Linear models: The least-squares method, The perceptron: a heuristic learning algorithm for linear classifiers, Support vector machines, obtaining probabilities from linear classifiers, going beyond linearity with kernel methods. **Distance Based Models:** Introduction, Neighbors and exemplars, Nearest Neighbors classification, Distance Based Clustering, Hierarchical Clustering.

Introduction: Geometric Models

⇒ In this unit we study the two geometric models of machine learning – Linear models, Distance based models.

⇒ Geometric models/feature learning is a technique combining machine learning and computer vision to solve visual tasks.

⇒ These models that define similarity by considering the geometry of the instance space[1].

⇒ Geometric models most often assume that instances are described by d real-valued features, and thus $X = R^d$.

- For example objects on the map can be described by their position, longitude and latitude (Here d=2) or in the real world by longitude, latitude, and altitude (Here d=3).

⇒ Thus, features could be described as points in two dimensions (x- and y-axis) or a three-dimensional space (x, y, and z).

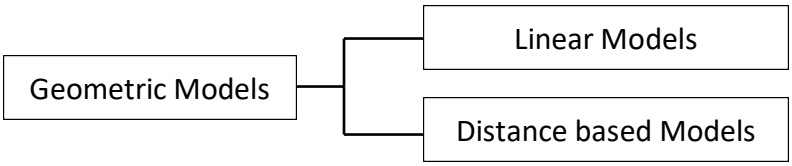
⇒ Even when features are not intrinsically geometric, they could be modeled in a geometric manner. For example, temperature as a function of time can be modeled in two axes.

3D models for printing **3D models for architecture**

Instance space[1]: An instance space is the space of all possible instances for some learning task. In other words, it is defined by a set of unique instances.

Note: The goal of model-based machine learning is a single modeling framework should support a wide range of models.

⇒ In geometric models there are two ways:



Linear Models:

⇒ Geometric concepts like lines or planes are used to segment (classify) the instance space.

⇒ Linear functions take particular forms, depending on the domain and codomain of function f.

⇒ If x and f (x) are scalars, it follows that f is of the form

f (x) = a + bx for some constants a and b; a is called the intercept and b the slope.

⇒ If x = (x1, . . .,xd) is a vector and f (x) is a scalar, then f is of the form

$$f(x) = a + b_1x_1 + \ldots + b_d x_d = a + b \cdot x$$

with b = (b1, . . .,bd). The equation f (x) = 0 defines a plane in R^d perpendicular to the normal vector b.

⇒ Linear models are parametric, meaning that they have a fixed form with a small number of numeric parameters that need to be learned from data.

- For example, in f(x) = a + bx, a and b are the parameters that we are trying to learn from data. This is different from tree or rule models, where the structure of the model (e.g., which features to use in the tree, and where) is not fixed in advance.

⇒ Linear models are stable, which is to say that small variations in the training data have only limited impact on the learned model. In contrast, Tree models tend to vary more with the training data, as the choice of a different split at the root of the tree typically means that the rest of the tree is different as well.

⇒ Linear models are less likely to over fit the training data than some other models, largely because they have relatively few parameters. The flipside of this is that they sometimes lead to under fitting: e.g., imagine you are learning where the border runs between two countries from labeled samples, and then a linear model is unlikely to give a good approximation.

⇒ Linear models exist for all predictive tasks, including classification, probability estimation and regression.

Few examples in real-life:

⇒ Hours spent studying Vs. Marks scored by students

⇒ Amount of rainfall Vs. Agricultural yield

⇒ Electricity usage Vs. Electricity bill

⇒ Suicide rates Vs. Number of stressful people

⇒ Years of experience Vs. Salary, and Demand Vs. Product price

The least-squares method

⇒ The "least squares" method is a form of mathematical regression analysis introduced by Carl Friedrich Gauss in 18th century, used to determine the line of best fit for a set of data, providing a visual demonstration of the relationship between the data points.

⇒ In statistics, linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.

⇒ In the case of one independent variable it is called simple linear regression or univariant. For more than one independent variable, the process is called multiple linear regressions.

⇒ The Line: Aim is to calculate the values m (slope) and b (y-intercept) in the equation of a line:

Y = mx + b

Where

y = how far up

x = how far along

m = Slope or Gradient (how steep the line is)

b = the Y Intercept (where the line crosses the Y axis)

Steps

⇒ To find the line of best fit for N points:

Step 1: For each (x,y) point calculate x^2 and xy

Step 2: Sum all x, y, x^2 and xy, which gives us Σx, Σy, $Σx^2$ and Σxy (Σ means "sum up")

Step 3: Calculate Slope m:

$$m = \frac{N \sum(xy) - \sum x \sum y}{N \sum(x^2) - (\sum x)^2}$$ where N is the number of points

Step 4: Calculate Intercept b:

$$b = \Sigma y - m \Sigma x N$$

Step 5: Assemble the equation of a line

$$y = mx + b$$

Example:

⇒ Find how many hours of sunshine vs. how many ice creams were sold at the shop from Monday to Friday:

"x" hours of sunshine	"y" ice creams sold
2	4
3	5
5	7
7	10
9	15

⇒ Let us find the best m (slope) and b (y-intercept) that suits that data Y = mx + b

Step 1: For each (x, y) calculate x^2 and xy:

x	y	X^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135

Step 2: Sum x, y, x^2 and xy (gives us Σx, Σy, Σx^2 and Σxy):

x	y	X^2	xy
2	4	4	8
3	5	9	15
5	7	25	35
7	10	49	70
9	15	81	135
$\sum x = 26$	$\sum y=41$	$\sum x^2 = 168$	$\sum xy = 263$

Step 3: Calculate Slope m:

$$m = \frac{N \sum(xy) - \sum x \sum y}{N \sum(x^2) - (\sum x)^2}$$

$$= \frac{5*263-26*41}{5*168-26^2}$$

$$= \frac{1315-1066}{840-676}$$

$$= \frac{249}{164} = 1.5183 \ldots$$

Step 4: Calculate Intercept b:

b = (Σy − m Σx) / N

= (41 − 1.5183 x 26) / 5 = = 0.3049...

