Machine Learning :

Introduction :

To solve a problem on a computer, we need an algorithm. An algorithm is a sequence of instructions that should be carried out to transform the input to output. For example, one can devise an algorithm for sorting. The input is a set of numbers and the output is their ordered list. For the same task, there may be various algorithms and we may be interested in finding the most efficient one, requiring the least number of instructions or memory or both.

We may not be able to identify the process completely, but we believe we can construct a good and useful approximation. That approximation may not explain everything, but may still be able to account for some part of the data. We believe that though identifying the complete process may not be possible, we can still detect certain patterns or regularities. This is the niche of machine learning. Such patterns may help us understand the process, or we can use those patterns to make predictions: Assuming that the future, at least the near future, will not be much different from the past when the sample data was collected, the future predictions can also be expected to be right.

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data or both .

Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from a sample. The role of computer science is twofold: First, in training, we need efficient algorithms to solve the optimization problem, as well as to store and process the massive amount of data we generally have. Second, once a model is learned, its representation and algorithmic solution for inference needs to be efficient as well. In certain applications, the efficiency of the learning or inference algorithm, namely, its space and time complexity, may be as important as its predictive accuracy.

Types of Learning :

1. Supervised Learning .
2. Unsupervised Learning .
3. Semisupervised Learning .

We don’t discuss much about Semisupervised Learning much in this document .

Supervised Learning :

Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value . A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier or a regression function .

The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias). The parallel task in human and animal psychology is often referred to as concept learning.

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the engineer should decide what kind of data is to be used as an example. For instance, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

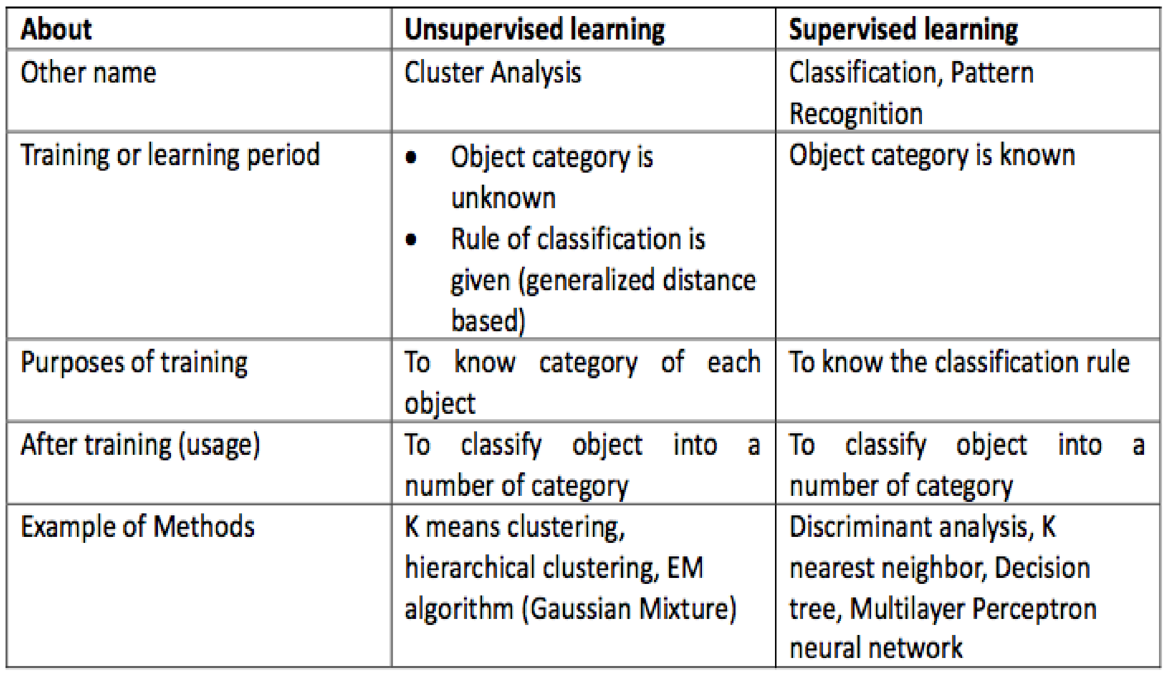
Unsupervised Learning :

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses . All data is unlabeled and the algorithms learn to inherent structure from the input data.

The most common unsupervised learning method is **Cluster Analysis** , which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance .These are called unsupervised learning because unlike supervised learning above there is no teacher .

