CSE 151 : Machine Learning

Machine learning is about extracting knowledge from data.
ML is the field of study that enables a system to learn from experience without being explicitly programmed.

AI: AI is the broadest term, applying to any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees and machine learning (including deep learning)

ML: The subset of AI that enables machines to improve at tasks with experience. The category includes deep learning.

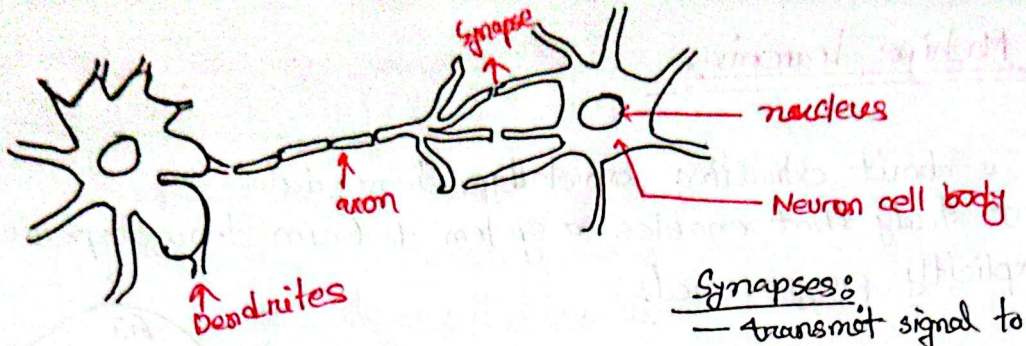
Deep learning: The subset of machine learning composed of algorithms to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amount of data.

Machine: A machine uses power to apply forces and control movement to perform an intended action. Machine include a system of mechanisms that shape the actuator input to achieve a specific application of output forces and movement.

Learning Human: learning is a process of acquiring new or modifying existing knowledge, behaviors, skills, values or preferences.

Human brain is the main element of learning.

Machine learning: Technique to give computer brain like learning ability through progressively update with data, without being explicitly program.

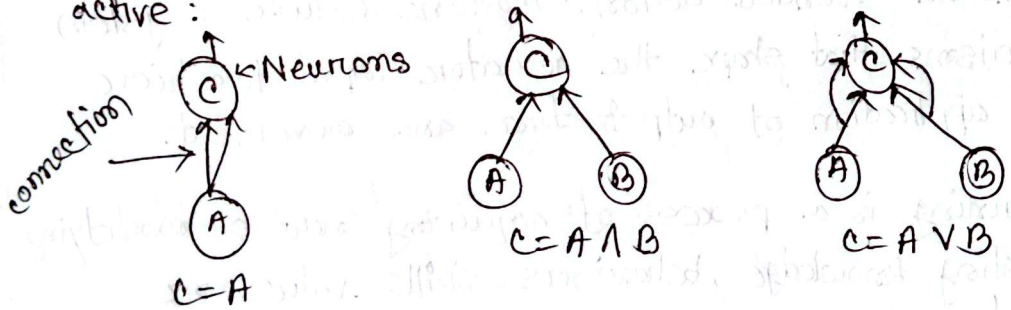


Synapses:
- transmit signal to neo neuron
- vary in strength
- change strength in response to use.

McCulloch – Pitts Model of a Neuron:

- The most fundamental unit of deep neural network is called artificial neuron / perceptron
- Basically, a neuron takes an input signal (dendrite), process it like the CPU (soma), passes the output to the other neuron's dendrite. This is the overview what is going on with a neuron in our brain.
- It has one one more binary input (on/off) and one binary output.
- With this simplified model, It is possible to build a network of artificial neurons that computes any logical proposition you want

Assume that a neuron is activated when at least two of its inputs are active:



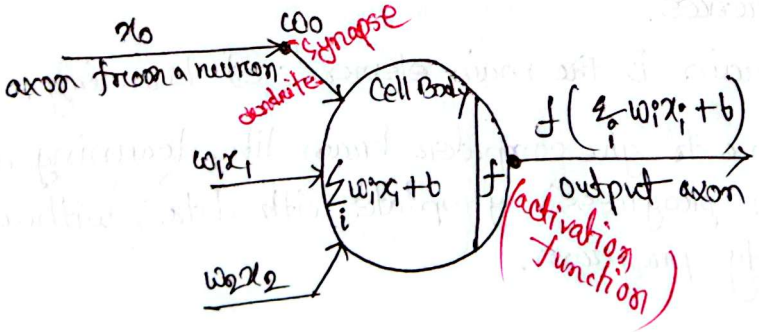


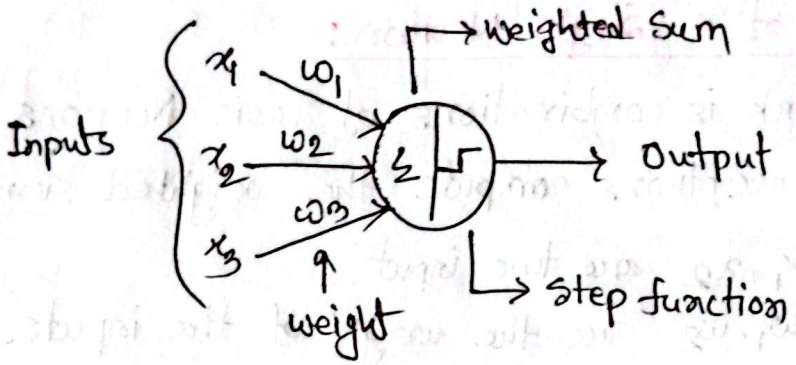
Fig: McCulloch – Pitts Model

The Perceptron: One of the simplest ANN architectures.

- It is based on slightly different artificial neuron called a linear threshold unit. (LTU)
- The input and output are now number instead of binary value
- each input connection is associated with a weight.
- LTU computes the weighted sum

$$z = w_1x_1 + w_2x_2 + \cdots w_nx_n = w^Tx$$

- Then apply a step function : $h_w(x) = \text{step}(z) = \text{step}(w^T.x)$



Step function / Activation function: It maps the output between (0, 1) or (-1, 1)

- Add a term called bias 'b' to this weighte sum to improve the models performance.

Here, weight = Feature importance

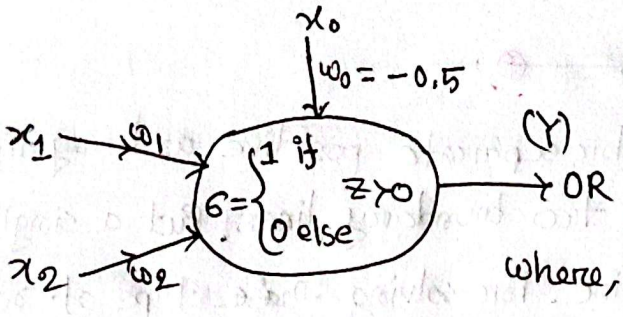
A single LTU can be used for simple linear binary classification. It computes a linear combination of the inputs and the output is in positive class if $z \geq T_H$, otherwise is in negative class. Just like a Logistic Regression Classifier or a linear SVM.

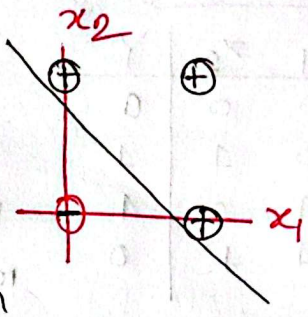
Training LTU means finding the right value of weights $(w_0, w_1, \cdots w_n)$

A perception is simply composed of a single layer LTU with each neuron connected to all the inputs. An extra bias feature is generally added $(x_0 = 1)$. This bias feature is typically represented using a special type of neuron called bias neuron which is a 1 all the time.

A single neuron performs logical operations.

OR operation: Here, inputs x_1, x_2. bias input x_0





where, $z = \sum_{i=0}^{n} w_i x_i$

x_0	w_0	x_1	x_2	$x_0 w_1$	w_2	Y
1	-0.5	0	0	1	1	$-0.5 \leq 0 = 0$
1	-0.5	0	1	1	1	$0.5 > 0 = 1$
1	-0.5	1	0	1	1	$0.5 > 0 = 1$
1	-0.5	1	1	1	1	$1.5 > 0 = 1$

chose weight such a way that the neuron work like OR gate.

$$Y = x_0 w_0 + x_1 w_1 + x_2 w_2$$

Here, $w_0 = -0.5$
$w_1 = 1$
$w_2 = 1$
are chosen.

