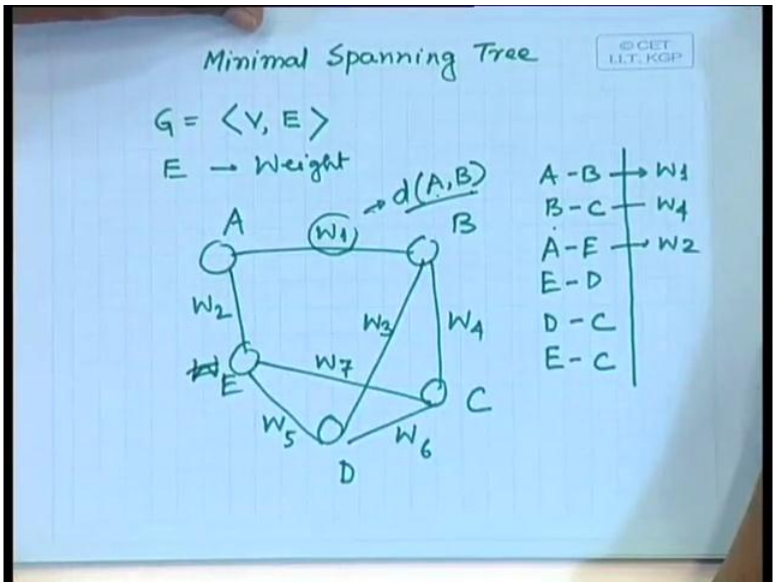
Pattern Recognition and Application
Prof. P.K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 36
Clustering Using Minimal Spanning Tree

Good morning. So, in the last class we have talked about the clustering using Batchelor and Wilkins algorithm, and then we have talked about a graphical approach, where the graph is represented by an adjacent symmetrix. And from the adjacent symmetrix we have seen that how we can cluster the given set of feature points. So, today we will discuss about another robust technique, which is spanning tree based clustering technique. So, when we talk about spanning tree or in our case we will be talking about minimal spanning tree. A spanning tree is nothing but a tree representation of a graph. So, you know that a difference between a graph and a tree. In case of a graph, you can have a cycles in a tree, you cannot have a cycle. So, given two nodes I can have only one path between two given nodes. I cannot have multiple path between two given nodes, which is allowed in case of a graphical representation. So, when we talked about the minimal spanning tree, there is the graph is represented as an weighted graph.

(Refer Slide Time: 01:32)



Minimal Spanning Tree

$G = \langle V, E \rangle$

$E \rightarrow$ Weight

A - B	$\rightarrow W_1$
B - C	W_4
A - E	$\rightarrow W_2$
E - D	
D - C	
E - C	

Weighted graph in the sense, when I say that of edges, every edge has an weight or a cost associated with the edge. So, if I have a graph something like this, say having a number of nodes. So, this is a graphical representation say any arbitrary curve something like this, the nodes are A, B, C, D, E. Then every node will have a weight associated with it, say W 1, W 2, W 3, W 4, W 5, W 6, W 7 and so on.

So, when we talk about the minimal spanning tree, as we said that a graph can be represented by in various ways. One of the representation that we have discussed in the last class is the adjacent symmetrix representation. That means if there is an edge between two nodes in a graph then the corresponding element in the adjacent symmetrix will be equal to 1. If the nodes are not connected, the corresponding elements in the adjacent symmetrix will be equal to 0. In case of a weighted graph I can have a matrix representation, where in the corresponding elements in the matrix I will have a value corresponding to weight of the corresponding edge. The other form of representation of the graph is by an edge list.

So, however I have an edge, I have edge between A and B. So, this graph can be represented in the form AB, then BC, then let us say this is W E, then AE, then ED, DC and we have EC. So, I have the set of edges and this set of edges along with the vertices represent the graph when I consider an weighted edge. Then for every edge I will have the corresponding weight. So, this will have W 1. Similarly, BC will have W 4, AE will have W 2 and so on, ok?

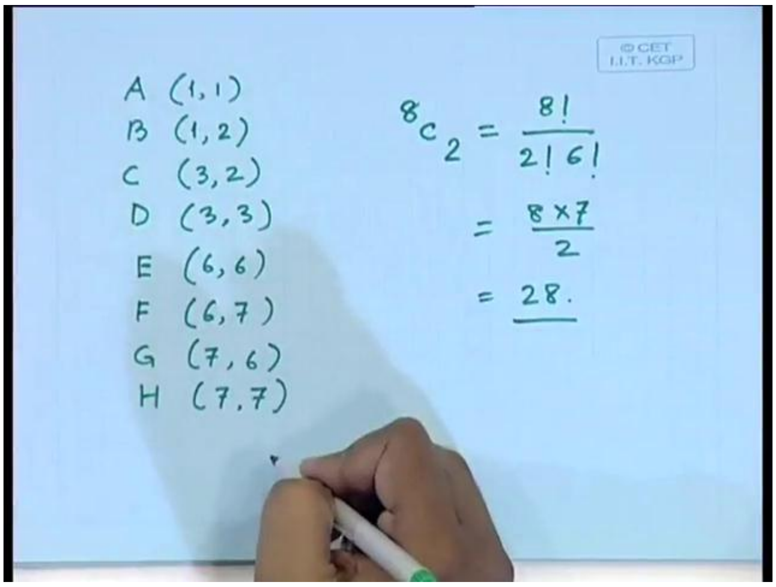
So, given a set of feature vectors, what I have to first to do is I have to represent that feature vector in the form of a weighted graph. So, if I assume that the graph is completely connected between, that means in every pair of nodes I have an edge, ok? The weight of the edge will be the distance between those two vertices or those two feature vectors, corresponding feature vectors. So, in our case this weight W is nothing but the distance between the nodes A and B, where node A represents one feature vector, node B also represents another feature vector. So, there is the distance between feature vectors A and B which represents, which will represent the weight of the corresponding edge, ok?

