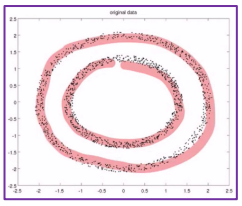
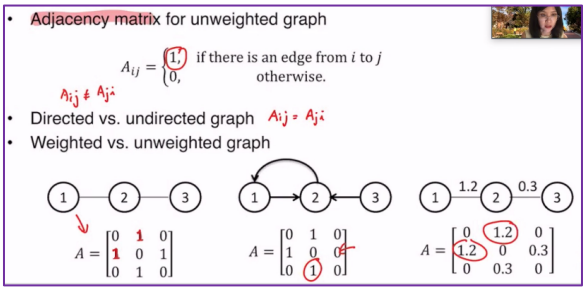
We'll talk about spectral clustering, which is another way of performing clustering of data points. So, again, this is an unsupervised learning method.

Review: Let's first review the last lecture we talk about so-called **k-means clustering**. Given m data points, and we're representing these data points using vectors. Each data point is an n-dimensional vector. And we have m such data points. **Our goal is to find groups of these data points**. And these groups are represented by the centers of each cluster, which we called the centroids. Then also based on a centroid, we can design data point to one of the centroids that gives assignment of data points as well. So, this is represented by this function, this Pi. So, based on that our problem becomes solving the following optimization problem. We want to find the optimal assignment, and the optimal centroids jointly minimizing the sum of the square errors, after this assignment. We talk about different choices of the similarity or distance function. The vanilla version of k-means studies the ll distance. So, it's going to be x minus c Pi i squared ll distance. You can also replace this with other choice of similarity function. It's going to give you different results. And one feature of this problem is that this optimization problem to solve here is simply hard. In principle, you can solve it by enumeration, but that requires that you enumerate. There were large number of combinations. So, instead of doing that, so we indirectly, the k-means algorithm is a heuristic that alternate between two procedures. Again, you start with random initialization of k cluster centers. And then we alternate between two steps, the cluster assignment, and the centroid adjustment. And keep doing these two alternating, until there is no further change to the cluster centers and then you terminate.



How about this dataset? So, k-means seems to be working pretty well. Then y introducing another clustering algorithm. Let's look at this dataset as an example. And this is a pretty simple simulated dataset, it consists of two rings. And human eyes can tell there are clearly two clusters, two groups organized by these two rings, inner and outer. Let's see if k-means can recognize these two clusters. And here I'm going to run a demo to illustrate the points. This is my demo code (test_tworings.m). Two rings. And the first part generates the data, as you can see, it's generated by synthetic calculations. We add some noise to it, then we're going to perform k-means clustering. And here we called it the function of camp k-means. And we set number of components in too. See if we can find the two components. And then we see what's the problem. And then I will introduce another new algorithm. Here's the result. This is the data, what it looks like, and let's keep running. So, this is the result of k-means. And as you can see interestingly, the k-means is not able to identify the corrected two components. It does put colors. Each color representing one component on these points. But as you can see, the k-means slice these two rings in half. This is not the correct way you wanted to find the groups or the clusters. So, clearly it is a problem for k-means. And let me show you another algorithm. This is the result of the spectral clustering which is the topic for today. And as you can see, this algorithm is able to find the correct cluster or grouping of data points. You can see the inner loop is colored in red and the outer loop is colored in blue. And there are some points you can use as outliers, seems to be connecting the two rings. And this has been separated correctly as well. So, magically, spectral clustering works. And on the right-hand side, this figure I'm showing a plot which is the spectrum or so-called **graph Laplacian**. And I'm going to explain the meanings of these plots in more detail in the next lecture. So, what's really happening is spectral clustering gives us the correct identification with the galactic clusters. Will start to explain this.

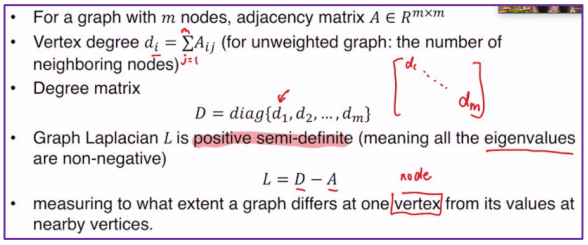
How about clustering nodes in social networks: So, getting back to here. Clustering nodes. What about clustering nodes in social networks? And so, again, this is the problem where you're given a pretty large network. And network means that you have the nodes in the network, and you have the edges between them. Different nodes, and edges between them representing their relationship. For example, you have each user in the social network being represented by a node. When the two users are friends, you're going to place an edge between them. This is a way to represent your data naturally. And so, one question is, can we find the clusters? clusters mean the group of users that belong to the same community. And one assumption we make is that users belong to the same community, tend to be more connected where they have more edges in between them. For example, I'm making a makeup example. Here I have this, belong to one community. This is another group nodes belongs to another community. And so, within the community, they have a tighter connection. Across the community, they have much weaker connection. Could we identify these communities or cluster of nodes in the node network? So, we briefly talk about the sick motivating example in our introduction slides as well. So, in this lecture, we'll talk about how do we achieve our goal finding the communities in this pretty large network.



