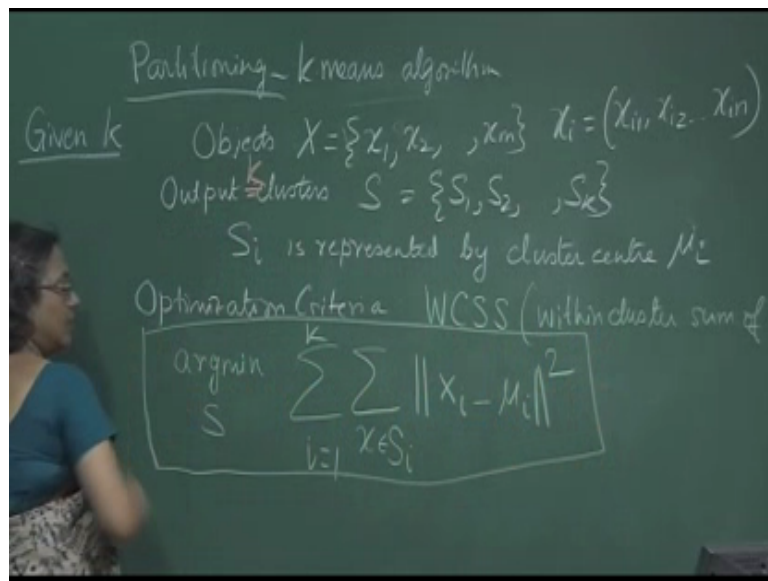


Introduction to Machine learning
Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 38
Kmeans Clustering

Welcome to today's class. Today we will talk about a particular clustering algorithm. In the last class we discussed about the various types of clustering outputs.

(Refer Slide Time: 00:34)



We mentioned partitioning as one of the methods of clustering. Today we will look at a partitioning based algorithm specifically the K-means algorithm. In this algorithm we are given k and we have to produce k clusters. So, k is given to us and we are given a set of objects or instances, let us call it x comprising of m objects x_1, x_2, \dots, x_n . And each object, let us assume is described in terms of n features. We can write that x_i , can be written in terms of the n features $x_{i1}, x_{i2}, \dots, x_{in}$.

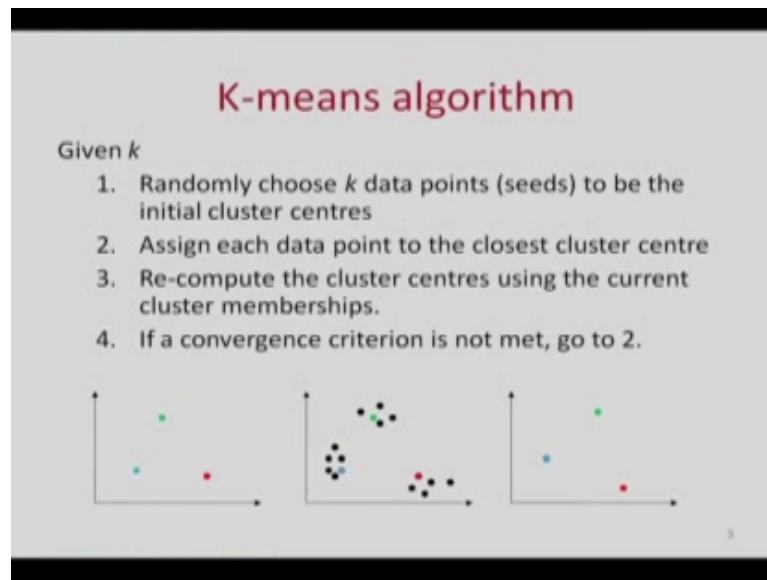
So now, what we have to do is we have to output k clusters, so S the set of clusters comprising of S_1, S_2, \dots, S_k . In this particular clustering algorithm the clusters that we output will form a partition of the objects, that is each part cluster is disjoint and together they cover the entire set of objects. And each cluster S_i is represented by the cluster center, let us call it μ_i . So, μ_i it notes the parameter with respect to the cluster s_i .

Now we have to find k clusters, and what could be our criteria for choosing the clusters. We want to find the clusters so that it can optimize some chosen criteria. One possible optimization criteria is the within clusters sum of square distance. We mention that a cluster is such that objects within a cluster are similar to each other. Objects belonging to different clusters are different from each other. So, one possible optimization criteria is Within Cluster Sum of Squares distance, which we can write as WCSS. Which can be computed as, we will right this later.

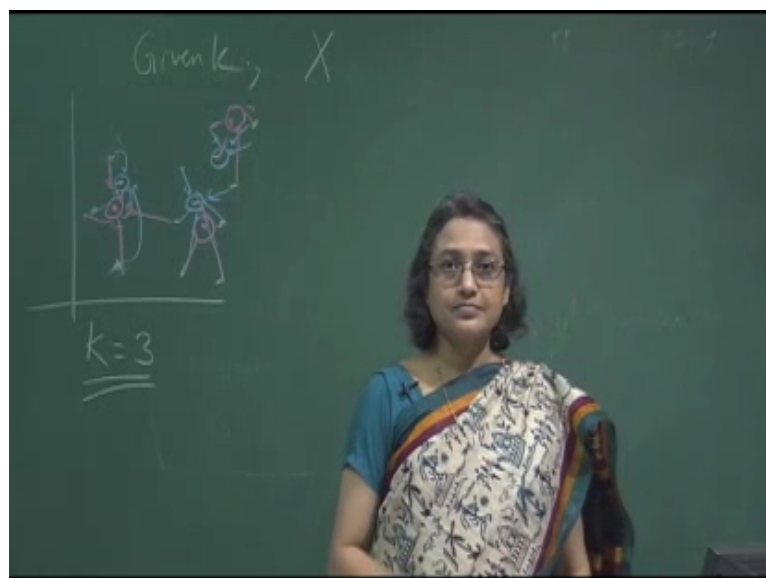
So, given a particular set of clusters the within clusters sum of distance is obtained as sum over i equal to 1 to k where k is the number of clusters, then summation x belongs to S_i . For all objects belonging to S_i , we find x_i minus μ_i also. We look at each of the k clusters for one cluster we look at all the objects in the cluster and find its distance from the cluster set. So, some of the square distance for all the elements in this clusters summed over all the clusters gives me the within cluster sum of square. Now for a given set of cluster we can compute this. We want that set of clusters for which this is minimized. So what we want is, find out that set of clusters for which this quantity is minimized.

