to be updated.

The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.

Simple reflex behaviors can be understood as follows: Imagine yourself as the driver of the automated taxi. If the car in front brakes and its brake lights come on, then you should notice this and initiate braking.

In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking." Then, this triggers some established connection in the agent program to the action "initiate braking."

We call such a connection a condition-action rule, written as:
if car-in-front-is-braking then initiate-braking.
(Also called situation-action rules, productions, or if-then rules)

Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence.

The Simple reflex agent will work only if the correct decision can be made on the basis of only the current percept - that is, only if the environment is fully observable. Even a little bit of unobservability can cause serious trouble.

For example, the braking rule given earlier assumes that the condition car-in-front-is-braking can be determined from the current percept - a single frame of video.

A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all.

Model-based Reflex Agent

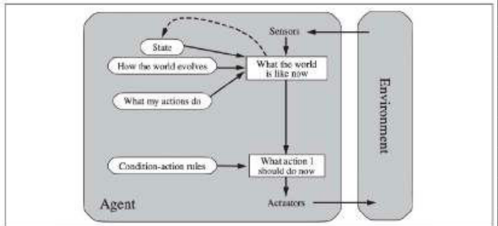


Figure 2.11 A model-based reflex agent.

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now. That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once. And for any driving to be possible at all, the agent needs to keep track of where the cars are.

First, we need some information about how the world evolves independently of the agent. For example, an overtaking car generally will be closer behind than it was a moment ago.

Second, we need some information about how the agent's own actions affect the world. For example, when the agent turns the steering wheel clockwise, the car turns to the right, or after driving for five minutes northbound on the freeway, one is usually about five miles north of where one was five minutes ago.

This knowledge about "how the world works"—whether implemented in simple Boolean logic or some other logic/scientific theories—is called a model of the world.

An agent that uses such a model is called a model-based agent.

Goal-based Agent

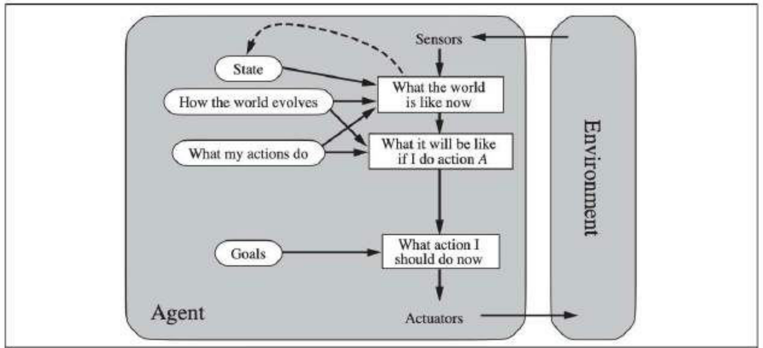


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Knowing something about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.

In other words, as well as a current state description, the agent needs some sort of goal information that describes desirable situations. For example, being at the passenger's destination.

The agent program can combine this goal information with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal.

Utility-based Agent

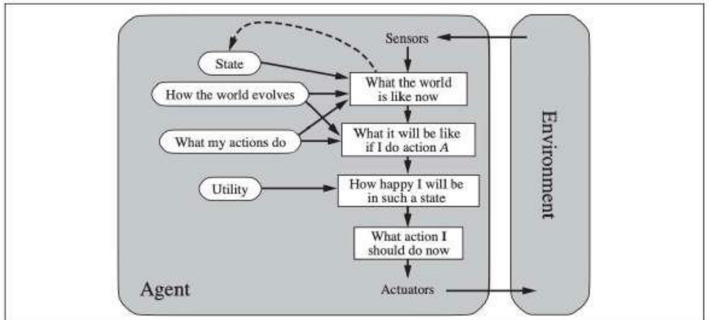


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

The word "utility" here refers to "the quality of being useful," not to the electric company or waterworks. Sometimes, goals alone are not enough to generate high-quality behavior in most environments. For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, safer, more reliable, or cheaper than others. Goals just provide a crude binary distinction between "happy" and "unhappy" states. A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent. Because "happy" does not sound very scientific, economists and computer scientists use the term utility instead.

Like goal-based agents, a utility-based agent has many advantages in terms of flexibility and learning. Furthermore, goals are inadequate, but a utility-based agent can still make rational decisions.

First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.

Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed against the importance of the goals.

Learning Agent

A learning agent can be divided into four conceptual components:

1. Performance Element
2. Critic Element
3. Learning Element
4. Problem Generator

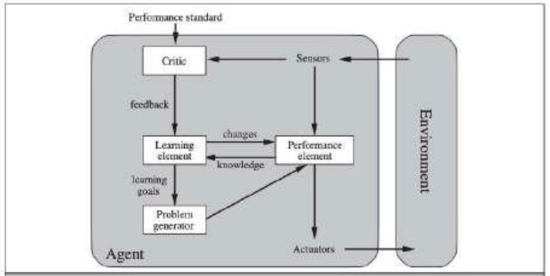


Figure 2.15 A general learning agent.

In many areas of AI, learning has become the preferred method for creating state-of-the-art systems.
Learning has another advantage, as we noted earlier: it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.

The key components of this system are the learning element, responsible for making improvements, and the performance element, responsible for selecting external actions.

The learning element utilizesss feedback from the critic on how the agent is performing and determines how the performance element should be modified to achieve better outcomes in the future.

The design of the learning element is heavily influenced by the design of the performance element. When attempting to design an agent that learns a specific capability, the initial question is not "How will I get it to learn this?" but rather "What kind of performance element will my agent require to execute this once it has learned how?"

With an agent design in mind, learning mechanisms can be developed to enhance each aspect of the agent.

The critic informs the learning element about the agent's performance relative to a predetermined performance standard. The critic's role is essential because the percepts alone do not offer any indication of the agent's success.

The problem generator serves as the final component of the learning agent. Its role is to propose actions that will result in novel and informative experiences.

The major points to recall are as follows:

1) An agent is something that perceives and acts in an environment. The agent function for an agent specifies the action taken by the agent in response to any percept sequence.

2) The performance measure evaluates the behavior of the agent in an environment. They can be fully or partially observable, single-agent or multiagentformance measure, given the percept sequence it has seen so far.

3) A task environment specification includes the performance measure, the external environment, the actuators, and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible. Task environments vary along several significant dimensions. They can be fully or partially observable, single-agent or multiagent, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.

4)The agent program implements the agent function. Various basic agent-program designs exist, reflecting the type of information made explicit and utilized in the decision process. These designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.

5)Simple reflex agents respond directly to percepts, while model-based reflex agents maintain internal state to track aspects of the world not evident in the current percept. Goal-based agents act to achieve their goals, and utility-based agents aim to maximize their own expected "happiness."

6)All agents have the capacity to enhance their performance through learning.

An agent is a learning agent if it improves it's performance on future tasks after making observations about the world.

Why would we want an agent to learn?

There are three main reasons:

1) First the AI designers cannot anticipate all possible situations that the agent might find itself in.

For example, a robot designed to navigate mazes (networks of paths and hedges designed as a puzzle through which one has to find a way) must learn the layout of each new maze it encounters.

2) Second, the AI Agent designers cannot anticipate all changes over time; a program designed to predict tomorrow's stock market prices must learn to adapt when conditions change from boom to bust.

3. Third, sometimes human programmers have no idea how to program a solution themselves.
For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

Any component of an AI agent can be improved by learning from data. The improvements, and the techniques used to make them, depend on four major factors:

1. Which component is to be improved.
2. What prior knowledge the agent already has.
3. What representation is used for the data and the component.
4. What feedback is available to learn from

LECTURE # 6&7 UNSUPERVISED LEARNING: K-MEANS CLUSTERING

K-Means Clustering

- The term K-means was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1956.
- The standard algorithm was first proposed by Stuart Lloyd of Bell Labs in 1957 as a teenique for pulse code modulation, though it wasn't published as a journal article until 1982.
- In 1965, Edward W Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy.
- K-means is one of the most popular "clustering" algorithms.
- K-means has K-centroids that it uses to define the number of clusters.
- A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid.
- K-Means find the best centroids by alternating between
  (1) assigning data points to clusters based on current centroids
  (2) compute centroids (points which are the centre of a cluster) based on the current assignment of data points to clusters.

K-Means Clustering : The Algorithm

Given the cluster number "K" for any data points as data set, the algorithm is carried out in four steps after initialisation:

STEP-1: Choose any K-centroids or cluster centres randomly.

STEP-2: Assign each data point to the nearest centroid or cluster centre. Use Euclidean or L_2 norm or square of Euclidean as distance metric

Given a set of data points $\{x_1, x_2, \ldots x_n\}$ in a dataset, where each pt. is a d-dimensional real vector, K-Means clustering aims to partition the n data points into K $(\leq n)$ clusters $C = \{C_1, C_2, \ldots, C_K\}$ so as to minimize the within-cluster sum of squares (WCSS) or Euclidean distance. Formally the objective function is to find

$$\text{Objective Function} = \underset{C}{\arg\min} \sum_{K=1}^{K} \sum_{i=1}^{n} \| x_i - c_K \|^2$$

where c_K = K^{th} cluster centre found by taking the average or mean of all the points assigned to c_K.