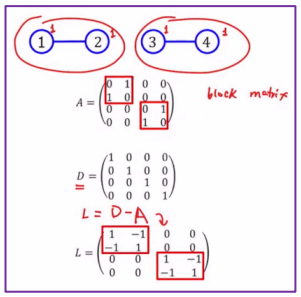
Representing graph using matrices: To describe the algorithm or first introducing a few concepts about graphs. The first one is called **adjacency matrix**. This is a fundamental measure about a graph, representing a graph. The adjacency matrix for animated graph is a binary matrix, meaning each entry is 0 or 1. If there's an edge between node I to node j, the corresponding entry is 1. Otherwise, it takes value 0. There's also the **distinction** between directed versus undirected graph. For the directed graph, the A_{ij} can be different from A_{ji}, meaning that there can be an edge pointing from ij, but not from ji. The edges have directions. For undirected graph, we're not going to distinguish

the directions of the edges. So, for undirected graph, the adjacency matrix are going to be symmetrical ($A_{ij} = A_{ji}$). There's also the **distinction** between weighted versus unweighted graph. For unweighted graph, the entries of the adjacency matrix are binary, either 0 or 1. For weighted graph, the entries for the non-zero values can be other than 1. So, below we gave a few examples of very simple networks and the corresponding adjacency matrix. For example, the first one is a three-node network and the corresponding adjacency matrix for this graph is given by this matrix. You can see the diagonal entries is 0 because there's no self-edge pointing to itself. So, the corresponding entry for diagonal is 0. For the off-diagonals, there's an edge between 1 and 2. So, we put our entry 1 here. We don't distinguish the direction of the edges, so the matrix is symmetrical. The second example is a directed graph, and in this case, there's an edge between 1 and 2 bi-directional and it's 1-D graph. Direction from edge pointing from node 3 to node 2. And I'm having the center here, but this corresponding entry is 0. And the last example is a weighted and directed graph. As you can see here, the entries for the non-zero edges can be other than 1. So, there's representing the weights and sometimes the unweighted graphs are used in machine learning algorithms to represent how tight the two nodes are connecting to each other.

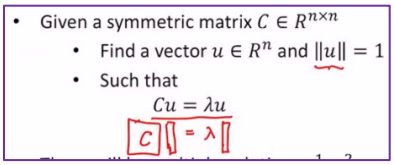
Graph Laplacian (undirected graph): The next fundamental quantity we'll define is a **graph Laplacian**. And here in this class, we'll focus on **undirected graph**. For a graph with m nodes, the adjacency matrix is going to be a m-by-m matrix. I first define a vertex degree. And a vertex degree for the i's node is the sum of A_{ij} for all the j's. So, j goes from 1 to m. And so, for the undirected graph, this essentially counts how many numbers of nodes are the neighbors of the i. So, if there's an edge connecting from node j to node i, the corresponding entry is 1, and otherwise it's 0. So, by summing this across all the j's, essentially, we're counting how many edges are connected to the i's node. So, we can count this for each of the nodes, and from there, each node is associated with a degree. The degree matrix is a diagonal matrix formed by placing the diagonal entries, the d's on the diagonal entries of this matrix. Now, we have the d and adjacency matrix. The graph Laplacian is the difference between the two matrices. L is D – A. And so, we can verify that Laplacian is positive semi-definite. And what that mean? It means that all the eigenvalues of graph Laplacians are non-negative. So, here we're talking about a concept called the **eigenvalues**. And in the next few slides, we will also reveal the basic concepts of Eigen decomposition. On a high level, intuitively, what does graph Laplacian mean? It means it measures roughly what extent a graft differs at one node or vertex form is valid and nearby vertexes. And here, vertex means a node and these two terms are used interchangeably when I describe the graph, which is very common.



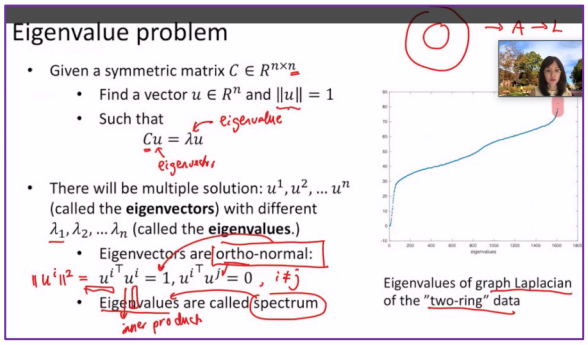
Graph Laplacian Example: Here we give a couple of examples illustrating what the graph Laplacian looks like. And you will start to feel why this defines the property of a graph. We start considering a graph is four nodes, and this is the adjacency matrix. As you can see, this graph have two clusters of nodes. There's no connection between these two clusters. One is connected to two and three is connected to four. If you write down the corresponding adjacency matrix, this is what it looks like. And so, here's the entry mark because there's edge between one and two. And these two entries are one because there's an edge between three and four and there's no other edges. And so, if you look at this adjacency matrix A, interestingly, we'll see that this A is a block matrix. It's a block matrix that can be decoupled into these two blocks. And so, representing the connection for each of the sub matrix form a group themselves. So, interesting, you can see A actually encodes the community or structure of this network or graph. And the degree matrix, in this case, is an identity matrix with 1's on the diagonal and 0 everywhere else. The reason is if you use that definition to find the degree for each of the nodes, the degree for all the nodes are equal to one. There's only one node connecting to each of the nodes. The Laplacian L is defined as D – A. Because D is an identity matrix and A is given by this expression. Simple algebra gives you this L expression. You will see you have the ones on the diagonals and minus ones at these locations because you have a –A. So, again interesting, you can see this graph Laplacian seem to encode the information that your network have two groups of nodes.