1. Clustering : A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

Difference between Supervised and Unsupervised Learning :



About	**Unsupervised learning**	**Supervised learning**
Other name	Cluster Analysis	Classification, Pattern Recognition
Training or learning period	• Object category is unknown • Rule of classification is given (generalized distance based)	Object category is known
Purposes of training	To know category of each object	To know the classification rule
After training (usage)	To classify object into a number of category	To classify object into a number of category
Example of Methods	K means clustering, hierarchical clustering, EM algorithm (Gaussian Mixture)	Discriminant analysis, K nearest neighbor, Decision tree, Multilayer Perceptron neural network

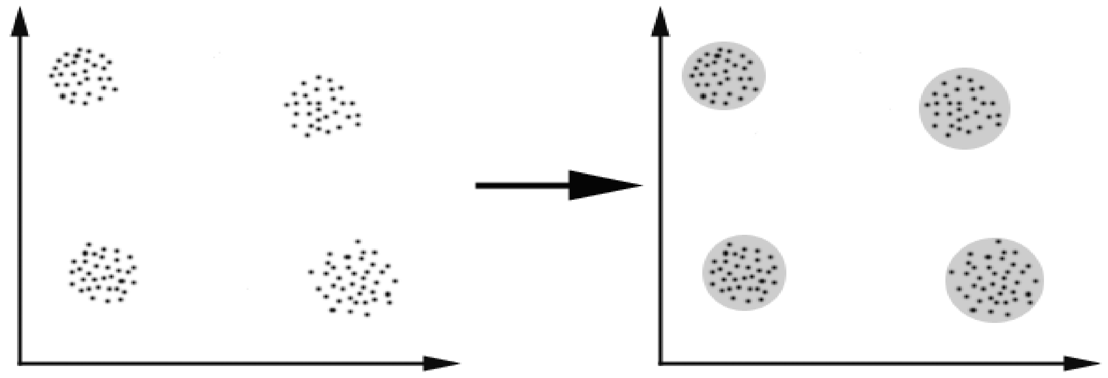
Clustering :

Clustering is one of the most primitive mental activities of humans, used to handle the huge amount of information they receive every day. Processing every piece of information as a single entity would be impossible. Thus, humans tend to categorize entities (i.e., objects, persons, events) into clusters. Each cluster is then characterized by the common attributes of the entities it contains .

As was the case with supervised learning, we will assume that all patterns are represented in terms of features,which form l-dimensional feature vectors.

Clustering can be considered the most important unsupervised learning problem so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data.A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way.

A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.



In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance: two or more objects belong to the same cluster if they are “close” according to a given distance (in this case geometrical distance). This is called distance-based clustering.

The basic steps that an expert must follow in order to develop a clustering task are the following:

1. **Feature selection** : Features must be properly selected so as to encode as much information as possible concerning the task of interest. Once more, parsimony and, thus, minimum information redundancy among the features is a major goal. As in supervised classification, preprocessing of features may be necessary prior to their utilization in subsequent stages. The techniques discussed Sum of Squared error d there are applicable here, too.

2. **Proximity measure :** This measure quantifies how “similar” or “dissimilar” two feature vectors are. It is natural to ensure that all selected features contribute equally to the computation of the proximity measure and there are no features that dominate others. This must be taken care of during preprocessing.

3. **Clustering criterion** : This criterion depends on the interpretation the expert gives to the term sensible, based on the type of clusters that are expected to underlie the data set. For example, a compact cluster of feature vectors in the l-dimensional space, may be sensible according to one criterion, whereas an elongated cluster may be sensible according to another. The clustering criterion may be expre Sum of Squared error sd via a cost function or some other types of rules.

4. **Clustering algorithms :** Having adopted a proximity measure and a clustering criterion, this step refers to the choice of a specific algorithmic scheme that unravels the clustering structure of the data set.

5. **Validation of the results :** Once the results of the clustering algorithm have been obtained, we have to verify their correctness. This is usually carried out using appropriate tests.

6. **Interpretation of the results :** In many cases,the expert in the application field must integrate the results of clustering with other experimental evidence and analysis in order to draw the right conclusions.

In a number of cases, a step known as **Clustering Tendency** should be involved. This includes various tests that indicate whether or not the available data is Sum of Squared errors of a clustering structure.