So, if I represent the set of feature vectors by a completely connected graph. Completely connected graph means between every pair of nodes I will have an edge and the corresponding edge will have an weight, which is nothing but the distance between the corresponding vertices, which are connected by the edge and this distance can be various distance measures. We will assume that Euclidean distance between the two vertices, ok?

So, given this how we can have a minimal spanning tree representation of this graph? The minimal spanning tree says that when I represent, when I find out the spanning tree corresponding to a graph, then the sum of the weights of the edges which are included in the spanning tree. So, you find that the spanning tree representation of this graph will obviously have all the vertices of the graph, but the edges that will be included in the spanning tree that will be a subset of the edges, which are there in the curve. So, when I consider that subset of the edges, the minimal spanning tree says that the sum of the weights of the edges which are included in the spanning tree that should be minimum. So, that is what is minimal spanning tree?

So, a minimal spanning tree algorithm can be very easily find out, that when I have this edge list. So, here we assume that we literal, we will use a representation of the graph which is an edge representation and every edge has a corresponding weight. So, when I have this edge list, this edges will be ordered in the ascending order of the weights. So, the first edge in my list will have the minimal weight and the last edge will correspond to maximum weight and in between the edges will be ordered in ascending order of the corresponding weights, ok? So, I will again take an example to explain how this minimal spanning tree can be constructed from a given graph and how this minimal spanning tree can be used for clustering purpose.

(Refer Slide Time: 07:23)



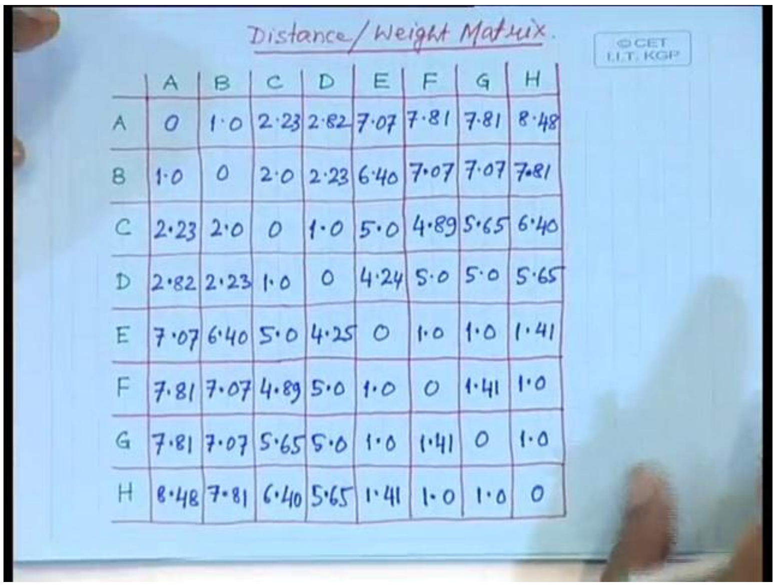
A (1, 1)
B (1, 2)
C (3, 2)
D (3, 3)
E (6, 6)
F (6, 7)
G (7, 6)
H (7, 7)

$$^{8}C_{2} = \frac{8!}{2!\,6!} = \frac{8 \times 7}{2} = \underline{28.}$$

So, I will use a set of feature vectors, say feature vector A which is 1 1, feature vector B which is say 1 2, feature vector C which is say 3 2, feature vector D which is 3 3, E which is 6 6, F 6 7, G 7 6, H which is 7 7. So, I will consider this eight feature vectors and as I said that when I represent this feature vectors by graph or a completely connected graph and that to an weighted completely connected graph. Then obviously I will have eight nodes. And the number of edges because the graph is completely connected is nothing but 8 C 2 and which you can compare, that this is the factorial 8 upon factorial 2 into factorial 6, which will be 8 into 7 by 2, it is equal to 28.

So, maximum I can have 20 edges within this graph as the graph is completely connected, and for each of this edge I will have a weight which is nothing but the Euclidean distance between the corresponding feature vectors. So, accordingly as we were doing in the last class we can form a distance matrix or in this case it is an weight matrix.

(Refer Slide Time: 09:13)



Distance/Weight Matrix.

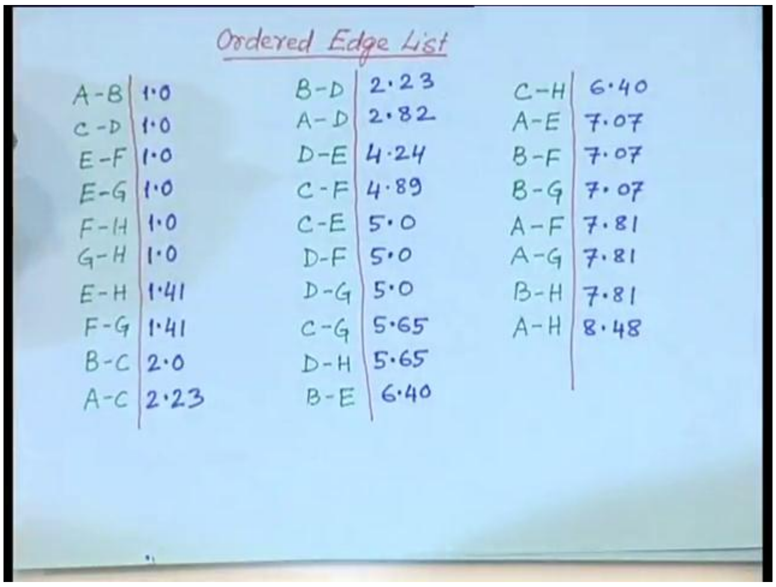
©CET I.I.T. KGP

	A	B	C	D	E	F	G	H
A	0	1·0	2·23	2·82	7·07	7·81	7·81	8·48
B	1·0	0	2·0	2·23	6·40	7·07	7·07	7·81
C	2·23	2·0	0	1·0	5·0	4·89	5·65	6·40
D	2·82	2·23	1·0	0	4·24	5·0	5·0	5·65
E	7·07	6·40	5·0	4·25	0	1·0	1·0	1·41
F	7·81	7·07	4·89	5·0	1·0	0	1·41	1·0
G	7·81	7·07	5·65	5·0	1·0	1·41	0	1·0
H	8·48	7·81	6·40	5·65	1·41	1·0	1·0	0

