NPTEL

NPTEL ONLINE CERTIFICATION COURSE

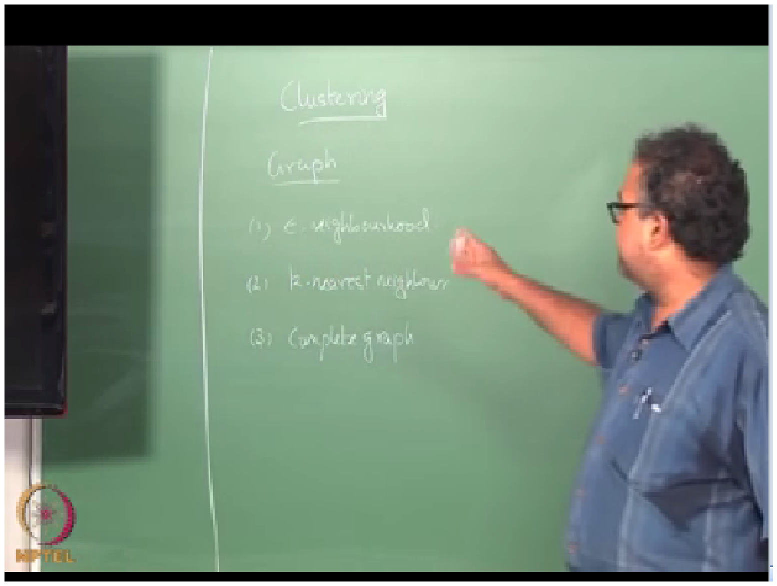
Introduction to Machine Learning

Lecture-79
Spectral Clustering

Pro. Balaraman Ravindran
Computer Science and Engineering
Indian Institute of Technology Madras

So the idea behind clustering is to group data points that are similar right and try to keep them as far away from things that are dissimilar etc., right.

(Refer Slide Time: 00:24)



And I said that a lot of convenience is to looking at the clustering problem as it is done on a graph right and we talked about you know hierarchical clustering you said you could do minimum spanning trees okay and then you could do all kinds of you know the single link complete link clusters you could form that using graphs right looking at threshold graphs etc. So we looked at a variety of different things you could do with the graphs right.

So it turns out that increasingly graph based clustering is becoming more and more popular right we talked about density based clustering and we said it could give you kind of arbitrary kind of clusters right and also we talked about cure and we said cure could also give you non convex clusters and things like that it turns out that if you use what are called spectral methods for clustering on graphs right.

You get all of these very in a very natural way right you get all kinds of weird clusters in a very natural way because essentially you're looking at graphs here you are not really looking at any kind of metric space right and as long as you are represented your data in a graph right you can do the clustering on it. So how do you get your data into a graph if you talk about constructing graph sort of your data so somebody tell me how you get this right.

So the first one is you can do something like you can do what is called epsilon neighborhood graph so what you do is you take a data point right draw a circle of radius epsilon around it and any point that is there in that radius you connect it right so this will be a symmetric relation so if A is within epsilon of B then B will be within epsilon of A so it would be a symmetric relation so you typically graphs that or constructed this way or undirected graphs.

Graphs that are constructed this way or undirected graphs and then second thing is typically graphs that are constructed this way are also unweighted graphs because epsilon is usually small right and the difference the differences in the distance also will be even smaller than epsilon so you do not really want to focus on those differences in the distances you just go ahead and assign them all the same weight okay.

So it is a clear so what do you do with epsilon neighborhood they still end up with and directed un-weighted graph typically. The second thing could do is call the k-nearest neighbor graph so what do I do I take a point I find the k-nearest neighbors to that point and correct them right now has a question is this relationship symmetric not necessarily right so a could be one of the k-nearest neighbors of be right but we need not be the k-nearest neighbors of a right.

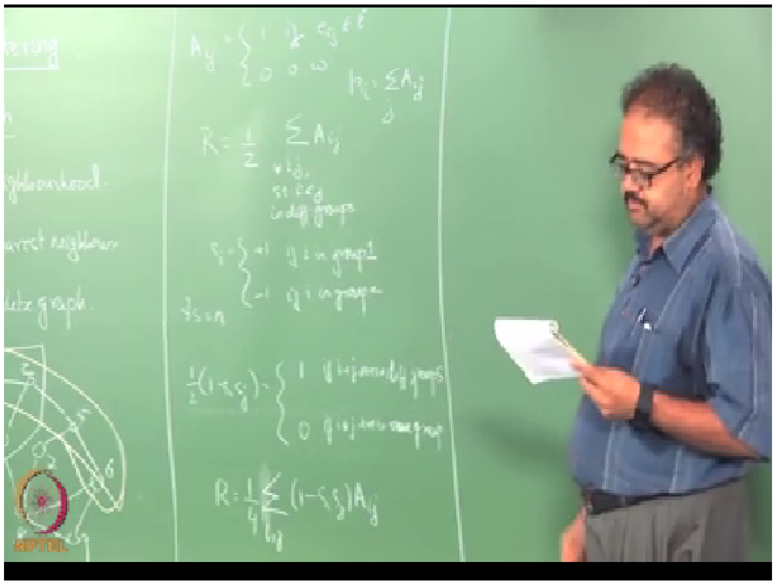
It could be slightly that could be more near neighbors that are clustered around a and so be might not fall in the k-nearest neighbor list so typically these crafts should be directed graphs you know there is an edge from A to B that means B is within the k-nearest neighbors list of a right so and then if there is edge from B to a also that means that the relationship is reciprocal but otherwise not right.