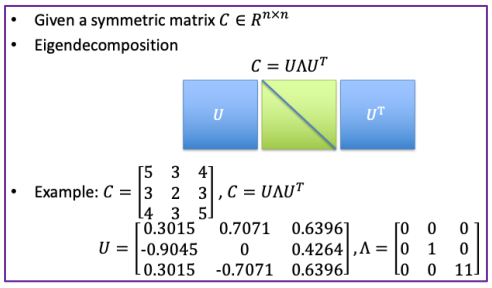
Eigenvalue problem: We talk about eigenvalues of Laplacian. So, what does it mean? Here I'm going to spend two slides or three slides talking about eigenvalue definitions. So, this is a very important concept in linear algebra. What is eigenvalue problem? Suppose you're given a symmetrical matrix C of dimension n by n. And we're interested in finding a vector that's n-dimensional whereas unit norm. So, the L to norm of u is equal to one is unit length. In this yield satisfy this equation: $Cu = \lambda u$. So, if we look closer, what does this equation mean? Basically, this C is the square matrix, and your u is the vector. And you multiply u by this C matrix. And by linear algebra, the output is also a column vector. And so, basically, this means you want to find the u such that this u is intrinsic to your matrix C. When you multiply C direct by this u, the result will be a scaled version of the same u. The direction doesn't change, you only change the magnitude. So, that's why it's called the eigenvalue, means is some intrinsic property associated with your matrix.



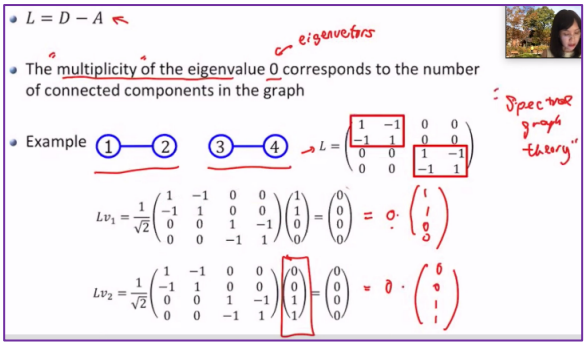
This u is called the eigenvector, and this Lambda is called the eigenvalue of C. And typically, when I say this is the Eigen pair, it's a pair of eigenvalue and eigenvector that are associated with each other. If we look at this equation, actually there are multiple solutions to it. And the number of solutions to it, it's going to be n, which is the dimension of your matrix. And you're going to have n eigenvectors for your matrix. Each of the eigenvector is having associated corresponding eigenvalue denoted by λI. And there are going to be n



eigenvalues for your matrix as well. And one thing to keep in mind is these lambdas can be 0, so you can have 0 eigenvalues and these λ's can also be equal to each other. You can have eigenvalues that are the same. So, what that means is basically sometimes you can have multiple eigenvectors associated with the same eigenvalues. And these eigenvectors also satisfy this requirement. The first one is your unit norm. So, the inner product, this is called inner product between this vector. So, transpose column vector, row times the column vector equals to 1. And recall that by basic linear algebra, this inner product means the L2 norm squared. So, this is actually the squared L2 norm of the eigen vector u. And so, this basically says all the eigenvector happen lens one. And besides that, for different eigen vectors, I doesn't equal to j, there are orthogonal to each other. So, $u^{iT}u^j = 0$, these are called ortho-normal vectors. Orthogonal means that their inner product is 0, and normal means that their lengths are equal to one. So, these sets of eigenvectors are called orthonormal. The spectrum is another name to call the set of eigenvalues because the spectrum contains a lot of information about the property of matrix C, we typically are quite interesting look at the eigenvalues and the corresponding spectrum. For example, here I am plotting the spectrum of the graph Laplacian for the touring data. This is the demo that I showed earlier in this lecture. You have this touring dataset. Then I can define graph associated adjacency matrix, and I can find the corresponding L. And then I compute the eigenvalues of my L Laplacian. And this is what the graph Laplacian eigenvalue look like. One thing that's interesting in this plot is you can realize there are two isolated, very large eigenvalues that are distinct from all the other eigenvalues. So, that is a quite interesting observation. In fact, that's one of the essential features indicating that you can probably play spectral clustering on the string dataset.

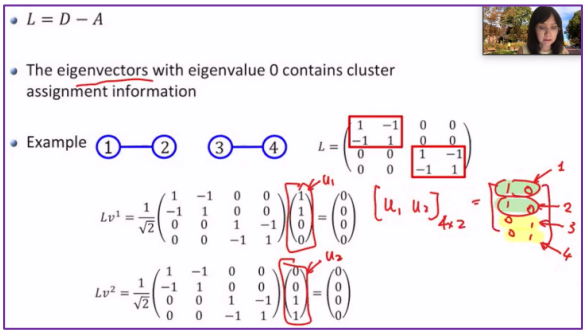


Eigendecomposition: Here is a definition of Eigendecomposition. We will focus our attention to symmetric matrices of dimension, for example, n by n. The Eigendecomposition of this symmetric matrix is given by $U\lambda U^T$, where the U is an orthonormal matrix and lambda is a diagonal matrix, contains all the eigenvalues and U^T is the transpose of U. And typically for symmetric matrices, the U is an orthonormal matrix. That means each column of U is a vector of L2 norm one. And the inverse of the U is equal to U^T. So, in this example, I'm giving here, that computes the Eigendecomposition of a very simple 3-by-3 matrix. The U is given by this matrix. You can verify each column is having L2 norm one. This is the diagonal matrix contains all the eigenvalues. You can see one of the eigenvalues is 0. So, as I would like to point out, this can happen that you will have 0 eigenvalues. And so, this means that your matrix C has Rank 2. So, it's low rank, it's ranked efficient because the rank is smaller than the size of the matrix. It also means that there is a linear dependence between the columns and the rows of the matrix.