STEP-3: Find new K-centroids (i.e. the K number of means based on the data points assigned to respective clusters) as the cluster centres of the current partitions.

$$C_{new\,i} = \frac{1}{n} \sum_{x_j \in S_i} x_j$$ ↳ average of members of cluster

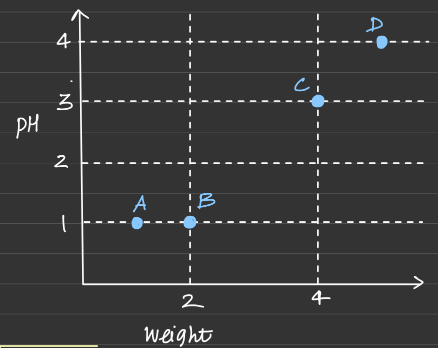
STEP-4: Go back to step-2, stop when centroids or cluster centres do not change or until convergence.

- In other words, the goal is to attain the smallest objective function.
- Does this depend on the initial seed value? YES!
- K-Means may not always converge at global minimum (optimal sol^n with lowest WCSS)
- K-Means converges to local minima and not neccesarily at the global minima (global optimum).
- Convergence depends on initial seed values.

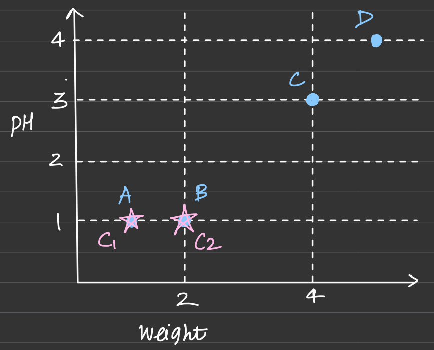
Example 1 (K-Means)

Problem: 4 types of medicines — each has two attributes (pH & weight index)
goal — group into 2 clusters (K=2)

Medicine	Weight_i	pH_i
A	1	1
B	2	1
C	4	3
D	5	4



Distance metrix
↳ Euclidean Distance $\sqrt{\Sigma |x_i|^2}$ — use this ✓
Manhattan Distance $\Sigma |x_i|$ } can use any.
l_p norm in general



Euclidean Distance

	A	B	C	D
X	1	2	4	5
Y	1	1	3	4

Step-1 Use initial seed points for positioning

$$\left.\begin{aligned} C_1 &= A = (1,1) \\ C_2 &= B = (2,1) \end{aligned}\right\} \text{(randomly)}$$

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}$$

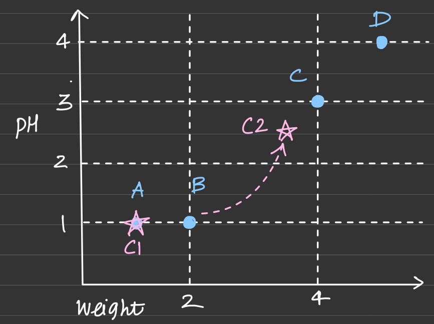
$C_1\ (1,1) \sim$ group 1
$C_2\ (2,1) \sim$ group 2

$$d_{A1} = \sqrt{(1-1)^2 + (1-1)^2} = 0$$
$$d_{C1} = \sqrt{(4-1)^2 + (3-1)^2} = \sqrt{13} = 3.61$$
$$d_{A2} = \sqrt{(2-1)^2 + (1-1)^2} = 1$$
$$d_{C2} = \sqrt{(4-2)^2 + (3-1)^2} = \sqrt{8} = 2.83$$
$$d_{B1} = \sqrt{(2-1)^2 + (1-1)^2} = 1$$
$$d_{D1} = \sqrt{(5-1)^2 + (4-1)^2} = \sqrt{25} = 5$$
$$d_{B2} = \sqrt{(2-2)^2 + (1-1)^2} = 0$$
$$d_{D2} = \sqrt{(5-2)^2 + (4-1)^2} = \sqrt{18} = 4.24$$

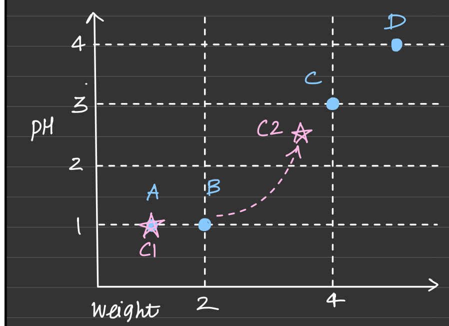
$A \rightarrow \min(0,1) = 0 \quad C_1$
$B \rightarrow \min(1,0) = 0 \quad C_2$
$C \rightarrow \min(3.61, 2.83) = 2.83 \quad C_2$
$D \rightarrow \min(5, 4.24) = 4.24 \quad C_2$

Assign each object to a cluster with the nearest seed point

Compute new centroids of the current partition
We know members of each cluster, we can find new centroid of each group by taking average of member values.



$$C_1 = (1,1)$$
$$C_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = \left(\frac{11}{3}, \frac{8}{3}\right) = (3.67, 2.67)$$



$C_1 = (1,1)$
$C_2 = (3.67, 2.67)$

- Renew membership based on new centroids
- compute distance of all objects to the new centroids

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \begin{matrix} c_1 \\ c_2 \end{matrix}$$

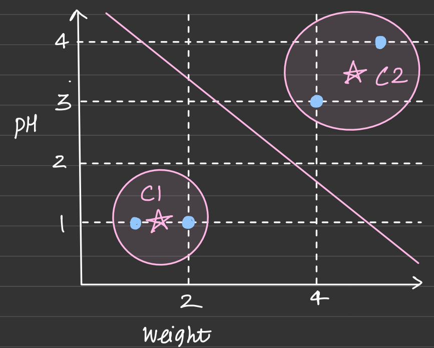
d_{i1} = same $\forall\ i = \{1,2,3,4\}$ since $c_1 = (1,1)$ is unchanged

$$d_{A2} = \sqrt{(3.67-1)^2 + (2.67-1)^2} = 3.14$$
$$d_{B2} = \sqrt{(3.67-2)^2 + (2.67-1)^2} = 2.36$$
$$d_{C2} = \sqrt{(3.67-4)^2 + (2.67-3)^2} = 0.47$$
$$d_{D2} = \sqrt{(3.67-5)^2 + (2.67-4)^2} = 1.89$$

$A \rightarrow \min(0, 3.14) = 0 \rightarrow C_1$
$B \rightarrow \min(1, 2.36) = 1 \rightarrow C_1$
$C \rightarrow \min(3.61, 0.47) = 0.47 \rightarrow C_2$
$D \rightarrow \min(5, 1.88) = 1.88 \rightarrow C_2$

New Centroids
$$C_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1)$$
$$C_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$$

New Distances
$$D_2 = \begin{bmatrix} 0.5 & 0.5 & 3.2 & 4.61 \\ 4.3 & 3.54 & 0.71 & 0.71 \end{bmatrix} \begin{matrix} c_1 \\ c_2 \end{matrix}$$
$\quad \hookrightarrow c_1 \quad \hookrightarrow c_1 \quad \hookrightarrow c_2 \quad \hookrightarrow c_2$



Stop due to no new assignment membership in each cluster no longer change.

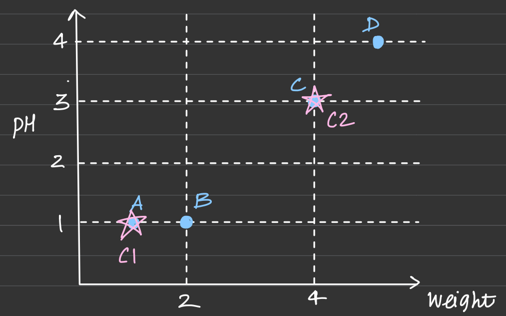
Example-2 (K-Means)

In previous problem, use $C_1 = A$, $C_2 = C$ } seed values.

Use K-means with Manhattan distance metric for clustering analysis by setting $K=2$.

1) How many steps required for convergence?
2) What are membership of two clusters after convergence?
3) What are centroids of two clusters after convergence?

Medicine	$weight_i$	pH_i
A	1	1
B	2	1
C	4	3
D	5	4



Norms

$f: \mathbb{R}^n \rightarrow \mathbb{R}$ is norm if
- $f(x) \geq 0 \ \forall x \in \mathbb{R}^n$ non negative
- $f(x) = 0 \Leftrightarrow x = 0$ definite
- $f(tx) = |t| f(x) \ \forall x \in \mathbb{R}^n, t \in \mathbb{R}$ homogenous
- $f(x+y) \leq f(x) + f(y)$ triangular inequality

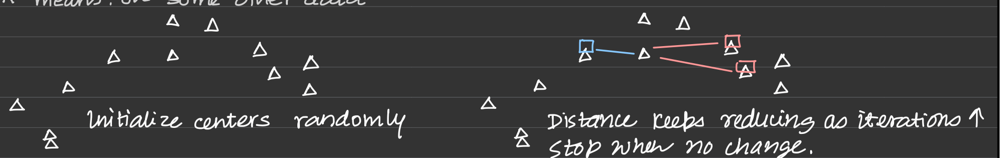
$\|x\|$ — general

$\|x\|_{symb}$ — specific norm

Norms facilitates length and distance measurements.

- $\|x\|_2$: Euclidean norm (l_2 norm)
- $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$ (l_1 norm or Manhattan distance)
- $\|x\|_p = \left[|x_1|^p + |x_2|^p + \cdots + |x_n|^p \right]^{1/p}$ (l_p norm)
- $\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$ (l_∞ norm)

K-means: on some other data

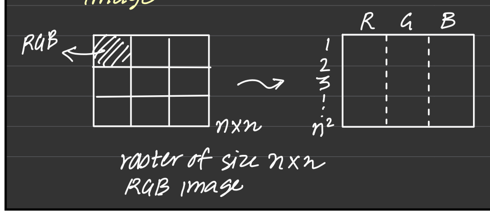


Iterate
- Assign / cluster each example to closest center.
- Readjust. → cluster centers changing → keep doing until no change
- Euclidean distance — distance metric unless stated (can be any norm)
- Stop when no further shift of centroid.