And also the nearest neighbors k-nearest neighbors could be very far away rather the distance need not be uniform so typically use a weight right so the most general form of this k-nearest neighbor graph should be a weighted directed graph right in the epsilon neighborhood case it will be an unweighted undirected graph the k-nearest neighbor case it will be a weighted directed graph but for reasons of convenience and because there are larger class of methods that operate with undirected graphs right.

We tend to treat the k-nearest neighbor graph also as an undirected graph essentially we ignore the direction on the arrows right so if there is an edge between A and B it means that either A is a k nearest neighbor of B or B is a k-nearest neighbor of a well or both I am incineration inclusive all right so both, both is fine so that is essentially what, what we will do normally right so we actually even treat the k-nearest neighbor graph as a as an undirected graph.

And so the third mechanism is we already given the graph right I told you that you are just given the similarity measures between their own so you are not really given the data points and from that you can construct the graph right so sometimes you essentially end up with the end up with a complete graph so in this case you have to give weights right otherwise it does not make sense so the complete graph will be a weighted graph right and it just says said okay

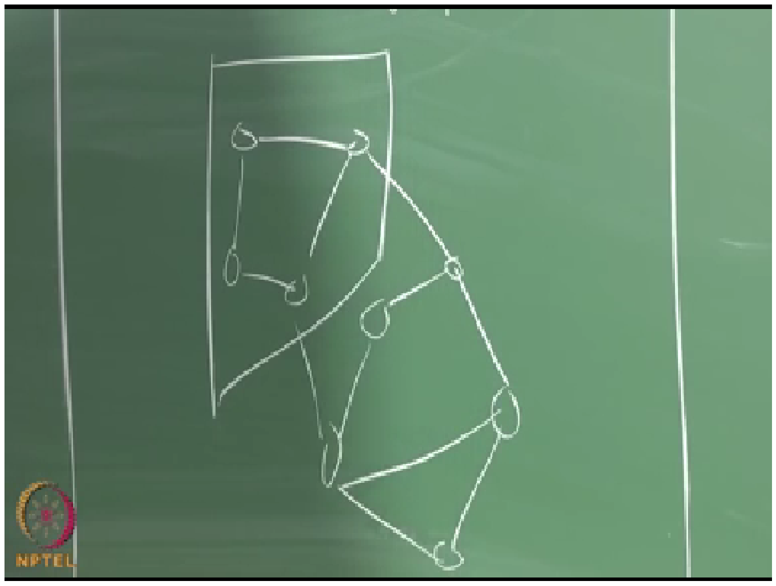
I will connect points a and Band the weight will be proportional to or inversely proportional to the distance between A and B right so it will be inversely proportional to the distance between a and B right so that means every point a and we will get connected if they are very very far away they will be connected with a very small weight but still they will all be connected okay. So this is essentially these are the three ways in which people typically take the data that you have unconverted into, into graphs okay.
(Refer Slide Time: 07:08)



So now what we do with this graphs so I will start off with a simple problem then of trying to split this into two two parts if I have data I want to split the data into two parts okay so little some notation is an adjacency matrix so I am going to assume that this is symmetric matrix or time be okay and so in fact all the three could be considered as symmetric matrices right if I am going if I were to convert all of them undirected graph all of them will be symmetric

So I am just going to think of this as a symmetric matrix right I am going to define the quantity called the cut value denoted by R so the cut value is essentially I will sum up all this A_{ij} such that i is in one group and j is in the other group so I have a graph now grab something like this let us say and I divide that into two groups that is one group that is another group right.

(Refer Slide Time: 08:59)



So I am going to sum up all of these just that i is in from i is in one group j is in the other group right let us give them numbers so that we can do some even some arbitrary numbers so what will I do so this sum will run over A_{15} A_{16} A_{17} A_{19} and A_{18} well all of them were meant to be 0 so we do not care right so likewise A_{21} no A_{25} A_{26} A_{27} and A_{29} so only A_{25} will be 1 so I will count that once right and then for 3 everything is 0 I do not care and 4 I will get the 1 I will count the once right.