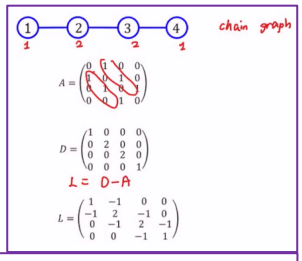


Property I of Graph Laplacian: So, now let's look at graph Laplacian we have just defined and see what are some interesting properties that can help us develop our reasons for finding clusters in the graph. So, recall the definition, Graph Laplacian is D - A. The first observation is that the multiplicity of eigenvalues 0. What does it mean? That means eigenvalue have the corresponding eigenvector. There can be more than one eigenvector for eigenvalue 0 to eigenvectors. The number of different eigenvectors corresponding to 0, associated with eigenvalue 0. How many eigenvectors correspond to eigenvalues 0, that's called the multiplicity of eigenvalue 0. And this is going to be equal to the number of connected components in the graph. So, what do I mean? Let's look at this very simple example. We have four nodes and we have found out before that the graph Laplacian is given by this expression. Maybe has two sub blocks of matrices that are non-zero, all the other entries are 0. And so, let's look at what is going to be the eigenvalue and eigenvector for this graph Laplacian. So, we find that the first eigenvector is looking like this. So, this eigenvector, if you perform simple linear algebra and can see. So, if I multiply L with what does it mean? I'm going to multiply this row with this column vector, excuse me. So, you can see after calculation the corresponding entry is zero. Why? Because essentially this is 1 times 1, 1 time minus 1 and you add them together, you get a 0. And same thing here and the corresponding entry is zero in here. Same thing is going to be 0 and 0. And so, you end up having a 0 vector. And what does it mean? This means that this is 0 times 1,1,0,0. So, what we have just shown from this calculation that this vector 1,1,0,0, is an eigenvector for zero eigenvalue. And you can find out that this is also true for another vector which is totally different, 0,0,1,1 also satisfy this property. If you multiply L to this vector you end up having a 0 vector. And so, that correspond to 0 multiply 0,0,1,1. So, this is another eigenvector associated with eigenvalue 0. So, interestingly, we have found two distinct eigenvectors for the same eigenvalue 0 and we have claimed that this Number 2 is the same as the number of connected components. As you can see, you have two connected components. You have two groups of nodes that are connected to each other, there is no connection in-between them. Interesting is that a coincident? Actually not. This is one of the very deep theoretical results you can prove mathematically called the spectral graph theory. And this result from spectral graph theory is going to be the theoretical foundation for us to develop spectral clustering.

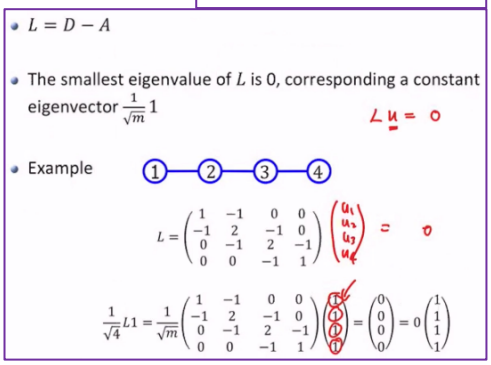
Property II of Graph Laplacian: And so, the second property of Graph Laplacian as see if you look closer at the eigenvectors associated with eigenvalue 0, you will find it contains cluster assignment information to the nodes. So, what do I mean to that? Again, look at the eigenvectors associated with eigenvalue 0. These are the two eigenvectors we find out. So, now let's put these two eigenvectors together, our U matrix. So, I want to put them together as this is my first eigenvector. This is my second eigenvector. So, I got U1, U2. I put them together to form a four by two matrix. And if I write it out, this is 1,1,0,0 0,0,1,1. So, if you look at this, and now I want to look at in this direction instead of, I want to look at the rows. You find out, well the first two rows are identical, and the second two rows are also identical. So, what does it mean? Essentially, if you say this each of the rows in this matrix formed by eigenvectors. If you look at the rows and you say this is the feature vector, I assign to Node 1 and Node 2, Node 3, Node 4. Essentially the node that belong to the same community, will have the same row associated with it. And you say it again. So, you can see that if I do it this way, that each row as a feature vector to the node, then the node that belong to the same community will have the same feature vector. You can see Nodes 1 and 2 have the same row vector associated with them, and Nodes 3 and 4 have the same row vector. Isn't that amazing?