Now finding a set of clusters which exactly minimizes these criteria is a hard problem. So what we do is that, we will describe a simple heuristic algorithm called K-means algorithm which is designed by Macqueen in 1967. K-means was proposed by Macqueen 1967 and this is a heuristic algorithm, which tries to come up with the set of clusters that tries to minimize this within clusters some of squares distance even though it may not be the exact minimum. Now let us describe the K-means algorithm.

(Refer Slide Time: 06:39)



(Refer Slide Time: 06:43)



In the K-means algorithm what we do is that here given k and we are given the set of objects x . So, assuming that we have two features we have different points representing the objects. And suppose we are given k equal to 3. So this is x these points are x and suppose we are given k equal to 3. And we need to find a set of 3 clusters. So, what we do in K-means algorithm is we identify randomly k cluster centers. We choose k of the data points, let us we choose this point, this point, this point, and this point. We randomly pick up 3 of the points and assign them as the cluster center or cluster seats.

After we have assigned that now what we do is that we look at each of the points and assign them to the closest cluster. So, this point gets assigned to this one, this gets assigned to this one, this gets assigned to this one, this point gets assigned here, this gets assigned here, this gets assigned here, this gets assigned here, this gets assigned here. We randomly pick up 3 seats and assign each of the data points to the closest seat. After we have done that now these have gone to the same seats, so we now take these points and recompute the cluster centers. Suppose, the new cluster center of these three points are found here, suppose the new center of these points is found here, and the new center of this point is found here.

Now on the second iteration, we take these as the cluster centers and repeat the process. That is now we again we look at the points and find out which cluster it will be, they will get assigned too. For example, now it may be possible that this is the assignment. Now based on the new cluster centers we again try to reassign the points to the clusters and we continue this process. And we continue until the convergence criteria are met.

So, the algorithm is very simple given k we randomly choose k data points to be the initial cluster centers. We assign each data point to the closest cluster center. We recompute the cluster centers using the current cluster memberships. And, if the convergence criteria are not met go to two we will soon see what type of convergence criteria we can look at. This example also illustrates we randomly choose these two points as seats based on that we assign points to cluster seats and then re-compute the cluster centers and this process is continued.

(Refer Slide Time: 10:16)

Stopping/convergence criterion

OR

1. no re-assignments of data points to different clusters
2. no (or minimum) change of centroids
3. minimum decrease in the *sum of squared error*

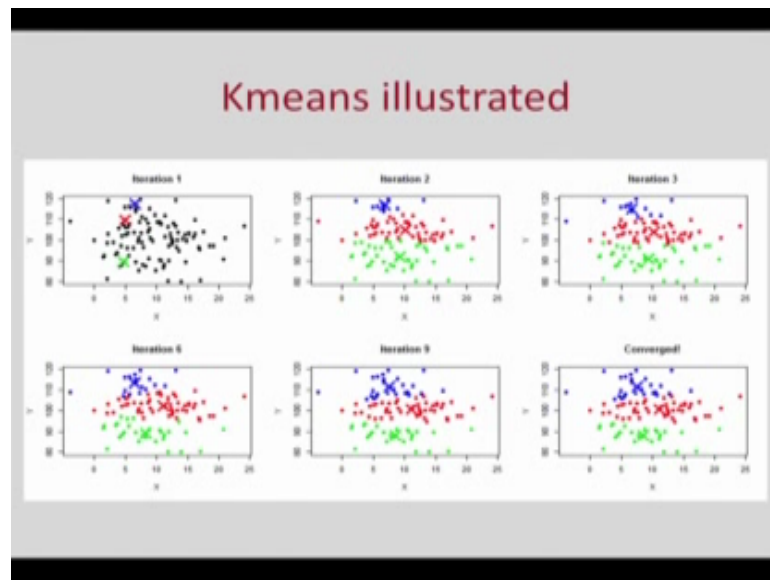
$$SSE = \sum_{i=1}^k \sum_{x \in S_i} \|x_i - \mu_i\|^2$$

4

So, when do we stop? We can stop when we reach a situation when between two iterations the cluster centers do not change. That is, there is no reassignment of points to the different clusters. We can stop if there is no change in the mean, or very little decrease in the sum of square errors. So, these are the possibilities under which we can stop.

Now this algorithm, we can show that within cluster the sum of squares distance will go on decreasing from iteration to iteration. And this process is bound to convert, because this is going on decreasing. So, this process is bound to convert. This is the converging process it will converge to local minima of within cluster sum of square distance which may or may not meet the global (Refer Time: 11:28).