And what then do I stop I keep going so 5 I will count it so what will happen 5 again will get a 1 so I will count it once right so 6 I do not care so 6 will be what so A_{62} A_{61} A_{64} A_{63} so all of them 0. Likewise I keep going and for 8 I get a 1okay so how many do I count I count 4 right. accounted for but truly how many edges have I cut 2 and that is why the half okay so this way I do I when I do this I count every edge twice right and some mode notation right.

So I am saying S_i is +1 if I is in group 1 is -1 if i is in group 2 just an indicator variable okay so one thing just remember that for all less S^T S will be n where n is the and this is number of nodes okay so n is a number of data points right so S_i is the ith entry in that vector S okay which indicates whether the higher data point belongs to class1 or group1 or group2 let us look at this expression

So when will be 1 sorry both in different groups right and they are both in different groups it will be 1 both in the same group that will be 0 so if i and j are in j are in the same group with me 0 if they are the different groups it will be1 right here okay. Just a bag and not a person load enough to be a person sorry now this just the assignment here right so we just taking some assignment when we are trying to evaluate it.

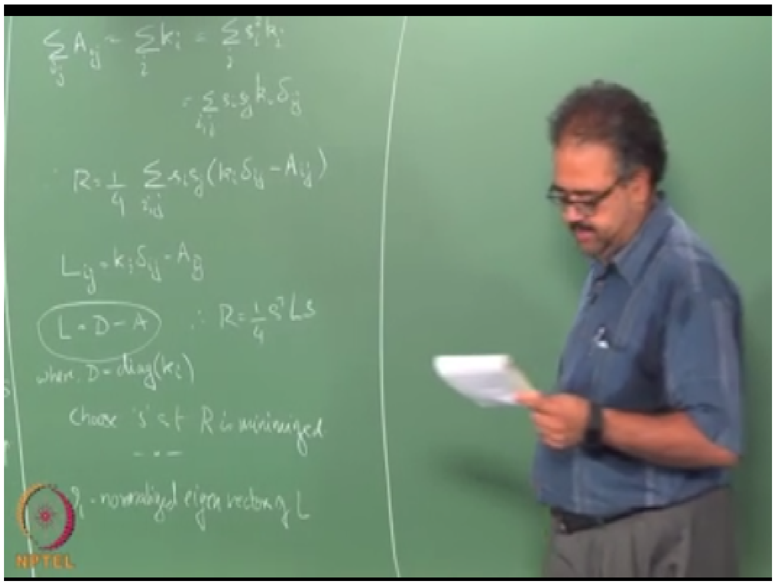
So like I said I have arbitrarily chosen this assignment that I cross in a different design like I could have said 1, 2, 5, 6 or in one cluster one group and the remainder or in another group and they could have evaluated that okay. Let us why do not we do that for a for fun right so what will be the R value get five right now R value if I so I can just take any clusters any grouping like this and I can evaluate and find our value right.

So my best is to find the R value is the smallest and would be the smallest all of you know the degenerate solution to this problem when all the nodes belong to group one or group to right so the cut will be zero. So we have to avoid the degenerate solution we will see that in fall out naturally from here a degenerate solution will come out come out as the best solution so we have to explicitly excluded great.

So so far so good so what are we doing here we have essentially come up with a mathematical expression that captures this right such that i and j are in different groups so instead of saying that I could just multiply the terms here by this and some over all ij. If they are in the same group the number will get added if they are in different groups I will get a zero right so instead of doing something number some like this.

That can do something more compact weight so I can write RS okay so that half I have another half here so they get a size a run over 1 to n okay so one other notation I should introduce what is that degree right the phone to get the degree of nodded I just sum over j A_{ij} right so that will give me the degree of notice so what will be the first term in here just a ij right is some over that will be the first term for R okay.

(Refer Slide Time: 16:51)



I am just going to take that and rewrite that into something more complex that what is sum over ij A_{ij} twice the number of edges but we will write it a little more complex form so that we can go back plug it in and derive one nice expression okay.

So this I will write it as summation over I of k_i right can do that the here Δ_{ij} is a function that is 1 if i = j 0 otherwise right sounds looks like this way but there is a purpose for doing it but all of you buy this but if you are wondering why S_i^2 is okay S_i^2 is just 1 okay so I am just multiplying it by 1 so it is fine and then I am just splitting that into S_i S_j and writing it into a more complex form.

Now I am going to substitute it back there right so essentially what I have done here is I have produced S_i S_j on both terms right right right. So this is just this lot of algebra here nothing nothing high school stuff nothing complicated so I am going to introduce a new matrix L_{ij} which so the ij at entry of the matrix L will be this right so if you think about it yet I can write this as D-A where D is a diagonal matrix where the ith entry in the diagonal is the degree of node i right.