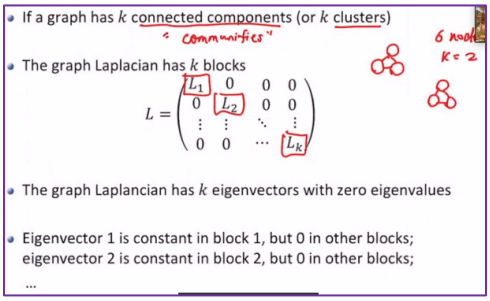
What if the graph has only 1 component: And we can see more examples like this. What if you only have one component in your graph? Let's see what's going to happen. For this case, this is a chain graph. Your nodes form a chain, and the corresponding adjacency matrix is given by this expression diagnosed are 0. The off-diagonals are two bends off one's. There are only one distance away from the diagonal and the degree matrix is 1,2,2,1 because the degree for 1 is 1, degree for Node 2 is 2, degree for Node 3 is 2, degree for Node 4 is 1. So, you put them together, Laplacian is D – A, and this is the corresponding L.



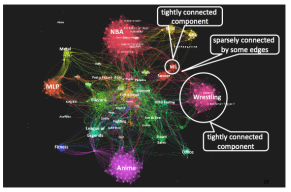
Special eigenvector with all 1's: And so, let's check out what is going to be the eigenvector associated with eigenvalue 0. So, what does it mean? I want to find a vector u satisfy this. L times u equal to 0. Because that's the eigenvector associated with 0. What is going to be the corresponding u? Let's try to solve this. And this is your L. Essentially, I want to find the u_1, u_2, u_3, u_4. These are the four entries in my eigenvector. And I want this to be 0. This gives me a linear system equation I can solve. And in fact, you will find that the only solution you can find from this, to satisfy this linear system equation is going to be this vector of all ones. It's going to be this outer one. And so, interestingly again, you can see the multiples of eigenvectors associated with eigenvalue 0 is 1. Because you only have one eigenvector for 0. So, again, it indicates it is consistent with our property one to just described. And a second, if you look at the eigenvector itself, and this time each of the row is just one number. If you again treat each of number as a feature design to my node, the odd nodes have the same feature. So, indicating that all the nodes belong to the same community. So, you only have one community in this example.



What if the graph has k components: So, we have developed some very useful property of the graph Laplacian. That seems indicating we can partition the graph to find connected components or sometimes called the communities based on the graph Laplacian. And these are the k clusters in your graph. So, if it generalize our observation from this simple example, you will find out if my graph Laplacian has k blocks, if graph Laplacian, and you look at the locations of the non-zero entries, you look at the patterns of your graph Laplacian. If you find that you have this structure. That will be an indicator that you have k communities or k clusters, k connected components in your graph. So, how do I know that? You should find the multiplicity of the 0 eigenvalue that will tell you how many communities likely to be near large graphs or networks. And I can also look at the eigenvectors associated with 0 and look at the rows formed by this set of eigenvectors like we did before. And so, by looking at that, we can possibly find which nodes belong to the same cluster. That's it. So, this seems to be very ideal. And this idealism is based on the assumption that you have exact k component. So, like this. For example, this is a group of 6 nodes, and you have 2 clusters. But in a real-world, let's think about what do you face.



Real world not perfectly block: In the real world, you have this large network, and you want to define these communities. These communities are tightly connected to each other. But between the communities, there's still edges. But these are just much fewer, sparsely connected edges. But within the same community, they're very tiny connected. This real-world network is not the perfect disconnected network that we're considering here. So, how do we generalize that very simple concept or principle into real-world networks?



High level idea of spectral clustering: So, we almost end up having this spectral clustering. Now let's try to develop this high-level idea and write it down more formally in mathematical terms. So, for spectral clustering, we'll start again by constructing the adjacency matrix of your graph. And then we'll construct the graph Laplacians. So, in the perfect case we have exactly disconnected clusters. You can look at the number of eigenvectors, multiplicities of eigenvectors for 0 eigenvalue. Then you look at eigen-vectors that tells you the cluster assignment. And then we use the same intuition for the imperfect case. And this time the eigenvectors are not going to be identical to each other if they belong to the same cluster, but they're similar to each other. The intuition is they're going to be similar to each other for the nodes that belong to the same community or same cluster. We'll do some post processing to assign the cluster assignment.

- Examine the properties of graph Laplacian for the perfect cases
  - The number of 0 eigenvalues corresponds to the number of connected components
  - Eigenvectors correspond to cluster assignment
- Then use the intuition from perfect cases to design algorithms for the imperfect case.
  - Eigenvectors not longer correspond exactly cluster indicator
  - Perform post processing to obtain cluster assignment

In general (imperfect case): So, this is the description of what's going to happen when you have imperfect case. If you have k tightly connected components or k cluster, with sparsely connected edges. And in this case the graph Laplacian will have approximately instead of exactly k blocks. And then the graph Laplacian will have k eigenvalue with small. Small means that very close to 0. We're close to 0 eigenvalues as instead of being exactly 0 eigenvalues. And then you look at the eigenvectors associated with these very small eigenvalues. Again, you form this matrix. Each column corresponding to the eigenvector, associated with a small eigenvalue. And you examining the rows. And each row is the feature design to a particular node. And then the similarities between these features, this way, are going to be roughly indicating which cluster this node belong to. If the two nodes belong to the same cluster, they are the corresponding features are going to be similar to each other.

