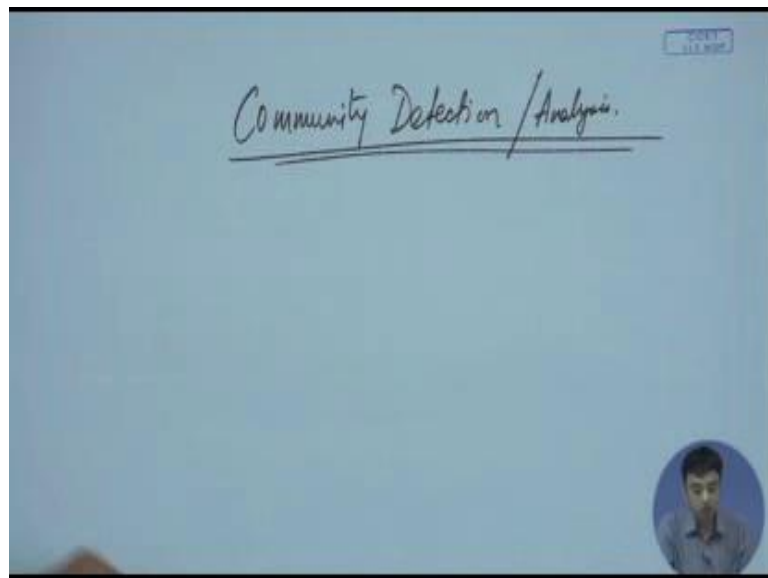


Complex Network: Theory and Application
Prof. Animesh Mukherjee
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 12
Community Analysis – I

So far we have been discussing about the basic matrix analysis of complex networks and then we have also looked into social network properties. Now, we will see into little bit more algorithmic methods especially related to what we call community detection or sometimes also called analysis; community analysis.

(Refer Slide Time: 00:39)



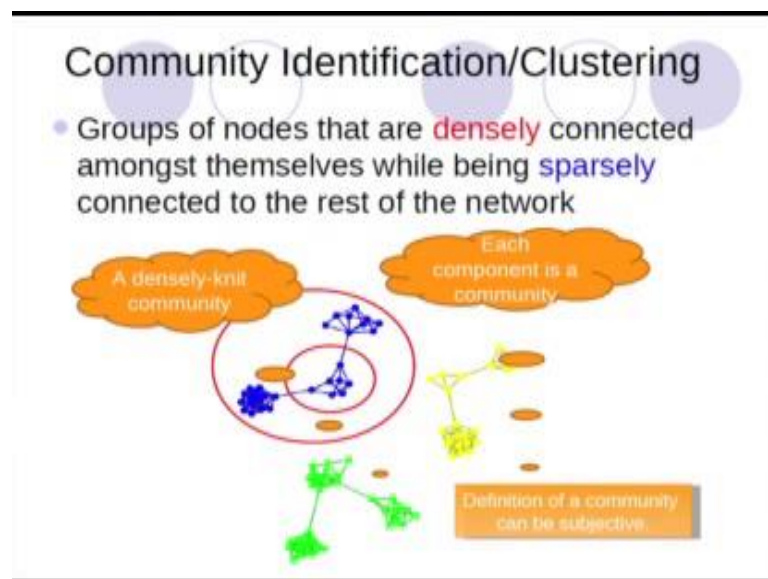
The basic idea is given a graph structure you want to find out dense sub-drafts which are very tightly connected among each other, but as sparsely connected to the rest of the network. These modules actually, if you can efficiently extract these modules then they are found to be very, very effective in various application designs later on. So, many unsupervised algorithms in machine learning actually depend on community or clustering as a first step.

So, basically the point that I am trying to make is that given a graph structure, if you can

identify these dense sub modules very efficiently then you can enhance the performance of various machine learning related applications into a large extent because many of these learning algorithms actually depend on the clustering as its input on the input clusters, and better the cluster algorithm the more efficient will your application on machine learning application be. So, in order to do this, the first thing that we need to do is to define what a community is? Now, as you can well understand that it is a very ill defined concept there is no hard and first or straight forward definition of what a community is.

So, it is mostly the concept is mostly philosophical if you wish so, but then if you have to do certain application development, if you have to write algorithms to identify communities, then you have to sum how quantify this philosophical notion and one very well accepted form of definition is that, if you look at the slides.