K-means Algorithm :

Given the cluster number K, the K-means algorithm is carried out in three steps after initialisation: Initialisation: set seed points (randomly)

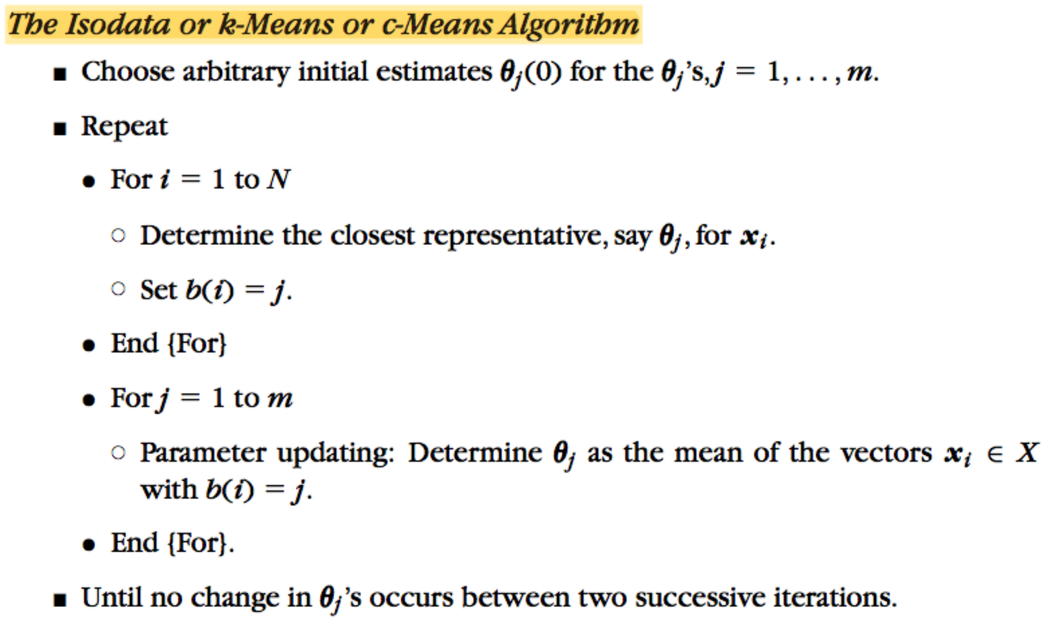
Step 1 : Assign each object to the cluster of the nearest seed point measured with a specific distance metric .

Step 2 : Compute new seed points as the centroids of the clusters of the current partition (the centroid is the centre, i.e., mean point, of the cluster)

Step 3 : Go back to **Step 1** , stop when no more new assignment (i.e., membership in each cluster no longer changes)

K Means is one of the most popular and well-known clustering algorithms . It can be viewed as a special case of the generalized hard clustering algorithmic scheme when point representatives are used and the squared Euclidean distance is adopted to measure the dissimilarity between vectors x_i and cluster representatives θ_j .

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} \|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|^2$$



The Isodata or k-Means or c-Means Algorithm

```
■ Choose arbitrary initial estimates $\theta_j(0)$ for the $\theta_j$'s, $j = 1, \ldots, m$.
■ Repeat
    • For $i = 1$ to $N$
        ○ Determine the closest representative, say $\theta_j$, for $x_i$.
        ○ Set $b(i) = j$.
    • End {For}
    • For $j = 1$ to $m$
        ○ Parameter updating: Determine $\theta_j$ as the mean of the vectors $x_i \in X$
          with $b(i) = j$.
    • End {For}.
■ Until no change in $\theta_j$'s occurs between two successive iterations.
```

Sum of Squared errors in K - Means Algorithm :

Sum of Squared errors is the sum of the squared differences between each observation and its group's mean. It can be used as a measure of variation within a cluster. If all cases within a cluster are identical then the Sum of Squared errors would then be equal to 0 .