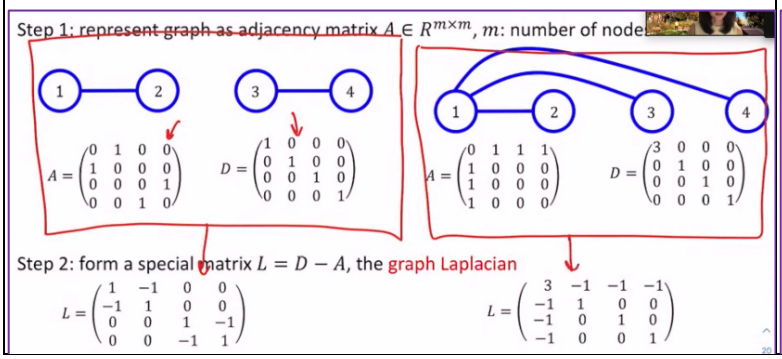
- If a graph has k tightly connected components (or k clusters) with sparsely connected edges
- The graph Laplacian has approximately k blocks
- The graph Laplancian has k eigenvectors with small eigenvalues (very close to 0)
- Eigenvector 1 is approximately constant in block 1, but 0 in other blocks; eigenvector 2 ...

Ideas of spectral clustering: we're getting closer to special clustering. There's indicating that we have the following four steps to perform partition of the graph, to find the clusters in the graph. The first is we will represent graph using adjacency matrix. And then we will construct the spectral matrix. The graph Laplacian. Then we will compute k eigenvectors. Correspond to smallest eigenvalue. This is important, smallest. You decide how many components or connected components communities roughly are, you're interested in finding out your network. Specify the k, k typically is much smaller than m, the number of nodes. So, you find this smallest eigenvalue. Then you find the corresponding eigenvectors put up together to form this matrix Z. What is the dimension of Z? Dimension of Z is mk. And then from there, you look at the rebels of Z and perform some clustering. Which we do by clustering to use k-means.

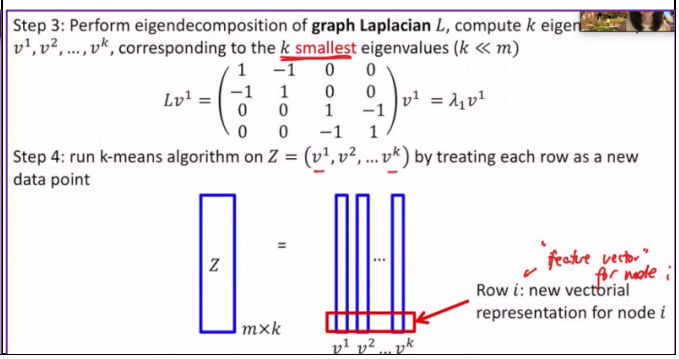
- Step 1: represent graph as adjacency matrix $A \in R^{m \times m}$
- Step 2: form a special matrix $L = D - A$, the graph Laplacian
- Step 3: compute k eigenvectors, $v^1, v^2, \ldots, v^k$, of L corresponding to the k smallest eigenvalues ($k \ll m$)
- Step 4: run kmeans algorithm on $Z = (v^1, v^2, \ldots v^k)$ ($\in R^{m \times k}$) by treating each row as a new data point

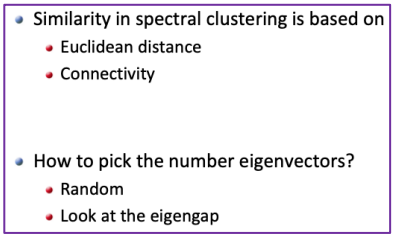
Spectral clustering algorithm: So, now let's see one example, how it renders. We're going to see two instances. I'm placing them next to each other to illustrate. **Step 1** is representing the graph as an adjacency matrix. So, on the left-hand side, I have four node network and I write the corresponding A in L. On the right-hand side, I have only one community. Everybody is connected to one. And so, I have my A and the corresponding D. **Step 2** form the graph Laplacian for each of them. And then **Step 3**, I will find the eigen decomposition of my graph Laplacian and find out k smallest eigenvalue. And I will put the small eigenvalues together to form my matrix Z. And then I will look at the rows of this matrix and treat each row as one feature vector for a node. The i's row is the i's feature where I treat this as a feature vector. Node i. So, based on this I can run K-means, because each node is associated with the feature vector. That's it.