K-means clustering is not global optimum (local optimum)

K-means — randomly choose centroids — partitioning data randomly
— clustering centers are based on seperation.

Image



n^2 entries with RGB values

total values = $n \times n \times 3 = 3n^2$ values

$\underbrace{}_{(R,G,B)}$

K-Means Clustering

Strength

- Relatively efficient : $O(i \cdot k \cdot n \cdot d)$ → computational complexity Big-O-Notation

  n = no. of objects or data points
  k = no. of clusters
  d = no. of features
  i = no. of iterations

Normally, $k, i \ll n$

Often terminates at a local optimum.

The global optimum may be found using tecniques such as: deterministic annealing and genetic algorithms

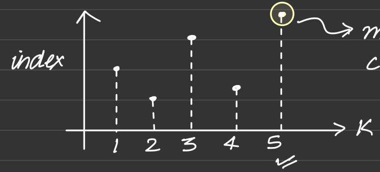
Weakness

- Need to specify K = no. of clusters in advance.
- Sensitive to initial seed points (i.e. random cluster centers).
- Unable to handle noisy data and outliers.
- Not suitable to discover clusters with non-convex shapes.

How to choose optimum value of K ? (V. Imp)

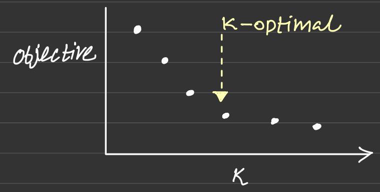
↳ a lot of research done !

- We have "quality indices" — measures of the clusters to access its quality or check efficiency. (eg. Silhouette index, Dunn index, etc) We use them to find optimal K.



max. value of quality index
choose corresponding K-value as optimal

- One other commonly used approach is elbow method.



Objective when K = n ?

~ Not for exam!

Applications of K-Means

Colour-Based Image Segmentation using K-Means

Step 1 : Load colour image of tissue stained with homotoxin & eosin (H&E)
~ Finding black doesn't mean exact black ~ similar ~ segment

Step 2 : Convert RGB color space ⟶ $L^* a^* b^*$ color space
- $L^* a^* b^* \equiv$
- $L^* a^* b^*$ is designed to approximate human vision unlike RGB.
- complicated transformation b/w RGB & $L^* a^* b^*$.

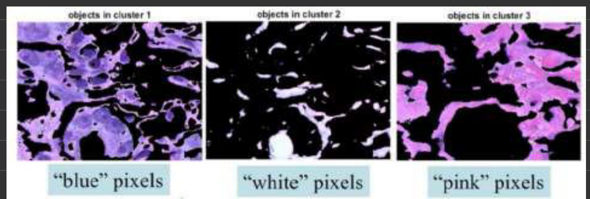
$(L^*, a^*, b^*) = T(R, G, B)$ $\quad (R, G, B) = T'(L^*, a^*, b^*)$

Step 3 : K-Means clustering with K = 3.

Step 4 : Label every pixel in image using K-means clustering result (three diff. grey levels)

Aim : to segment the image based on colors.
Segment to roughly / approx. best match colors and not exactly.

Step 5 : Create images that segment H&E image by color
Apply label and color info of each pixel to seperate color images corresponding to three clusters.



Some Relevant Data Clustering works that you may try :

[1] Nishchal K. Verma and M. Hanmandlu, Color segmentation via. Improved Mountain Clustering Technique, International Journal of Image and Graphics, vol.7, no. 2, pp. 407-426, Apr. 2007.

[2] Nishchal K. Verma and A. Roy, Self-Optimal Clustering Technique Using Optimized Threshold Function, IEEE Systems Journal, vol. 99, pp. 1-14, Jul. 2013.

↳ SOC → search google → find & implement using codes on site.

Summary (K-Means)

- K-Means algorithm is simple yet popular method for clustering analysis.
- It performance is determined by initialisation and appropriate distance measure.
- There are several variants of K-Means to overcome its weaknesses
  - — K-Medoids : resistance to noise and outliers
  - — K-Modes : extension to categorical data clustering analysis
  - — CLARA : extension to deal with large datasets
  - — Mixture models (EM algorithm) : handling uncertainty of clusters

LECTURE # 8 FUZZY C-MEANS CLUSTERING

Fuzzy C-Means (FCM)

- Fuzzy C-Means (FCM) clustering is an extension of the K-Means clustering developed by J.C. Dunn in 1973 and improved by J.C. Bezdek in 1981.
- FCM clustering allows data points to be assigned into more than one cluster.
- This algorithm works by assigning membership to each data point corresponding to each cluster on the basis of distance b/w the cluster centre and data point. Data near to the cluster center more is its membership for the particular cluster center. Clearly summation of membership of each data point should be equal to 1.

- To introduce this algorithm we define a sample set of n data point that we want to cluster

$$\overline{X} = \{\overline{x}_1, \overline{x}_2, \overline{x}_3, \ldots \overline{x}_n\}$$

each data point x_i is defined by d features i.e.

$$\overline{x}_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iD}\}$$ where D = no. of features

Dataset $X \rightarrow$; Features $\rightarrow D$

	d_1	d_2	d_3	d_j	D
x_1	~	~	.	. . .	~
x_2					
x_3					
.					
.					
x_i	x_{i1}	x_{i2}	. . . . . .	x_{ij} . .	x_{iD}
.					
.					
x_n					

→ bold (vector) multiple data pts.

Datapoints

$n \times D$ matrix

data points ; features

- Define a family of fuzzy sets A_j where $j = 1, 2, \ldots, c$ is a fuzzy c-partitions in the universe of data points $\overline{X}$.
  c_j is the d-dimensional center of the j^{th} cluster.
- Assign a membership degree to various data points $(\overline{x})$ in each fuzzy set (fuzzy class). Hence, a single data point can have partial membership degree in more than one cluster.
  for eg:- the i^{th} data point in the j^{th} cluster have membership degree $\mu_{ij} \in [0,1]$
- The condition is that the sum of all the membership degrees for a single data point in all the clusters has to be unity (1).
  i.e.

$$\sum_{j=1}^{n} \mu_{ij} = 1 \quad \forall\ i = \{1, 2, 3, \ldots n\}$$

total no. of data points

- Define a fuzzy c-partition matrix $\overline{U}$ for grouping a collection of n data points into c-clusters. The objective function J for a fuzzy c-partitions is given as

$$J(U, c) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m (d_{ij})^2$$

Objective Function

weights introduced to k-means

$$d_{ij} = d(x_j - c_j) = \| (x_i - c_j) \| = \left[\sum_{d=1}^{D} (x_{id} - c_{jd})^2 \right]^{1/2}$$

μ_{ij} = membership of the i^{th} data point in the j^{th} cluster
d_{ij} = Euclidean distance b/w j^{th} cluster center and i^{th} data point
x_{id} = d^{th} feature of the i^{th} data set
$m \in [1, \infty]$ weighing parameter controls the amount of fuzziness in the clustering process (usually $m = 2$)
c_j is the j^{th} cluster center described by D features is represented in the vector form

$$c_j = \{c_{j1}, c_{j2}, \ldots, c_{jD}\}$$

Each cluster coordinates for every cluster can be calculated as follows

$$c_{jd} = \frac{\sum_{i=1}^{n} (\mu_{ij})^m x_{id}}{\sum_{i=1}^{n} (\mu_{ij})^m}$$

where d is a variable on the feature space i.e. $d = 1, 2, 3, \ldots D$
Optimum fuzzy c-partitions will be obtained by

$$J^*(U^*, c) = \min(J(U, c))$$

Fuzzy C-Means Clustering Algorithm

STEP-1 : Initialize $U = [\mu_{ij}]$ matrix, $U^{(0)}$ randomly.
(Initial membership matrix)

STEP-2 : Calculate the cluster centers c_j for $j = 1, 2, \ldots c$ with $U^{(0)}$

$$\tilde{c}_j = \frac{\sum_{i=1}^{n} (\mu_{ij})^m \tilde{x}_i}{\sum_{i=1}^{n} (\mu_{ij})^m}$$

n = no. of data points
c = no. of clusters

Each cluster coordinates for every cluster can be calculated as follows:-

$$c_{jd} = \frac{\sum_{i=1}^{n} (\mu_{ij})^m x_{id}}{\sum_{i=1}^{n} (\mu_{ij})^m} \quad \text{(weighted average)}$$

where d is a variable on the feature space $d = \{1, 2, \ldots, D\}$

STEP-3 : Update $U^{(k)}, U^{(k+1)}$

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}} \quad \text{(proportion calculation)}$$

$$d_{ij} = d(x_i - c_j) = \| x_j - c_j \| = \left[\sum_{d=1}^{D} (x_{id} - x_{jd})^2 \right]^{1/2}$$

where $d = 1, 2, 3, \ldots D$

STEP-4 : If $\| U^{(k+1)} - U^{(k)} \| < \varepsilon$ then STOP.
Otherwise return to step 2.

- Use $m=2$ (fuzziness parameter) unless stated.
- More the value of m, better information, lesser fuzziness
- like K-means, FCM also reaches at local optimum and not the global optimum.

