K mediods algorithm

Input:
K
DataSet (X)
Max_iter

Output:
Set of 'k' Clusters

k-mediod(k, {xi}):

Randomly initialize k cluster medoids $\mu=(\mu 1,\mu 2,...,\mu k) \in Rk+1$
Should_continue = true
prevCost=0
while(should_continue)
//this is the first iteration
if prevCost!=0
//choose a random μj for previous_μk
// cluster assignment step:
for i=1 to m:
for j=1,p=1 to k:
// (Use ManhattanDistance dist=|xi−μj|)
//Cp← Assign **xi** to the cluster **Cp** which is closer to medoid **μj**
Cp ← min_manhattan_dist(xi, **μj**)

for all clusters in Ck:
for all points pi in Ck:
//calculate cost of all the points in Ck to the medoid μk
//cost means the distance
cost_k=cost_k+calculate_cost(pi,μk)

//Compare previous and current cost of the **replaced μk**
//get the minimum of the medoids
if prev_cost!=0 and prev_cost<costk
min_cost = prev_cost
else
min_cost=cost_k

prevCost=minCost

```
        //for all clusters, assign prevCost
        previous_μk = μk;

        iter = iter + 1;

  If no change in previousCk and currentCk   or  iter = Max_iter:
       Should_continue = false;
```

Advantages :-

1. K-Medoids method is more robust than k-Means in the presence of noise and outliers

Disadvantages:-

1. K-Medoids is more costly that the k-Means method
2. Like k-means, k-medoids requires the user to specify k
3. It does not scale well for large data sets

K means Algorithm

Input:
K
DataSet (X)
Max_iter

Output:
Set of 'k' Clusters

k-means(k, {xi}):

```
Randomly initialize k  cluster centroids μ=(μ1,μ2,...,μk) ∈ Rk+1
Should_continue = true;
while(should_continue)
  // cluster assignment step:
        for i=1 to m:
             for j=1,p=1 to k:
                // (Use Euclidean Distance dist= // xi−μj|^2)
                Cp ← min_euclidean_dist(xi, μj)
                //Cp← Assign xi to the cluster Cp which is closer to centroid μj
```

```
  previous_μk = μk;
// move centroids step:
        for i=1 to k:
                μk←average of all points assigned to ck
  Iter = iter + 1;
  If  μk =  previous_μk   or  iter = Max_iter:
       Should_continue = false;
```

Advantages

1. K-means is relatively scalable and efficient in processing large data sets
2. The computational complexity of the algorithm is O(nkt)

n: the total number of objects
k: the number of clusters
t: the number of iterations
Normally: k<<n and t<<n

Disadvantage

1. Can be applied only when the mean of a cluster is defined
2. Users need to specify k
3. K-means is not suitable for discovering clusters with non convex shapes or clusters of very different size
4. It is sensitive to noise and outlier data points (can influence the mean value)

Naive Bayes

Input:

- Let D be the training data set which has samples for ‘m’ classes C1,2,3 …..m.
- Let X be the input data that has to be classified.

Output:

- The class label for the input data X

```
naive_bayes_classify(D, X):-
        create a list posterior_probabilities to store posterior probabilities
        // C1,2,3 …..m.
```

```
    for each class_name in class_names:
        posterior_probabilities[class_name] = compute_posterior(class_name, X):
    end for
    classified_class = max(posterior_probabilities);
    return classified_class;

compute_posterior(C, X):
    'n' is the number of features present in the dataset
    P(X|C) = P(X1|C) * P(X2|C) * …….*P(Xn|C);
    P(C) = (Number of samples in C) / (Total number of samples in the training set D)
    P(C|X) = (P(X|C)*P(C)/P(X)
    return P(C|X)
        // P(C) is the prior probability
        // P(C|X) is the Posterior Probability of Class  for a given data X.
        // P(X|C) is the conditional probability of the class for the given X.
```

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Hierarchical clustering algorithm

It is of two types:

i) Agglomerative Hierarchical clustering algorithm or AGNES (agglomerative nesting) and

ii) Divisive Hierarchical clustering algorithm or DIANA (divisive analysis).

Both this algorithm are exactly reverse of each other. So we will be covering Agglomerative Hierarchical clustering algorithm in detail.