AND operation

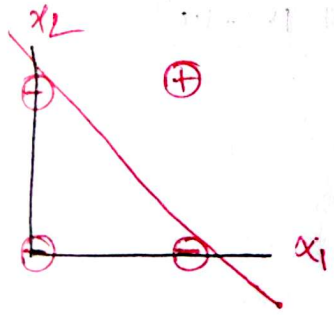
$w_0 x_0 + w_1 x_1 + w_2 x_2$

$w_0 = -1.5$
$w_1 = 1$
$w_2 = 1$
$x_0 = 1$

Not operation

$x_0 = 1$
$w_0 = 1$
$x_1 =$
$w_1 = -1$

$\frac{w_0 x_0 + w_1 x_1}{1}$

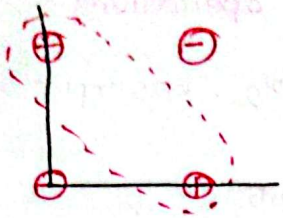


Linear Separability is a concept in ML that refers to the ability to distinguish between ~~dimentions~~ classes of data points using a straight line (in 2D) and hyperplane (in higher Dimension).

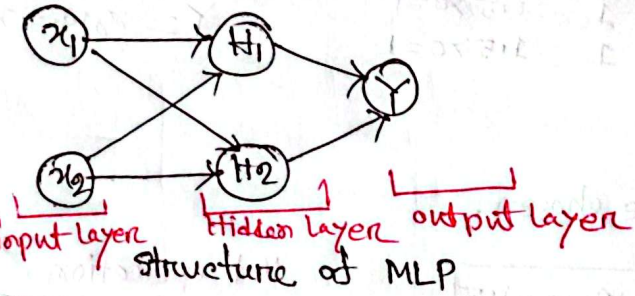
XOR Gate:

Truth table:

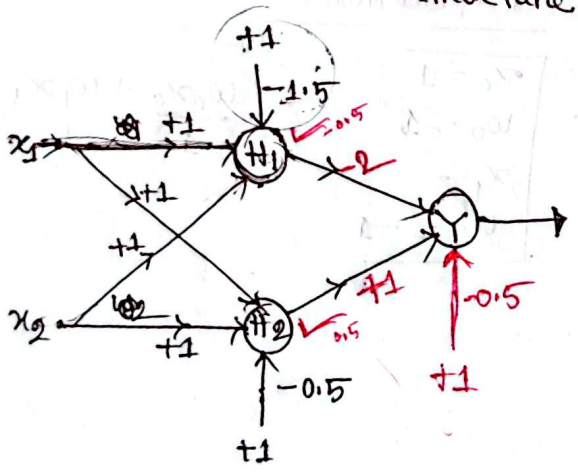
x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



Here, for separate positive and negative output required two boundary lines. But a single neuron can draw only a single line. For solving type of non-linear problem, required Multilayer-Perceptron (MLP). MLP consists of multiple layers of perceptrons, allowing it to model more complex, non-linear functions.



Structure of MLP



$0\times1+0\times1+1\times(-1.5)$

$=$

Here, The Hidden Neuron H_1 act as AND logic gate
H_2 act as OR logic gate

x_1	x_2	H_1	H_2	Y
0	0	0 -1.5	0 -0.5	0
0	1	0 -0.5	1 0.5	1
1	0	0 -0.5	1 0.5	1
1	1	1 0.5	1 1.5	0

The outputs of the two hidden layer becomes the input of output layer. By taking appropriate weight, output neuron generates the output like X-OR logic gate.

This is how MLP worcks as X-OR Gate using two lines.

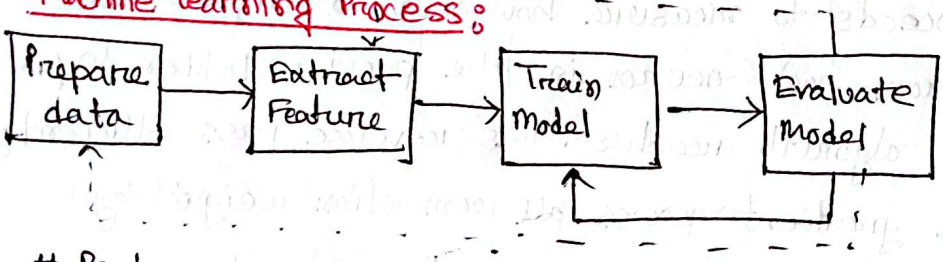
Application of ML :

Supervised Learning
- **Classification :** Fraud detection, Image classificat
  Diagnostics, Customer Retention
- **Regression :** Forecasting, Predictions,
  Process optimization, New insights

Unsupervised Learning
- **Dimentionality Reduction :**
  Feature Elicitation, structure discovery,
  Meaningful compression, Big data visualisation
- **Clustering :** Recommended systems,
  Targetted marketing,
  Customer segmentation.