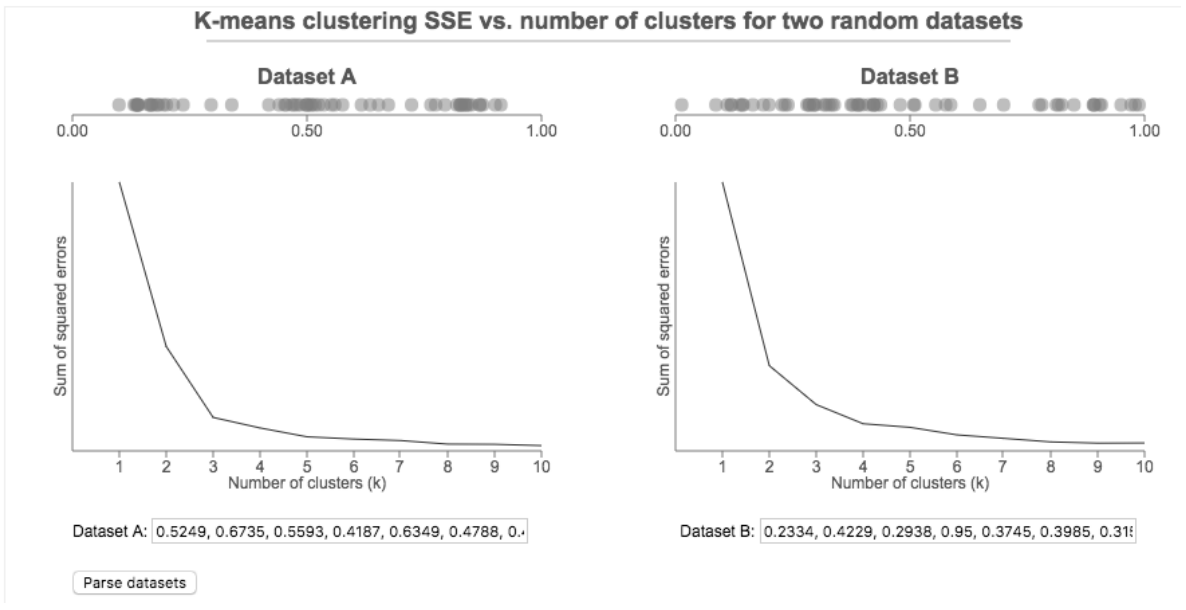
It is mathematically defined as :

$$\sum_{i=1}^{n} (x_i - \overline{x})^2$$

Elbow method for determining optimum number of clusters :

The Elbow method is a method of interpretation and validation of consistency within cluster analysis designed to determine appropriate number of clusters for a dataset . This method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data.

More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance) , but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the **Elbow Criterion** .



Dataset A: 0.5249, 0.6735, 0.5593, 0.4187, 0.6349, 0.4788, 0.

Dataset B: 0.2334, 0.4229, 0.2938, 0.95, 0.3745, 0.3985, 0.31

A plot a line chart of the Sum of Squared errors for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small Sum of Squared errors, but that the Sum of Squared errors tends to decrease toward 0 as we increase k .So our goal is to choose a small value of k that still has a low Sum of Squared errors , and the elbow usually represents where we start to have diminishing returns by increasing k .

Note Dataset A on the left. At the top we see a number line plotting each point in the dataset, and below we see an elbow chart showing the Sum of Squared errors after running k-means clustering for k going from 1 to 10. We see a pretty clear elbow at k = 3, indicating that 3 is the best number of clusters.

Elbow method is a method which looks at the percentage of variance explained as a function of the number of clusters. This method exists upon the idea that one should choose a number of clusters so that adding another cluster doesn't give much better modelling of the data.

The percentage of variance explained by the clusters is plotted against the number of clusters.

The first clusters will add much information but at some point the marginal gain will drop dramatically and gives an angle in the graph.

The correct k i.e. number of clusters is chosen at this point, hence the "elbow criterion". The idea is that Start with K=2, and keep increasing it in each step by 1, calculating your clusters and the cost that comes with the training.

At some value for K the cost drops dramatically, and after that it reaches a plateau when you increase it further. This is the K value you want. The rationale is that after this, you increase the number of clusters but the new cluster is very near some of the existing. In the fig. 1, the distortion J () goes down rapidly with K increasing from 1 to 2, and from 2 to 3 , and then W reach an elbow at K=3 , and then the distortion goes down very slowly after that. And then it looks like maybe using three clusters is the right number of clusters, because that’s the elbow of this curve.

Distortion goes down rapidly until K=3 , and really goes down very slowly after that hence number of cluster needed for this data set is 3.

Algorithm : Elbow method to determine K value in K means algorithm.

Step 1 : Initialize k=1 .

Step 2 : Start

Step 3 : Increment the value of k

Step 4 : Measure the cost of the optimal quality solution .

Step 5 : If at some point the cost of the solution drops dramatically .

Step 6 : That’s the true k .

Step 7 : End .

Limitations of K means Algorithm :

Given an integer K, K-means partitions the data set into K non overlapping clusters. It does so by positioning K centroids in densely populated regions of the data space. Each observation is then assigned to the closest centroid (Minimum distance rule) . A cluster therefore contains all observations that are all closer to a given centroid than to any of the other centroids .

1 . Handling Empty Clusters :

One of the problems with the basic K-means algorithm given earlier is that empty clusters can be obtained if no points are allocated to a cluster during the assignment step. If this happens, then a strategy is needed to choose a replacement centroid, since otherwise, the squared error will be larger than necessary.

One approach is to choose the point that is farthest away from any current centroid. If nothing else, this eliminates the point that currently contributes most to the total squared error. Another approach is to choose the replacement centroid from the cluster that has the highest Sum of Squared errors. This will typically split the cluster and reduce the overall Sum of Squared errors of the clustering. If there are several empty clusters, then this process can be repeated several times.