Agglomerative Hierarchical clustering -This algorithm works by grouping the data one by one on the basis of the nearest distance measure of all the pairwise distance between the data point. Again distance between the data point is recalculated but which distance to consider when the groups has been formed? For this there are many available methods. Some of them are:

1) single-nearest distance or single linkage.
2) complete-farthest distance or complete linkage.
3) average-average distance or average linkage.
4) centroid distance.
5) ward's method - sum of squared euclidean distance is minimized.

This way we go on grouping the data until one cluster is formed. Now on the basis of dendogram graph we can calculate how many number of clusters should be actually present.

Algorithmic steps for Agglomerative Hierarchical clustering

Let $X = \{x_1, x_2, x_3, ..., x_n\}$ be the set of data points.

1) Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.

2) Find the least distance pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

3) Increment the sequence number: m = m +1.Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to L(m) = d[(r),(s)].

4) Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster(k) is defined in this way: d[(k), (r,s)] = min (d[(k),(r)], d[(k),(s)]).

5) If all the data points are in one cluster then stop, else repeat from step 2).

Divisive Hierarchical clustering - It is just the reverse of Agglomerative Hierarchical approach.

Advantages

1) No apriori information about the number of clusters required.

2) Easy to implement and gives best result in some cases.

Disadvantages

1) Algorithm can never undo what was done previously.
2) Time complexity of at least O($n^2 \log n$) is required, where *'n'* is the number of data points.
3) Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:
   i) Sensitivity to noise and outliers
   ii) Breaking large clusters
   iii) Difficulty handling different sized clusters and convex shapes
4) No objective function is directly minimized
5) Sometimes it is difficult to identify the correct number of clusters by the dendogram.

Pesudo-Code:

Input:

- DataSet X = $\{x_1, x_2, x_3, ..., x_n\}$

Output

- Single Cluster of All points

Let X = $\{x_1, x_2, x_3, ..., x_n\}$ be the set of data points.
Agnes_clustering():

Initialize level of Clustering L(0) = 0 and sequence number m = 0.
Should_continue = true;
Represent

while(should_continue)
    //where the minimum is over all pairs of clusters in the current clustering.
    find_least_distance_cluster (r), (s) ← d[(r),(s)] = min d[(i),(j)]
    //Increment Seq number
    m = m +1

    Merge clusters (r) and (s) into a single cluster to form the next clustering m.

    Set the level of this clustering to L(m) = d[(r),(s)].
    //find centroid of the points present in the new cluster; centroid represents the new cluster;
    compute_centroid_for_the_new_cluster()

    update_distance_matrix()
    //by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and //column corresponding to the newly formed cluster. The distance between

the new cluster, denoted (r,s) and old cluster(k) is defined in this way: //d[(k), (r,s)] = min (d[(k),(r)], d[(k),(s)]).
//or It Could be based on Centroid Approach

If all_data_points in Single_cluster
Should_continue = false;
// else repeat; The loop will continue.

Hierarchical cluserring:

1. Bottom Up(Agglomerative) - Each point is a cluster of its own.
In each step we find two closest cluster and combine them into one cluster.

Questions:

1. How do you represent a cluster of more than one point
represent the location of each cluster, to tell wch pair of clusters is closest.
- We can do that by its centroid- avg of its points