Step 5: Assemble the equation of a line: y = mx + b

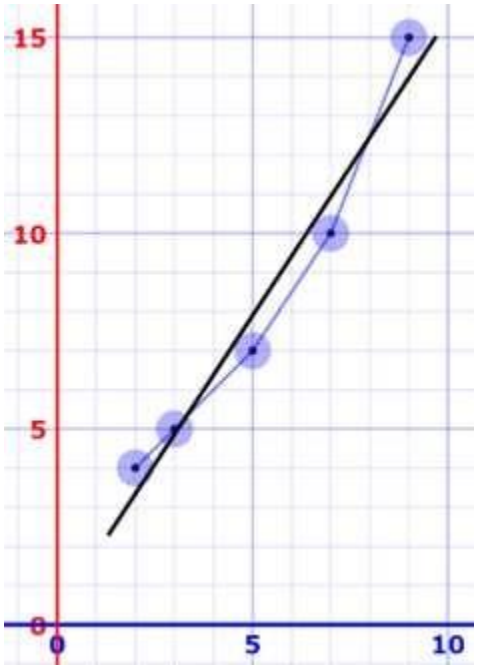
y = 1.518x + 0.305

Let's see how it works out:

x	y	Y= 1.518x + 0.305	error
2	4	3.34	-0.66
3	5	4.86	-0.14
5	7	7.89	0.89
7	10	10.93	0.93

9	15	13.97	-1.03

Here are the (x,y) points and the line **y = 1.518x + 0.305** on a graph:



Mr. Sairam hears the weather forecast which says "we expect 8 hours of sun tomorrow", so he uses the above equation to estimate that he will sell

y = 1.518 x 8 + 0.305 = 12.45 Ice Creams

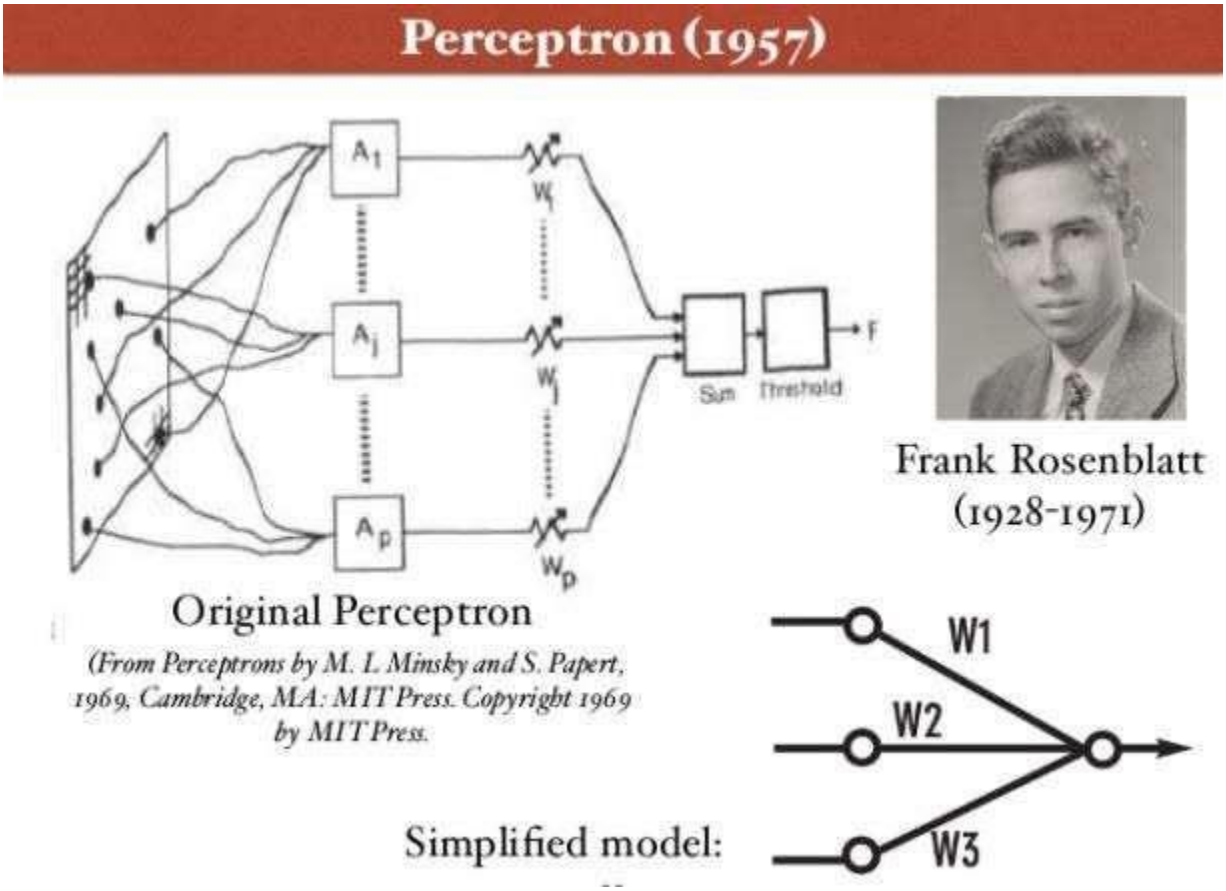
Problem:

⇒ Study the relationship between the monthly e-commerce sales and the online advertising costs. You have the survey results for 7 online stores for the last year.

Online Store	Monthly E-commerce Sales (in 1000 s)	Online Advertising Dollars (1000 s)
1	368	1.7
2	340	1.5
3	665	2.8
4	954	5
5	331	1.3
6	556	2.2
7	376	1.3

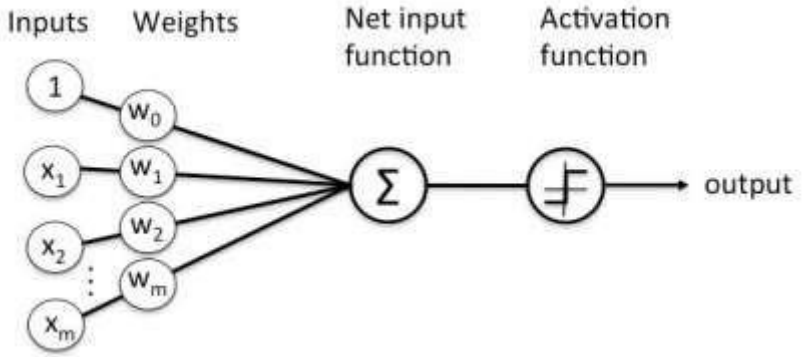
Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.