Reinforcement Learning
- Real time Dicision
- Game AI
- Learning Task
- Skill Aquisition
- Robot Navigation

Machine learning Process :



Performance depends on data-preprocessing.

When to use deep learning?

- DL outperforms with large dataset.
- DL really shines when it comes to complex problems such as image classification, NLP and speech recognition.
- DL doesnot need feature engineering. DL technique outshines others as you don't have to worry about feature engineering.

2.

Why DL is generally better than others methods on image, speech ?

- DL means using a neural network with several layers of nodes between input and output.
- The series of layer between input and output do feature identification and processing in a series of stages, just as our brains seem to.

Machine Learning Vs deep learning

1. Data Dependencies :
2. Hardware Dependencies: ML (CPU) , DL (GPU)
3. Feature Engineering
4. Training Time (ML → less , DL → more.
5. Interpretability : DL → Black box (not good choice)
6. Execution time: DL takes more.

Supervised Learning: Machines are trained using well "labeled" training data and on basis of that data, machines predict the output.

labeled data → some input having output.

Supervised Vs Unsupervised

Supervised

- labeled dat
- producess accurate result
- simple

Unsupervised

- unlabeled data
- less accurate
- complex

Energy demand Predictor :

$n=4$ instance and two attribute

Wind speed	People inside Building	Energy Requirement
100	2	5
50	42	25
45	31	22
60	35	18

we would form the matrix form with $n=4$ and $d=3$

$$X=\begin{bmatrix}1 & 100 & 2\\ 1 & 50 & 42\\ 1 & 45 & 31\\ 1 & 60 & 35\end{bmatrix} \quad \theta=\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix}, \quad Y=\begin{bmatrix}5\\ 25\\ 22\\ 18\end{bmatrix}$$

Suppose that: $\theta=[1 \;\; 0 \;\; 0.5]^T$

Then, $X\theta = \hat{y}$

$$=\begin{bmatrix}1 & 100 & 2\\ 1 & 5 & 42\\ 1 & 45 & 31\\ 1 & 60 & 35\end{bmatrix}\begin{bmatrix}1\\ 0\\ 0.5\end{bmatrix}$$

$$=\begin{bmatrix}1+0+1\\ 1+0+21\\ 1+0+16.5\\ 1+0+17.5\end{bmatrix}$$

$$=\begin{bmatrix}2\\ 22\\ 16.5\\ 18.5\end{bmatrix}$$

Predict on test data: say : $x=[50, 20]$

$$y'=\begin{bmatrix}1 & 50 & 20\end{bmatrix}\begin{bmatrix}1\\ 0\\ 0.5\end{bmatrix}=[1+0+10]=11$$

Describe: Here, given a typical dataset with $n=4$ instances and 2 attributes we would like to learn a model that how inputs effects the outputs. And then, by this model, we can predict for any input x_{n+1}, output $\hat{y}(x_{n+1})$

$$\hat{y}(x_i) = 1\theta_1 + x_i\theta_2$$

$$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \theta_1 - x_i\theta_2)^2$$

In general, the linear model is expressed as: $\hat{y}_i = \sum_{j=1}^{d} x_{ij}\theta_j$

$$= x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{in}\theta_n$$

$$= \theta_1 + x_{i2}\theta_2 + \cdots x_{in}\theta_n$$

Here assumed that $x_{i1} = 1$

So that θ_1 = intercept of the line with the vertical axis.

known as bias or offset

In matrix form, $\hat{y} = X\theta$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & & \\ x_{n1} & \cdots & x_{nd} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

Cost function, $J\theta = (y - X\theta)^T (y - X\theta) = \sum_{i=1}^{n} (y_i - x_i^T\theta)^2$

Optimization approach : Our aim is to minimise the quadratic cost between the output labels and the model predictions.

cost function : $J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$$= \sum (y_i - \theta_1 - x_i\theta_2)^2$$

If we convert i/o as a matrix $\Rightarrow (y - X\theta)^T (y - X\theta)$

$$= \sum (y_i - x_i^T\theta)^2$$

$\theta_1 - x_i\theta_2 = X\theta$

[in matrix form]