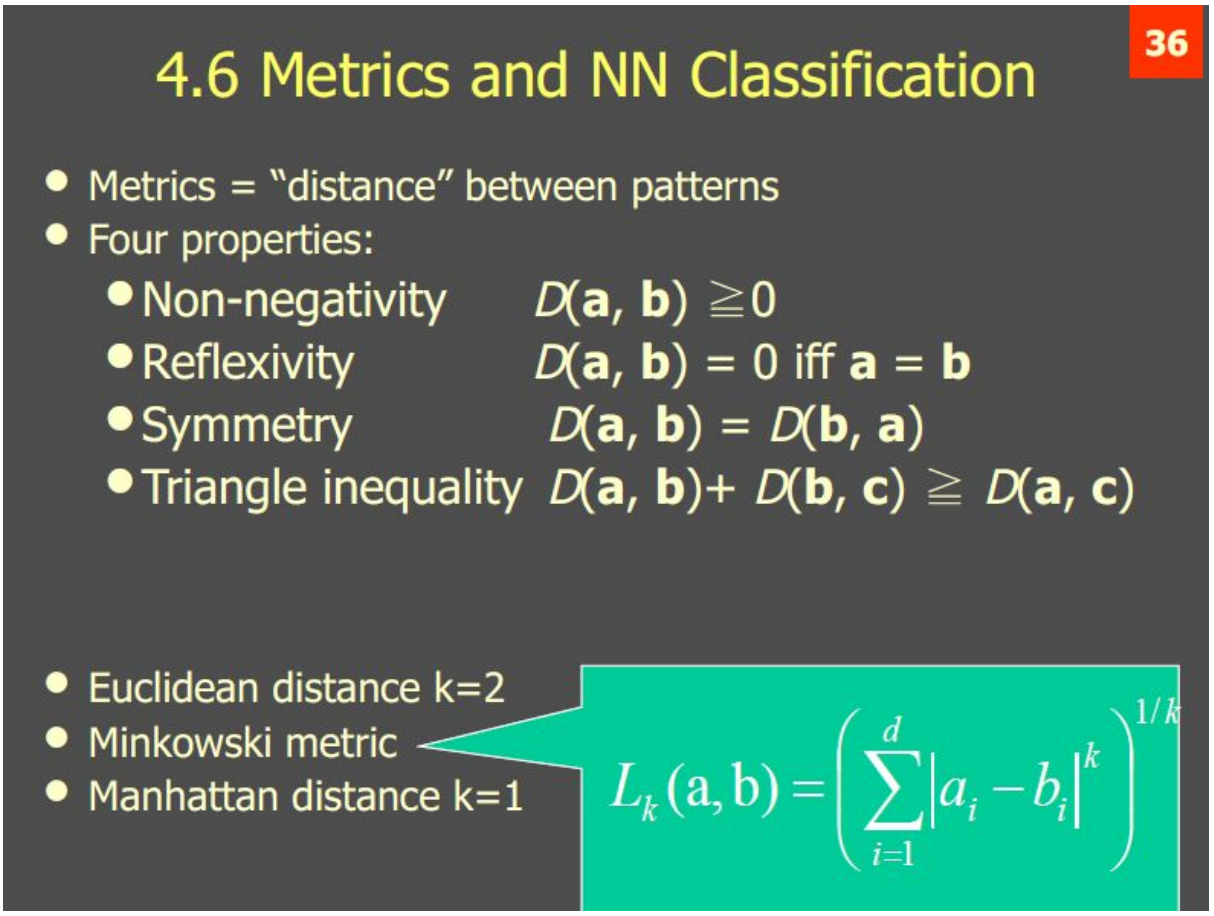
2. Determine the nearness of clusters
- Measure the cluster distance by distance of centroids.

3.When to stop

-In the non-euclidean space the representation of the cluster will be one of the point in the cluster and is called Clustroid.

2.Top Down(Divisive) - All datapoints are in single custer
we recursilvely split the cluster

Properties of Distance :



EM Algorithm :

Expectation Maximization

Model Learning using EM

1: **Input**:Dataset $\mathcal{X} = \{\vec{X}_1, \ldots, \vec{X}_N\}$, t_{min}.
2: **Output**: Θ.
3: {**Initialization**}: K-Means Algorithm.
4: **while** relative change in log-likelihood $\geq t_{min}$ do
5: {[**E Step**]}:
6: **for all** $1 \leq j \leq K$ do
7: Compute $p(j|\vec{X}_i)$ for $i = 1, \ldots, N$.
8: **end for**
9: {[**M step**]}:
10: **for all** $1 \leq j \leq K$ do
11: Update the mixing parameter π_j.
12: Update the mean $\vec{\mu}_j$.
13: Update standard deviation Σ_j.
14: **end for**
15: **end while**

Template for Question 4:

Data preprocessing

Feature extraction

Mixture Model Distribution-MLE

- Input:Dataset
- Check dataset type-missing/not missing
- Identify the distribution based on features.
- Identify whether univariate or multivariate.
- Do MLE for the respective distribution
- Mixture Model (Output: maximum parameters)

EM algorithm- Optimizing maximum parameters

- Input:Dataset, threshold, Mixture Model
- Output:optimized maximum parameters

AIC/MDL- (To choose the best value of k)

- Input : Mixture Model
- We compute the best value of k

(OR)

Any Clustering algorithm

- Explain the clustering algorithm
- Tell the distance used
- Why we use it in this context?
    - If hierarchical use Agnes
    - If k is given, use k-means,k-medoids
    - Use k-medoids when outliers are more
- Pseudo Code with Input and output

Divergence-distance between two mixture model distributions, **do only if every class is a distribution.**

Postprocessing