⇒ Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.

⇒ A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



⇒ There are two types of Perceptron: Single layer and Multilayer.

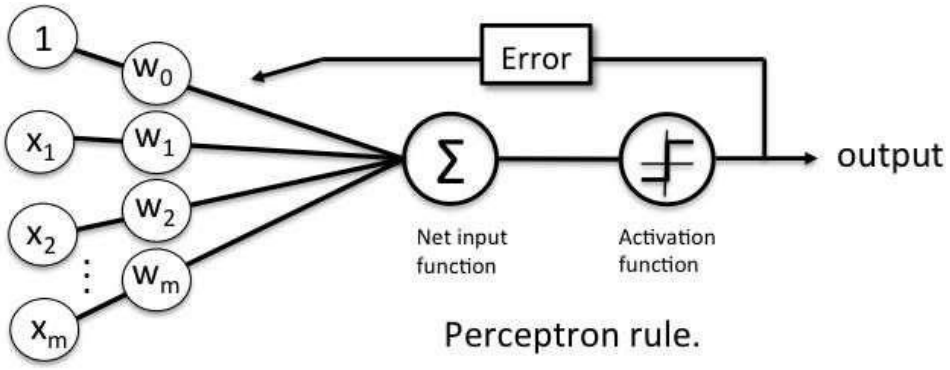
- Single layer Perceptron can learn only linearly separable patterns.
- Multilayer Perceptron or feedforward neural networks with two or more layers have the greater processing power.

⇒ The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary. This enables you to distinguish between the two linearly separable classes +1 and -1.

Note: Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data.

Perceptron Learning Rule

⇒ Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.



⇒ The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

Perceptron Function

⇒ Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

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

⇒ In the equation given above:

- "w" = vector of real-valued weights
- "b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)
- "x" = vector of input x values

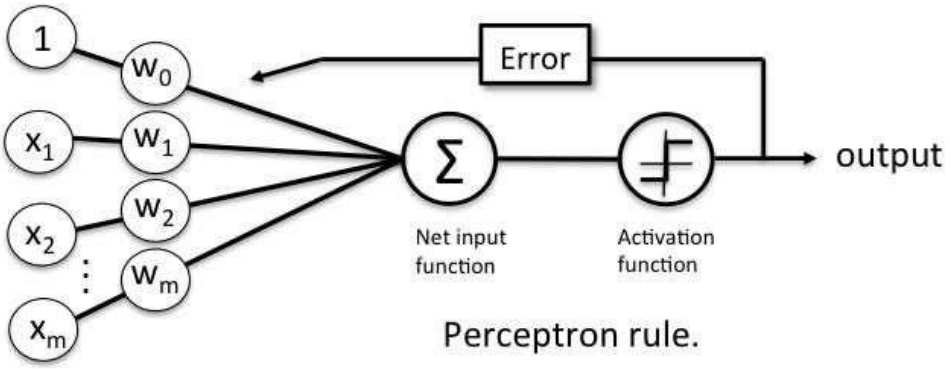
$$\sum_{i=1}^{m} w_i x_i$$

- "m" = number of inputs to the Perceptron

⇒ The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

Inputs of a Perceptron

⇒ A Perceptron accepts inputs, moderates them with certain weight values, and then applies the transformation function to output the final result. The above below shows a Perceptron with a Boolean output.

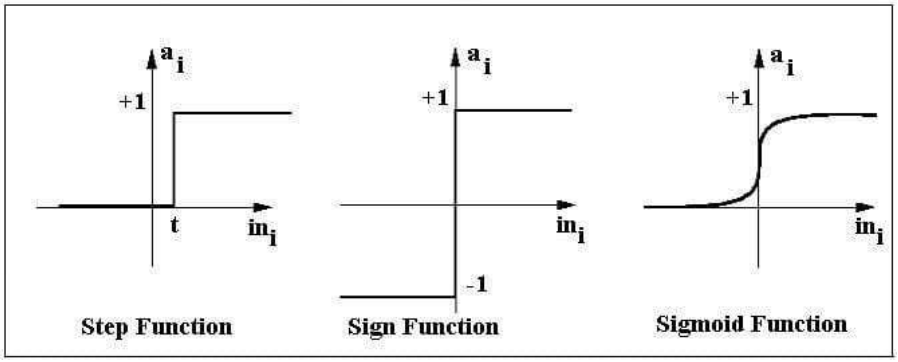


⇒ A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Activation Functions of Perceptron

⇒ The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.



For example:

If $\sum w_i x_i > 0$ => then final output "o" = 1 (issue bank loan) Else, final output "o" = -1 (deny bank loan)

⇒ Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

Perceptron with a Boolean output:

Inputs: x1…xn

Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

Weights: wi=> contribution of input xi to the Perceptron output;

w0=> bias or threshold

If ∑w.x > 0, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.

"sgn" stands for sign function with output +1 or -1.

Error in Perceptron

⇒ In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen.

Perceptron: Decision Function

⇒ A decision function ϕ(z) of Perceptron is defined to take a linear combination of x and w vectors.

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

⇒ The value z in the decision function is given by:

$$Z = W_1 X_1 + \ldots + W_m X_m$$

⇒ The decision function is +1 if z is greater than a threshold θ, and it is -1 otherwise.

$$\phi(z) = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$$

Bias Unit

⇒ For simplicity, the threshold θ can be brought to the left and represented as w_0x_0, where w_0= -θ and x_0= 1.

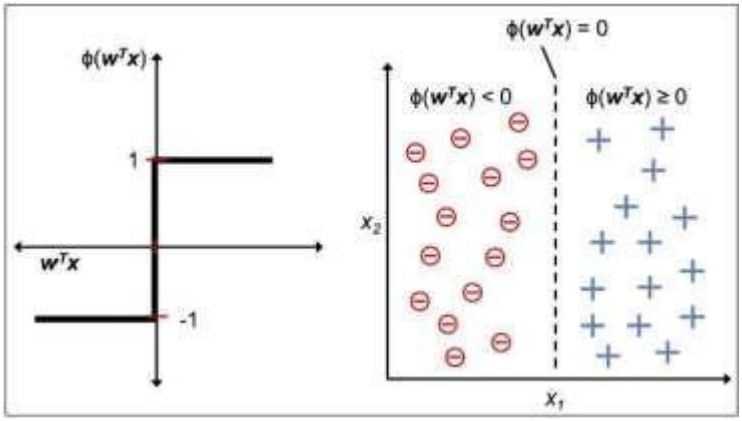
$$z = w_0x_0 + w_1x_1 + \ldots + w_mx_m = \boldsymbol{w}^T\boldsymbol{x}$$

⇒ The value w_0 is called the bias unit. The decision function then becomes:

$$\phi(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ -1 \text{ otherwise} \end{cases}$$

Output

⇒ The figure shows how the decision function squashes wTx to either +1 or -1 and how it can be used to discriminate between two linearly separable classes.