(Refer Slide Time: 11:32)



This is an illustration of the K-means algorithm on a particular data set, on this data set of the black points carried over for 9 iterations. In the first iteration, the red blue and green are chosen as cluster centers. As a result this is the assignments of the points and based on this assignment it shows the new cluster centers iteration 2. Iteration 3 these are the cluster centers. Iteration 4, 5 and 9, and then we see that it has converged. This is an example of the K-means algorithm applied.

(Refer Slide Time: 12:12)

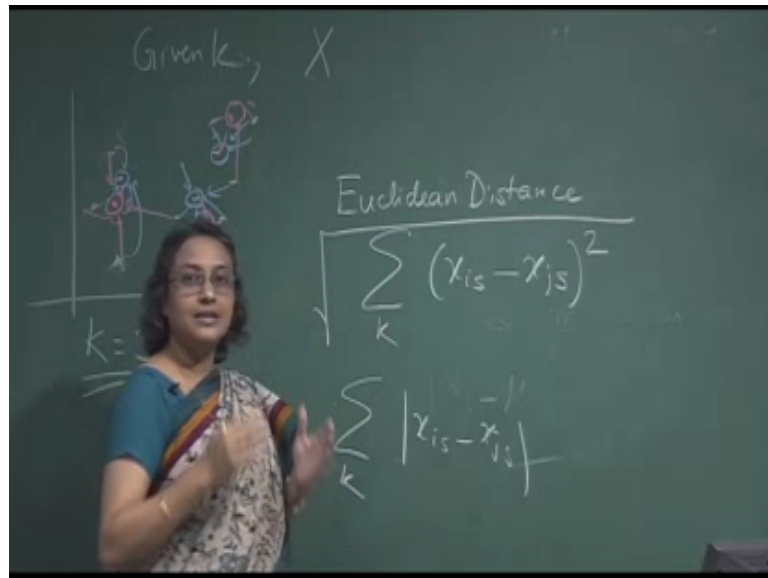
Similarity / Distance measures

- Distance metric (scale-dependent)
 - Minkowski family of distance measures
$$d(x_i, x_j) = \left(\sum_{s=1}^n |x_{is} - x_{js}|^p \right)^{1/p}$$

Manhattan (p=1), Euclidean (p=2)
 - Cosine distance

Now, when we use the K-means algorithm we have to choose a measure of similarity or a measure of distance or dissimilarity. Now there are different distance measures that one could use, most of you will be familiar to the Euclidean Distance.

(Refer Slide Time: 12:32)



So, Euclidean distance is the sum of the squares. The Euclidean distance is given by x_i minus x_j square root over, if there are k clusters for each clusters it will look at the points that is belonging to that cluster and the sum of square distance between the points. So, this is what you are familiar with.

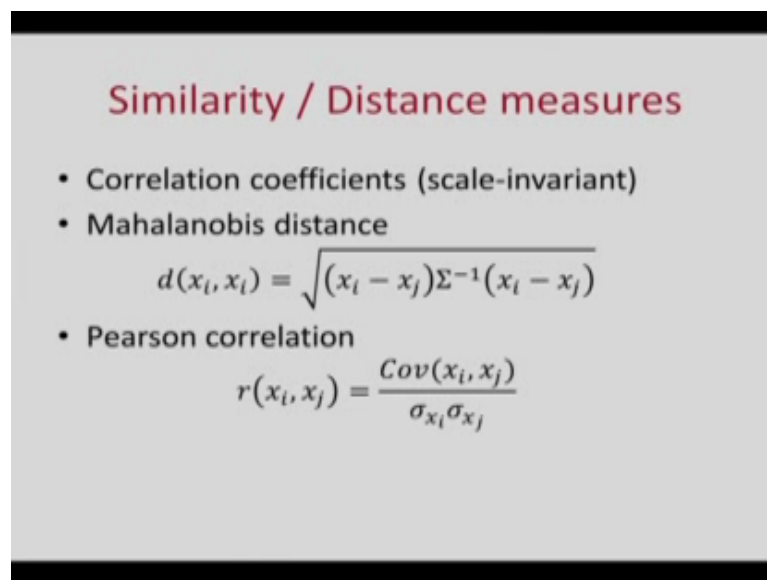
Another variation of the Euclidean distance function that one can use is the Manhattan distance, which is simply $\sum_k x_i$ minus x_j . In fact, these two and something similar can be brought under Minkowski measures. Minkowski family of distance measure is a more general measure of distance between two points, x_i and x_j where there are n features. x_i and x_j are two points there are n features and the distance between them is equal to, I think I made a mistake so this not k this is sigma over s equal to 1 to n , where n is the number of features.

So what we do is that we look at, this is the distance between two points each having n features and we take the distance from every dimension. So, x_i minus x_j is the difference in a particular dimension of these two objects. In Euclidean distance we take x_i minus x_j whole square summation and the whole thing root over. In Manhattan

distance called city block distance we just take x_i minus x_j absolute value and we do the summation.

There is another measure called the Cosine measure, which is a very popular measure when we are dealing with text documents. In text documents, we often represent a simple representation of a text document is as a bag of words. We look at the words in the document and the number of words. We look at the set of words and the frequency of each word. So, when we represent the documents as a vector the distance between two different documents can be used as a cosine of the two vectors representing the documents. The cosine measure is very popular when we work with text documents, Euclidean distance does not work well with text documents.

(Refer Slide Time: 16:02)



Similarity / Distance measures

- Correlation coefficients (scale-invariant)
- Mahalanobis distance
$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}$$
- Pearson correlation
$$r(x_i, x_j) = \frac{\text{Cov}(x_i, x_j)}{\sigma_{x_i} \sigma_{x_j}}$$

Then there are other types of distance measures. For example, the Mahalanobis distance between two x_i and x_j is given by $(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)$. Then you have correlation coefficient, for example Pearson correlation which is given by covariance of x_i, x_j divided by the standard deviation of x_i multiplied by standard deviation of x_j .