So, given these feature vectors I can find out what will be the weight matrix for this particular graph graphical representation. So, this matrix gives you the weight matrix. So, you find that the this is the same one which we were trying in the last class, that is the distance between AB is 1.0, distance between AC is 2.23, distance between AD is 2.82. So, the edge between A and B will have a corresponding weight, which is 1 and edge between A and C will have the weight, which is equal to 2.23. Edge between A and D will have the weight which is 2.82. Edge between A and E will have the weight 7.07 and so on, ok?

So, I can find out the edges within the graph using this particular weight matrix. And once I have this weight matrix, from this weight matrix I can find out the ordered list of the edges which are put in ascending order of weights, ok? So, as I said that the weight between these two vertices A and B is 1.0 and when you look at this weight matrix, you find that this is the minimum weight. Similarly, I will have other edges where also the weight is 1.0 say for example, here E and F, the edge between that also has an weight 1.0. Edge between E and G, that also has a weight 1.0, ok?

(Refer Slide Time: 10:55)


Ordered Edge List

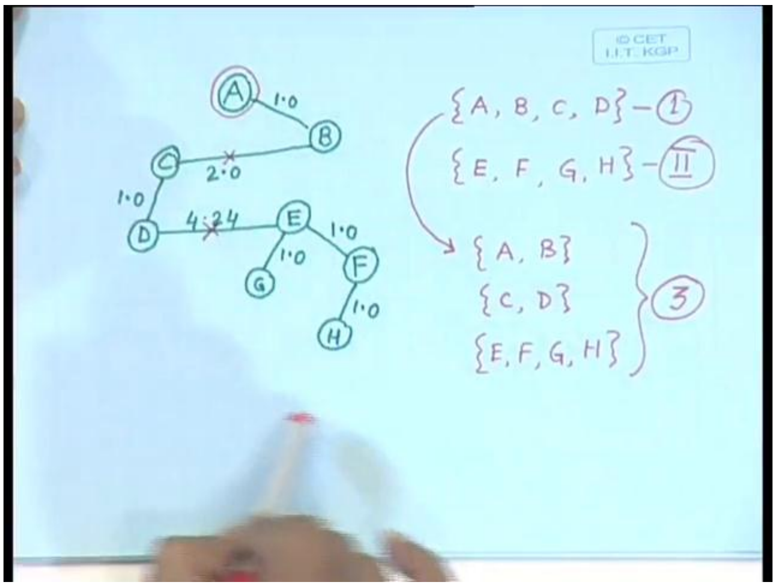
A-B	1·0	B-D	2·23	C-H	6·40
C-D	1·0	A-D	2·82	A-E	7·07
E-F	1·0	D-E	4·24	B-F	7·07
E-G	1·0	C-F	4·89	B-G	7·07
F-H	1·0	C-E	5·0	A-F	7·81
G-H	1·0	D-F	5·0	A-G	7·81
E-H	1·41	D-G	5·0	B-H	7·81
F-G	1·41	C-G	5·65	A-H	8·48
B-C	2·0	D-H	5·65		
A-C	2·23	B-E	6·40		

So similarly, I can find out an edge list and based from the weights I can put the edges in ascending order of the corresponding weight values. So, I get the edge list like this AB is 1.0, CD is 1.0, EF is 1.0, like this EH which is 1.41, FG is 1.41, BC is 2.0, AC is 2.23 and so on, ok? So, once I have this ordered layered edge list using this edges, I have to form the spanning tree. So, what is the algorithm for finding out the spanning tree? The algorithm says that you take the first stage because initially my tree is empty to take the first stage. One of the edges you take as root of the spanning tree. One of the vertices you take as root of the spanning tree and the other vertices is connected by an edge, whose weight is same as the weight of the edge as in the graph. Next, we consider the next stage if they have a common vertex, then you simply add another node which are connected to one other edges of already existing tree by the edge that we are considering.

Now, while doing so it may so happen that I can create number of tree, because the first edge and the second edge they may not be connected. So, effectively what I will have is two trees, but subsequently when one of the edges connecting one of the nodes of quantity and other node of the other tree, you cannot count of that you connect those two trees by the corresponding edge. And while forming this tree, always you have to find that if you encounter an edge which is already placed in one of the trees. Then you should not consider that edge anymore, not only that if any edge produces a cycle within the tree you should not consider that edge also, ok?

So, following this rules you can form the spanning tree and as in this case the spanning tree is formed by considering the edges. First which have got the minimum weights so that will ensure that the spanning tree, the two constant that will also be a minimal spanning tree. So, from this example let us try to construct the spanning tree. So, the first edge that I have is an edge AB which has an weight 1.6. So, I will take this particular edge to start my spanning tree and I assume that out of this edge let A be the root node, ok?

(Refer Slide Time: 13:52)



So, I will have node A as the root node and AB forms a tree. Other weight of this edge is 1.6, the next edge is CD, you find that you do not have any common vertex. So, CD will form another tree whose weight is also 1.0. So, let me take CD somewhere over here, it is weight is also 1.0. The next one is EF, again there is no common node of EF with AB or CD. So, this will be another tree. So, EF let me consider this EF, this edge is also weight of this edge is also 1.0, then EG. Now, you find that between EF and EG the node E is common. So, I will include this EG and connect it to this existing tree by this edge EG and it is weight is also 1.0. The next one is FH whose weight is also 1.0, F I already have. So, this is another node FH and this weight is also 1.0.