2. Outliers :

When outliers are present, the resulting cluster centroids may not be as representative as they otherwise would be and thus, the Sum of Squared errors will be higher as well.

To remove this it is often useful to discover outliers and eliminate them beforehand. It is important, however, to appreciate that there are certain clustering applications for which outliers should not be eliminated. When clustering is used for data compression, every point must be clustered , and in some cases , such as financial analysis , apparent Outliers.

3. Reducing the Sum of Squared errors with Post processing :

In k-means to get better clustering we have to reduce the Sum of Squared errors that is most difficult task. There are various types of clustering methods available which reduces the Sum of Squared errors.

An obvious way to reduce the Sum of Squared errors is to find more clusters , i.e., to use a larger K. However, in many cases, we would like to improve the Sum of Squared errors, but don’t want to increase the number of clusters. This is often possible because K-means typically converges to a local minimum. Various techniques are used to “fix up” the resulting clusters in order to produce a clustering that has lower Sum of Squared errors. The strategy is to focus on individual clusters since the total Sum of Squared errors is simply the sum of the Sum of Squared errors contributed by each cluster.

Cluster Validity Indices :

1 . Davies- Bouldin index or DB index :

Davies- Bouldin index or DB index is a metric for evaluating cluster algorithms . This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset .

Let Si be a measure of dispersion of a cluster Ci (i.e., a measure of its spread around its mean vector) and D(Ci,Cj) ≡ D_{ij} the dissimilarity between two clusters , using an appropriate dissimilarity measure . **Lower DB Index value indicates better clustering .**

Based on these, a similarity index R_{ij} between C_i and C_j is defined to satisfy the following conditions **.**

Condition 1 : $R_{ij} = R_{ji}$.

Condition 2 : If $S_i = 0$ and $s_j = 0$ then $R_{ij} = 0$.

Condition 3 : If $S_j > S_k$ and $d_{ij} = d_{ik}$ then $R_{ij} > R_{ik}$.

Condition 4 : If $S_j > S_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$.

- These conditions state that R_{ij} is nonnegative and symmetric .
- If both clusters, C_i and C_j , collapse to a single point, then R_{ij} 0 .
- A cluster C_i with the same distance from two other clusters , C_j , C_k , is more similar to the cluster with the largest dispersion (**Condition 3**) .
- For the case of equal dispersions and different dissimilarity levels, the cluster C_i is more similar to the closer of the two (**Condition 4**).

A simple choice for an *Rij* that satisfies these conditions is the following :

R_{ij} is defined as :

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

provided that d_{ij} is symmetric.

R_i is defined as :

$$R_i = \max_{j=1,\ldots,m, j\neq i} R_{ij}, \quad i = 1, \ldots, m$$

DB index is defined as

$$DB_m = \frac{1}{m}\sum_{i=1}^{m} R_i$$

2. Dunn - Index :

The Dunn index **is a metric for evaluating** cluster algorithms . As do all other such indices, the aim is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance.

For a given assignment of clusters, a **higher Dunn index indicates better clustering**. One of the drawbacks of using this is the computational cost as the number of clusters and dimensionality of the data increase.

There are many ways to define the size or diameter of a cluster. It could be the distance between the farthest two points inside a cluster, it could be the mean of all the pairwise distances between data points inside the cluster, or it could as well be the distance of each data point from the cluster centroid.

$$\Delta_i = \frac{\sum_{x \in C_i} d(x, \mu)}{|C_i|}, \mu = \frac{\sum_{x \in C_i} x}{|C_i|}$$

$\delta(C_i, C_j)$ be this intercluster distance metric, between clusters C_i and C_j.

With the above notation, if there are *m* clusters, then the Dunn Index for the set is defined as:

Dunn Index is defined as :

$$DI_m = \frac{\min\limits_{1 \leqslant i < j \leqslant m} \delta(C_i, C_j)}{\max\limits_{1 \leqslant k \leqslant m} \Delta_k}.$$

Types of Distances :

1 . Euclidean Distance :

The Euclidean distance function measures the distance between two points of same dimensions . The formula for this distance between a point X (X_1 , X_2 , ….. , X_i) and a point Y (Y_1, Y_2, ….. , Y_i) is:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2 . Mahalanobis distance :

The Mahalanobis distance of an observation $\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_i})^T$ from a set of observations with mean $\mu = (\mu_{1,}\mu_2 \ , \ \ldots\ldots \ ,\mu_i)$ [where X and μ are vectors] and covariance matrix S is defined as