Step 1 & 2



Step 3 &. 4

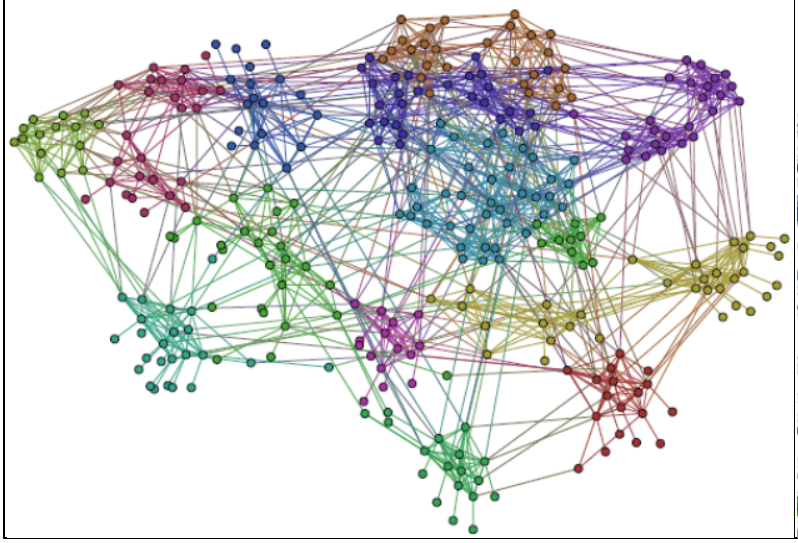




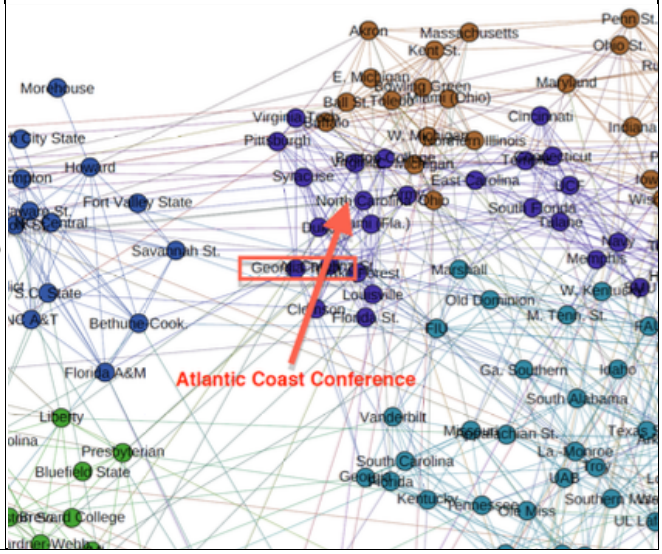
Questions: So, we end up having a description of our algorithm, and here's some questions for you to think about. The first is the similarity in spectral clustering is based on Euclidean distance or based on connectivity. And the question is, based on connectivity. Why? Because we have this A matrix, adjacency matrix for the network. And this captures the connectivity between the nodes in your data. So, here so far, we have assumed that the data given to you is a graph. For example, social networks, users are connected to each other. How are they connected to each other? Is now basically in distance. And remember, this is the basis of vanilla version K-means that we talked about in the last lecture. And the next question is, how do we pick out the number of eigenvectors? And why it is important? So first of all, it is important. Why? Because the number of the smallest eigenvector you pick in here correspond to how many communities you believe there are in your network. So, of course, we shouldn't do that by random guess, and we should have a way to identify how many small eigenvalues you should have your data.

Run demo test_football.m: So, here we're going to run a demo to show how this works on a real dataset. So, to start with, what is this data given to you? The data given to you is this network. This network represent the matches between different teams in college football. This is, I guess, in one particular year, and on the left-hand side are the schedules, on the right-hand side are which league these teams belong to, for example, the Big East, the Big West, and so, on and so, forth. So, you have this large network. So, we want to be able to find, given this network, the communities of these different college sport teams belong to. And essentially, you can think about this as putting color on each of the teams. Each node represent team in this example. And as you can see, this is one result, running spectral clustering. Each node is assigned to a cluster and the node with the same color belong to the same cluster. And if you look closer, the result makes a lot of sense based on spectral clustering. For example, if you look at all the purple nodes, and I can zoom in a little bit, you can see this contains Georgia and so, on. And this turns out to be the teams belong to Atlantic Coast Conference. So, how do we interpret the results? Essentially, by looking at the connectivity of your nodes, in this case, connectivity correspond to the matches between a team and the teams in the same conference are going to have more games with each other. So, they have more edges connected to each other. And you discovered this by looking at this matching network instead of knowing beforehand that they belong to the same conference. So, that's quite interesting in this study.

Visualization using Gephi



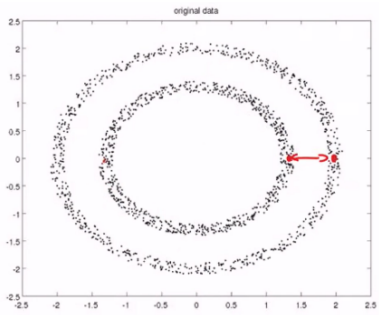
Look closer

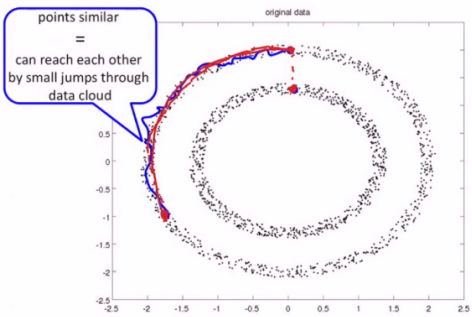


Demo: Show you how this is being implemented in my lab. So, in this example we have 321 teams, and I want to partition this into 10 clusters. And first I'm going to read in the data. This is going to be contained in this adjacency matrix, then given to you as adjacency matrix. And I want to first make this an undirected graph. So, I perform this operation, $(A + A^T) / 2$. That make my A symmetrical matrix. And I check the pattern of my A, so you can see my A is an unweighted graph, and you have a non-zero entry and location when the two teams have a match, and zeros correspond to no match between the two teams. And the next step is to find the graph Laplacian and perform K-means to find assignment of the teams. So, I would like to comment that here we uses a variant of the spectral clustering, which I will discuss in the end of this lecture but is equivalent to the spectral clustering we discussed so far. And so, I formed the degree matrix and then I find this out, defined as DAD, and then perform the eigen decomposition. This time I extract the largest eigenvalue and largest eigenvectors. This is the variant that I mentioned. And then this is the result. And then I will run K-means. Find the assignment,

and then this part will show the clusters. So, this is the result. You can see I have found 10 clusters and each cluster contains a bunch of teams. And it could be interesting if you look closer into the clusters, for example, let's see, Georgia Tech. So, this is where the cluster Georgia Tech line, so you can see it's in the same cluster with Georgia State, and Wake Forest, and North Carolina and so, on, indicating we're in the same conference in Florida. So, it makes lot of sense and belong to the same cluster.