Next, what I have is FG. So, here you find that if I consider this edge F and G, because both the nodes are already existing in one tree. So, this will form a cycle. So, I should not consider this node at all. The next one is BC, so you find that C is in one tree and B is in another tree. So, I should connect these two trees using this edge BC and the weight of this edge BC is 2.0.

The next one is AC, you find that both A and C after inclusion of this edge B and C, both A and C are in the same tree. So, if I connect this A and C by this edge, in that case there will be a cycle within the tree. So, I should not consider this edge A and C, next comes BD. So, by using BD again I have similar equation, if I connect B and D there will be a cycle. So, I should not connect B and D.

Next comes AD, again the same situation if I connect A and D there will be a cycle. So, it will not remain a tree anymore. So, I should not connect AD, next comes DE and here you find that D is in one tree and e is in another tree. So, I should consider this node, this particular HDE and connect D and E by the HDE and you find that weight of this edge is 4.24. So, that is the weight of this edge and here you find that we started with eight nodes, all the eight nodes have been put in this particular tree. I can go on checking each and every other edge, but if I try to include any other edge within this tree, there will be a circle.

So, I can stop creation of the tree when I have a single tree and all the nodes in the graph are included in that single tree, so that tree formation can be stopped at that particular point. So, here what I have is I have formed a tree where node A is the root node. This is the root node and if you take any pair of vertices or any pair of nodes, you will find only one path existing between these pair nodes and as the edges have been considered in ascending order of the weight values. So, this algorithm shows that the sum of the weights which are included in this tree, that will be the minimum possible sum. Do you have any question? So, that is the minimum possible sum.

However, this minimal spanning tree is not unique, that is the reason why I said minimal, not minimum. The reason is I started with the node A as the root node if I take any other node as the root node. Let us start formation of the tree with any other node of the root node, then I can have another form of tree, but this ensures that assuming one of the node to be root node, your spanning tree will be unique, but for various nodes as root nodes, the spanning tree will be different.

So, that is why it is the minimal spanning tree. Now, the question is once I have this minimal spanning tree, then how this minimal spanning tree can be used for the clustering purpose? The operation is very simple, you just check the weight of the edges within the minimal, this minimal spanning tree and because it is a tree structure, it is quite obvious, that if I remove any of the edges then the tree will be broken into two sub trees.

If I remove two edges the tree will be broken into three sub trees and so on. Unlike in case of a graph, if I have a completely connected graph, removal of one the edges does not ensure that the graph will be partitioned into two unconnected sub graphs, which is not true in case of a graph. But in case of a tree because there is no cycle, if I remove any of the edges then the tree will be partitioned in two and so on.

So, by deleting the number of edges I can attain the desired number clusters. So, what you can do once I have this minimal spanning tree is that you find out an edge whose weight is maximum. And as I said that in this, in our case the weight of an edge is nothing but the distance between the corresponding feature points or the distance between the corresponding nodes. So, it simply says the weight represents that what is the degree of dissimilarity between two different points. So, if I choose an edge whose weight is maximum or whose host is maximum that simply says that the distance between the corresponding vertices is maximum.

So, coming to this spanning tree you find that the maximum weight of an edge in this spanning tree is 4.24. So, if I remove this edge then you find that the tree is partitioned into two sub trees. The nodes ABCD the other tree contains, the other sub tree contains EFG and H. So, obviously I get two clusters, one cluster containing nodes A B C D. This is my cluster number 1 and the other cluster is EFG and H, this is cluster number 2. If I want to go further subdivided, this into more number of clusters then within each of them you find out which is the next maximum.

So, here you find that the next maximum weight is between edges C and B which is equal 2. So, I can remove this, if I remove this then this cluster ABCD is now broken into two clusters, one containing AB, the other containing CD. So, I have cluster containing feature vectors AB CD and the other cluster remains as it is because here none of the edges is having the same value, ok?

So, other cluster is EFG and H, so total I have three number of clusters in this case. So, I can approach one of the two criteria’s, that is I can say that this is the number of clusters

I want or minimum number of clusters that I want. Or I can also say that what is the distance between two given clusters I should tolerate, so that those sets of points can be put into a semi cluster. So, like in this case when I consider this cluster considering EFG edge and I consider this cluster ABCD, you find the distance between these two clusters is 4.24.

Similarly, over here between the cluster AB and the cluster CD, the distance between these two cluster is 2. So, I can have either or I can either select the number of clusters that I want or I can also select that what is the minimum distance between the two clusters which can be tolerated. So, I can either have distance ratio or weight ratio or I can have number of clusters.

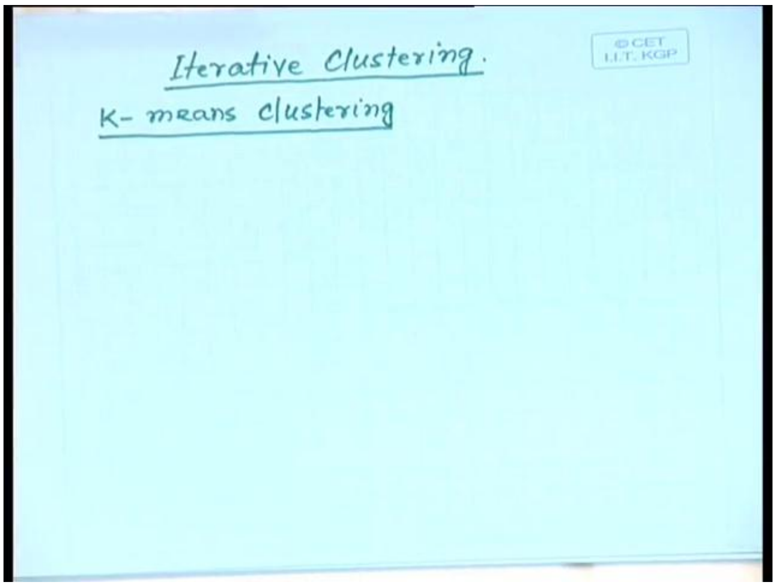
So, accordingly I can form the number clusters. So, here you find that because it is a graphical technique and the graph is of global approach. So, the number of clusters that you get or the points which are put in the clusters, will also be independent of order in which unlike in case of any other sequential, pardon cluster 1 and cluster 3. That will obviously be more than 4.24, the reason is when have I have constructed the spanning tree, all the edges are taken are considered in ascending order of weight values. So, had there been any edge between any of the points or any of the nodes A and any of the nodes, within this cluster, had there been any edge whose weight was less than 4.24, that edge would have been considered before this edge, is that ok?