Example 1

2 3 4 5 6 7 8 9 10 11

initial centroids (3 & 11)

μ_{ij} = degree of membership of x_i in the cluster j.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}} \quad m=2$$

$$d_{ij} = \| x_i - c_j \| = \sum_{d=1}^{D} (x_{id} - x_{jd})^P$$

$$\mu_{11} = \frac{1}{\left(\frac{2-3}{2-3}\right)^2 + \left(\frac{2-3}{2-11}\right)^2} = 0.99 \qquad \mu_{12} = \frac{1}{\left(\frac{2-11}{2-3}\right)^2 + \left(\frac{2-11}{2-11}\right)^2} = 0.01$$

$$\mu_{21} = \frac{1}{\left(\frac{3-3}{3-3}\right)^2 + \left(\frac{3-3}{3-11}\right)^2} = 1 \qquad \mu_{22} = \frac{1}{\left(\frac{2-11}{2-3}\right)^2 + \left(\frac{2-11}{2-11}\right)^2} = 0$$

$$\vdots \qquad\qquad \vdots$$

$$\mu_{n1} = \frac{1}{\left(\frac{x-3}{x-3}\right)^2 + \left(\frac{x-3}{x-11}\right)^2} = \sim \qquad \mu_{n2} = \frac{1}{\left(\frac{x-11}{x-3}\right)^2 + \left(\frac{x-11}{x-11}\right)^2} = \sim$$

Initial Membership values $U^{(0)}$

	c1	c2	total (sum)
2	0.99	0.01	1.00
3	1.00	0.00	1.00
4	0.98	0.02	1.00
5	0.90	0.10	1.00
6	0.74	0.26	1.00
7	0.50	0.50	1.00
8	0.26	0.74	1.00
9	0.10	0.90	1.00
10	0.02	0.98	1.00
11	0.00	1.00	1.00

Updated cluster centers

$$c_j = \frac{\sum_{i=1}^{n} \mu_{ij}^{m} x_j}{\sum_{i=1}^{n} \mu_{ij}^{m}} \quad m=2$$

$$C_1 = \frac{0.99^2 \times 2 + 1^2 \times 3 + 0.98^2 \times 4 + 0.9^2 \times 5 + 0.74^2 \times 6 + 0.5^2 \times 7 + 0.26^2 \times 8 + 0.1^2 \times 9 + 0.02^2 \times 10 + 0^2 \times 11}{0.99^2 + 1^2 + 0.98^2 \times 0.9^2 + 0.74^2 + 0.5^2 + 0.26^2 + 0.1^2 + 0.02^2 + 0^2}$$

Similarly $C_2 = \frac{\sum \mu_j^2 \times x_j}{\sum \mu_j^2}$

	1	2	1_new	2_new
2	0.99	0.01	0.93	0.07
3	1.00	0.00	0.98	0.02
4	0.98	0.02	1.00	0.00
5	0.90	0.10	0.95	0.05
6	0.74	0.26	0.75	0.25
7	0.50	0.50	0.40	0.60
8	0.26	0.74	0.12	0.88
9	0.10	0.90	0.01	0.99
10	0.02	0.98	0.01	0.99
11	0.00	1.00	0.05	0.95
	c1	4.00	c1 new	3.98
	c2	9.46	c2 new	9.26

C1 & C2 at K=1   C1 & C2 at K=2

$c_1' = 4$

$c_2' = 9.46$

Example 2 : Given fuzzy clusters, convert into crisp clusters.

Fuzzy Clusters (Soft clusters)

$$\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ \hline C1: & 0.991 & 0.986 & 0.993 & 0 \\ C2: & 0.009 & 0.014 & 0.007 & 1 \end{array}$$

Data point with partial membership to multiple clusters

Crisp Clusters (Hard Clusters)

$$\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ \hline C1: & 1 & 1 & 1 & 0 \\ C2: & 0 & 0 & 0 & 1 \end{array}$$

Data points exclusively belong to one cluster

To convert fuzzy clusters into crisp clusters, assign each data point to the cluster with the highest membership value.

C1 : $\{x_1, x_2, x_3\} \rightsquigarrow \mu_{i1} = 1 \quad \mu_{i2} = 0$

C2: $\{x_4\} \rightsquigarrow \mu \rightsquigarrow \mu_{i1} = 0 \quad \mu_{i2} = 1$

Answer: Crisp cluster $C1 = \{x_1, x_2, x_3\}$

$C2 = \{x_4\}$

Example 3 : Apply FCM on D-Dimensional feature space data sets

$c = 2 \quad D = 4$

	x_1	x_2	x_3	x_4
d_1	1	5	2	1
d_2	0	5	0	0
d_3	4	9	9	0
d_4	5	9	0	2

$\rightsquigarrow$ transposed matrix

$X = \{\bar{x}_i\}$

$= \begin{Bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{n1} & x_{n2} & & x_{nd} \end{Bmatrix}$

Kept to confuse.

$x_1 = \{1, 0, 4, 5\}$

$x_2 = \{5, 5, 9, 9\}$

$x_3 = \{2, 0, 9, 0\}$

$x_4 = \{1, 0, 0, 2\}$

Step1: Initialisation matrix

$$U^{(0)} = \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \left[\quad \right] \begin{array}{c} c_1 \\ c_2 \end{array}$$

Assign randomly.

proceed.

In Bi-clustering, we take subset of the feature.

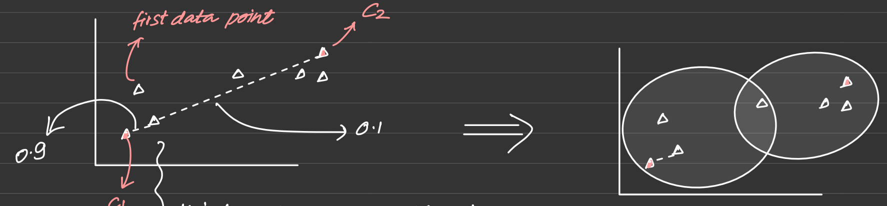
Types of questions in exam:

1. K-Means Numerical
2. FCM Numerical
   - initial cluster centers or initialisation matrix given (seed values fixed so as to get same final clusters)
   - after n^{th} iteration, what will be the cluster values using Fuzzy c Means algorithm?

same seed values
- initial centers or
- initialisation matrix U^0

Understand FCM well ~ solve examples to understand.

SOC Paper ~ implement the paper ~ take MATLAB code write the python code for this paper. submit!



this has less membership with C2 & more " " C1

CLUSTER VALIDITY INDICES

- Used to evaluate the quality of clustering results. Helpful to understand how well the data has been clustered.

1 Silhouette Index

- The silhouette coefficient measures how similar an object is to its own cluster compared to other clusters.
- Ranges from -1 to 1, where a high value indicate that the object is well matched to its own cluster and poorly matched to neighbouring clusters.
- The silhouette coefficient can be calculated for individual data points and then averaged over all data points to get a global measure.

$$\boxed{s = \frac{b-a}{\max(a,b)}}$$

Example: Dataset with 3 clusters
For each data point the silhouette coefficient is calculated by computing

a = average distance b/w the point and all other points in the same cluster

b = average distance b/w the point and all points in the nearest neighbouring cluster

- A higher silhouette coefficient indicates better clustering eg: $s=0.6$ indicates cluster assigned is apt for data pts.

- The silhouette index is a measure of how similar an object is to its own cluster compared to other clusters. It provides a way to access the quality of clustering results by measuring the cohesion within clusters and the seperation b/w clusters. A high silhouette index indicates object is well clustered while a low silhouette index indicates object may be better assigned to a different cluster.

Procedural steps on how to compute the silhouette index for clustering

1) **Compute cluster Assignments:** Begin by clustering your dataset using a chosen clustering algorithm, such as K-Means, FCM, hierarchical clustering or DBSCAN. Assign each point to its corresponding cluster.

2) **Compute cluster centroids (optional):** For some clustering algorithms, such as K-Means, FCM, compute the centroid of each cluster. This step is necessary if your clustering algorithm requires centroid based distances.

3) **Compute Average Distance to other points in the same cluster (a):** For each i^{th} data point, calculate the average distance (similarity) to all other data points within the same cluster. This distance measure can vary depending on the application, but commonly used distance metrics include Euclidean distance, Manhattan distance, or cosine similarity. Let's denote the value as a_i.

4) **Compute Average Distance to Data Points in Neighbouring clusters (b):** For each i^{th} data point, calculate the average distance to all data points in the nearest neighbouring cluster. This means excluding data points from the same cluster as i^{th}. Let's denote this value as b_i.

5) **Compute Silhouette Index for each data point:** For each data point i^{th}, compute the silhouette index using

$$\boxed{s_i = \frac{b_i - a_i}{\max(b_i, a_i)}}$$

- If $a_i \approx b_i \approx 0$, $s_i \approx 0$ indicating data point is on or very near the decision boundary between clusters.
- If $a_i << b_i$, $s_i \approx 1$ indicating data point is well clustered.
- If $b_i << a_i$, $s_i \approx -1$ indicating data point may be assigned to wrong cluster.

6) Compute overall silhouette index: once you have computed the silhouette index for each data point, calculate the average silhouette index across all data points to obtain the overall silhouette index for the clustering result.