(Refer Slide Time: 02:47)

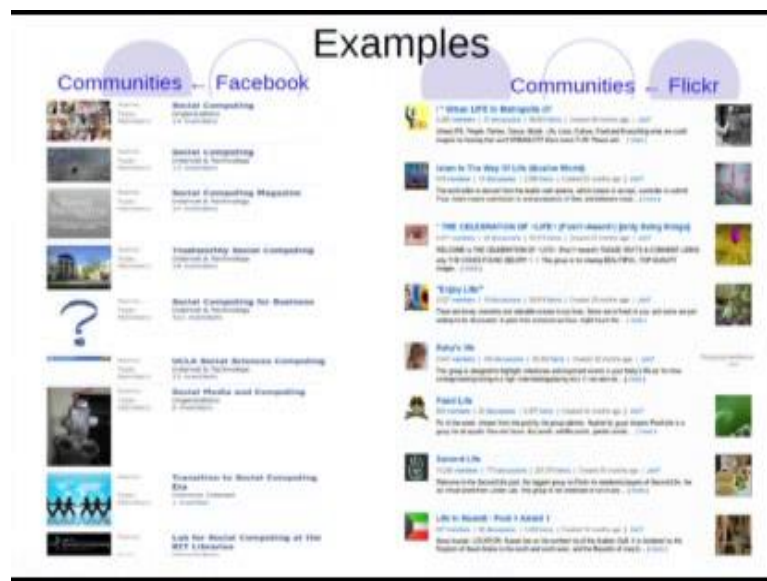


It is a group of nodes that are densely connected amongst themselves while being sparsely connected to the rest of the network. So, for instance, let us take this example here. This set of blue nodes here which I have circled by a red circle they seem to be densely connected, similarly there is this set which is again densely connected and also this set which is densely connected, but then there are sparse connections in between

these densely connected sub modules.

So, basically each of this can be thought of as a community then these communities are sparsely connected by a very few edges. Now, the question is like how to go about and design efficient algorithms to extract such communities even large social graphs.

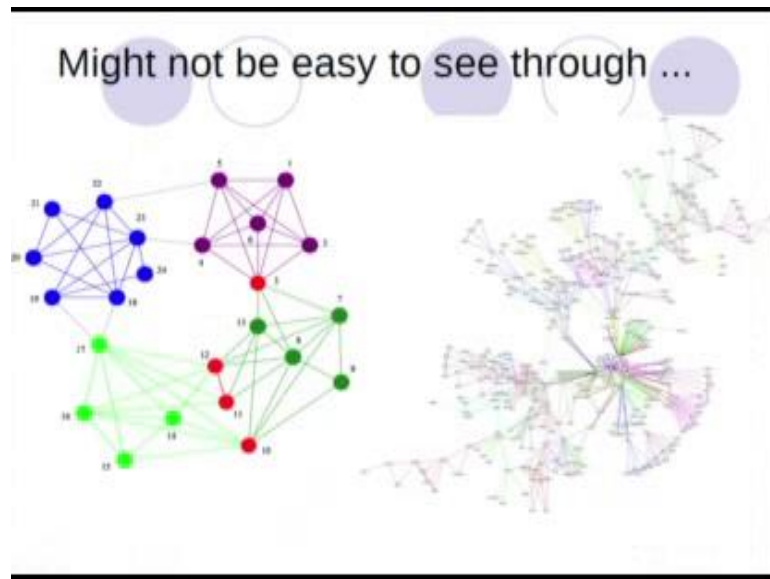
(Refer Slide Time: 03:41)



For example, the communities in Facebook graph or communities in Flickr social network, here you have, for instance in Facebook you have probably all of you have seen there are various communities like communities like social computing, and communities like transition in social communities. So, these are all communities about social computing, where as in Flickr you have similar communities like communities like enjoy life, baby's life, pond life.

So, all communities related to life. So, you can have such communities in various social networks. Now, the point is like how given set of nodes and a bunch of connections between this set of nodes how one can efficiently design algorithms in order to find out meaningful community structures, meaningful sub structures in this graph.