So this expression is sometimes called the laplacian is called the laplacian of the matrix A right is also called the un normalized laplacian of the matrix A so the normalized laplacian you actually do a transformation on this and has got better properties in terms of both in terms of clustering and also in terms of other things which people use the laplacian for but I am just going to show you how to work with the un normalized laplacian.

In form right the actual algorithm for working with the normalized laplacian is exactly the same as working with a normalized laplacian except that proving things are slightly different showing that you are getting a good clustering is slightly different right so I will give you material to read up on both normalization and un normalized laplacian but we will look un normalized in the class right.

So this makes sense so the laplacian is D-A and it is not an arbitrary choice we arrived rated by trying to say that I am looking at the cut size right and I want something that characterizes the cut size then we did a lot of algebra after that we ended up with D-A it is not the only way to derive the laplacian there are many ways in which many different fields in which people are actually independently come up with something that looks like the laplacian.

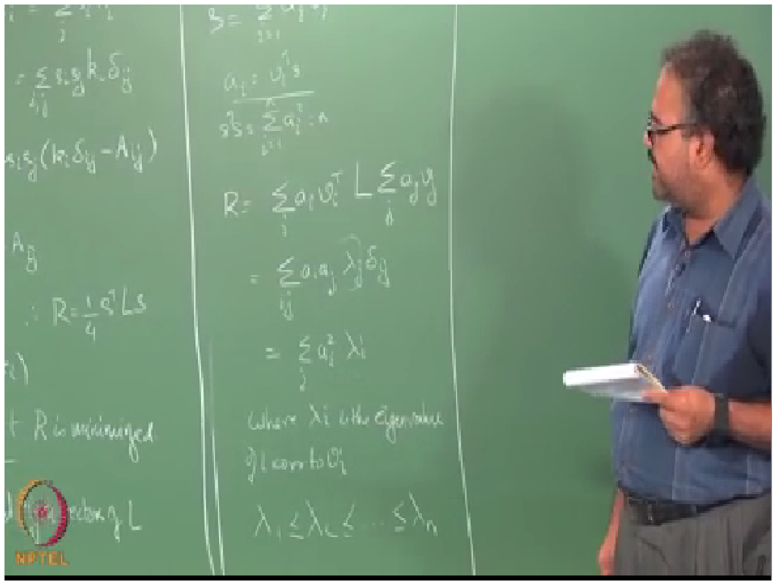
And then it is got wide applicability so how many actually use laplacians before of graphs this not it not in matrices okay I am not yeah it is very closely related to your laplacian in the continuous domain actually so it is its yeah now I am not going to get into that is too much so we do over matrix notation transformation right so I can say that might cut this just 1/4 S^T LS right so S_i S_j x L_{ij} right.

So essentially S^T LS so now the goal is choose S is that R is minimized so the the development I am following comes from Neumann who is one of the pioneers in looking at graph partitioning and what the social network people called community direction and soon so for right I mean he did not come up with a spectral clustering right spectral clustering is older but so Neumann came up with this very nice way of introducing spectral clustering right.

Without ever going into real analysis or functional analysis or anything just going with little bit of algebra and a little bit of linear algebra so far we are not even done linear algebra right except for that very rudimentary transformation right so just looking at very basic concepts he kind of motivates how you do the partitioning that if you typically go read other things right they will start off with say okay.

Here are the properties of the laplacian and we apply these properties and therefore we can solve this like in one short and here is an answer and things like that so it becomes little more tricky so I am going to make you read all of that but I just want you to appreciate that the idea behind this is fairly straightforward so whatever we are doing is sadly straightforward right so now so now we will enter into the spectral domain right. I want to say Vi denotes a kind of a normalized at the Vi is denote the normalized eigenvectors of L right.

(Refer Slide Time: 25:55)



I take the S and I can right it as right so I have I have eigenvectors that they are going to span my n-dimensional space so I can just take any point S which is in the n dimensional space I can write it as combination of the eigenvectors right so you about it A_i is just essentially and s'f' sure why I need this anyway yeah so note that S^TS is what n and S^TS is what sum of A^2 right so that should be equal to n that is a constraint that we should have in remember that.

So now let us go back and make the R look complex again the let us A^T S right so here comes R the fact that V_i and these are eigenvectors of L right so what would be L will be j λjvj right so I can just write it as λjvj and right what about V_i^T V_j Δ_{ij} right A^T V_j will be 0 i is not equal to j right so essentially what I will be left out with this then I mean I Creighton λ at λ_j which over it does not matter.