(Refer Slide Time: 16:31)

Convergence of K-Means

- Recomputation monotonically decreases each square error since (m_j is number of members in cluster j):
 $\sum (x_i - a)^2$ reaches minimum for:
$$\sum -2(x_i - a) = 0$$
$$\sum x_i = \sum a = m_j a$$
$$a = 1/m_j \sum x_i = c_j$$
- K-means typically converges quickly

So, different distance measures can be used depending on the dominium which applying this algorithm. Now as I mentioned the K-means algorithm is guaranteed to converge, because as you go to the iterations the within cluster sum of square distance will go on monotonically decreasing, and this is the proof.


Suppose, you have k clusters m_1, m_2, m_j are the means of the clusters, and suppose m_j is the number of members in cluster j . Now $\sum x_i - a$ whole square this quantity reaches minimum under such following conditions. These conditions are given as; if you take the differentiation of this you get $\sum -2(x_i - a) = 0$. From this we will find that $\sum x_i = \sum a$, and because the cluster has m_j members this is equal to $m_j a$. Or, you take a equal to $1/m_j \sum x_i$.

So, if you take a equal to $1/m_j \sum x_i$ that you take as the center of the cluster. This is what you are doing in K-means; we are re-computing the cluster center as the center of the points which belongs to the cluster. And if we do that this will give you a reduction in the quantity of within cluster sum of squares.

(Refer Slide Time: 18:19)

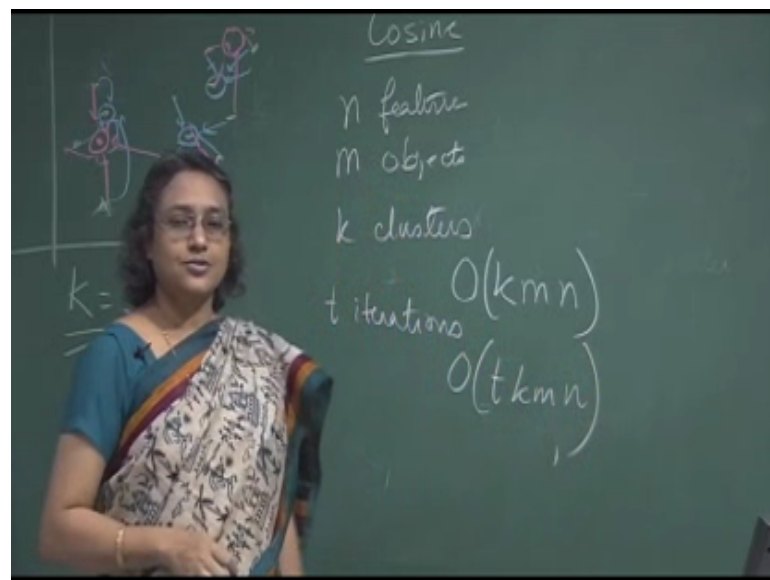
Time Complexity

- Computing distance between two items is $O(n)$ where n is the dimensionality of the vectors.
- Reassigning clusters: $O(km)$ distance computations, or $O(kmn)$.
- Computing centroids: Each item gets added once to some centroid: $O(mn)$.
- Assume these two steps are each done once for t iterations: $O(tknm)$.



Now, what is the time complexity of this algorithm? The time to compute the distance between two objects is $O(n)$, where n is the number of features. The reassignment of the cluster takes $O(km)$. So, you (Refer Time: 18:39) so many distance computations, why?

(Refer Slide Time: 18:44)



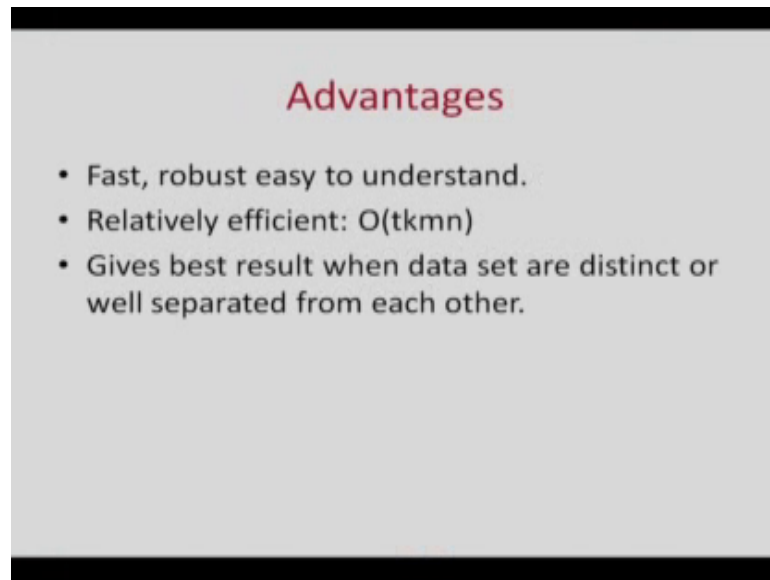
The chalkboard contains the following handwritten text:

- Cosine
- n features
- m objects
- k clusters
- t iterations
- $O(kmn)$
- $O(tknm)$

Every point there are m object there are n features, there are m objects there are k clusters. In the reassignment steps every object. You check its distance with each of the cluster centers and reassign it to the closest. So, the reassignment face will take $O(km)$ computations of distance and each computation of distance is takes time n . So, each

iteration takes $O(kmn)$ time. If there is t iteration then the total time taken is $O(tkmn)$. For K-means algorithm usually they converge quite quickly and t is usually quite small. So, the time to run this algorithm is $O(tkmn)$.