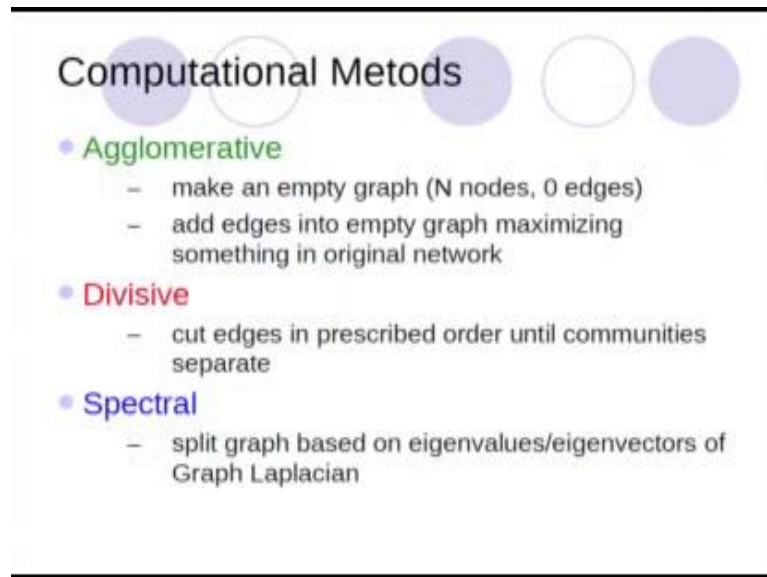
(Refer Slide Time: 04:38)



The problem might not be always very easy, for instance, if you look at the left hand side in this slide the communities are very nicely well marked and well separated each color here would represent. You can immediately see that each color in this graph would represent a separate community whereas, if you look at the right hand side graph it is not so easy to just look at the graph and tell what are the important communities hidden in this graph.

As and when the graph structure grows it is more and more it becomes more and more difficult to identify sub modules just by looking at the graphs. So, looking at the graphs it is not always easy to identify or infer the densely connected sub modules. So, therefore, you should have some computational techniques, some algorithmic approaches to handle this problem.

(Refer Slide Time: 05:35)

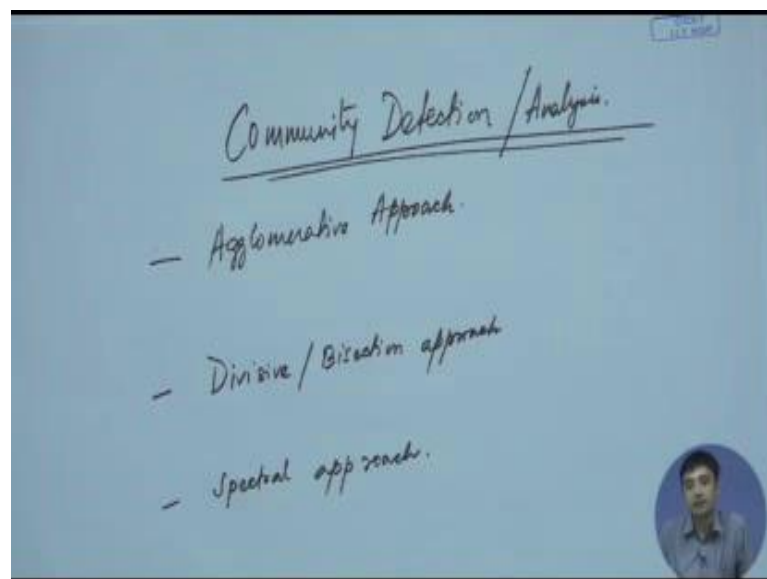


Computational Methods

- **Agglomerative**
 - make an empty graph (N nodes, 0 edges)
 - add edges into empty graph maximizing something in original network
- **Divisive**
 - cut edges in prescribed order until communities separate
- **Spectral**
 - split graph based on eigenvalues/eigenvectors of Graph Laplacian

And in order to do that the computational methods that people have suggested in the literature actually can be classified into at least 3 different types; the first one is called the agglomerative approach.

(Refer Slide Time: 05:46)



Community Detection/Analysis.

- Agglomerative Approach.
- Divisive/Bisection approach
- Spectral approach.

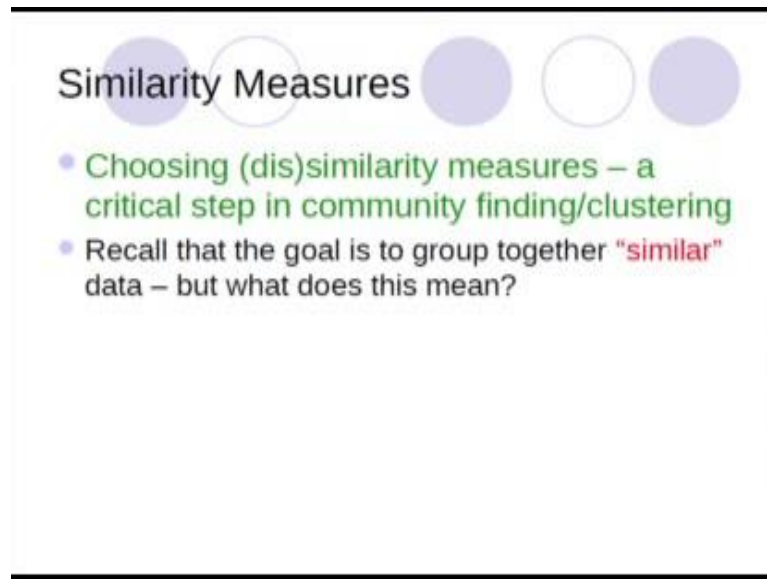
The second one is called the divisive or bisection approach and the third one is called the