So, that is quite obvious that the obviously there is an edge between A and E. There is an edge between A and F, edge A and H, there is an edge, but the weight of this edge weight of such edge is much more has to be more than 4.24. You come to this weight matrix. Let us take AE, see the weight is 7.07. Consider AF, weight is 7.81. Similarly, AG 7.81 is 8.48. It has to more than four point, this 4.24. Otherwise, those edges would have been considered before this edge, ok? So, naturally the distance between this cluster AB and the cluster EFG, edge will be more than 4.24, that is how this spanning tree is constant, is that correct?

During the course of formulation of spanning tree I do not know that is not necessarily, see what can be done is if I have as you have done in this case. We have put a threshold on value. So, while construction of the spanning tree, because I know the maximum weight, edge is the last one which is been considered while forming this minimum spanning tree. So, if I put a threshold on the weight, so the moment I get an edge of a particular weight I will stop construction of the spanning tree, but even then the problem is there might be other vertices which has not been included in the tree at all.

So, if there are more number of edges whose weights might have been more than 4.24. So, if I put a threshold on this 4.24 when this edge is considered, I stop construction of the spanning tree. There might other vertices which has not been included the spanning tree at all, ok? So, then you have to go on checking whether every node has been considered. Now, let us talk about one iterative technique, iterative cluster.

(Refer Slide Time: 29:10)



Iterative Clustering.

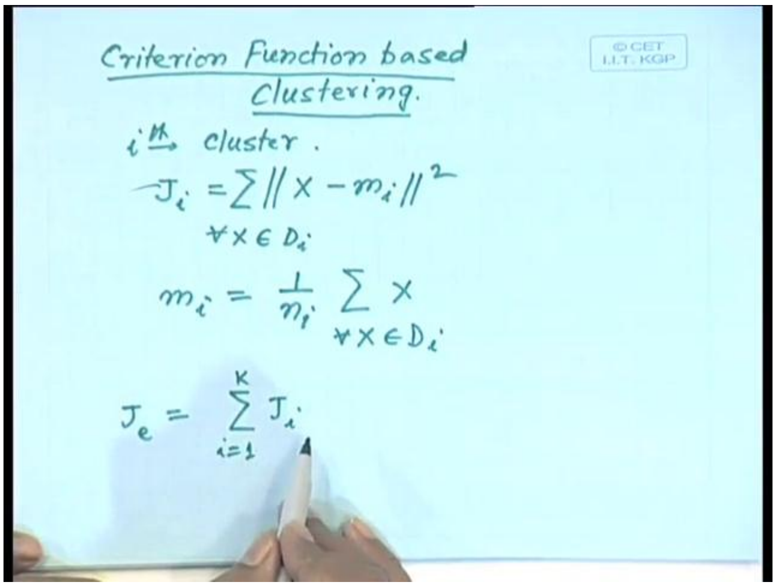
K- means clustering

So, one of the very popular iterative cluster technique is called k-means clustering. So, it is concept is very simple, I assume that whatever be the set of given feature points I will construct k number of clusters out of this feature points. So, the approach is something like this. Initially or arbitrarily partitioned the set of data into k number of clusters, so that is quite arbitrary once you partition the data into k number of clusters. For every cluster you find out the cluster mean which is the mean of the feature factors?

So, I get k number of mean vectors, once I get this the k number of mean vectors, then you try to reassign the set of points into various clusters based on minimum distance. It may so happen that initially because your clustering was totally arbitrary, so a feature vector which has been put into say cluster 1. But after computation of the cluster centres you may find that for this feature vector it is distance to cluster centre of cluster 5 is less than it is distance centre from cluster 1, and it is distance from centre cluster 5 is minimum among all other distances.

So, you reassign this feature vector into cluster number 5. So, likewise you try to reassign all the feature vectors using minimum distance based on minimum distance from all the cluster centres. So, after reassignment you recompute the cluster centres and then reassign the feature vectors again. So, your recomputation of the cluster centres and reassignment of the feature vectors goes on iteratively until and unless you come to an stage, when the clusters have not changes. So, that is the condition of vectors that in a pass no point has been or no feature vector has been reassigned to any other cluster, ok? So, that is a condition of iterative problem. So, this is an I can have a formal way original way of this iterative problem, where I can define that or I can define a criteria function or this criteria function is based on sum of squared error. So, if I say let us take ith cluster.