The formula for average silhouette index s is:

$$\boxed{s = \frac{1}{N} \sum_{i=1}^{N} s_i}$$

where N = total no. of data points

Some other quality indices

② Davies – Bouldin Index

$$DB_i = \left(\frac{1}{n_i}\right) \sum_{j=1}^{K} \left[\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right]$$

inner cluster distance b/w c_i & c_j.

③ Calinski Harabasz Index (Varience Ratio criterion)

$$CH = \frac{\text{B/w cluster dispersion}}{\text{within cluster dispersion}} \times \frac{n-K}{K-1}$$

④ Dunn Index   compactness and seperation

$$D = \min_{1 \le i \le K} \left(\min_{j \ne i} \left(\frac{\text{Inter cluster distance}}{\text{Intra cluster distance}} \right) \right)$$

⑤ Gap Statistics
compares total intra-cluster variation for diff. values of K (no. of clusters) with their expected

⑥ Within – Cluster Sum of Squares (WCSS) also known as inertia

$$WCSS = \sum_{1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

lower WCSS indicates better clustering

⑦ Adjusted Rand Index (ARI) measures similarity b/w true label & . –

⑧ Fowlkes-Mallows Index (FMI)

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$$

LECTURE # 9 MACHINE LEARNING AND SUPERVISED LEARNING

So far unsupervised learning in artificial intelligence

- Finding clusters or group or category labels and the no. of clusters or groups or categories directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

Machine Learning

Process of developing or obtaining a model (AI Agent) by learning from data (i.e. examples, experiences, etc).

Normally any ML-based model is obtained by

1. Learning its parameters (supervised learning, semi-supervised learning)
2. Learning structure (eg. no. of hidden layers of ANN, no. and type of rules for fuzzy models, graphs, etc) (supervised learning, semi-supervised learning)
3. Learning hidden concepts based on certain attributes (e.g. clustering, biclustering, etc. in unsupervised learning)

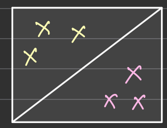
Types of Machine Learning

- Unsupervised Learning : Learning only from examples or data or experience, no corresponding labels (clustering / Biclustering)
- Supervised Learning or Inductive Learning : Learning from examples or data or experience with corresponding labels (classification / regression)
- Semi-Supervised Learning : Learning from examples or data or experience with only some not all the corresponding labels (classification)
- Reinforcement Learning: An agent interacting with the world makes observations, takes actions, and is rewarded or punished, it should learn to choose actions in such a way as to obtain a lot of reward (classification / Regression)

Supervised Learning

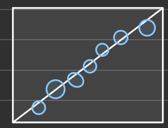
1) Classification

learn a discrete function / labelled data
boolean / binary / multi-class



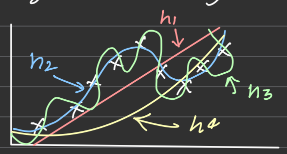
2) Regression

Learn a continuous function



- Task of supervised learning : find a fn 'h' that approximates 'f' given training set $\{x_1,y_1\}\{x_2,y_2\} \dots \{x_n,y_n\}$

hypothesis → $h(x) \approx f(x)$ ← true fn



Best-fit

Occam's razor's principle

- prefer simpler hypothesis over complex ones
- choose explanations with fewer assumptions

classification : a two step process

1) Model Development / Training
   - define pretermined classes

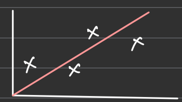
2) Model Usage / Testing and Validation
   - classify unseen / test samples
   - test set never part of training set
   - validation : more accuracy value, the better

$$\text{Accuracy} = \frac{\#\text{ of correct classifications}}{\text{Total }\#\text{ of test cases}}$$

CURVE FITTING

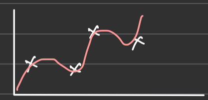
1) Least Squares Regression

- single curve representing trend
- data with errors / noise



2) Interpolation

- fit cures passing directly thro' data pts.
- precise data ~ exact fit



Least Squares Regression

Simple Linear Regression

Data
$\{x_i, y_i\} \quad \forall i = 1, \dots, n$

- $y = a_0 + a_1 x$

$a_0 \downarrow$ intercept, $a_1 \searrow$ slope

- $e_i = y_i - \hat{y}_i = y_i - (a_0 + a_1 x_i)$

y_i ↓ observed, $\hat{y}_i$ ↓ model/predicted

- criterion for best fit

$$\min S_r = \min_{a_0, a_1} \sum_{i=1}^{n} e_i^2 = \min_{a_0, a_1} \sum_{i=0}^{n} (y_i - a_0 - a_1 x_i)^2$$

Find $a_0, a_1 = ?$

$$\frac{\partial S_r}{\partial a_0} = -2 \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) = 0 \quad \text{———} \ (1)$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) x_i = 0 \quad \text{———} \ (2)$$

$(1) \Rightarrow \quad \Sigma y_i = \Sigma (a_0 + a_1 x_i)$

$\Sigma y_i = n a_0 + a_1 \Sigma x_i$ ——— (3) ← Normal eq^ns

$(2) \Rightarrow \quad \Sigma y_i x_i = a_0 \Sigma x_i + a_1 \Sigma x_i^2$ ——— (4) ←

$(3) \Rightarrow \quad a_0 = \frac{1}{n} \Sigma y_i - \frac{a_1}{n} \Sigma x_i = \bar{y} - a_1 \bar{x}$

$\bar{y}, \bar{x}$ ↓ average

$(4) \Rightarrow \quad \Sigma y_i x_i = a_1 \Sigma x_i^2 + \left(\frac{\Sigma y_i}{n} - \frac{a_1}{n} \Sigma x_i \right) \Sigma x_i$

$$a_1 = \frac{\Sigma x_i y_i - \frac{\Sigma x_i \cdot \Sigma y_i}{n}}{\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}}$$

$$a_1 = \frac{n \Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n \Sigma x_i^2 - (\Sigma x_i)^2}$$

Standard error after find a_0 and a_1

$$\boxed{S_r = \Sigma (y_i - a_0 - a_1 x_i)^2}$$

standard error of estimate $\quad \boxed{S_{y/x} = \sqrt{\frac{S_r}{n-2}}}$

— spread around linear regression

Std. Dev. of data points

$$\boxed{S_y = \sqrt{\frac{S_t}{n-1}} = \sqrt{\frac{\Sigma (y_i - \bar{y}_i)^2}{n-1}}} \qquad \boxed{S_t = \Sigma (y_i - \bar{y}_i)^2}$$

corelation coefficient

$$\boxed{r = \sqrt{\frac{S_t - S_r}{S_t}}}$$

— improvement or error reduction due to describing the data in terms of straight line rather than avg.

Practice Problem – 1

x	1	2	3	4	5	6	7
y	0.5	2.5	2.0	4.0	3.5	6.0	5.5
y_cap	0.91	1.75	2.59	3.43	4.27	5.11	5.95
ei=y-y_cap	-0.41	0.75	-0.59	0.57	-0.77	0.89	-0.45
y-y_bar	-2.9	-0.9	-1.4	0.6	0.1	2.6	2.1
	sum(xi)	28			LINEAR REGRESSION		
	sum(yi)	24					
	sum(xi^2)	140					
	sum(xi*yi)	119.5					
	a1	0.8393					
	ao	0.07143					
	Model	y=0.07143+0.8393x					
	Sr=sum(ei^2)	2.9911					
	St=sum(y-ybar)^2	22.714					
	Sy=SQRT(St/(n-1))	1.946					
	Sy/x=SQRT(Sr/(n-2))	0.773					
	r	0.932					

Polynomial Regression

Data $\{x_i, y_i\}\ \forall i=1,\dots,n$

- $y = a_0 + a_1 x + a_2 x^2$
- $e_i = y_i - \hat{y}_i = y_i - (a_0 + a_1 x_i + a_2 x_i^2)$
- criteria for best fit $\min S_r = \min_{a_0,a_1,a_2} \sum_{i=1}^{n} e_i^2$

$$= \min_{a_0,a_1,a_2} \sum (y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$$

Find $a_0, a_1, a_2 = ?$

$$\frac{\partial S_r}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i - a_2 x_i^2) = 0 \quad \text{—— (1)}$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i - a_2 x_i^2)\,x_i = 0 \quad \text{—— (2)}$$

$$\frac{\partial S_r}{\partial a_2} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i - a_2 x_i^2)\,x_i^2 = 0 \quad \text{—— (3)}$$

$$(1) \Rightarrow \quad \sum y_i = n a_0 + a_1 \sum x_i + a_2 \sum x_i^2 \quad \text{—— (1')}$$

$$(2) \Rightarrow \quad \sum x_i y_i = a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 \quad \text{—— (2')}$$

$$(3) \Rightarrow \quad \sum x_i^2 y_i = a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 \quad \text{—— (3')}$$

equivalent to solving a system of 3 simultaneous linear eq^ns

In general, to fit m^{th} order polynomial

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_m x^m$$

using least-squares regression is equivalent to solving a system of $(m+1)$ simultaneous linear eq^ns.

standard error $\boxed{S_{y/x} = \sqrt{\dfrac{S_r}{n-(m+1)}}}$

Multiple Linear Regression

Given data $\{x_i, y_i\}\ \forall n=1,\dots,n$

$y = a_0 + a_1 x_1 + a_2 x_2$

$e_i = y_i - \hat{y}_i = (y_i - a_0 - a_1 x_1 - a_2 x_2)$

$S_r = \sum e_i^2$

Find a_0, a_1, a_2 to minimize S_r.