spectral approach. So, there can be various other variants of these approaches, but roughly these are the kind of 3, you can say the pillars of computational techniques that you find in this area. So, the agglomerative methods are one of the earliest methods for community detection or clustering what you do is you make an empty graph, you have n nodes and 0 edges and it is a bottom of approach. At every, iteration you kind of add edges between nodes that are most similar and you keep on agglomerating in this way. So, that is what is called the agglomerative approach.

Whereas the divisive approach is the other side of the coin, where you have the whole graph given to you., you find out the most probable edge that should be removed from the graph to divide the graph into 2 parts and in this way you keep on removing again and again the repeatedly the most probable edges, until and unless the graph gets segregated into smaller dense components and the third approach is this spectral approach where you actually resort to the eigenvector analysis, eigenvector space analysis in order to identify communities.

We will get to know more about this when we talk about spectral clustering. So, as we will see there that the second eigenvector actually contains important information about the community structure or the components or the dense connected components in a graph. So, we shall see that when we deal more in details with the spectral bisection method.

(Refer Slide Time: 08:03)



The slide features the title "Similarity Measures" at the top, with the word "Similarity" in a light purple circle and "Measures" in a white circle. To the right of the title are three circles: a solid purple one, a white one with a purple outline, and another solid purple one. Below the title are two bullet points:

- Choosing (dis)similarity measures – a critical step in community finding/clustering
- Recall that the goal is to group together "similar" data – but what does this mean?

So, the first thing that one needs to actually think about or ponder about when designing a clustering algorithm is the similarity or a dissimilarity matrix based on which you are going to say for instance you are trying to do agglomerative clustering. So, you start with all nodes in separate clusters every node is in a single cluster.

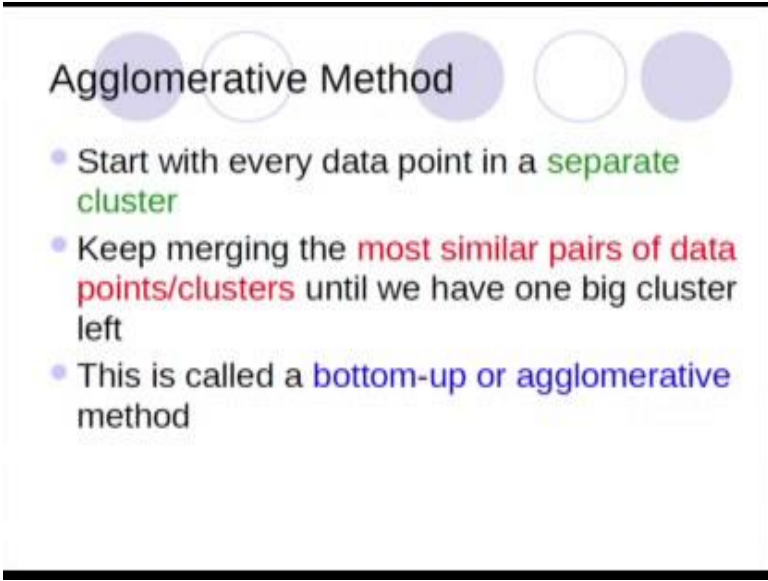
Now, you have to based on some similarity measure actually establish links between nodes and in this way you slowly grow up the cluster structure or the community structure. Now, for this you have to first of all the first and the foremost task that needs to be solved is to identify a way to define similarity between a pair of nodes in a graph, and there could be various different ways actually in order to do this there is no one single sacrosanct way in which you can actually define the similarity between a pair of nodes.

So, basically that is why people often say that clustering is actually more often art than of a science because these design of similarity measure actually is in many cases more important than the algorithm that you are using in order to do the clustering. The design of the similarity function is more important in many cases than the algorithm itself. So, this is something that cannot be overlooked in the first place that is this is one of the most crucial steps in devising a clustering algorithm or a community detection algorithm.