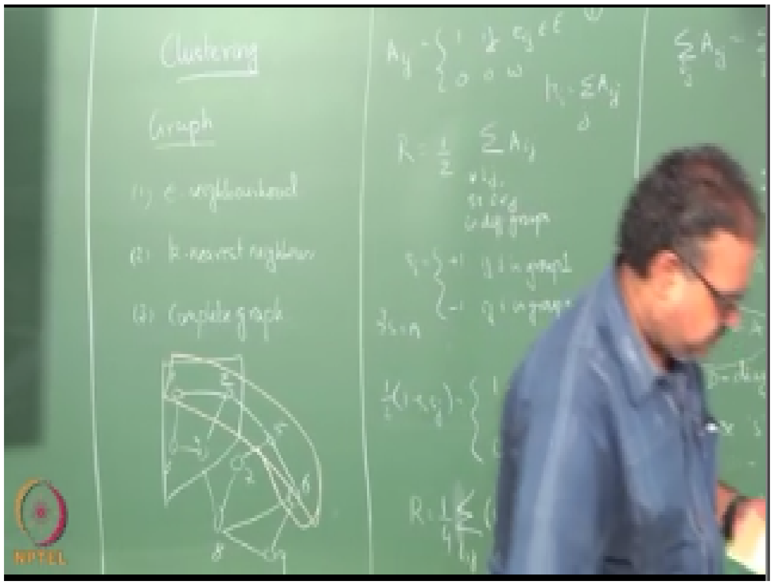
We had write λ_j because this stick with this notation that I had the j on the right-hand side i is it fine so this is essentially equal to what is Aij so S is a vector right so I am trying to express S in the coordinate space defined by the eigenvectors of L so essentially what I do is take a look at the projection of S on each of those dimensions and then write it as a sum of those right so that is essentially what Aij is.

This is again basic linear algebra the only only place where we use the spectra is when we went from here to here right so I am using the fact that my VS are going to give me a basis like and in fact we are giving me an orthogonal well they are giving me an ortho normal basis right because I am assuming the normalized so giving me also normal basis and that is why I get this Δ_{ij} here right and then the only place where I used it is the introduction of this λ here λ_j here.

So this is K right so why I can collapse the summation back into this right great now what we do so what is λ here so I am going to assume that the λ sour index is that λ 1 is the smallest Eigen value λ2 is the next highest and so on so for right okay. So now if I want to minimize this expression right all I really want to do is minimize my R now found to minimize the expression so what should I do.

So I should choose my A says that the maximum weight is scale placed on the smallest λ right in fact I should pace all the weight of the smallest λ right and what is all the weight right so we have that somewhere right so summation A_i^2 is n right so I need to keep all the weight on what is the smallest value that is λ1 right what is λ1 what is that if you are wondering what that symbol meant was L.

(Refer Slide Time: 32:52)



So then they summing up one row of the laplacian so what do I get called race come on so this will be what k_i totally listening okay so each row of the laplacian would sum to 0 and if you think about it what is the diagonal element the degree of that node and all the off-diagonal elements or negatives of the edges rate so whenever there is an edge there will be a -1 there is no edge there will be nothing so the diagonal entries degree and then I will have as many -1 as they are our neighbors.

So adding up the row will give me 0 likewise adding a column will give me symmetry when you are thinking so long symmetric okay so it should give you the same thing right great so now what does this really mean the fact that the row sum is 0 what does it mean all one is an eigenvector with Eigen value of 0 and with a little bit of additional work you can show that this is the smallest right.

Because its laplacian this is symmetric right and it also turns out to be positive semi-definite and that for all the Eigen values will have to be non-negative so you can you can easily verify that L will be positive semi-definite so you know how to verify it will be positive semi-definite right take an arbitrary F so F^T LF should be greater than or equal to 0 so if you can show that for any arbitrary choice of F that will happen then you are all set.

So it is for L is positive semi definite that for 0 will be the lowest Eigen value so all you need to do is make sure that you put all your weight on the first Eigen value okay you are all set right so what is your first eigenvector in this case and so by V_1choice is actually normalized so it will be divided by 1√n 1√n 1√n 1√n will be the eigenvector that I am choosing for this in this case right so how will I minimize this I essentially have to align my S with this eigenvector right so I will get the maximum A_1 correct.

We will see the thing so what is hey one what is the A1 A1 is $V_i{}^T$ S so if I want all the way to go to a1 essentially all I need to do is choose sin the direction of V1 and what will be S what will be the inner product of s with any of the other v is 0 so if I choose it to be in the direction of V1 I get my assignment but choosing it to the direction of e 1 means what I assign it to all once right this is exactly the degenerate solution we are talking about right.

So taking s to be all once means I am putting everything in group 1 at all I can put everything in group group2 does not matter one of those things right so choosing yes to be all ones essentially is a degenerate solution I was selling it avoid right so what we should do is we should say that I am going to exclude this because this is a degenerate solution so find an S for me right that is orthogonal to all once right.

Because I have to exclude this direction I have to look at the space spanned by the rest of the eigenvectors so that space will be orthogonal to the original is this dimension that we are excluding so essentially I have to put an additional constraint I am not only interested in minimizing R but I want a minimize that is orthogonal to the all ones so we want actually exclude this solution right.