$$\frac{\partial S_r}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_{1i} - a_2 x_{2i}) = 0 \quad \text{—— (1)}$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_{1i} - a_2 x_{2i})\,x_{1i} = 0 \quad \text{—— (2)}$$

$$\frac{\partial S_r}{\partial a_2} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_{1i} - a_2 x_{2i})\,x_{2i} = 0 \quad \text{—— (3)}$$

$$n a_0 + a_1 \sum x_{1i} + a_2 \sum x_{2i} = \sum y_i \quad \text{—— (1)'}$$

$$\sum x_{1i} a_0 + a_1 \sum x_{1i}^2 + a_2 \sum x_{1i} x_{2i} = \sum x_{1i} y_i \quad \text{—— (2)'}$$

$$\sum x_{2i} a_0 + a_1 \sum x_{1i} x_{2i} + a_2 \sum x_{2i}^2 = \sum x_{2i} y_i \quad \text{—— (3)'}$$

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i} x_{2i} \\ \sum x_{2i} & \sum x_{1i} x_{2i} & \sum x_{2i}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{1i} y_i \\ \sum x_{2i} y_i \end{bmatrix}$$

standard error $S_{y/x} = \sqrt{\dfrac{S_r}{n-(m+1)}}$

General linear least squares

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2 + \cdots + a_m z_m = \sum_{i=0}^{m} a_i z_i$$

$(m+1)$ different functions

Special cases
1. Simple linear LSR $\quad z_0 = 1,\ z_1 = x \quad z_i = 0\ \forall\ i \geq 2$
2. Polynomial LSR $\quad z_i = x^i \quad (z_0 = 1,\ z_1 = x,\ z_2 = x^2 \ldots)$
3. Multiple linear LSR $\quad z_0 = 1 \quad z_i = x_i$ for $i \geq 1$

"linear" $\Rightarrow$ model's dependence on a_i's is linear
the functions can be highly non-linear.

$$S_r = \sum e_i^2 = \sum (y_i - \hat{y}_i)^2$$

min S_r to get a_j, $j = 0, 1, 2 \ldots m$ = ?

$$\frac{\partial S_r}{\partial a_k} = -2 \sum_{i=1}^{n} (y_i - \sum a_j z_{ji}) \cdot z_{ki} = 0$$

$$\sum y_i z_{ki} = \sum \sum z_{ki} z_{ji} a_j \quad k = 0, 1, \ldots m$$

$$\sum_{j=0}^{m} \sum_{i=1}^{n} z_{ki} z_{ji} a_j = \sum_{i=1}^{n} y_i z_{ki}$$

$$Z^T Z A = Z^T Y$$

$$\boxed{A = (Z^T Z)^{-1} Z^T Y}$$

$$Z = \begin{bmatrix} z_{01} & z_{11} & \cdots & z_{m1} \\ z_{02} & z_{12} & \cdots & z_{m2} \\ \vdots & \vdots & & \vdots \\ z_{0n} & z_{1n} & \cdots & z_{mn} \end{bmatrix}_{m \times n}$$

Interpolation

Given $(n+1)$ data points (x_i, y_i) $i = 0, 1, \ldots n$
there is one and only one polynomial of order n
that passes through all the points.

(A) Newton's Divided Difference Interpolating Polynomials

Linear Interpolation

Given (x_0, y_0) and (x_1, y_1)

$$\frac{y_1 - y_0}{x_1 - x_0} = \frac{y - y_0}{x - x_0}$$

$$f_1(x) = y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right) \times (x - x_0)$$

$\hookrightarrow$ first order interpolation

Given: $\ln 1 = 0$, $\ln 6 = 1.791759$, use linear interpolation to find $\ln 2$.

$$\frac{\ln 2 - \ln 1}{1} = \frac{\ln 6 - \ln 1}{5} \Rightarrow f_1(2) = 0.3583518$$

But $\ln 2 = 0.6931$ (true soln)

$$\left(\frac{f_1(2) - \ln 2}{\ln 2}\right) \times 100 = \frac{0.358 - 0.6931}{0.6931} \times 100 = 48.3\%$$

a smaller interval provides better estimate

(B) Quadratic Interpolation

$(x_0, y_0), (x_1, y_1), (x_2, y_2) \sim$ second order poly^n

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$$

$$f_2(x_0) = b_0$$

$$f_2(x_1) = b_0 + b_1(x_1 - x_0) \rightsquigarrow b_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

$$f_2(x_2) = b_0 + b_2(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1)$$

$$b_2 = \frac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}$$

$$\boxed{b_0 = y_0 \qquad b_1 = \frac{y_1 - y_0}{x_1 - x_0} \qquad b_2 = \frac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}}$$

Given:- $\ln 1 = 0$ find $\ln 2 = ?$
$\ln 4 = 1.386294$
$\ln 6 = 1.791759$

Sol^n:-
$b_0 = y_0 = 0$

$$b_1 = \frac{y_1 - y_0}{x_1 - x_0} = \frac{1.386294 - 0}{4-1} = 0.4621$$

$$b_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} = \frac{\frac{1.7917 - 1.386}{6-4} - 0.4621}{6-1}$$

$$= -0.0518731$$

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_1)(x - x_0)$$
$$\hookrightarrow = 0 + 0.4621(x-1) - 0.0518(x-4)(x-1)$$

$$f_2(2) = 0.565844$$

$$\frac{f_2(2) - \ln 2}{\ln 2} \times 100 = 18.4\%$$

Straightforward Approach

$y = a_0 + a_1 x + a_2 x^2$ (quadratic function)

$$\begin{Bmatrix} a_0 + a_1 x_0 + a_2 x_0^2 = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 = y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 = y_2 \end{Bmatrix}$$

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

(C) General Form of Newton's Interpolating poly^n

Given $(n+1)$ data points
fit n^{th} degree poly^n

$$f_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \cdots + b_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$
$$= \sum_{i=0}^{n} b_i \prod_{j=0}^{i-1} (x - x_j)$$

find $b_0, b_1, \ldots b_n$.
$x = x_0$
$y_0 = b_0$ or $b_0 = y_0$
$x = x_1$ $y_1 = b_0 + b_1(x - x_0) \Rightarrow b_1 = \frac{y_1 - y_0}{x_1 - x_0}$

$$b_1 = f[x_1, x_0] = \frac{y_1 - y_0}{x_1 - x_0}$$
$\vdots$
$x = x_n$

$$b_n = f[x_n, x_{n-1}, \ldots, x_1, x_0] = \frac{f[x_n, \ldots x_1] - f[x_{n-1}, \ldots x_0]}{x_n - x_0}$$

(D) Lagrange Interpolating Polynomials

reformation of the Newton's interpolating poly^n that avoids the computation of divided diff.

$$\boxed{f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)}$$

where $L_i(x) = \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$

Linear interpolation $(n=1)$
$$f_1(x) = L_0(x) f(x_0) + L_1(x) f(x_1)$$
$$= \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

Second order interpolation $(n=2)$
$$f_2(x) = L_0(x) y_0 + L_1(x) y_1 + L_2(x) y_2$$
$$= \left(\frac{x - x_1}{x_0 - x_1}\right)\left(\frac{x - x_2}{x_0 - x_2}\right) y_0 + \left(\frac{x - x_2}{x_1 - x_2}\right)\left(\frac{x - x_0}{x_1 - x_0}\right) y_1 + \left(\frac{x - x_1}{x_2 - x_1}\right)\left(\frac{x - x_0}{x_2 - x_0}\right) y_2$$

eg:- $\ln 1 = 0$ $\qquad$ $\ln 2 = ?$

$\ln 4 = 1.386$

$\ln 6 = 1.791$

$$f_2(x) = \left(\frac{x-4}{1-4}\right)\left(\frac{x-6}{1-6}\right) 0 + \left(\frac{x-1}{6-1}\right)\left(\frac{x-4}{6-4}\right)\ln 6$$

$$+ \left(\frac{x-1}{4-1}\right)\left(\frac{x-6}{4-6}\right)\ln 4 = 0.565$$

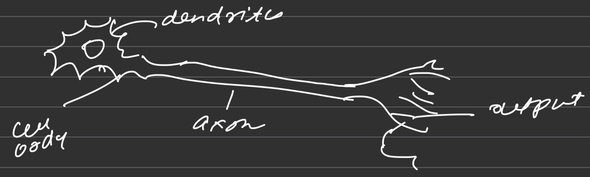
$$f_1(x) = \left(\frac{x-4}{1-4}\right) 0 + \left(\frac{x-1}{4-1}\right) 1.386 = 0.46209$$

Supervised Learning : Artifitical Neural Network (ANN) based Classifiers

Artificial Neural Network

- Human Brain
- basic computation unit in nervous system contains a nerve cell (Neuron) + synaptic links (synapses)

- A natural neuron has three major components
  - Dendrites (Receptor or Input node)
  - Cell Body (soma)
  - Axon Endings (Transmitter buds or Output Nodes)



Properties and Capabilities of ANN

- Non-linearity
- Input – Output mapping
- Adaptivity
- Degree of correctness of Response / output
- Fault Tolerance
- Implementability (using VLSI)
- Uniformity of Analysis and Design
- Neurotiological Analogy
- Contextual Information

1943 ~ McCulloch and Pitts ~ earliest mathematical models

☆ (1943 ~ McCulloch and Pitts)