So, you can have various different measures. So, some of the ways you can some of the ways we have already looked into, for instance, you can think of that 2 nodes are similar if they are structurally equivalent. So, and we have looked into various notions of defining the extent of structural similarity in a graph and you can say, you can use any one of these measures to quantify the notion of similarity between a pair of nodes.

So, in such ways, not only structural similarity you can use a lot of other different methods say, for instance, just the number of common neighbors between them or say the assortativity that the mixing coefficient that we talked about. So, if both of them are rich they should be placed in the same community, the mixing parameter between them. So, all these different factors can be used to modular similarity function to model the similarity function that you will use in order to put nodes within a cluster call them similar to each other.

(Refer Slide Time: 10:50)



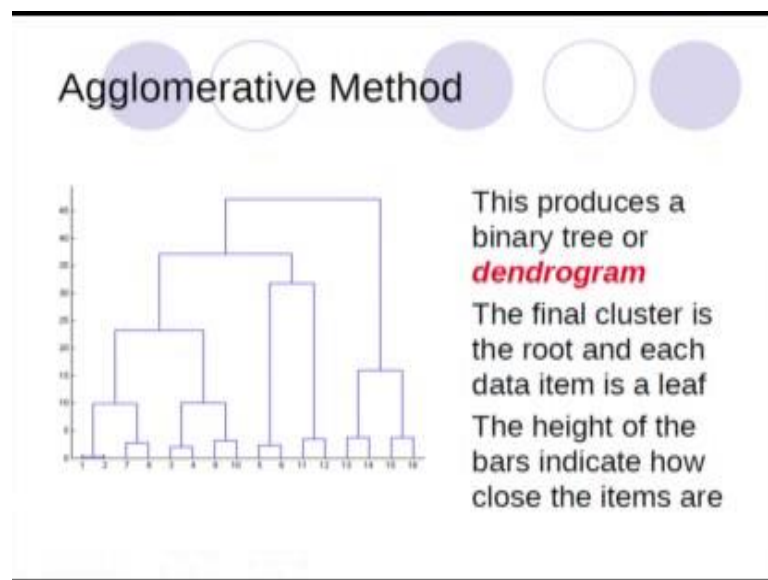
Agglomerative Method

- Start with every data point in a **separate cluster**
- Keep merging the **most similar pairs of data points/clusters** until we have one big cluster left
- This is called a **bottom-up or agglomerative method**

So, in a nutshell agglomerative methods actually can be divided into 3 steps. It is actually a 3 step process, as you see in the slides the first step is that you start with every data point in a separate cluster each and every data points. If there are n data point n nodes in the graph then you start with n different clusters.

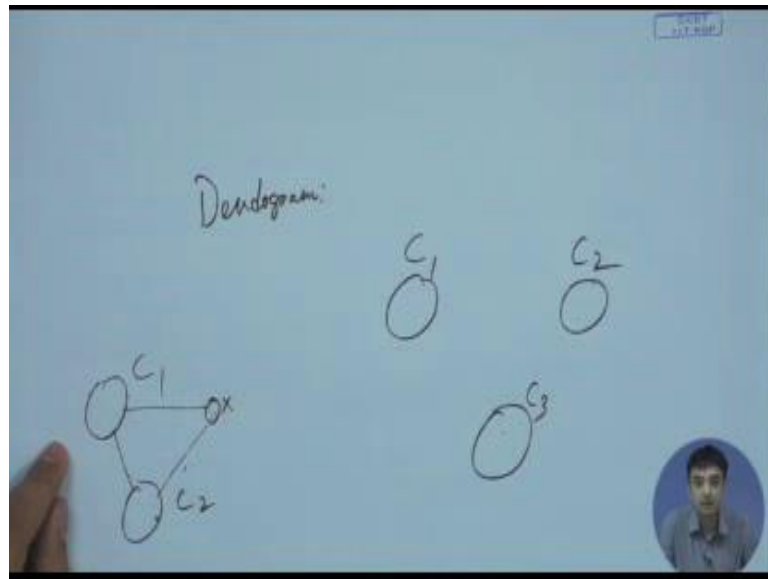
Now, keep merging the most similar pair of data points or clusters until you have one big cluster where you have all the nodes where you have taken all the nodes together. So, at every step what you do you merge a pair of nodes or 1 cluster with a node or 2 clusters and you keep iteratively doing this until and unless you have 1 big cluster which actually contains all the nodes in the graph. So, this is actually a bottom of agglomerative method its bottom of fashion, you grow from the bottom. So, you look into pairwise cluster distances and then based on it you actually merge the most similar clusters.