(Refer Slide Time: 32:26)



Criterion Function based Clustering.

i^{th} cluster.

$$J_i = \sum_{\forall x \in D_i} \| x - m_i \|^2$$

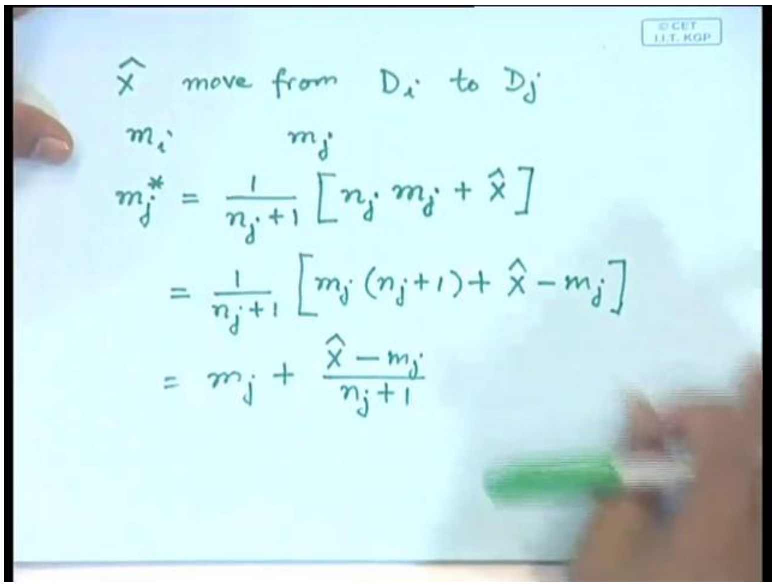
$$m_i = \frac{1}{n_i} \sum_{\forall x \in D_i} x$$

$$J_e = \sum_{i=1}^{K} J_i$$

So, criteria function based like clustering, so again in this case I assume that I will form k number of clusters. So, initially all the feature vectors will be arbitrary partitioned into k number of clusters. Now, if I take ith cluster, in ith cluster the error in the ith cluster I will be computed as x minus m i square, sum of this for all x which belongs to partition d i. However, m i is nothing but the mean of this feature vectors which are put into this cluster d i. So, m i is simply 1 upon say ni. ni is the number of feature vectors put into say di and summation of x for all x belonging to di. So, this I compute for every cluster i, where i is from 1 to 8. As I am saying that I consider k number of clusters and the total error is given by J e is equal to sum of J I, however i varies from 1 to k that is the.

So, you find that this term tells you within clusters some of square area and this tells you what is the overall? So, this clustering technique tries to find out the clusters for which this total error will be minimum. So, how I can do it iteratively? Suppose, after that initial partitioning I will take one feature vector from one cluster and try to push into another cluster. And then try to find out that if I push this feature vector from one cluster to another cluster, whether the total error can be reduced or not, ok?

(Refer Slide Time: 35:15)



$\hat{x}$ move from D_i to D_j

m_i $\qquad m_j$

$$m_j^* = \frac{1}{n_j+1}\left[n_j m_j + \hat{x}\right]$$

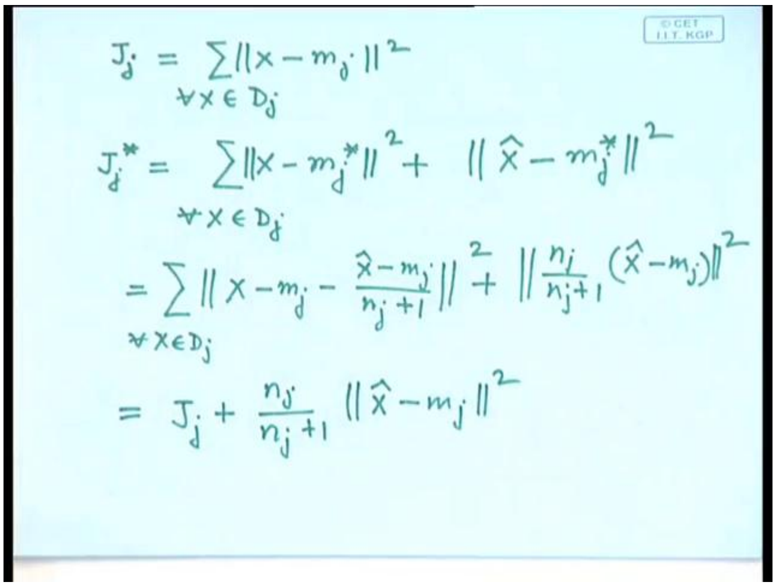
$$= \frac{1}{n_j+1}\left[m_j(n_j+1) + \hat{x} - m_j\right]$$

$$= m_j + \frac{\hat{x} - m_j}{n_j+1}$$

So, I take a feature vector x hat and try to move it from subset D i to subset D j, you take this feature vector x i from the ith cluster, try to push it into jth cluster. So, when I try to do this, then the ith cluster initially had mean off m i. Similarly, jth cluster had mean of m j. So, as I am pushing x hat from ith clusters to jth cluster, this in j mean of jth cluster will be changed. Let us put it as m j star, that is the new mean of the jth cluster which will be nothing but initially before putting this x hat into this jth cluster, the number of points in the jth cluster was the m j as put this new point into the jth cluster, the number of points in this jth cluster will be m j plus 1, ok?

So, this will simply be 1 upon m j plus 1 into previous number of points, m j into previous mean, m j plus x hat is it, so this is the mean new mean of the jth cluster D j. And if I simplify this, you find that it will be 1 upon j plus 1 into, I can rewrite this expression as m j into m j plus 1 plus x hat minus m j and which is nothing but m j plus x hat minus m j upon m j plus 1. So, as this new feature vector x hat has been pushed from the ith clusters to jth cluster, the mean of the jth cluster which before this x hat was pushed into this m j. After this the x hat is pushed into the jth cluster, this m j has been changed to m j star. However, you find that there is an increment and in m j which is given by x hat minus m j upon n j plus 1, ok? So, in the same manner the total error I can also compute, what is the total error change in total error of the jth cluster. So, the change in total error of the jth cluster will be something like this.