$x^T\theta^T y = y^T x\theta$

Here, $J(\theta) = (y - X\theta)^T (y - X\theta)$

$$= y^T y - y^T X\theta - x^T\theta^T y + x^T\theta^T x\theta$$

$$= y^T y - y^T X\theta - y^T X\theta + x^T\theta^T x\theta$$

differentiate:
$\theta\theta^T = \theta^2$
$= 2\theta$

Apply partial derivatives wrt θ.

$$\frac{\delta J(\theta)}{\delta \theta} = \frac{\delta}{\delta \theta}\left(yy^T - 2y^T x\theta + x^T\theta^T x\theta\right)$$

$$= 0 - 2y^T x - 2x^T y + 2x^T x\theta$$

Hence, $\frac{\delta A\theta}{\delta \theta} = A^T$ and $\frac{\delta \theta^T A\theta}{\delta \theta} = 2A^T\theta$

$$= -2x^T y + 2x^T x\theta$$

As we want to calculate θ accurately,

$$\frac{\delta J(\theta)}{\delta \theta} = 0$$

$$\Rightarrow -2x^T y + 2x^T x\theta = 0$$

$$\Rightarrow 2x^T x\theta = 2x^T y$$

$$\Rightarrow \theta = \frac{x^T y}{x^T x} = (x^T x)^{-1} x^T y$$

Now we can calculate the correct θ using I/O matrix
However, this linear prediction can lead to problems in poor data conditions. We need to add a regularizer to solve this issue.

Chapter-2 : Linear Classifier

Classification:

A binary classifier is a mapping from $\mathbb{R}^d \to \{-1, +1\}$

for classifier, use the letter : h where (h = hypothesis)

$$h(\text{input}) = \text{output}$$

So, the classification looks : $x \to \boxed{h} \to y$

Defining a function $\phi(x)$ whose domain is $\mathbb{R}^d$

and where ϕ represents the features of x

and let : $h: \phi(x) \to \{-1, +1\}$

The training data sets form in supervised learning:

$$D_n = \{(x^{(1)}, y^{(1)}), \ldots (x^{(n)}, y^{(n)})\}$$

where, $x^i = d \times 1$ column vector

So, if we give input $x(i)$, the learned hypothesis should generate output $y(i)$

We have to assume a connection between the training data and testing data. typically, they are drawn independently from the same probability distribution

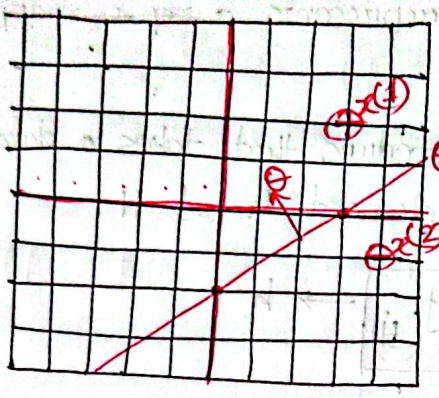
let, training dataset: D_n, classifier h,

So training error of $(h) = \varepsilon_n(h) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 1 & \text{if } h(x^{(i)}) \neq y^{(i)} \\ 0 & \text{otherwise} \end{cases}$

and test error $\varepsilon(h) = \frac{1}{n'} \sum_{i=n+1}^{n+n'} \begin{cases} 1 & \text{if } h(x^{(i)}) \neq y^{(i)} \\ 0 & \text{otherwise} \end{cases}$

Here, $n' \to$ new example for testing purpose.

Example: Let h be the linear classifier defined by $\theta = \begin{bmatrix} -1 \\ 1.5 \end{bmatrix}$, $\theta_0 = 3$



$\theta = \begin{bmatrix} -1 \\ 1.5 \end{bmatrix}$, $\theta_0 = 3$

let,

$-1X + 1.5Y + 3 = 0$

$\Rightarrow \boxed{\frac{x}{3} + \frac{y}{-2} = 1}$

The diagram shows several points, classified by h.

let $x^{(1)} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$, $x^{(2)} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$

So, $h(x^{(1)}; \theta, \theta_0) = \text{sign}\left(\begin{bmatrix} -1, 1.5 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} + 3\right) = \text{sign}(3) = +1$

$h(x^{(2)}; \theta, \theta_0) = \text{sign}\left(\begin{bmatrix} -1, 1.5 \end{bmatrix}\begin{bmatrix} 4 \\ -1 \end{bmatrix} + 3\right) = \text{sign}(-2.5) = -1$

learning linear classifier :

Given a data set and the hypothesis class of linear classifier, our objective is to find the linear classifier with the smallest possible training error.