(Refer Slide Time: 11:48)



And while doing so, you can generate something called dendrogram.

(Refer Slide Time: 11:54)



So, this dendrogram actually gives you an idea, whenever you look at this dendrogram you actually get an idea of which clusters are most similar to one another and at what stage the 2 clusters have joined. So, for instance if you look at the slide here you see that there are a bunch of clusters that we can see and say there are nodes numbered from 1 to 16, the first 2 nodes, the height of bars in this dendrogram actually represents how close the 2 items are.

So, you see immediately by looking at the dendrogram you can infer that nodes 1 and 2 are most similar because they have the least height. So, they actually join first. Then the nodes 3 and 4 join together, then the nodes 5 and 6 join together, then in similar fashion 7 and 8, 9, 10 and then after a certain point in time there are clusters, say 5 and 6 and 11 and 12 join together at this particular point.

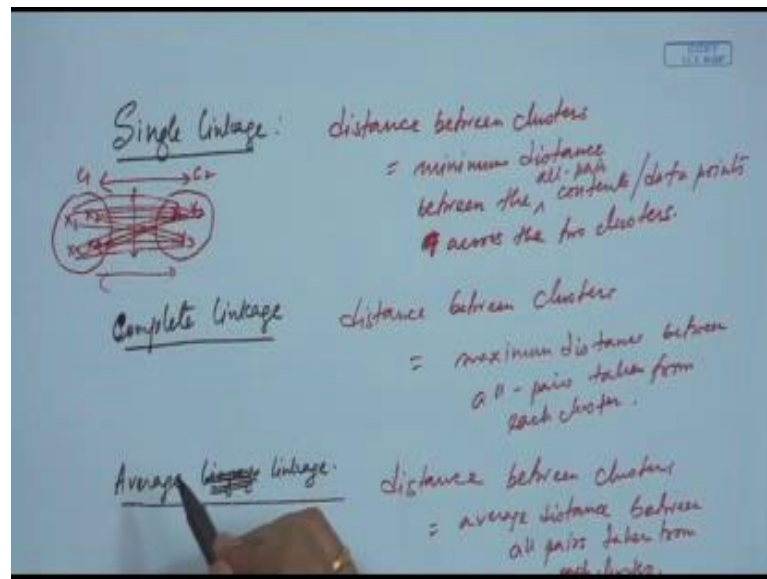
So, the height of this line actually shows the point of iteration in the iterative agglomerative algorithm where the 2 clusters have actually joined to form a new cluster. In this way, basically you have smaller clusters in an earliest stage of the iteration which actually join to form larger clusters until and unless you have 1 single cluster at the end of the full execution of the algorithm. Now, we have been continuously talking of the idea that 2 clusters need to be joined based on their similarity. So, you can estimate the

similarity between a pair of nodes by some similarity function, but the question is how to estimate the similarity between a pair of clusters or say 1 cluster and the node, say for instance if you have 2 clusters c_1 and c_2 here and say c_3 here.

Now, in the next iteration which are the 2 clusters that you should merge, you have to define somehow a similarity notion between all pair of clusters or say you have 2 clusters c_1 c_2 and the node x . So, now, in the next iteration whether x combines with c_1 or x combines with c_2 or c_1 and c_2 combines has to be decided. Now, these has to be decided based on some sort of similarity notion between c_1 and x or c_2 and x or c_1 and c_2 right. So, this idea is called actually the idea of linkage and again as you can imagine there could be various ways of defining the similarity between a pair of clusters or a node and a cluster.

So, if you can define the similarity between a pair of clusters that automatically gets translated to a similarity between a single node and a cluster. So, if you somehow can define the similarity between 2 clusters say c_1 and c_2 then automatically the notion of similarity between 1 single node and the cluster is also define because you can imagine that c_2 is just consisting of 1 single node. So, then the idea is that you have to somehow define the similarity notion between a pair of clusters. Now, in order to do that we come up with the idea of linkages and there could be at least 3 different forms of linkages.

(Refer Slide Time: 15:30)



The first form is called single linkage, the second one is called complete linkage and the third one is called average linkage. In the single linkage idea, what you do is that the distance between the clusters is set to the minimum distance between all pairs across the 2 clusters, this distance between clusters is equal to the minimum distance between the contents or the data points between the all pair contents or data points across the 2 clusters.