So there are many ways in which you can do this one simple fix is to say that I will fix fix the sizes of the two groups to n1 and n2 now fix the sizes of the two groups to n1 and n2 so essentially now I am saying find two groups that minimizes the cut size say step one group as n1 elements other group has n 2 elements right so if there is some prior knowledge that you have that lets you decide on this n1 n2 grade.

Otherwise what you can do is you can start off by saying okay. I am going to assume I wonder 50-50 split okay and then search around that to get a better thing or you do not have to do this at all you can type something else okay which I will talk about right now just a second so once we have excluded right one what is the next thing you can possibly try to do align with v2 I try to put all the weight on λ2 right.

Because λ1 is excluded forest so we have to go to λ to try to put all the weight on λ a 2 right so this is also called the also called the fiddler vector Fiddler Fiddler vector. So v2 v2 essentially the λ the Eigen vector corresponding to the second smallest Eigen value of the laplacian is called the figure vector so what you try to do is you choose your s to lie in the direction of V2 okay can we do that we cannot why wait stick-to-itiveness one exactly.

So s is already restricted rate this can be have only either + or - one entries so I cannot really choose a your S arbitrary early just to lie in the direction of V 2 so I have to look at this pace of + or -1 is and figure out which is the closest to be too right so essentially what I want to do is I want to maximize right I want to maximize that right when this is Swift for S that is either +or -1 right so I can get this so essentially the maxima will be achieved with this is the maxima right.

You cannot get better than this the maximum will be achieved when I am making sure that these signs are all positive shape basically signs are all same right so we'll make sure that signs are all positive or all negative right so that is essentially what I have to ensure here so what I do now is then I look at me to write and so this is essentially the higher component of V2 I look at the add component of V2 if it is greater than 0.

I make SI + 1which lesser than 0 I make SI -1so that is basically so now I have a way of dividing it into two groups and hopefully okay I have a sufficient number that r plus 1 and a sufficient number that are -1 therefore I do not end up getting everything in one class or the other usually you will find that that is a very nice in fact yeah I will put up some materials on the non model right.

So where you can see pictures of the the eigenvectors it is what what does well what does v1 look like what does be to look like and so on so forth so what would be one look like straight line right v1 will be a straight line assuming that your graph is connected if we graph as multiple components right so what will happen is your Eigen values 0 will have a higher multiplicity then 01 right.

Rates of Eigen so the number of components in your system number of components in your graph essentially is given by the multiplicity of the Eigen value 0 right and the Eigen vectors like the first Eigen vector will essentially be an indicator function saying which node belongs to the first component right the second Eigen vector will be an indicator function that says which nodes belong to the second component right.

So all the Eigen vectors corresponding to the Eigen value zero will essentially be indicated functions of which components in the graph their underlying thing belongs two sessions will be something like this one for all this way and 0 elsewhere right and then next one will go like this for a while and then it will be 0 elsewhere the third one will be so it will be like that so so so that you can you will see that so it is not that the first eigenvector will always be flat right.

If you have multiple components you actually see indicator functions there okay and in that case right how will you do clustering your components already give you clusters the only tricky part is now within the clusters within the components do you want to do clustering in which case you will not actually look at the appropriate Eigen vectors right so essentially you could either say that okay I am going to take the designator component separately it.

And then find out the Fiedler vector there can then assign it within that component right so if you should get this is Libby doing that before somebody asked me what we do for equal to zero just put it somewhere else great and the second λ to write is also sometimes called the algebraic connectivity of the graph it tells you how well-connected the graph is right and then things like that there were a whole bunch of other things associated with the interpretations associated with each of the Eigen values of the laplacian it.

But the second one is the most important one okay great so so far we have been talking about dividing it into two right so what if you want to do more than two what is previous kind of clustering methods but this requires one possible nothing you take the adjacency matrix okay the whole thing will reduce to this take the adjacency matrix okay compute the laplacian compute the Eigen vectors of the laplacian do not even compute all the eigenvectors of laplacian.

And I want you to compete only the second eigenvector of the laplacian so there are incremental methods that actually give you the laplacian one at a time instead of actually doing the I mean the eigenvectors one at a time instead of doing it completely their incremental methods which can work on very large graphs right and then give you the eigenvectors one at a time where you just compute the second eigenvector right.

And then do this that is basically it right so this is step one step two right well I do not have a proper step three anywhere step three is essentially finding these finding the spectra and step four is this now if you are doing it namely finding the Eigen vectors of the laplacian is a very expensive operation right so that is that is a problem so one is not cheap step one is not cheap right when it is a onetime computation.