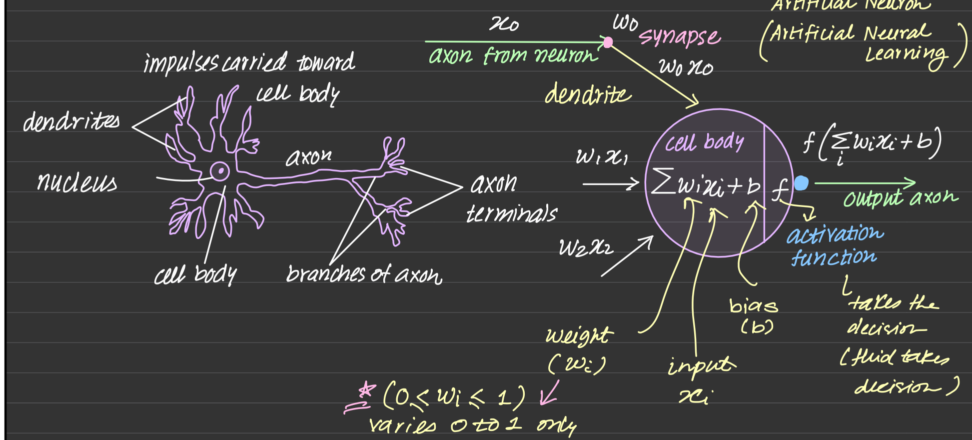
Perceptron simplest feed forward linear binary classifier ANN

$$f(x) = \begin{cases} 1 & ; w \cdot x + b > 0 \\ 0 & ; \text{otherwise} \end{cases}$$

How does the ANN Learn?

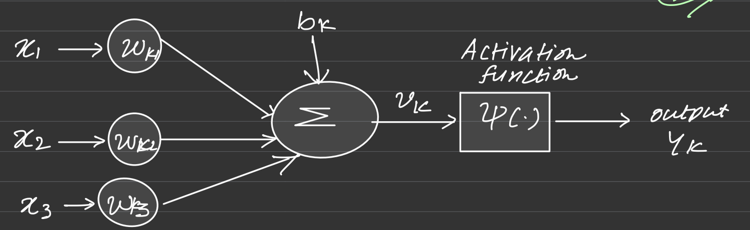
Imp. Diagram

understanding learning



Non-linear model of a neuron

Imp. Diagram



without bias

$$v_k = \sum_{j=0}^{m} w_{kj}\, x_j$$

$$y_k = \varphi(v_k) = \varphi(\Sigma w_j x_j)$$

with bias

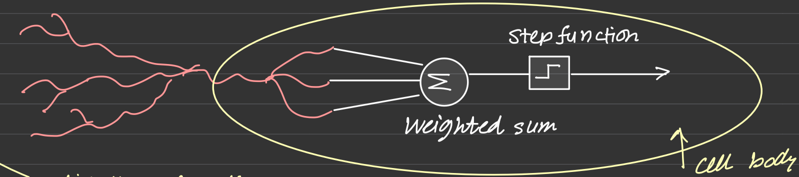
$$u_k = \sum_{j=1}^{m} w_{kj}\, x_j$$

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$$

$$y_k = \varphi(v_k) = \varphi(u_k + b_k)$$

linear Artificial Neuron or the Perceptron

Perceptron (A binary linear classifier) ~ simplest mathematical model of a biological neuron

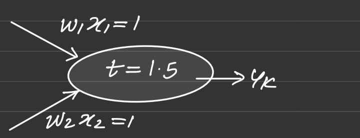


activation function

↪ step function.

Perceptron Training

$$\text{Output} = \begin{cases} 1 & \text{if } \Sigma w_i x_i > t \\ 0 & \text{otherwise} \end{cases}$$

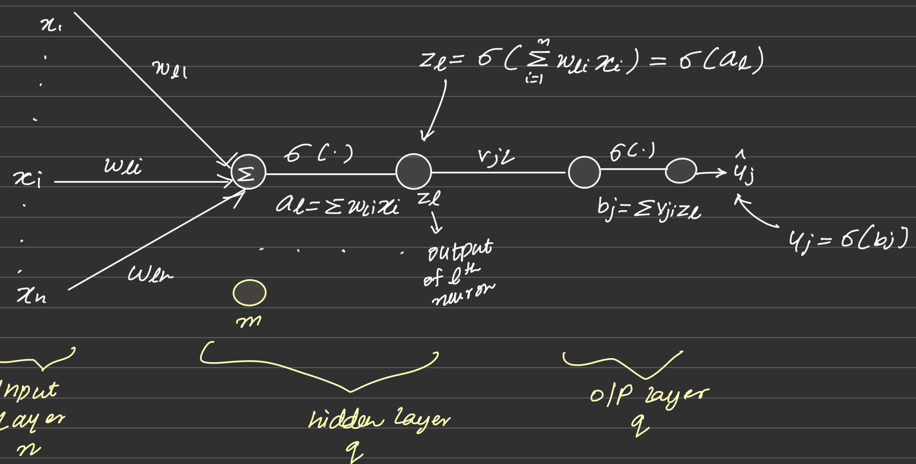


$1 + 1 = 2 > 1.5 \Rightarrow y_k = 1$

Bias can also be added in $\Sigma w_i x_i$.

Backpropagation Algorithm

$$z_l = \sigma\left(\sum_{i=1}^{m} w_{li} x_i\right) = \sigma(a_l)$$



$\sigma(x) = x \quad \therefore \sigma'(x) = 1$

$a_\ell = \sum_{i=1}^{n} w_{\ell i} x_i \qquad z_\ell = \sigma(a_j)$

$b_j = \sum_{i=1}^{n} v_{ji} z_i \qquad \sigma(b_j) = b_j \Rightarrow \sigma'(b_j) = 1$

$e_j = y_j - \hat{y}_j$

$w(k+1) = w(k) + \Delta w(k)$ b/w I/P & hidden

$v(k+1) = v(k) + \Delta v(k)$ b/w hidden & o/p

$E = \frac{1}{2} \sum_{j=1}^{n} (e_j)^2$

overall error of ANN

$\sigma'(a_\ell) = z_\ell(1 - z_\ell)$

$$\boxed{\begin{array}{ll} \sigma(a_\ell) = \dfrac{1}{1+e^{-a_\ell}} & \text{(sigmoid)} \\ \sigma(a_\ell) = \dfrac{1-e^{-a_\ell}}{1+e^{-a_\ell}} & \text{(tanh(a) or tansigmoid)} \end{array}}$$

weight change b/w hidden and o/p layer ~ Gradient Descent Method

$$\Delta v_{j\ell} = -\eta \frac{\partial E}{\partial v_{j\ell}} = -\eta \frac{\partial \frac{1}{2}\Sigma e_j^2}{\partial v_{j\ell}} = -\eta e_j \frac{\partial e_j}{\partial v_{j\ell}}$$

$v_{j\ell}$ contributes only for e_j.

$$= -\eta e_j \frac{\partial (y_j - \hat{y}_j)}{\partial v_{j\ell}} = -\eta e_j \frac{\partial}{\partial v_{j\ell}} \left(y_j - \sum_{\ell=1}^{n} v_{j\ell} z_\ell \right)$$

$\boxed{\Delta v_{j\ell} = +\eta e_j z_\ell}$ or $v_{j\ell}(k+1) = v_{j\ell} + \eta e_j z_\ell$

In general,

$\boxed{v_{j\ell}(k+1) = v_{j\ell}(k) + \eta e_j \sigma'(a_j) z_\ell}$

$\Delta v_{j\ell} = \eta e_j \hat{y}_j (1 - \hat{y}_j) z_\ell$

$= +\eta e_j z_\ell$

$\sigma(b_j) = b_j$

$\sigma(b_j) = \text{sigmoid} = \dfrac{1}{1+e^{-b_j}}$

weight change b/w I/P and hidden layer ~ Gradient Descent method

$$\Delta w_{\ell i} = -\eta \frac{\partial E}{\partial w_{i\ell}} = -\eta \frac{\partial E}{\partial z_\ell} \times \frac{\partial z_\ell}{\partial a_\ell} \times \frac{\partial a_\ell}{\partial w_{\ell i}}$$

$$= -\eta \frac{\partial (y_i - \hat{y}_i)}{\partial z_\ell} \times \sigma'(a_\ell) \times x_i$$

$$= -\eta \Sigma e_j \frac{\partial (y_\ell - \hat{y}_\ell)}{\partial z_\ell} \times z_\ell (1 - z_\ell) x_i \quad \text{for sigmoid}$$

$$= -\eta \Sigma e_j \left(\frac{\partial}{\partial z_\ell} \left(y_j - \sum^{m} v_{j\ell} z_\ell \right) \right) z_\ell (1 - z_\ell) x_i$$

$$= +\eta \Sigma e_j v_{j\ell} z_\ell (1 - z_\ell) x_i$$

$\Delta w_{\ell i} = \eta \delta_\ell x_i \qquad \delta_\ell = \left[\Sigma v_{j\ell} e_j (1 - \hat{y}_j) \hat{y}_j v_{j\ell} z_\ell (1 - z_\ell) \right]$

When output layer has activation function then

$\Delta v_{j\ell} = \eta e_j z_\ell \sigma'(a_j)$ where $\sigma'(a_j) = \hat{y}_j (1 - \hat{y}_j)$

$\Delta w_{\ell i} = \eta \delta_i x_i$ where $\delta_\ell = \sum_{\ell} e_j \sigma'(a_j) v_{j\ell} \sigma'(a_\ell)$

or $\delta_\ell = \left[\sum_{j=1}^{q} e_j \hat{y} (1 - \hat{y}) v_{j\ell} \right] z_\ell (1 - z_\ell)$

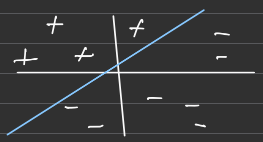
- Classification and Regression using ANN is also two-step process.

$$\text{process} \rightarrow \text{training},\ \text{testing}$$

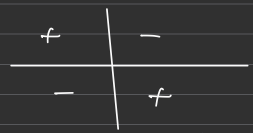
- Supervised Learning

Decision boundaries

1. Divide feature space/input by drawing a hyperplane across it.
   hyperplane → decision boundary
2. Discriminant function returns diff. values on diff. sides of straight line.
3. classified problems are linearly seperable



linearly seperable → AND → representable



non-linearly seperable → XOR → not representable

How to ensure ANN has been trained well?