(Refer Slide Time: 38:28)



$$J_j = \sum_{\forall x \in D_j} \|x - m_j\|^2$$

$$J_j^* = \sum_{\forall x \in D_j} \|x - m_j^*\|^2 + \|\hat{x} - m_j^*\|^2$$

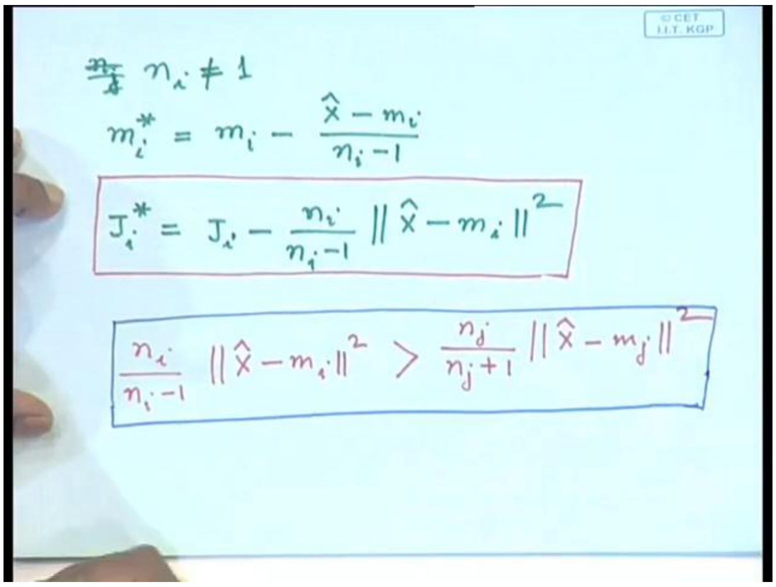
$$= \sum_{\forall x \in D_j} \left\|x - m_j - \frac{\hat{x} - m_j}{n_j + 1}\right\|^2 + \left\|\frac{n_j}{n_j + 1}(\hat{x} - m_j)\right\|^2$$

$$= J_j + \frac{n_j}{n_j + 1}\|\hat{x} - m_j\|^2$$

So, before pushing this x hat the error in the jth cluster was x j which is nothing but sum of x minus m j square, for all x belonging to cluster D j. So, this was the total error in the jth cluster before putting this exact. After I put this x hat, then the total error in the jth cluster is changed to x j star and which is nothing but summation of x minus m j star, because now the new mean m j star, it is not m j anymore, square where for all x belonging to cluster D j, ok? Plus as this new feature vector x hat has been included in the jth cluster. So, it will be x hat minus m j star square, this is the total error in the jth cluster and again I can simplify this.

So, this will be and this can be simplified as n j by n j plus 1, so because of inclusion of this x hat into the jth cluster. The mean has been moved from m j to m j plus x hat minus m j by n j plus 1 and the total error squared error has been increased from x j to x j plus n j by n j plus 1 into x hat minus m j square. So, as there has been change in the mean and error some of squared error in the jth cluster. Similarly, there will also be change in the mean and sum of the squared error in the ith cluster, ok?

(Refer Slide Time: 41:44)



$n_i \neq 1$

$$m_i^* = m_i - \frac{\hat{x} - m_i}{n_i - 1}$$

$$\boxed{J_i^* = J_i - \frac{n_i}{n_i - 1} \| \hat{x} - m_i \|^2}$$

$$\boxed{\frac{n_i}{n_i - 1} \| \hat{x} - m_i \|^2 > \frac{n_j}{n_j + 1} \| \hat{x} - m_j \|^2}$$

So, if I consider the ith cluster assuming that n i is not equal to 1. Why I am considering this n i is not equal to 1 because if n i is equal to 1, that means the ith cluster or say D i had a single point. If it has a single point, if I want to push that into another cluster then the number of clusters will be reduced, ok?

So, if any clusters contains only one feature vector, that cluster should not collapsed. So, that is the reason that this condition n i not equal to 1. So, if n i is greater than 1, then only I can take a feature vector from the ith cluster, trying to push it into another cluster. So, consider having this particular condition that n i is not equal to 1, the cluster centre for the ith cluster will be modified to m i star will be equal to m i minus x hat minus m i upon n i minus 1.

Similar, such calculation you can find out this and similarly the error within the ith cluster will be changed to x i cluster, which is j i minus n i. So, it is by similar calculation that we have done earlier. I can find out that what is the change in the mean of the ith cluster and what is the change in the sum of squared error in the ith cluster.

So, now if you compared these two, you find that for the jth cluster, so this is my j j star. So, for the jth cluster where I have pushed this feature vector x hat, there the total error has been increased by amount in n j upon n j plus 1 x hat minus m j square. This is the amount of increase in the error in the jth cluster.

Similarly, in the ith cluster the total error has been reduced, it has been decreased by n i upon n i minus 1 into x hat minus m i square. So, here I have a net increase over here, there is an increase by this amount and over here there is a reduction by this amount. So, this transfer of x hat that the ith cluster to jth cluster will be profitable, in the sense that it will reduce the total error because total error is nothing but sum of these two errors.

So, it will reduce the total error, if the reduction in the error in the ith cluster is greater than the increase of error the jth cluster. So, where ever I have error reduction, the amount of reduction has to be more than the amount of increase that is happening in the other cluster. Then only when I take the sum then I will have a net reduction in the total error. So, I will take a feature vector from one of the cluster, try to push it into another cluster and this transfer of the feature vector from one cluster to another cluster will be effective will be done. Only when the decrease in error in one cluster is less is more than the increase in error in the other cluster, ok?