You do that at the beginning right so you do that at the beginning is not iterative it is it at the beginning you can store this side so if you have one some of the other methods also become little easier but not k-means because in K means you have an arbitrary point somewhere even it in every iteration you have an arbitrary point which is centroid and now we have to find the distance to the arbitrary point.

So you will not be able to minimize this but here you are not introducing any arbitrary points only the fixed end points that you have and we can find the distances between the fixed end points and use any one of those methods and get my adjacency matrix right and once I just do a this is this is fairly trivial right this is fairly trivial and then after that I do this this is again time-consuming but once I have done that this is again fairly trivial and I get the clusters okay.

So the advantages are it tends to give you a lot more varied cluster you know I mean there is no restriction on the shapes of the clusters that you are finding right so you can find all kinds of different shapes in the classes there is no implicit assumption about that and that turns out to be incredibly robust to small noise and things so there are lots of nice properties to spectral clustering right.

And therefore this one of the reasons is becoming more popular nobody says lots of tools and packets are available it is not like you're doing multiple passes over the data in fact over the data you will do one pass when you construct your A_{ij} right after that is all operations with the adjacency matrix so does not matter what how larger dimension your data was in right you do one pass over the data right or well however cleverly you want to organize it in right you do one pass over the data right or well however cleverly you want to organize it right you fiddle around with the data once and you get a IJ now that becomes a dimensionality of the data so data could lie in a P dimensional space right.

So P could be far larger than n right but if will be working only with the N dimensional data typically when is far larger than P sorry image data yeah an image data if P is for far larger than in is it yeah depends on how you count p and in suppose yeah but they are typically PPis far larger than end in liquid working with image data so in many of these domains where you have this kind of structure strictly clustering is lot more helpful right.

And I also got a lot of cool theory it is you in fact you know exactly where you are approximating right even though you cannot tell by how much you are approximating it by at least you have some kind of understanding of what is going on and K means well you have a.m. to guide you there but so you need the reversion of k-means ray right so you have to get you there but it still it is a rough approximation right.

So all of this is confusing but you really have to do only these three steps to get your clusters so one thing I forgot to mention here before I go on to multiple clusters so what if you have the sizes fixed to n1 and n2right so how will what will you do then so can you just do this essay equal to plus one if is greater than or equal to0 and minus 1 if it is less than zero you cannot do that right.

I might have more in I might not be able to match then one into so what do I do in that case so I start off from the most positive end right and I keep marking everything as one until they reach n one right and the remainder I say is into which is the cluster 2 will make sense start off I order the points from most positive to most negative Eigen vector component right start at one end right I start at the positive most positive end right and I say everything is +1 until reach n 1 points and then the remainder will say is – 1.

Alternatively I can start from the most negative end I can keep going until I reach n 1 points now not in two points that's the something okay until I reach n 1 points right and then assign the remaining n 2to plus 1 these are actually two different ways of splitting the graph but if I had gone from the most negative point and gone up to n 2 and then assign the remaining 10 well that will be the same right but if I choose the n1 either from the top or from the bottom.

So I will actually get two different split points right I will get two different clustering so which one do you pick which one which of whichever one has the lower arm right so now that I have a method of comparing I just compare our which one has the lowest are I will pick that ok so this VI 2 is there right so I will order it the most positive VA to the most negative so we what is VI to the IH component of the second Eigen vector.

So I will take the component wise and I will sort it so that the most positive is at the top and the most negative is at the bottom okay so how do you do multiple clusters one way to do it is to repeat divide the two clusters right divided into two now I have two graphs right so once I have done the separation I have two graphs and in each of those curves I can go and divide it into two again right can I reuse the same Eigen vectors whatever thing I have done no typically no right so I that essentially reduce everything all over again.

Right I would have changed even though a lot of the connections are still the same right would still be having some sub matrix preserve right I will still have some sub matrix pressured but because I have chopped off some of the entries right so I have to redo the computation again so I sensed have to redo this again find the fielder value for that reduced graft not proceed okay instead a satisfactory solution may be not now.

When I say divided in two I mean you might have made choices right which you do not want to-do should have made a different choice if we say divided into four in the first place not divided into three in the first place so 3 is even worse right so what would you do if you have three clusters so which one of the two will you split into two so I start off by dividing something into to right and then I want three cluster so I have to pick one of them.