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

Let X = (X_1 , X_2 …… , X_i) and Y = (Y_1 , Y_2 , …… ,Y_i) then the distances are defined as follows :

3 . L_1 Norm :

$$L_1 = | X_1 - Y_1 | + | X_2 - Y_2 | + \ldots\ldots + | X_i - Y_i | .$$

4 . L_2 Norm :

$$L_2 = (| X_1 - Y_1 |^2 + | X_2 - Y_2 |^2 + ... + | X_i - Y_i |^2)^{1/2}.$$

5 . L_p Norm :

$$L_p = (| X_1 - Y_1 |^p + | X_2 - Y_2 |^p + ... + | X_i - Y_i |^p)^{1/p}.$$

K Means++ .

In K-means algorithm the initialization of centroids is random selection of points. We propose a specific way of choosing centers for the k-means algorithm which is called as k-means++ algorithm.

Algorithm :

Step 1 : Take one centroid **C_1** , chosen uniformly at random from the dataset.

Step 2 : Take a next centroid **C_i** , choosing $x \in X$ with probability

$$\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}.$$

Where D(x) is the distance between the centroid C_1 and the remaining data points .

Step 3 : For the point having the maximum probability is assigned the next centroid.

Step 4 : Repeat Step 2 until all cluster centers are assigned.

Step 5 : Continue with the standard K-means algorithm after all cluster initialization.

DBSCAN - Density Based Spatial Clustering Algorithms with Noise.

In this section, we present the algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) which is designed to discover the clusters and the noise in a spatial database. Given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

Suppose we have a dataset of n-dimensional data points.

1. For each point in the dataset we make an n-dimensional sphere of radius **epsilon** around the point and count the number of data points within the sphere.
2. If the number of points within the sphere are more than **min_points** then we mark the center of the sphere to be belonging to a cluster.
3. We also mark the points inside the sphere to be belonging to the same cluster. We then recursively expand the cluster by applying the same criteria to the points inside the sphere, except the center.
4. Incase the number of points inside the sphere are less than **min_points**, we ignore it and proceed to the next point in the dataset.
5. But if it's isolated cannot form the cluster cannot be reachable by other clusters, we call this one **outlier** or **noise**.

The parameters **epsilon** and **min_points** can be determined for the best possible clustering using the dataset itself. This method is called adaptive DBSCAN, which I'm not going to deal with over here. However, for practical purposes we may initialize the values manually if we know the kind of data we will run it on.

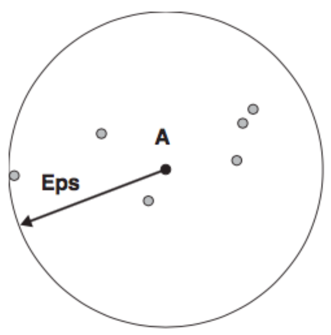


Figure (a) Center-based

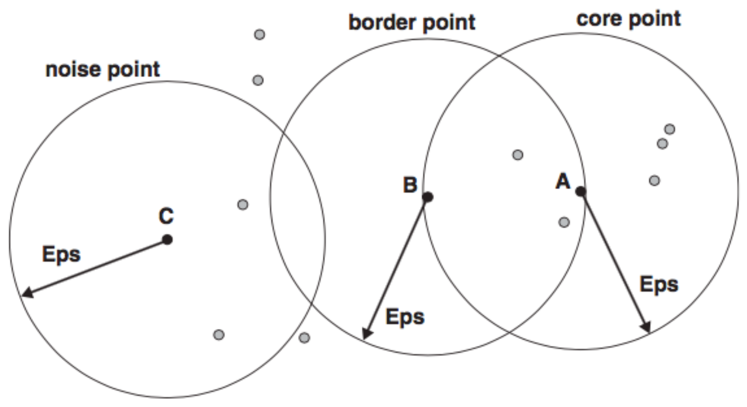


Figure (b) Core, border, and noise points.

Core points: These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user- specified distance parameter, **epsilon**, exceeds a certain threshold, **min_points**, which is also a user-specified parameter. In Figure 8.21, **point A is a core point**, for the indicated radius **epsilon**.(if min_points ≤ 7).

Border points: A border point is not a core point, but falls within the neighborhood of a core point. In Figure 8.21, point B is a border point. A border point can fall within the neighborhoods of several core points.

Noise points: A noise point is any point that is neither a core point nor a border point. In Figure 8.21, point C is a noise point.

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p wrt. Eps and MinPts. If p is a core point, this procedure yields a cluster wrt. **Eps** and **MinPts** . Ifp is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.