So, basically what we mean say for instance if you have 2 clusters c_1 and c_2 and there are data points x_1, x_2, x_3 and x_4 here and there are data points y_1, y_2 and y_3 here. So, you find out pairwise distance between $x_1, y_1, x_1, y_2, x_1, y_3, x_2, y_1, x_2, y_2, x_2, y_3, x_3, y_1, x_3, y_2, x_3, y_3$. Similarly $x_4, y_1, x_4, y_2, x_4, y_3$ and among all these you find out the minimum, among all these distances you find out the minimum distance that is actually set to the distance between the 2 clusters. So, the distance between the 2 clusters is equal to the minimum of all these distances.

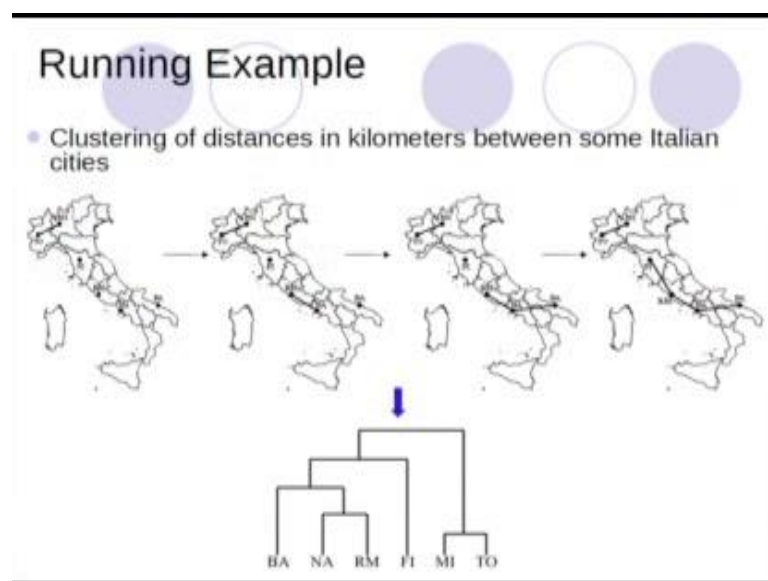
Complete linkage is actually just the opposite. So, distance between clusters is equal to the maximum distance between all pairs taken from each cluster. So, basically here again if you have the same example, you find out the pairwise distances between the entries of the cluster c_1 and the cluster c_2 and among them you find out the maximum distance

that is said to be the distance between the 2 clusters, that is what is complete linkage and the average linkage is nothing, but the average distance, average pairwise distance between the 2 clusters.

So, here distance between clusters is set to the average distance between all pairs taken from each cluster. So, again in the third case what you do is you find out the all pairs distances from the entries of c 1 to entries of c 2. Now, you find out the average of all these distances and that is set to be the distance between the cluster c 1 and c 2. So, these are the 3 different ways in which you can define the notion of similarity or distance between a set of 2 clusters.

So, using this now, we can envisage to do a clustering. So, we will take one example of hierarchical clustering hierarchical of agglomerative clustering and see how it works.

(Refer Slide Time: 20:06)



So, for instance let us take this map of Italy here and we have a bunch of cities. So, we have a bunch of cities, which are separated by some distance in say kilometers from 1 another. Now, given this matrix of distances can we find out an agglomerative clustering of these cities based on the notion of closeness of the cities? So, in order to do these steps are as follows. First of all let us say the initial distance matrix is like the following


between the pairs of cities.

(Refer Slide Time: 20:47)

Initial distance matrix between the pairs of cities of Italy.

	BA	FJ	MI	NA	RM	TO
BA	0	662	877	255	412	996
FJ	662	0	295	468	268	407
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	407	138	869	669	0

← Single linkage

← 

So, that is given by the following matrix by Bari, Florence, Milan, Naples, Rome and Torrain, similarly, Bari, Florence, Milan, Naples, Rome and Torrain. Now, we can fill in this matrix by the distance between these cities in kilometers. So, as you see the diagonal should be equal to 0. So, now, Bari from Florence is 662, 877, 255, 412 and 996; 662, 295, 468, 268, 400; 877, 295, 754, 564, 138; 255, 468, 754, 219, 869; 412, 268, 564, 219, 669; 996, 400, 138, 869, 669. So, in this way we have information about the pairwise distance between every pair of cities in Italy in kilometers.

Now, given this input distance matrix, we have to iteratively generate the agglomerative clustering algorithm; based on the agglomerative clustering algorithm we have to iteratively we generate the communities and using the measure of single linkage. So, if you use the measure of single linkage then the first 2 candidates that should actually get merged, the most similar pair if you see here the is the minimum distance is between Milan and Torrain, this is 138 kilometers only which is the smallest. So, the first 2 nodes that get joined are Milan and Torrain these are most similar nodes they get joined in the first iteration. Now, once they get joined we can reduce this matrix we can shrink this matrix because now Milan and Torrain will form 1 single entity.