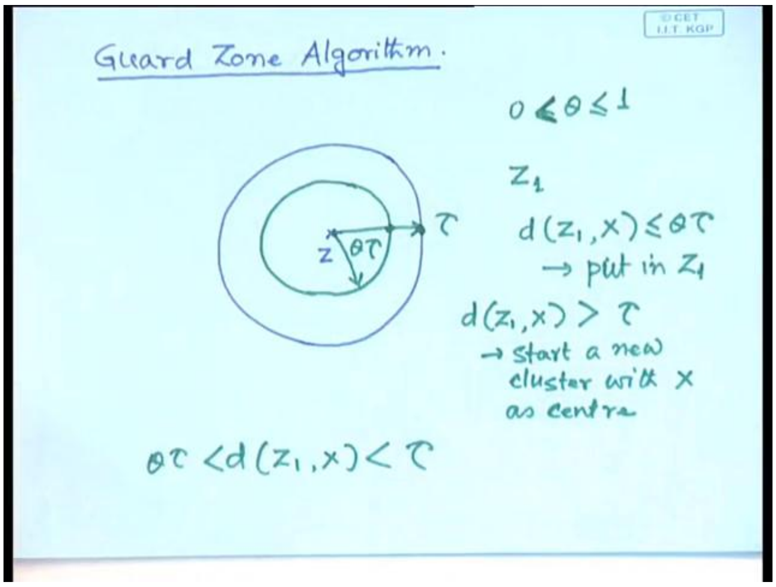
So, my condition for transfer will be that n i upon this n i minus 1 into x hat minus m i square, if this is greater than n j. So, only when this condition is true then only I can push this x hat from this ith cluster to jth cluster. Now, again you find that I am taking x hat from the ith cluster, that means this amount is fixed, but ith cluster is one of k number clusters. So, I have k minus 1 number of other clusters and this j can take on any of this k minus 1. This jth cluster can be any of this k minus 1 number of clusters, right? So, what I have to do is I have to compute this value for all those jth clusters which is not ith cluster, ok?

So, for all values of j not equal to, I have to compute this value and because this is the amount of increase, naturally I should select that particular cluster where the increase is minimum. If I have two such clusters, cluster number 2 and cluster number 3, I find that the increase in the error in cluster number 2 is more than the increase in error in cluster number 3.

So, obviously I should push this x hat into cluster number 2, not in cluster number 3 because in that case the total amount of reduction that will be that I will getting is not minimum. So, this ensures that by pushing this x hat from ith clusters to jth cluster you have an net reduction in the total error. And to maximise that reduction I should push x hat, push x hat into cluster which gives minimum increase in the within class within clusters error, ok?

So, out of all those A minus 1 number of clusters for whichever cluster the increase in the sum of squared error is minimum, the x hat should be pushed into that. So, for the first condition without this I should not try to push x hat from one cluster to another cluster. And the next condition is I should try to find out the cluster where if I put x hat, the increase in the sum of squared error will be minimum, ok? So, this is another iterative algorithm following which we can also find out the cluster for a given set of feature vectors. There is another algorithm which is called guard zone algorithm.

(Refer Slide Time: 49:41)


Guard Zone Algorithm.

$0 \leq \theta \leq 1$

Z_1

$d(z_1, x) \leq \theta\tau$

$\rightarrow$ put in Z_1

$d(z_1, x) > \tau$

$\rightarrow$ start a new cluster with x as centre

$\theta\tau < d(z_1, x) < \tau$

So, guard zone algorithm is something like this. Suppose, somehow I get a cluster centre x z said and then around this cluster centre z I define a zone, ok? So, if a point falls outside this zone, then that feature vector will not be put into same cluster as that cluster centre z and within this zone I have a sub zone something like this, ok? So, suppose this zone is defined by this parameter tau and this is a fraction of this parameter. So, let us put it as theta times tau, ok? So, the algorithm works like this, you take one of the feature vectors, assume that to be the cluster centre. Take the next feature vector, if the next feature vector falls or the distance between the next feature vector and this cluster centre is less than theta times tau, you put that feature vector into the same cluster as this cluster centre. If it is between theta times tau and tau do not take any immediate action or immediate decision about that feature vector, and if the feature vector is beyond tau then that feature vector should not be clustered into this bed, ok?

So, normally this tau it is decided by the designer. Similarly, what fraction of tau I should consider for putting a feature vector into the same clusters, that is also decided by the designer. So, theta is obviously between 0 and 1. So, if it is equal to 1 then whenever the feature vector falls within this region tau, it will be put into a centre stage. I should not put it as 0, but it is greater than 0, 0 not inclusive, but 1 maybe inclusive. So, the algorithm is like this, if you take the first feature vector consider that as your first cluster centre, z 1. Take the next feature vector, find out the distance between the feature vector and this cluster centre z 1.

So, x is the next feature vector, distance between z 1 and z. If this distance is less than or equal to theta times tau, if this is less than or equal to theta times tau put it into the centre stage if distance, this distance is greater than tau. So, in this case put in the same clusters z 1, if this distance is greater than tau, then you can start a new cluster with x as centre. So, that says that x is somewhere over here or some over here that is far away from z. So, you start a new cluster with z as the new cluster centre. However, as if d z 1 x, if this is between theta times tau and tau that means it is within this region. Do not take any immediate decision about this, you keep the case pending.

So, like this when all the feature vectors are considered, some of the feature vectors will be put into some of the clusters. Some of the feature vectors this is what is called guard zone will be in the guard zones of other clusters, which are already formed. Then what I can do is you can find out that, what is the number of points which are there in the guard zone of a particular feature vector. If the number of points within the guard zone of a cluster is not significant is quite small. Then you put them into the same cluster, but if the number of points is quite significant then I can do this interoperation once more with a modified value of tau, maybe with a modified value tau for to try to recluster all those points which are there in the cluster.

So, this depends upon how many points in the guard zone. If the number of points in the guard zone is quite small, then try to put them into same cluster in whose guard zones the points are existing. But if the number of points in the guard zone are quite high, then it is quite logical that I can try to form more number of clusters using those points as their degree of belongingness within the same clusters in guard zones. They are belonging, they are existing is not that strong. So, you can try to form other clusters using this one.

So, again you find that this a iterative algorithm and as it is iterative algorithm, the final clustering output as in case of any other iterative algorithm will depend upon what is the order in which the points are presented, ok? Unlike in case of graph graphical algorithm, where we said that the when I form a graph it is a global structure. So, in case of a graphical algorithm, the final output does not depend upon the order in which the points are presented. But in case of any of the iterative algorithms, the final output may vary depending on the order in which the feature vectors are presented tau. So, we will stop here today. Next day we will talk about some other topic.

Thank you.