Perceptron at a Glance

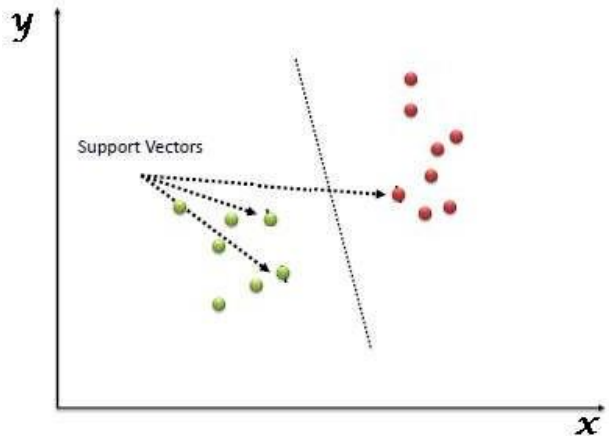
⇒ Perceptron has the following characteristics:

- Perceptron is an algorithm for Supervised Learning of single layer binary linear classifier.
- Optimal weight coefficients are automatically learned.
- Weights are multiplied with the input features and decision is made if the neuron is fired or not.
- Activation function applies a step rule to check if the output of the weighting function is greater than zero.
- Linear decision boundary is drawn enabling the distinction between the two linearly separable classes +1 and -1.
- If the sum of the input signals exceeds a certain threshold, it outputs a signal; otherwise, there is no output.

Types of activation functions include the sign, step, and sigmoid functions.

Support Vector Machines

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

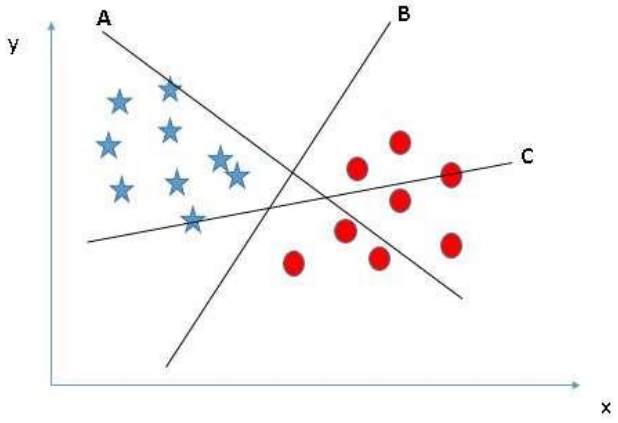


Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).
You can look at support vector machines and a few examples of its working here.

How does it work?

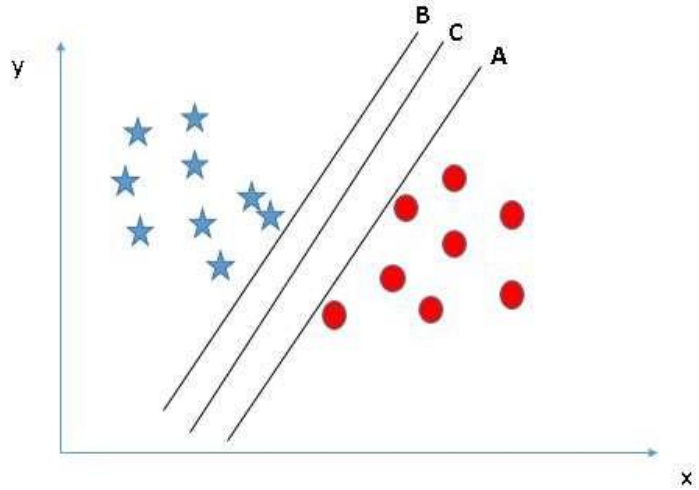
Let's understand:

⇒ **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

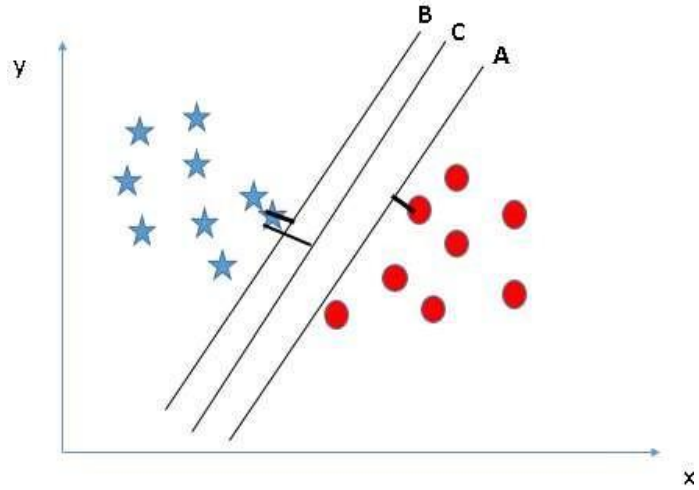


- You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

⇒ **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

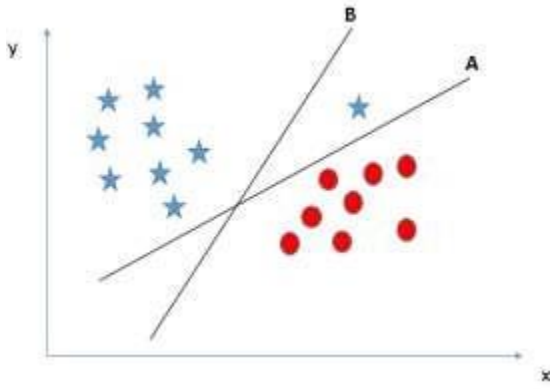


- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let's look at the below snapshot:



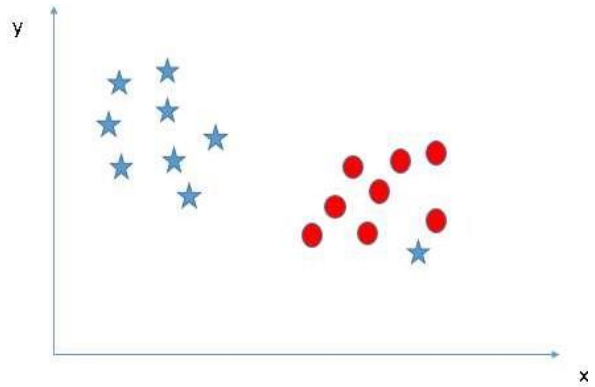
- Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

⇒ **Identify the right hyper-plane (Scenario-3):**Hint: Use the rules as discussed in previous section to identify the right hyper-plane

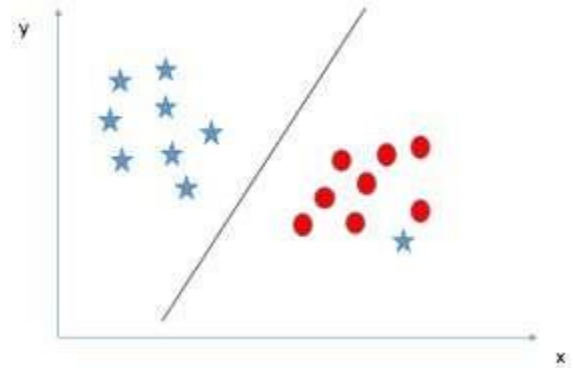


- Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch; SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

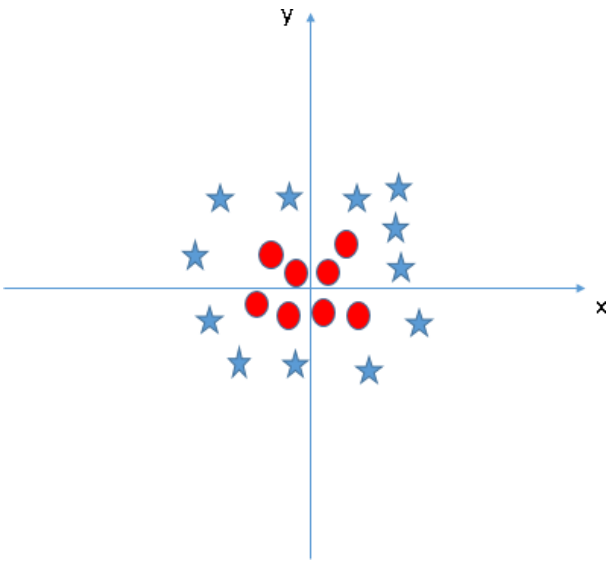
⇒ **Can we classify two classes (Scenario-4)?:** Below, I am unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.



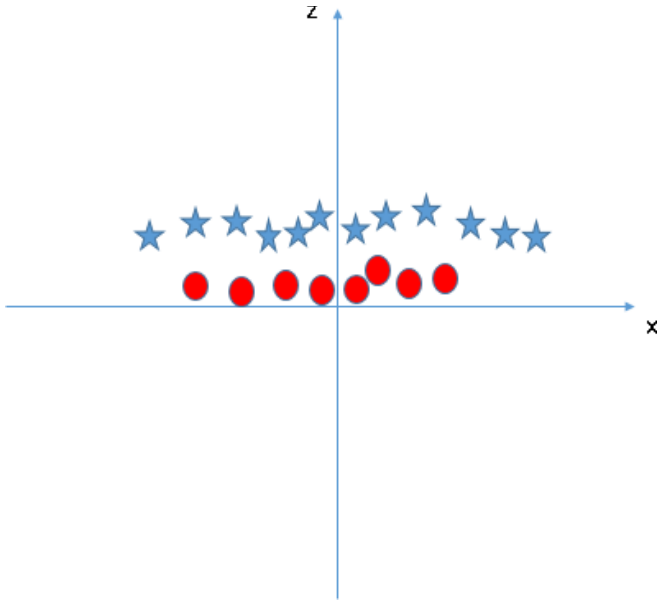
- As I have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.



⇒ **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

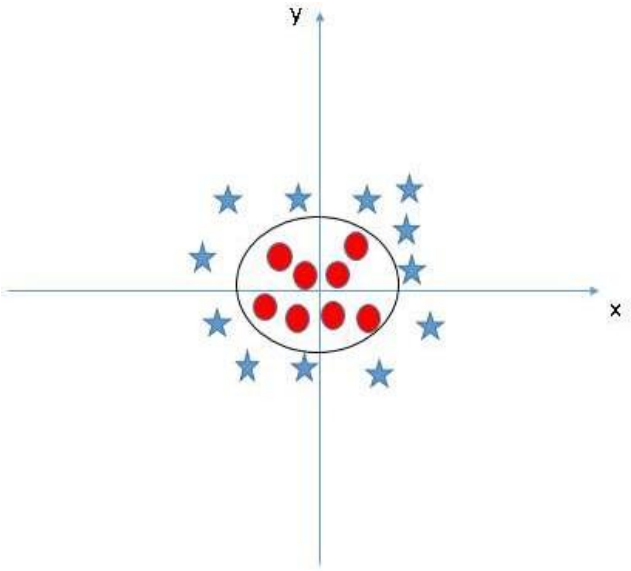


⇒ In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

⇒ In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

⇒ When we look at the hyper-plane in original input space it looks like a circle:



Kernel in Machine Learning

⇒ A kernel is a great tool to transform non-linear data to (almost) linear. The shortcoming of this method is it computationally time-consuming and costly.

⇒ In machine learning, kernel methods are a class of algorithms for pattern analysis, whose best known member is the support vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets.

⇒ Kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

⇒ Kernel methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called "kernel trick". Kernel functions have been introduced for sequence data, graphs, text, images, as well as vectors.

⇒ The magic of the kernel is to find a function that avoids all the trouble implied by the high-dimensional computation. The result of a kernel is a scalar, or said differently we are back to one-dimensional space

⇒ After you found this function, you can plug it to the standard linear classifier.