(Refer Slide Time: 19:46)

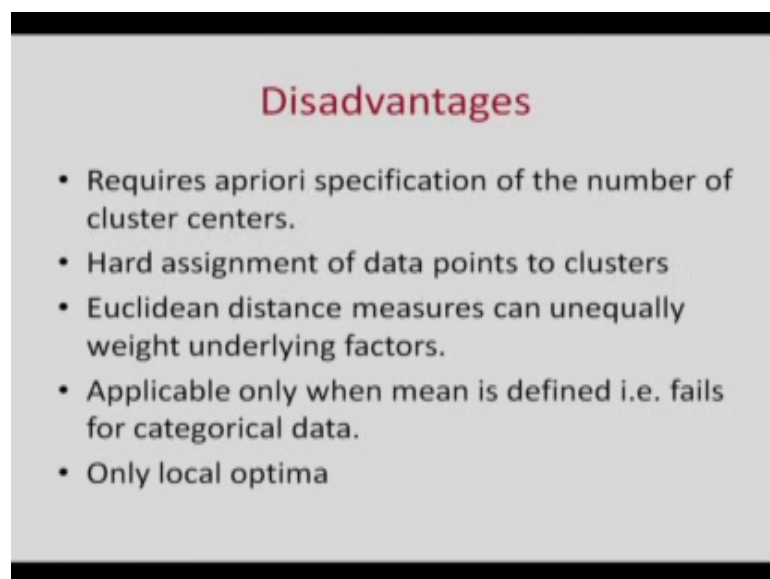


Advantages

- Fast, robust easy to understand.
- Relatively efficient: $O(tkmn)$
- Gives best result when data set are distinct or well separated from each other.

So, what are the advantages of K-means algorithm? It is the fast algorithm, robust algorithm easy to understand, relatively efficient, and it gives good result when the data set adjusting or well separated from each other.

(Refer Slide Time: 20:00)



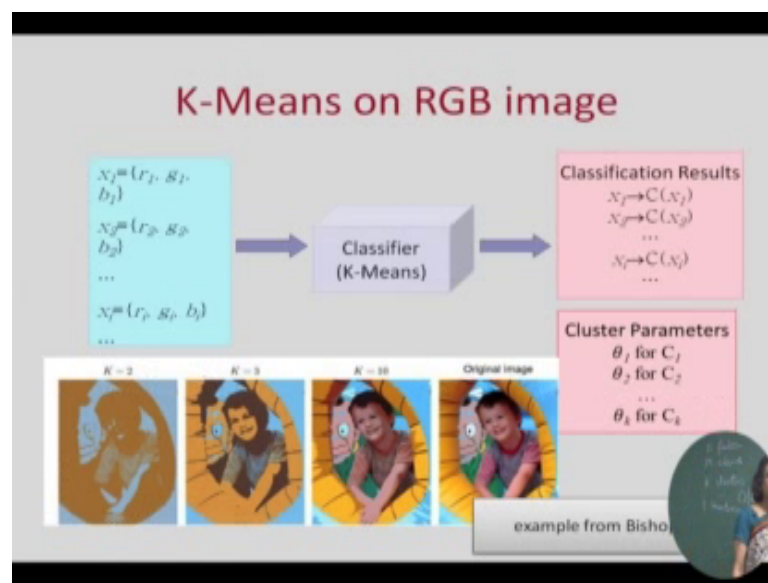
Disadvantages

- Requires apriori specification of the number of cluster centers.
- Hard assignment of data points to clusters
- Euclidean distance measures can unequally weight underlying factors.
- Applicable only when mean is defined i.e. fails for categorical data.
- Only local optima

But there are certain disadvantages. It requires apriori specification of the number of cluster that is k has to be given. Also the K-means algorithm that we have described there is a hard assignment or partitioning of the objects. So, each object has to belong to exactly one cluster. There are situations where one object may have similarity to two different clusters, and there is soft clustering algorithm which assigns an object to multiple clusters. But in K-means there is a hard assignment to a single cluster. The distance measured, depending on the domain that the distance present has to be carefully chosen.

If we would use Euclidean distance measured it leads to unequal weighting of the underlying factors, and it is not a very good distance function to be used for some types of data. Also K-means algorithm will only work when the means of the objects can be defined. So, if the features that we are using have lot of categories then you cannot find the means, so K-means is not very appropriate. And we have seen K-means only gives you a local optimal not the global optimal.

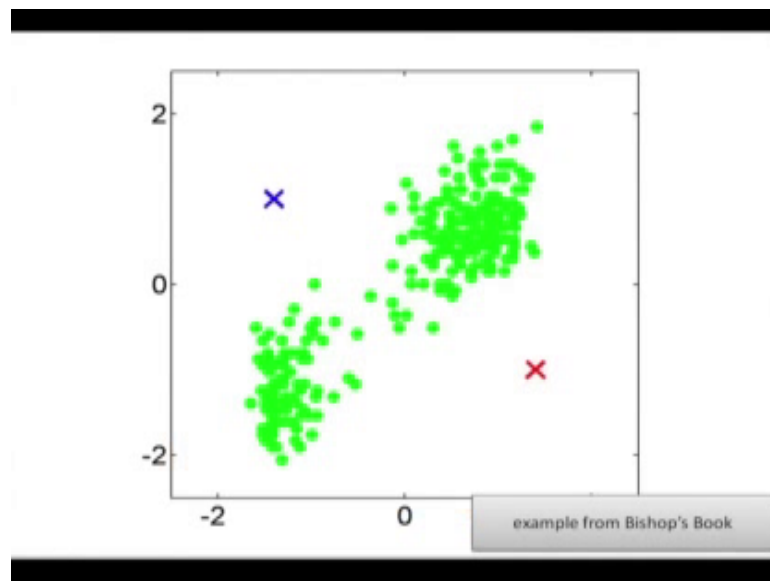
(Refer Slide Time: 21:26)