Algorithm :

```
DBSCAN (SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point,
                ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

However, there are two key parameters for DBSCAN algorithm : **eps** and **Minpts**. Though users do not need to know the number of clusters, these two parameters are user-determined. **eps** play an important role in DBSCAN, and the result may vary significantly according to different eps value..Different density should have different eps and corresponding Minpts value. Determine the eps value automatically according to the distribution of data under the multi-density situation is a desired property. In order to determine the **eps** and **minpts** we here see the **Adaptive DBSCAN**.

```
ExpandCluster(SetOfPoints, Point, ClId, Eps,
               MinPts) : Boolean;
  seeds:=SetOfPoints.regionQuery(Point,Eps);
  IF seeds.size<MinPts THEN // no core point
    SetOfPoint.changeClId(Point,NOISE);
    RETURN False;
  ELSE   // all points in seeds are density-
         // reachable from Point
    SetOfPoints.changeClIds(seeds,ClId);
    seeds.delete(Point);
    WHILE seeds <> Empty DO
      currentP := seeds.first();
      result := SetOfPoints.regionQuery(currentP,
                                            Eps);
      IF result.size >= MinPts THEN
        FOR i FROM 1 TO result.size DO
          resultP := result.get(i);
          IF resultP.ClId
              IN {UNCLASSIFIED, NOISE} THEN
            IF resultP.ClId = UNCLASSIFIED THEN
              seeds.append(resultP);
            END IF;
            SetOfPoints.changeClId(resultP,ClId);
          END IF; // UNCLASSIFIED or NOISE
        END FOR;
      END IF; // result.size >= MinPts
      seeds.delete(currentP);
    END WHILE; // seeds <> Empty
    RETURN True;
  END IF
END; // ExpandCluster
```

Results :

Dataset	Epsilon (Radius)	Min-Points	Number of Clusters formed
Internet Advertisement	25	30	2
Internet Advertisement	25	35	2
Internet Advertisement	25	40	2
Internet Advertisement	20	30	2
Internet Advertisement	20	35	3
Internet Advertisement	20	40	4
Internet Advertisement	15	30	7
Internet Advertisement	15	35	3
Internet Advertisement	15	40	2

Adaptive DBSCAN (Modified DBSCAN).

There are two key parameters for DBSCAN algorithm : **eps** and **Minpts**. Though users do not need to know the number of clusters, these two parameters are user-determined. Eps play an important role in DBSCAN, and the result may vary significantly according to different eps value. Sometimes it is hard to decide an appropriate eps because dataset are different from each other. Also DBSCAN has difficulties in finding clusters for dataset with different densities, because the eps value is a global parameter.

Determining the eps value :

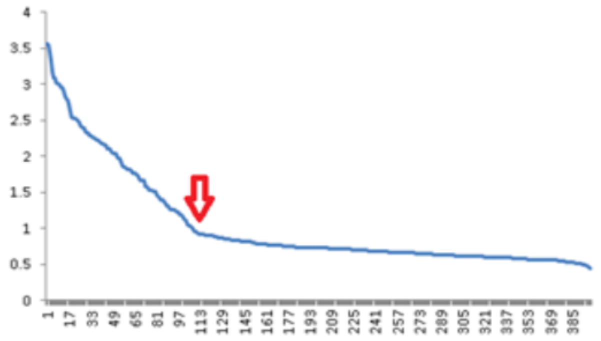
This section explains Adaptive DBSCAN. The first step is to draw a k-dist graph. The distance function employed in DBSCAN is the Euclidian distance where data object x and y are two n-dimension data points:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

The **k-dist** is a function that calculates the distance between a **point** and its **kth nearest point.**

After getting the k-dist value of every point in the dataset and sorting it in descending order, we get a sorted k-dist graph. A k-dist graph gives some clues about the density distribution of the dataset. A k-dist graph shows the k-dist value of every point in descending order.

The k-dist returns the distance from the point to its kth-nearest points, **if there is a sudden increase distance in the k-dist graph, that mean the kth-nearest neighbor of a point becomes farther, and the density changes**. If the points whose **kdist is higher than the threshold**, they are regarded as **Noise**.



A Sample k-dist Graph.

We sort the k-dist in ascending order.We can find several smooth curves connected by rising curves connected by a sharply rising **Line b**, indicating a and c are two density levels. **Line c** and **Line e** are similar case. Three smooth curves refers to the three density level and **Line f** shows the k-dist of the noise since it does not connect the two smooth lines.