⇒ Let's see an example to understand the concept of Kernel. You have two vectors, x1 and x2. The objective is to create a higher dimension by using a polynomial mapping. The output is equal to the dot product of the new feature map. From the method above, you need to:

1. Transform x1 and x2 into a new dimension
2. Compute the dot product: common to all kernels
3. Transform x1 and x2 into a new dimension

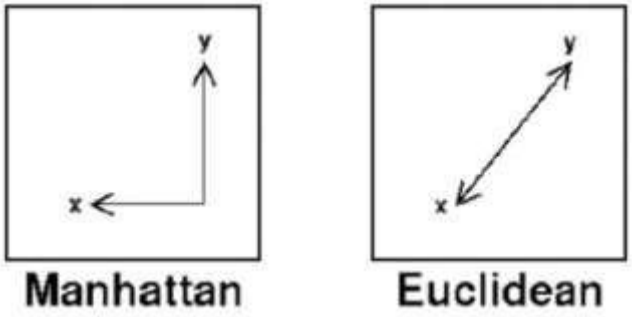
⇒ Application areas of kernel methods are diverse and include geostatistics kriging, inverse distance weighting, 3D reconstruction, bioinformatics, chemoinformatics, information extraction and handwriting recognition.

Popular kernels:

⇒ Fisher kernel
⇒ Graph kernels
⇒ Kernel smoother
⇒ Polynomial kernel
⇒ Radial basis function kernel (RBF)
⇒ String kernels
⇒ Neural tangent kernel

Distance Based Models: Introduction

⇒ **Distance-based models** are the second class of Geometric models. Like Linear models, distance-based models are based on the geometry of data. As the name implies, distance- based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points. Instead, we could think of the distance between two points considering the **mode of transport** between two points. Travelling between two cities by plane covers less distance physically than by train because a plane is unrestricted. Similarly, in chess, the concept of distance depends on the piece used – for example, a Bishop can move diagonally. Thus, depending on the entity and the mode of travel, the concept of distance can be experienced differently. The distance metrics commonly used are **Euclidean**, **Minkowski**, **Manhattan**, and **Mahalanobis**.



⇒ Distance is applied through the concept of neighbours and exemplars. Neighbours are points in proximity with respect to the distance measure expressed through exemplars. Exemplars are either centroids that find a centre of mass according to a chosen distance metric or medoids that find the most centrally located data point. The most commonly used centroid is the arithmetic mean, which minimises squared Euclidean distance to all other points.

Notes:

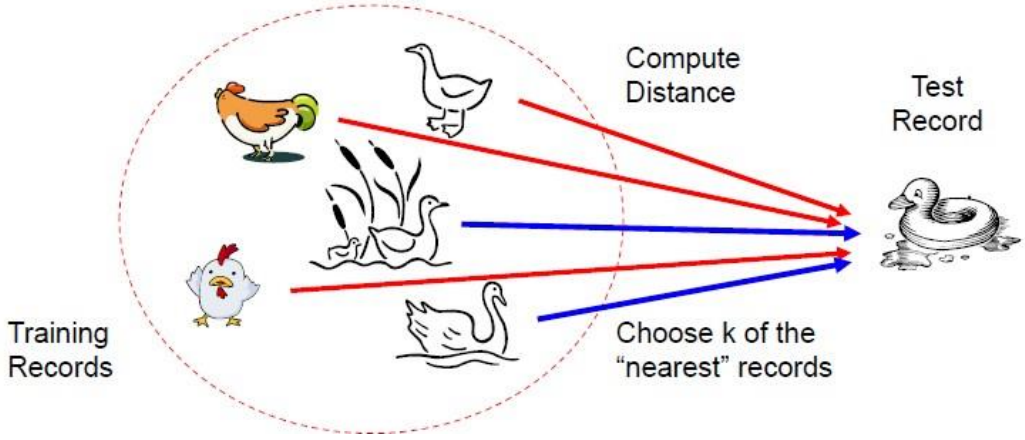
⇒ The **centroid** represents the geometric centre of a plane figure, i.e., the arithmetic mean position of all the points in the figure from the centroid point. This definition extends to any object in *n*-dimensional space: its centroid is the mean position of all the points.

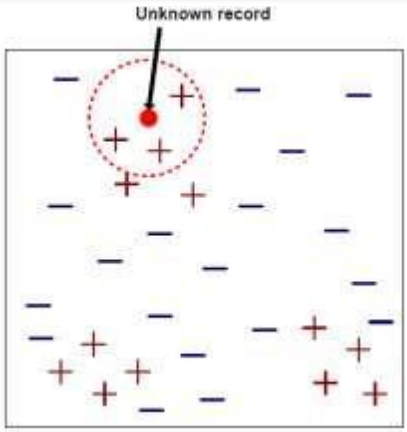
⇒ **Medoids** are similar in concept to means or centroids. Medoids are most commonly used on data when a mean or centroid cannot be defined. They are used in contexts where the centroid is not representative of the dataset, such as in image data.

⇒ Examples of distance-based models include the **nearest-neighbour** models, which use the training data as exemplars – for example, in classification. The **K-means clustering** algorithm also uses exemplars to create clusters of similar data points.

Nearest Neighbor Classifiers

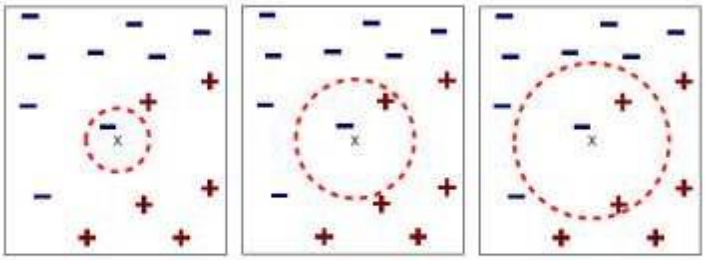
- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck





- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

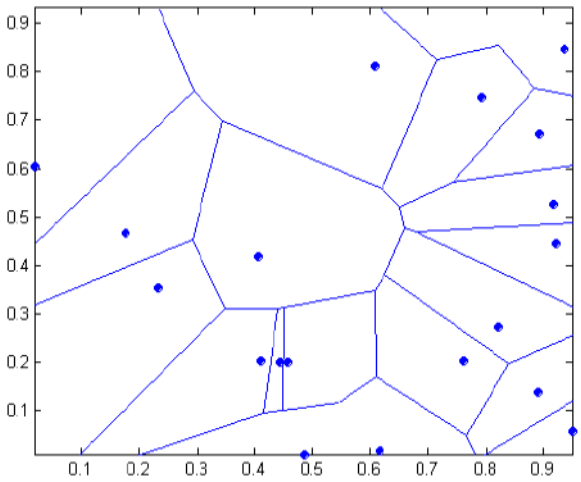


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1 nearest-neighbor

Voronoi Diagram



Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance
$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$
  - Manhatten distance
$$d(p,q) = \sum_i |p_i - q_i|$$
  - q norm distance
$$d(p,q) = (\sum_i |p_i - q_i|^q)^{1/q}$$

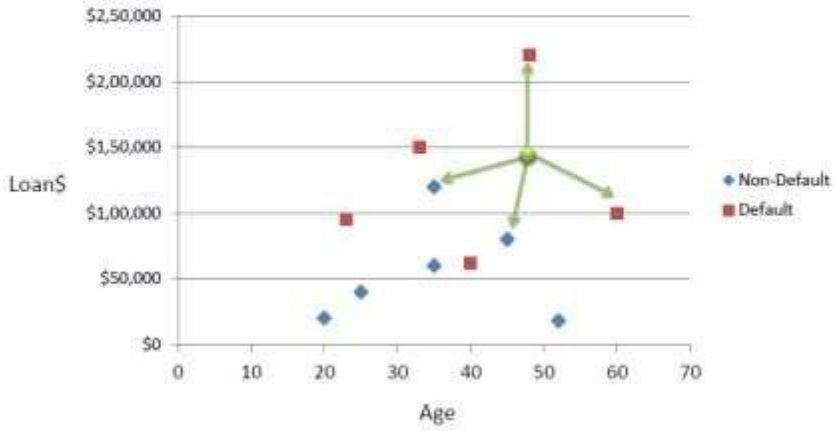
- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D_z} I(v = y_i)$$
where D_z is the set of k closest training examples to z.
  - Weigh the vote according to distance
$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D_z} w_i \times I(v = y_i)$$
    - weight factor, $w = 1/d^2$

The KNN classification algorithm

Let k be the number of nearest neighbors and D be the set of training examples.

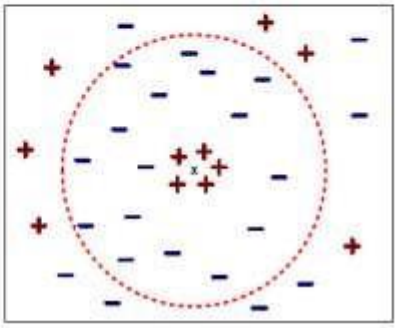
1. **for** each test example z = (**x**′,y′) **do**
2. Compute d(**x**′,**x**), the distance between z and every example, (**x**,y) ϵ D
3. Select $D_z \subseteq D$, the set of k closest training examples to z.
4. $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D_z} I(v = y_i)$
5. **end for**

KNN Classification



Nearest Neighbor Classification...

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 60 KG to 100KG
    - income of a person may vary from Rs10K to Rs 2 Lakh

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality: all vectors are almost equidistant to the query vector
  - Can produce undesirable results

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
	vs		
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		d = 1.4142	

  - Solution: Normalize the vectors to unit length
- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Clustering: Introduction

⇒ Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

⇒ Let's understand this with an example. Suppose, you are the head of a rental store and wish to understand preferences of your costumers to scale up your business. Is it possible for you to look at details of each costumer and devise a unique business strategy for each one of them? Definitely not. But, what you can do is to cluster all of your costumers into say 10 groups based on their purchasing habits and use a separate strategy for costumers in each of these 10 groups. And this is what we call clustering.

⇒ Now, that we understand what is clustering. Let's take a look at the types of clustering.

Types of Clustering

Broadly speaking, clustering can be divided into two subgroups :

⇒ **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.

⇒ **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

Types of clustering algorithms

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the '*similarity*' among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly, let's look at them in detail:

⇒ **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

⇒ **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.

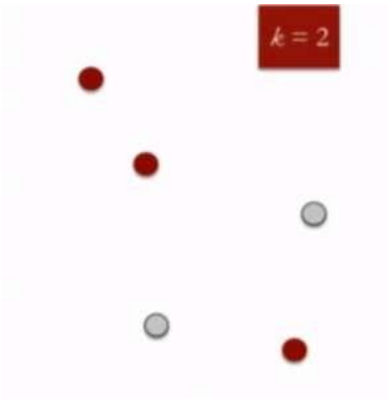
⇒ **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.

⇒ **Density Models:** These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

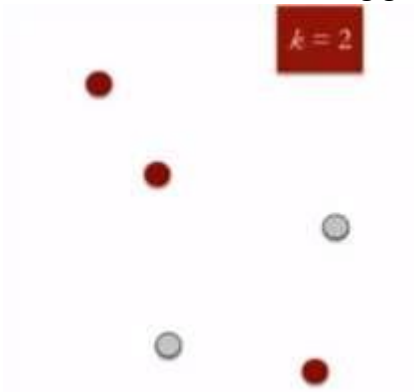
K Means Clustering

K means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps :

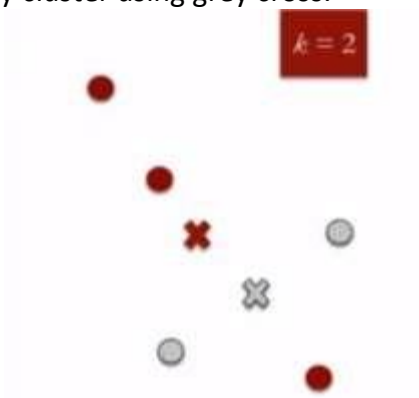
1. Specify the desired number of clusters K : Let us choose k=2 for these 5 data points in 2-D space.