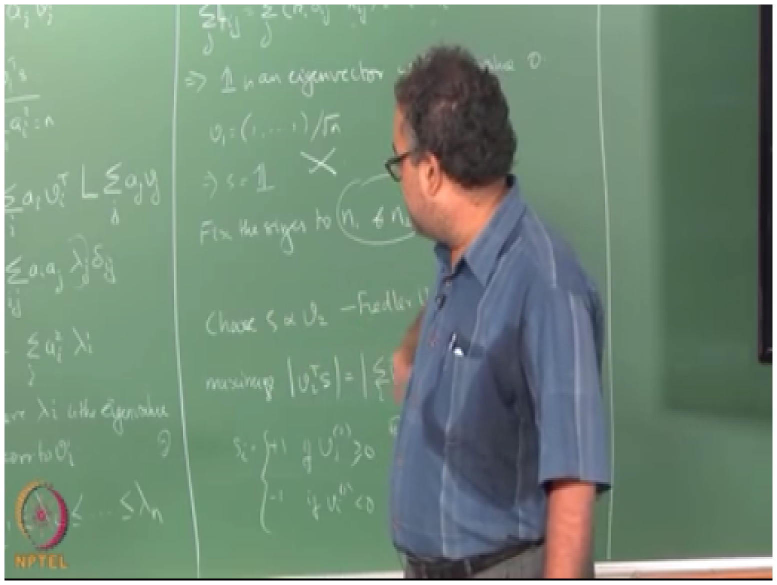
So I can pick the larger one and divided into two but that not my that might not be the right choice right then you can argue saying that kind of pic divide both into to whichever gives you the better cut value take that right and leave the other ones a whole cluster you could do the thing of all kinds of heuristics to do this so typically this repeated divisionism done right when you are looking to split it into some nice powers of two right.

And quite often this is something which people use when they are looking at vise layouts right when they are trying to look at layouts on the chip and then they want to do some kind of segmentation of the circuits that they want to put on the ship but they typically end up doing something like this so the reason they want to do this is so I want to put two things far apart on the chip I better have few wires going cross right.

So these are the line sat I am cutting right so they want to keep this as small as possible the number of wires longer wires I have to draw they want to keep these things as small as possible So they essentially do this kind of repeated clustering and they use spectral clustering a lot in that so what people do is I am just going to tell you what people are not going to actually explain why that is a good what way of doing it I am going to let you read it up so what people essentially does they do k-means clustering I want Clusters okay they do k-means clustering but in a different space what they do is they take the data points right they do all of these things and then they compute the top k I when vectors.

If they compute the top k eigenvectors and each data point right each node in the graph that we had originally will get transformed to a point in ask-dimensional space so what would that mean suppose I have data point I data point I will be represented by v1 I v2iv3i all the way to VK a right so essentially I will be arranging the eigenvectors and then reading of throws to get my data points so set clear it.

(Refer Slide Time: 59:04)



So I will essentially be doing this so V 1 V 2 The column vector say okay this is actually a matrix right so v1 is v1 runs like this we Torrance like this we see lunch like so then the first row here is essentially a representation for data point one the second dose are presentation for data point to the third row is a representation for data point 3 and so on so forth so I take all of these things and then run k-means on that space.

It turns out that if the graph originally had good separation it is the data itself was originally nicely separated into K clusters right if there is some way of separating them into K clusters then the data points in this projected space that will be well separated I will very clearly see k groups so it is easy for me to recover this with k-means so k-means gets confused if we have all kinds of if your representation is not is not proper right so right so the usual example is this.

That is one okay forget or the other one starts but let us do it this way that is another right so our density based clustering methods like DB scan well written this as one cluster and that has another cluster right but k-means will completely mess-up right k-means will do something like okay this is one cluster and that is another cluster or something very weird right.

But then what Happens in the spectral domain is that when intake these data points right and then try to project them into my the eigenvector space they actually project into different parts in the space right these points will project to somewhere here these points will project to somewhere there and I can run by K means very easily and recover these data points.

I they occur these clusters sorry right so that is essentially the idea behind doing multiple clusters with the spectra and so this is where the un-normalized normalized business becomes important right so far we didn't have to worry about the difference between normalization normalized and there are different definitions for the normalized laplacian right so one definition is essentially it is whatever the power minus half meets people number d right so d is a diagonal matrix so L is this one.

So you take this is one definition for a normalized laplacian is also sometimes called the centered laplacian and then the other definition I believe is sorry I which is essentially I - D inverse when I- D inverse aright sorry guys back there and I was blocking whatever I was written here so the firestone is called the centered laplacian the second one is called the random walk laplacian.

So why is it called the random walk laplacian think of what d inverses means is for every entry where if there is an edge between that node and J that entry will be 1 by degree of I so if I am doing a random walk from I that is a probability with which I will take that edge right so I look at all my neighbors it will be taking edge with equal probability right so that is what the random walk test so 1by d IJ 1 by did will be the probability with which I will take any particular edge therefore the second thing is called the random walk laplacian and then they use that also right.

And apart from the definitely difference in the definitions of the Laplace jeans the algorithms are typically the same right so you construct the adjacency matrix find the Laplace shape find the eigenvectors then if you are looking at two clusters you do this if you are looking at multiple clusters you form that matrix a so that will be an N by K n by K matrix so form that and then you do-means on the data points and the only thing that will change is which laplacian you are using through water right