This picture taken from Bishop's book on machine learning shows, K-means algorithm applied to an image. So, if you look at an image and image can be described in terms of the pixels. There are various of pixels in an image, and each pixel is described by the RG and B values. Now a typical image can have very many colors. Suppose you want to denote the image in terms of a few numbers of colors.

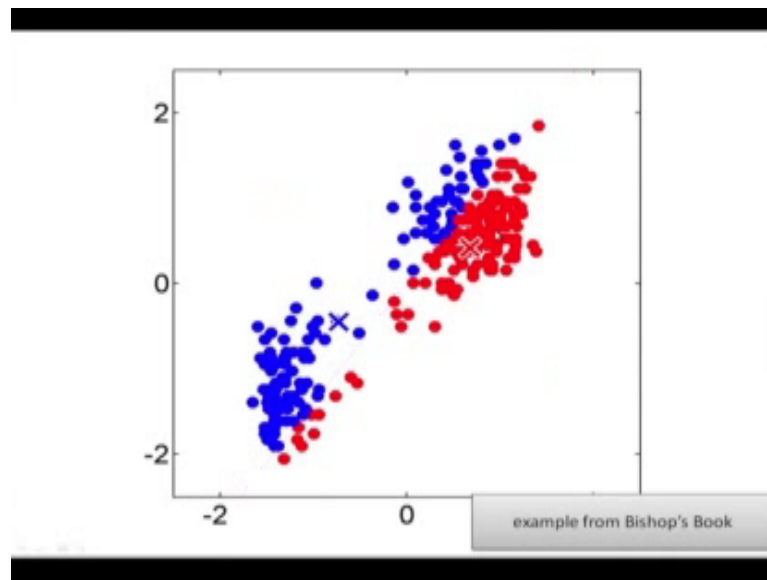
So what you can in that, you can take every color that is there in the image and cluster the color. After that suppose you take k equal to 10 you can redraw the image using only 10 colors. If you look at the slide we have x_1, x_2, x_i as the different pixel colors, we use K-means algorithm which comes up with k clusters of colors. And this shows when k equal to 2, when we use 2 colors we get this image, 3 colors we get this image, 10 colors you get this image where as this is the original image. This is an example of application of K-means.

(Refer Slide Time: 22:59)



I will also quickly running to another example from the book, where these are the green points are the objects, the red and blue are the initial cluster centers.

(Refer Slide Time: 23:19)



This shows the reassignment of the point to the cluster centers, the new cluster centers, again the reassignments of the points, the new cluster centers, reassignments of the points, new cluster centers, reassignments of the points, and there is convergence.

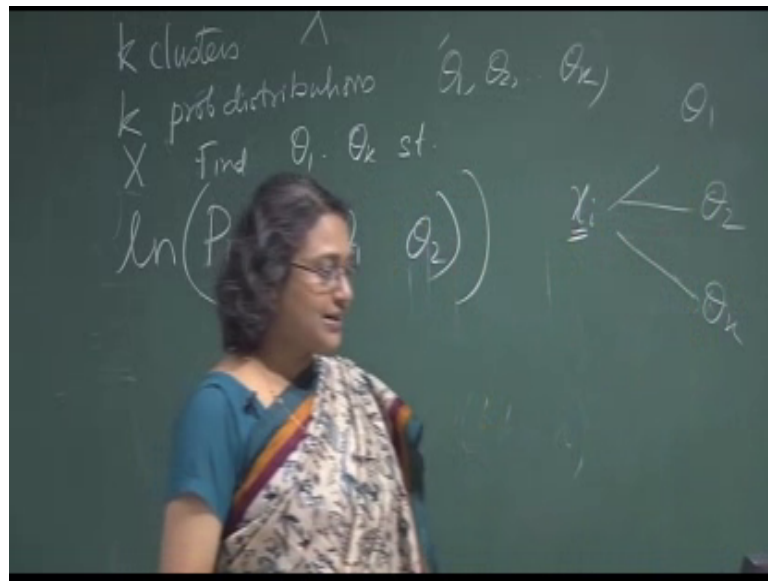
(Refer Slide Time: 23:37)

Model-based clustering

- Assume k probability distributions with parameters $\theta_1, \theta_2, \dots, \theta_k$
- Given data X , compute $\theta_1, \theta_2, \dots, \theta_k$ such that
 $Pr(X|\theta_1, \theta_2, \dots, \theta_k)$ [likelihood] or
 $\ln Pr(X|\theta_1, \theta_2, \dots, \theta_k)$ [log likelihood]
is maximized.
- Every point $x \in X$ may be generated by multiple distributions with some probability

I will very briefly talk about Model-based cluster. In K-means we have an iterative algorithm. Now I will very briefly describe a model-based clustering algorithm. In model-based clustering what we do is that we assume that there is a model from which the data is being generated.