1. large dataset → training 70% + testing 30%.
2. small dataset → 90% | 10% — Repeat 10 times for diff. train + test
   cross-validation

Performance Measures for Classifiers

1) Accuracy
2) PPV (Precision or positive predictive value)
3) Recall or sensitivity or Hit Rate
4) confusion matrix
5) F1 score
6) specificity or True Negative Rate (TNR)
7) Receiver operating characteristic (ROC) curve
8) Area under ROC curve (AUC)
9) Efficiency
10) Robustness — deal with noise and missing value
11) Scalability — able to change scale
12) Interpretability
13) compactness of the model — size of decision tree

In Regression → RMSE → is used mostly to check quality.

① Accuracy $= \dfrac{\#\text{ of correct classification}}{\#\text{ of total test cases}}$

② Precision $= \dfrac{TP \uparrow}{TP + FP \downarrow}$ {True Positive, False Positive}

③ Recall or sensitivity or hit rate $r = \dfrac{TP}{TP + FN}$

④ Confusion Matrix $= \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$

TP	FP
FN	TN

eg:- find confusion matrix? find precision, recall?

$$CM = \begin{bmatrix} 30 & 0 \\ 30 & 100 \end{bmatrix} \qquad p = \frac{30}{30+0} = 1 \rightarrow 100\% \qquad r = \frac{30}{30+30} = \frac{1}{2} \rightarrow 50\%$$

⑤ $PPV = 1 - FDR$

$$p = PPV = \frac{TP}{TP+FP} = 1 - FDR$$

$TPR = 1 - FNR$

$$r = TPR = \frac{TP}{P} = \frac{TP}{TP+FN} = 1 - FNR$$

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

where, PPV positive predictive value
FDR false discovery rate
TPR true positive rate
FNR false negative rate

Endsem Exam
↓
3 hours exam
↓
Full syllabus
(before + after MS)

Exam
Confusion Matrix is for Binary Classifier
eg:- in exam given confusion matrix, find precision, recall, accuracy, PPV, sensitivity, hit rate, FDR, TPR, FNR.

$$CM = \begin{bmatrix} 120 & 15 \\ 25 & 90 \end{bmatrix} \begin{array}{c} 135 \\ 115 \end{array}$$
$$\begin{array}{cc} 145 & 105 \end{array}$$

$p = 0.88 = PPV$
$r = 0.827$ = sensitivity = hit rate
$A = 0.84$

$FDR = 1 - p = 0.12$

⑥ F1 Score (popular)

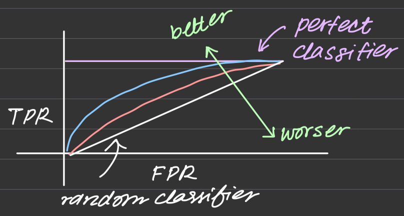
F1 = harmonic mean of precision and sensitivity

$$F_1 = \frac{2}{1/p + 1/r} = \frac{2pr}{p+r} = \frac{2TP}{2TP+FP+FN}$$

⑦ Specificity $= TNR = \frac{TP}{P} = \frac{TP}{TP+FN} = 1 - FPR = \frac{TN}{TN+FP}$

⑧ Receiver operating characteristic
ROC curve
plot of TPR against FPR

$$\frac{TP}{TP+FN} \qquad \frac{FP}{FP+TN}$$



ROC curve

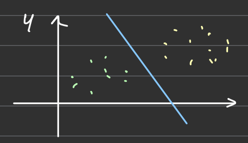
→ AUC = 1, closer to 1 is better
Area under ROC curve

→ AUC = 0.5 for random classifier

Support Vector Machine (SVM)

~ Supervised learning ~ Classification & Regression Analysis
~ Developed by Vladimir Vapnik
~ Before SVM, ANN was most eff. But now SVM >> ANN.
~ SVM gives global optimum unlike ANN that gives local optimum

Problem Statement



$x_1, \dots x_i \dots, x_n$ n data pts.

x_i has m dimensions

each x_i belongs to one of the two classes labelled by +1 and −1.

training:- $(x_1, y_1) \dots (x_n, y_n) \ \forall \ x_i \in \mathbb{R}^m, y_i \in \{+1, -1\}$
$x_i = \{x_1^i, x_2^i, \dots x_m^i\}$

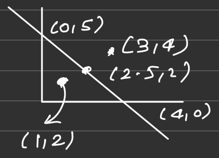
Find best seperating hyperplane $\underset{\sim}{w} \cdot \underset{\sim}{x} + b = 0$ hyperplane

$$f(x) = w_1 x_1 + w_2 x_2 + \dots + w_m x_m + b = 0$$

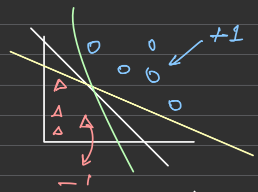
(Assume plane is a classifier hyperplane for linearly seperable data) for any x_i, that x_i may be located on

1) one side of classifier plane : $f(x_i) > 0$
2) other side " " " : $f(x_i) < 0$
3) on the classifier plane : $f(x_i) = 0$

$\text{sign}(f(x) = \underset{\sim}{w} \cdot \underset{\sim}{x} + b)$ ⟶ Decision function



$f(x_1, x_2) = 5x_1 + 4x_2 - 20 = 0$



$f(1,2) = -7 < 0$ left side of plane
$f(3,4) = 11 > 0$ right side of plane
$f(2.5,2) = 0$ exactly on p

$f(x) = wx + b = 0$
$\text{sign}(f(x))$

there can be many such hyperplanes.

Which classifier is best suited?
$\text{sign}(f(x) = (\underset{\sim}{w} \cdot x + b)$

Hyperplane should be as far as possible from any sample data point.

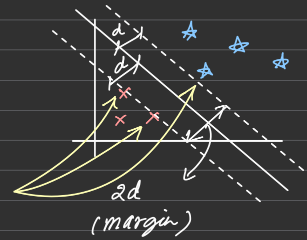
SVM Approach: linearly seperable case

max the distance b/w hyperplane and closest sample pt.

dist. of hyperplane from +ve nearest pt. and -ve nearest pt. should be equal.

Goal: maximize margin $= 2 \times d$

Support vectors: sample data pts. closest to seperating hyperplane



Linear seperable case Formula for margin

① Total margin $= \dfrac{2}{||w||}$

② Solve constrained optimization

$$\min L(w,b,\alpha) = \frac{1}{2}||w||^2 - \Sigma \alpha_i (y_i(x^{iT} w + b) - 1)$$

(A) $\dfrac{\partial L}{\partial w} = 0 \Rightarrow w^* = \Sigma \alpha_i y_i x^{iT}$

(B) $\dfrac{\partial L}{\partial b} = 0 \Rightarrow \Sigma \alpha_i y_i = 0$

(C) $\alpha_i^* \ldots \alpha_n^* = \max_{\alpha_1 \ldots \alpha_n} L(w^*, b^*, \alpha)$

$$L = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x^{iT} x^j + \Sigma \alpha_i$$

s.t. $\alpha_i \geq 0 \ \forall i$ and $\Sigma \alpha_i y_i = 0$

(D) Compute partial derivative of dual w.r.t. all α & set them equal to zero.

$$\frac{\partial L}{\partial \alpha_i} = 0_n \quad \forall i$$

(E) Find b^*
$y = [x^T w + b] = 1$ ⟵ from any segment vector given

Kuhn-Tucker theorem (KT)

Step 1: Solve primal minimization problem

$\min_{\text{primal variables}} L : w, b$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w^* = \sum_{i=1}^{n} \alpha_i y^i x^i \quad \text{— (1)}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i = 0 \quad \text{— (2)}$$

y_i takes ± 1 only.

primal variables are $w, b \rightarrow \min \Rightarrow \left.\begin{array}{l} \frac{\partial L}{\partial w} = 0 \\ \frac{\partial L}{\partial b} = 0 \end{array}\right\}$

Step 2: Solve dual maximization problem

$\max_{\text{Lagrangian multipliers}} L(w^*, b^*, \alpha)$

$\frac{\partial L}{\partial \alpha} = 0_n$ for $\alpha_1^*, \alpha_2^*, \ldots \alpha_n^* = \max_{\alpha_1, \ldots \alpha_n} L(w^*, b^*, \alpha)$

dual Lagrangian at optimal parameters w^*, b^* for max. under +ve constraints. $\alpha_i \geq 0$, $i \in 1, 2, \ldots n$

$\sum_{i=1}^{n} \alpha_i y_i = 0$

$$L(w^*, b^*, \alpha) = \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, x^{iT} x^j$$

Q. Write conditions of KT theorem to find the SVM? (Exam question)