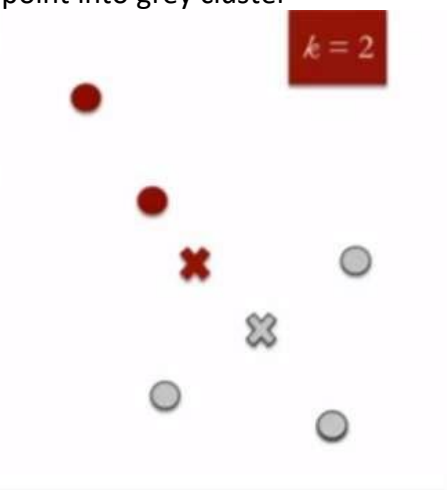
2. Randomly assign each data point to a cluster : Let's assign three points in cluster 1 shown using red color and two points in cluster 2 shown using grey color.



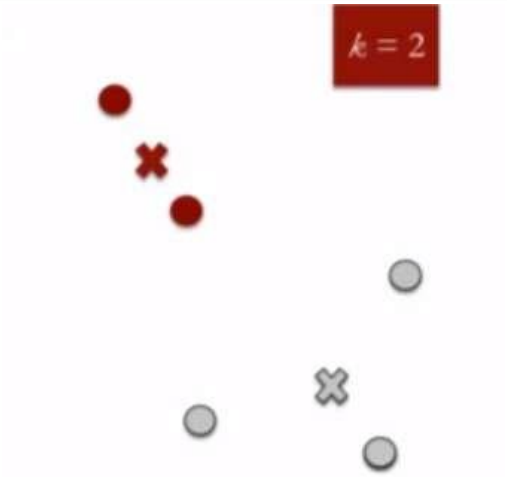
3. Compute cluster centroids : The centroid of data points in the red cluster is shown using red cross and those in grey cluster using grey cross.



4. Re-assign each point to the closest cluster centroid: Note that only the data point at the bottom is assigned to the red cluster even though its closer to the centroid of grey cluster. Thus, we assign that data point into grey cluster



5. Re-compute cluster centroids: Now, re-computing the centroids for both the clusters.

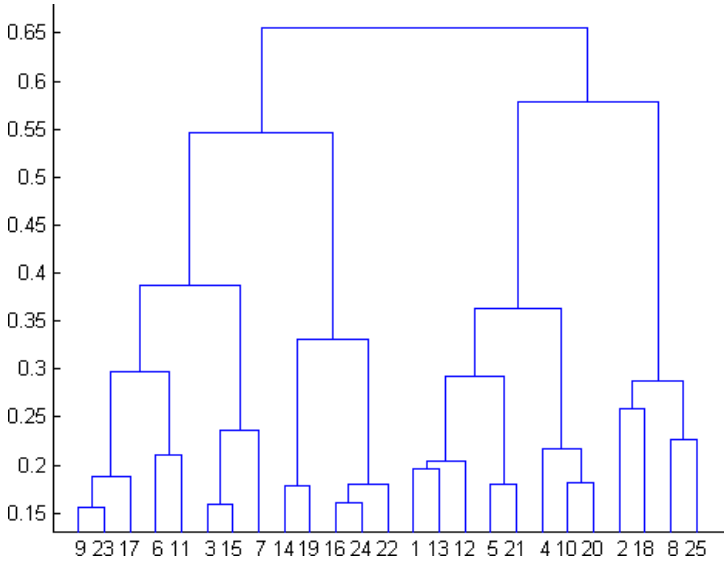


6. Repeat steps 4 and 5 until no improvements are possible : Similarly, we'll repeat the 4th and 5th steps until we'll reach global optima. When there will be no further switching of data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentioned.

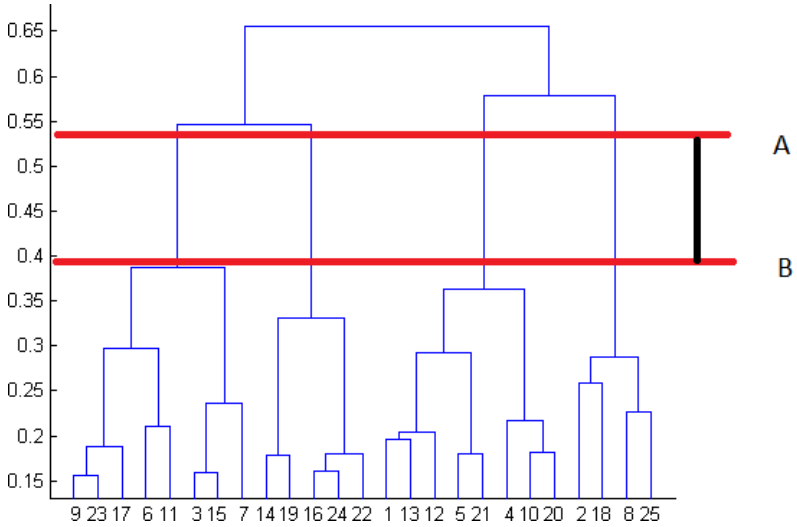
Hierarchical Clustering

Hierarchical clustering, as the name suggests is an algorithm that builds hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

The results of hierarchical clustering can be shown using dendrogram. The dendrogram can be interpreted as:



At the bottom, we start with 25 data points, each assigned to separate clusters. Two closest clusters are then merged till we have just one cluster at the top. The height in the dendrogram at which two clusters are merged represents the distance between two clusters in the data space.
The decision of the no. of clusters that can best depict different groups can be chosen by observing the dendrogram. The best choice of the no. of clusters is the no. of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.
In the above example, the best choice of no. of clusters will be 4 as the red horizontal line in the dendrogram below covers maximum vertical distance AB.



Two important things that you should know about hierarchical clustering are:

⇒ This algorithm has been implemented above using bottom up approach. It is also possible to follow top-down approach starting with all data points assigned in the same cluster and recursively performing splits till each data point is assigned a separate cluster.

⇒ The decision of merging two clusters is taken on the basis of closeness of these clusters. There are multiple metrics for deciding the closeness of two clusters :

- Euclidean distance: $||a-b||_2 = \sqrt{(\Sigma(a_i-b_i))}$
- Squared Euclidean distance: $||a-b||_2^2 = \Sigma((a_i-b_i)^2)$
- Manhattan distance: $||a-b||_1 = \Sigma|a_i-b_i|$
- Maximum distance: $||a-b||_{INFINITY} = max_i|a_i-b_i|$
- Mahalanobis distance: $\sqrt{((a-b)^T S^{-1} (-b))}$ {where, s : covariance matrix}

*******END********