(Refer Slide Time: 24:11)



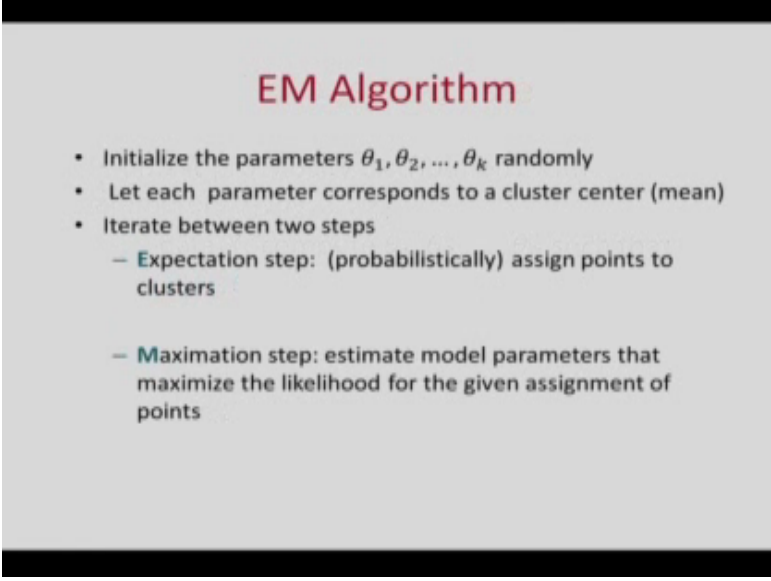
So, we want to find k clusters, we assume that the model of the clusters is given by k probability distributions, and the model parameters are θ_1 , θ_2 , θ_k . For example, we can assume that the model each cluster is modeled by a Gaussian distribution and we can think of θ_1 as the mean of the Gaussian distribution corresponding to cluster one. We can assume that all the Gaussians have a same variance. And suppose, we fix the variance of one, we could also say that the parameters are μ and variance, mean and variance of the Gaussian distribution.

So, let us assume that θ_1 , θ_2 , θ_k is parameters of the probability distribution corresponding to the k clusters. Now we are given data x , x is my set of objects. So, we have to find θ_1 , θ_2 , θ_k . In model best clustering we assume that the data has been the data is being generated from a model, we have given the data; we have to find the parameters of the model. So, we have to find θ_1 , θ_2 , θ_k .

Let us call this big theta, so we have to find the parameters θ_1 , θ_2 , θ_k such that the likelihood of the data is the maximum. For that we have to find that set of parameters for which the probability of generating x is highest. That is, we want to find θ_1 , θ_2 , θ_k such that, probability x given θ_1 , θ_2 , θ_k is the highest or equivalently such that the log of this probability is the highest. So, this is called the likelihood of the data this is called the long likelihood of the data and we want to maximize this long likelihood of the data

Also in this model we have a slight deviation from the earlier model. We assume that a particular point x_i has, x_i need not belong only one cluster rather x_i has a probability of belonging to different clusters. So, x_i may have certain probability belonging to the cluster θ_1 , another probability of belonging to the cluster θ_2 , another probability of belonging to the cluster θ_k . So, that the sum of this probability is 1. So, every point can be generated by multiple distributions with some probability this is what we assume.

(Refer Slide Time: 27:34)

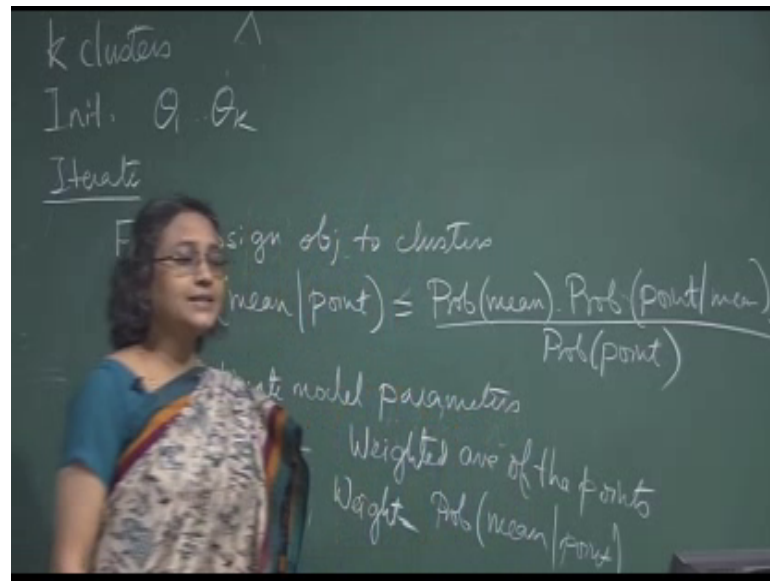


EM Algorithm

- Initialize the parameters $\theta_1, \theta_2, \dots, \theta_k$ randomly
- Let each parameter corresponds to a cluster center (mean)
- Iterate between two steps
 - Expectation step: (probabilistically) assign points to clusters
 - Maximization step: estimate model parameters that maximize the likelihood for the given assignment of points

Now for this, there is a standard algorithm in order to do this and that algorithm is called the EM Algorithm.