The idea is to generate k possible hypothesis by generating their parameter vectors at random. Then evaluate the training error on each hypothes and return the hypothesis with lowest training error.

The Perceptron :

Based on McCulloch and Pitts and by Hebb modern on 1943

We have: training dataset D_n with $x \in \mathbb{R}^d$

and $y \in \{-1, 1\}$

The perceptron algorithm trains a binary classifier $h(x; \theta, \theta_0)$ to find θ and θ_0 using τ iterative steps.

Algorithm :

Perceptron (τ, D_n) :

1. $\theta = [0 \quad 0 \ldots . \quad 0]^T$
2. $\theta_0 = 0$
3. for $t = 1$ to τ
4. for $i = 1$ to n
5. if $y^{(i)}(\theta^T x^{(i)} + \theta_0) \leq 0$
6. $\theta = \theta + y^{(i)} x^{(i)}$
7. $\theta_0 = \theta_0 + y^{(i)}$
8. return θ, θ_0

K-Means Clustering

K-means clustering is an unsupervised learning Algorithm. There is no labeled data for this clustering. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster. Here K= number of cluster.

How works?

1. Initialization: start by randomly selected k points from dataset. These points are act as initial cluster centroids.
2. Assignment: calculate distance ~~from~~ of each datapoint from all centroids and assign that cluster whose ~~cluster~~ centroid is closet to it.
3. Update centroids: find mean of all data point and make it as new centroid for each cluster.
4. Repeat: repeat 2, 3 until convergence. (centroid no longer change significantly)
5. Final Result: Once convergence is achieved, algorithm outputs the final cluster centroids.

What is clustering?

Cluster analysis is a technique used in data mining and machine learning to group similar objects into cluster.

Advantages:

- Easy and simple
- Fast and efficient

Disadvantages

- sensitive to initial centroid
- Requires specifying number of cluster
- sensitive to outliers

Example: We have 4 data points in 2D space. Apply K-means cluster (where $k=2$); $X = \{(1,1), (2,1), (4,3), (5,4)\}$

Step-1: Initialization: We randomly select two initial cluster centroids.

$c_1 = (1,1)$

$c_2 = (4,3)$

Step-2: Assign points to the nearest cluster (use Euclidean distance)

$(1,1) \Rightarrow d((1,1),(1,1)) = \sqrt{(1-1)^2+(1-1)^2} = 0$ ✓ (cluster-1)

$d((1,1),(4,3)) = \sqrt{(1-4)^2+(1-3)^2} = 3.61$

$(2,1) \Rightarrow d((2,1),(1,1)) =$ = 1 ✓ (cluster-1)

$d((2,1),(4,3)) =$ $= \sqrt{8}$

$(4,3) \Rightarrow d((4,3),(1,1)) =$ $= \sqrt{13}$

$d((4,3),(4,3)) =$ $= 0$ ✓ (cluster-2)

$(5,4) \Rightarrow d((5,4),(1,1)) =$ $= \sqrt{5}$

$d((5,4),(4,3)) =$ $= \sqrt{2}$ (cluster-2)

So, cluster-1: (1,1), (2,1)

cluster-2: (4,3), (5,4)

Step-3: Update centroids: $c_1 = \frac{(1,1)+(2,1)}{2} = (1.5, 1)$

$c_2 = \frac{(4,3)+(5,4)}{2} = (4.5, 3.5)$

Repeat step-2;

Again calculate the distance from centroids

distance from c_1:

$\sqrt{(1.5-1)^2+(1-1)^2} = 0.5$

$\sqrt{(1.5-2)^2+(1-1)^2} = 0.5$

$\sqrt{(1.5-4)^2+(1-3)^2} = 3.2$

$\sqrt{(1.5-5)^2+(1-4)^2} = 4.6$

distance from c_2:

$\sqrt{(4.5-1)^2+(3.5-1)^2} = 4.30$

$\sqrt{(4.5-2)^2+(3.5-1)^2} = 3.53$

$\sqrt{(4.5-4)^2+(3.5-3)^2} = 0.7$

$\sqrt{(4.5-5)^2+(3.5-4)^2} = 0.7$

Repeat step-3; new centroid: $c_1 = \frac{(1,1)+(2,1)}{2} = (1.5, 1)$

$c_2 = \frac{(4,3)+(5,4)}{2} = (4.5, 3.5)$

since consecutive 2 operation has same centroid, so KMC is converged and iteration is stoped.