(Refer Slide Time: 24:45)

	BA	FI	MI/TO	NA	RM
BA	0	662	877	255	412
FI	662	0	295	468	260
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	260	564	219	0

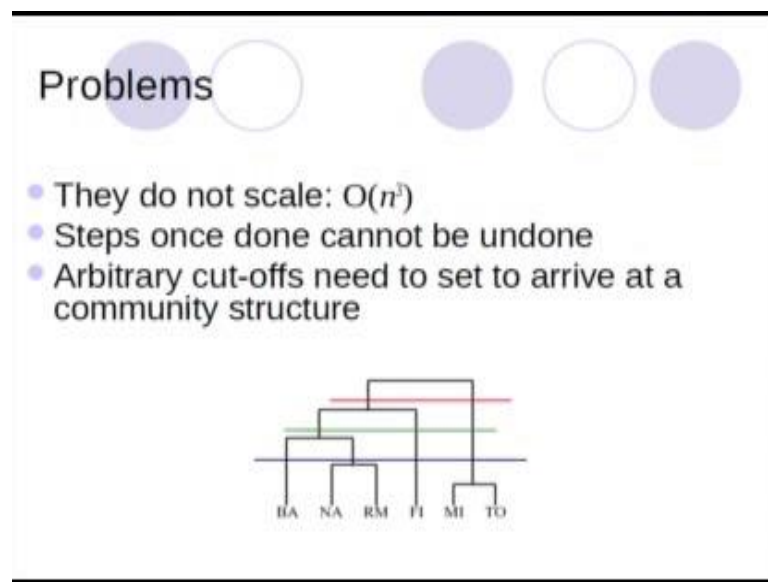
They will form 1 single node in the graph. Now, in the with that idea we can have a revision of the original distance matrix and we can construct the new distance matrix as follows Bari, Florence; now Milan and Torrain comes together, Naples and Rome.

Similarly, here Bari, Florence, Milan and Torrain together, Naples and Rome, again we can fill in the entries of this matrix 255, 412, 662, 295 468 260. Now, you see from Milan and Torrain we have to estimate. So, this is 1 cluster now, 1 single node, now we have to estimate the distance between this cluster and Bari. So, we find out the distance between Milan and Bari and Torrain and Bari and out of this, we put the smallest distance. So, in this case the smallest distance turns out to be 877, since we are using single linkage clustering. So, we put the smallest distance here, similarly we find out the distance between Milan and Florence and Torrain and Florence and we put the smallest among this which is 295 in this case. Similarly, 754 here and 564 here in this way you construct the matrix in the second iteration.

So, if you keep on doing this you will find that at every step there will be 1 cluster joining with a node or 2 clusters joining based on their similarity using single linkage or there will be 2 nodes joining. So, in this way if you keep on merging clusters, at the end if you look at the slides you will get a dendrogram like this; Milan and Torrain joins first

as we have already seen, then there is napoleon Rome which joins together with Bari and then that joins together with Florence and then this sub segment Milan and Torrain joins with the rest. So, in this way you can construct the hierarchical clustering of this distance matrix. So, this actually gives you an idea of which cities are very close to each other and this can come handy in making a trip plan. Suppose you are making a vacation to Italy what are the places you are you can visit you can make a nice choke out of that looking into this clusters.

(Refer Slide Time: 27:50)



Now, we can cut the clusters, we can cut this dendrogram at different places. So, you can cut here, you will get a different set of clusters, you will Bari separately in Napoleon, Rome separately in a separate cluster Florence as a separate cluster and Milan and Torrain. So, you can cut at different places like this, like the blue line or the green line or the red line based on different thresholds, based on different distance thresholds you can cut this dendrogram and accordingly you will get different set of clusters the generality of the clusters will depend on what threshold you use for cutting the dendrogram and so, this particular algorithm as we see is a bit inefficient because if you have say n such cities you will have n square matrix.

At every step you will have to process an order n square matrix in order to compute the

similarities. Now, then if there are n such nodes you can go in the worst case up to n iterations. So, then the total complexity of the algorithms scales as order of n cube, every step you have to process a n square matrix and this you have to do at least ten times, at most n times. So, then the worst case complexity would be order of n cube and imagine if it is a very large graph like a billion node graph then this kind of an algorithm is very, very inefficient.

Thank you very much.