(Refer Slide Time: 27:58)

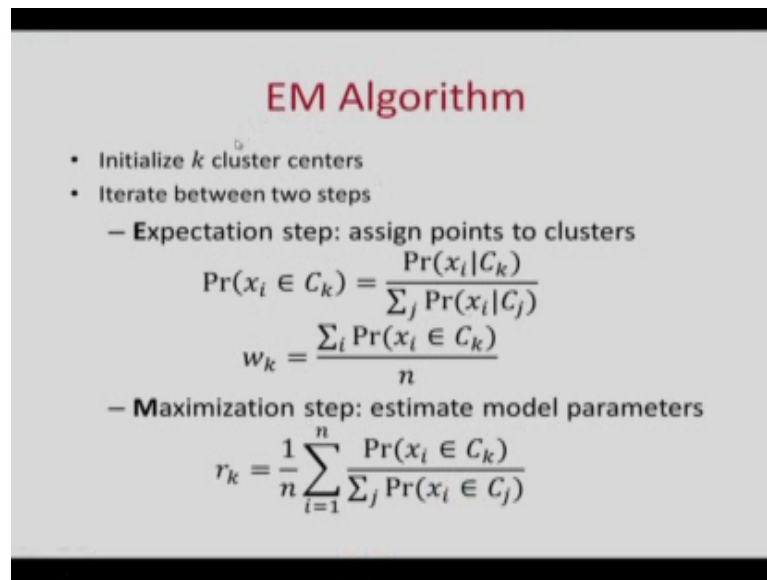


The EM algorithm proceeds as follows. We initialize theta 1, theta 2, theta k randomly. So we initialize randomly, when we assume that theta 1, theta 2, theta k in a simplest way case we assume that they represent the cluster means. Now we iterate between two steps and these two steps are the E-step and the M-step. In the E-step, we assign the objects to clusters. And in the M-step, we estimate the model parameters based on this assignment so as to maximize the likelihood of the given assignments of points. So, we do a maximum likelihood estimate of the model parameters. So, this is the EM algorithm and these two steps are iterated.

For example, in the E-step in order to assign objects to clusters as we said that we need not do a hard assignment rather we do a probabilistic assignment. We probabilistically assign points to clusters. So, we want to find probability of the cluster mean given the point or given the objects. For example, for this we can use base rule which tells us that this equal to probability of the mean times probability of point given mean times divided by probability of the point. And from the data we can find out probability of point given.

In the M-step we find the weight. We find that each mean the maximum likelihood estimate of the mean is the weighted average of the points belonging to each cluster. And what is the weight? Weight is the probability of the point belonging to that cluster. Where, weight is equal to probability of mean given point. So, this is how the EM algorithm process.

(Refer Slide Time: 31:17)



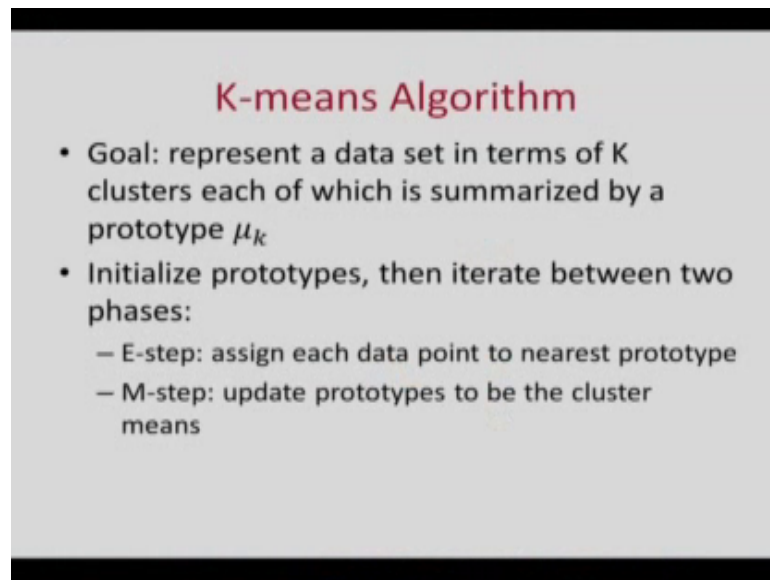
EM Algorithm

- Initialize k cluster centers
- Iterate between two steps
 - Expectation step: assign points to clusters
$$\Pr(x_i \in C_k) = \frac{\Pr(x_i | C_k)}{\sum_j \Pr(x_i | C_j)}$$
$$w_k = \frac{\sum_i \Pr(x_i \in C_k)}{n}$$
 - Maximization step: estimate model parameters
$$r_k = \frac{1}{n} \sum_{i=1}^n \frac{\Pr(x_i \in C_k)}{\sum_j \Pr(x_i \in C_j)}$$

For example, in this particular case if you look at the slide in this EM algorithm we initializing k cluster centers. In the E-step we assign points to clusters probability that the point x_i belongs to the cluster C_k is given by probability of x_i given C_k divided by summation over j probability x_i given C_j . And the weight can be computed as summation over i probability x_i into let C_k by n where n is the points belonging to the cluster.

In the maximization step, we estimate the model parameters as r_k , is r_k is the 1 by n summation i equal to 1 to n probability x_i included in C_k divide by sigma j probability. So, this is one particular instance of the EM algorithm. As you can see that the K-means algorithm is also an example of the EM algorithm, but with certain restrictions so that we allow the point belonging to only 1 cluster.

(Refer Slide Time: 21:26)



K-means Algorithm

- Goal: represent a data set in terms of K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
 - E-step: assign each data point to nearest prototype
 - M-step: update prototypes to be the cluster means

So, in a K-means algorithm we represent a data set in terms of k clusters each of which is summarized by the cluster center μ_k . We initialize the means iterate between two faces. In the K-means algorithm in the E-step we assign each data point to the nearest prototype, M-step we update prototypes to be the clusters. This is an example of the EM algorithm, but in general EM algorithm is more general and we can use model base Clustering.

With this I stop today's lecture.

Thank you.