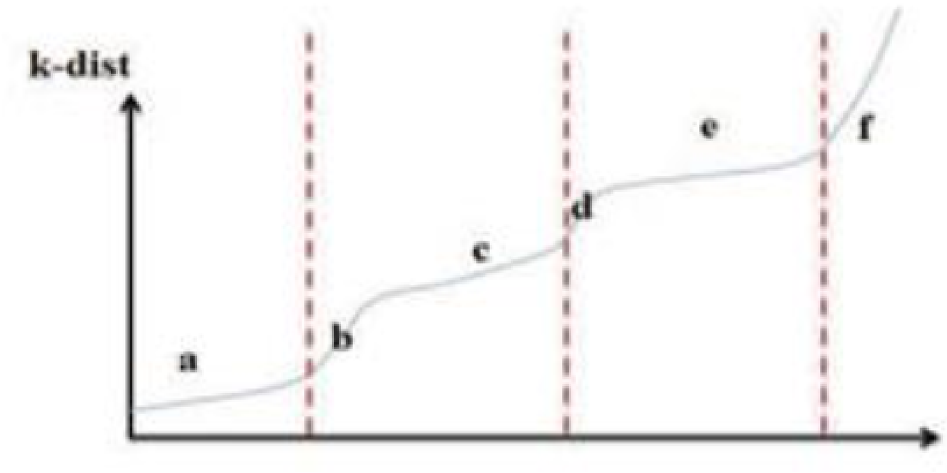


Fig. 2: k-dist graph in ascending order

To find the **eps** value for each density level, we need to calculate the slopes and to check if there exists a sharp difference between slopes. If we detect the slopes of two lines, it is the threshold for the certain density level.

When detecting a large change between two neighboring slopes and making sure the slope is really increasing, the point connected to the two lines is the threshold for the density level and we select it as the **eps** value. Finding all these kind of points in the k-dist graph, we find the value of **eps** for each density levels. The Adaptive DBSCAN starts by clustering from the **lowest eps** to the **highest eps**.

The **lower eps** value we get in k-dist graph indicates the **denser region** because the average distance between a point and all of its k-nearest neighbors is small. First we cluster with **lowest eps,** the points belonging to that densest density level will be clustered together and the lists of noise are those points belonging to sparser region. Next, we apply DBSCAN on the noise list with the next eps and the points that have been clustered will be ignored in the following process.

The **eps value** is the threshold to certain density level, so applying DBSCAN on the noise list obtained before with remaining eps can cluster points that belong to similar density level.

Evaluating Minpts :

After determining the eps values, the next step is to calculate the minpts value. To determine the minpts values, calculate the number of data points in the eps neighborhood of every point in the dataset

$$\text{Minpts} = \frac{1}{n}\sum_{1}^{n} P_{\text{i}}$$

P_i - Number of data point in the eps-neighborhood of point i

n - Total number of data points in the dataset.

So firstly, the number of data objects in **Eps** neighborhood of every point in dataset is calculated one by one. And then mathematical expectation of all these data objects is calculated, which is the value of **MinPts**.

So for each different value of **Eps** we will get corresponding **Minpts** value. We cluster with smaller eps and then cluster remaining data points with larger eps value, so we will get Minpts with lower eps first.

Because the k-dist is sorted in ascending order, we get the eps value from smaller value to larger value, which cause the Minpts value to increase. We cluster with smaller eps and then cluster remaining data points with larger eps value, so we will get Minpts with lower eps first. However, the larger eps indicated sparser density level, when we get the Minpts for larger eps values, the Equation 2 above will also consider the points in the density region since it calculate the Minpts with all data points.

With larger eps, the points in dense region will collect more points, causing the Minpts to become larger. But the remaining points in the noise list after applying DBSCAN are the points with sparser density level when compared to the points that are just clustered in the last process. Thus the situation that sparser region have larger Minpts will happen. Since Minpts refer to the number of points in the eps-neighborhood of a point, it is not helpful to cluster sparser region with larger Minpts values.

$$\text{Minpts}_{\text{j}} = \frac{1}{n}\sum_{1}^{n} P_{\text{i}}\,,\ \text{eps}_{\text{j-1}} \le \text{kdist(i)} \le \text{eps}_{\text{j}}$$

References :

1. Alsabti, Khaled, Sanjay Ranka, and Vineet Singh. "An efficient k-means clustering algorithm." (1997).
2. Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.
3. Paliwal, Priyamvada, and Meghna Sharma. "Enhanced DBSCAN Algorithm." International Journal of Advanced Research in Computer Science and Software Engineering (2013).
4. DA, Priyadharshini, T. Sudha, and G. Usha. "Implementation of Adaptive DBSCAN for Cluster Analysis."