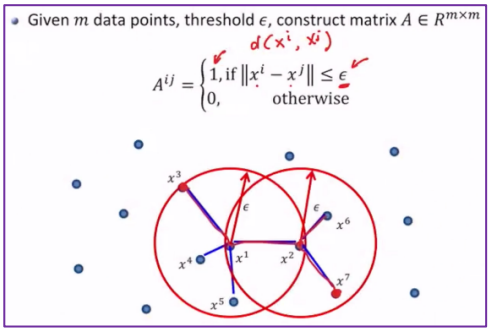
How about this dataset? So, that's the demo, presented how to perform partition of the graph., if I gave you a graph which is represented by adjacency matrix. And now, there's a remaining question we haven't answered, which is towards the question we posed at the beginning of this lecture. How do we perform clustering for this two-ring dataset? We have the two clusters in here, two rings, and remember we apply K-means, it didn't perform well, but there's a magic algorithm, which is actually the spectral clustering that is able to find the two clusters correctly. How is that being performed? So, if we look at this problem, the data are not in the form of the graph. So, the data given to you are these points. Each point on this plane is a two-dimensional vector. They're not a graph. So, to use the idea of spectral clustering, the first step is we're going to try to construct a graph based on these data points. And this graph that we're going to construct will capture similarities between the feature points. And why? Because the similarity between data points is going to tell us how the data are connected to each other. So, we're interested in how they're connected to each other, instead of the actual clinging distance. For example, let me make my point even more clear. For example, these two data points are very close to each other, if you look at their clinging distance, but they're not connected to each other, because they belong to different clusters. So, we want to be able to capture this connectivity through this graph we're going to construct based on the data points.



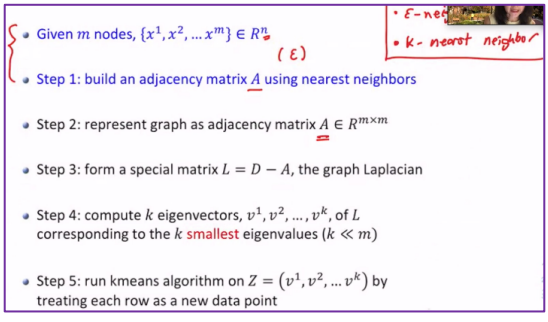


What's a reasonable similarity measure? So, this further illustrate my points, and these two points are not connected. We should have their similarity being very large, because they don't belong to the same cluster. But these two they have pretty long Euclidean distance from each other. But they belong to the same cluster, so they are connected to some paths in the same datasets. And so, we want to be able to capture this geometry of the data by constructing some graph.

Nearest neighbor graph: The idea is captured through the nearest neighbor graph. This nearest neighbor graph can tell us about the local information in how data points are connected to each other in your dataset. And this nearest neighbor graph is constructed as follows, you will look at the distance between two data points, x^i and x^j are two data points in your dataset. Here, I illustrate this using the L2 distance between the two data points. In principle, you can also choose any other similar measures between $d(x^i, x^j)$, and two data points are closer to each other if the distance is small. I will calculate this distance, then I will set the corresponding entry A^{ij} in my adjacency matrix to be 1 if their distance is smaller than sum of Epsilon. This is a hyperparameter neighbor can be tuned in your obviousness well. So, intuitively what happens is you will look at each data point, you will draw a circle with radius Epsilon, and you basically are going to set an arch between any points that fall within the Epsilon neighborhood from this current data points. So, each data point is going to have a bunch of neighbors in this nearest neighbor graph and these neighbors are close to this particular data point in terms of symmetric, it can be L2 distance, it could be other distance of your choice. This can be quite general. So, overall, what does nearest neighbor graph defines is how two data points are similar to each other within one hop, one hop means that, it's directly connecting to each other. And what about this data point? To this data point, they don't fall into each other's neighborhood, but you can still get there because you have neighbors, common neighbors to each other so this captures the connectivity of data points. **What's a reasonable similarity measure?** So, getting back to this point here, these two are far from each other, but they're still connected because you can get there through many hops to neighbors. But these two data points are not connected at all because there's no way you can be jumping to the neighbors and eventually get to this point. **Nearest neighbor graph:** So, this similarity graph is going to capture the geometry and connectivity in this particular dataset.

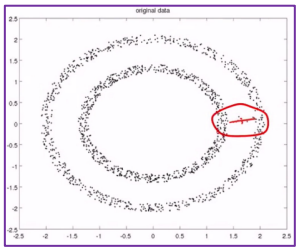


Spectral clustering for vector data: Once we have the nearest neighbor graph, the rest of these steps are going to be completely the same as before. We have this, so I'm highlighting the steps in blue that are additional steps in here. You have n nodes, each of them is n-dimensional and will first build an adjacency matrix and this adjacency matrix can be the nearest neighbor graph. So, you can see you can choose the parameters, for example Epsilon, and then the rest of the step is the same. You will use this nearest neighbor graph and find the spectral clustering based on this adjacency matrix. Also, just to comment that the way we construct the graph here based on Epsilon radius, this is one way of doing it.

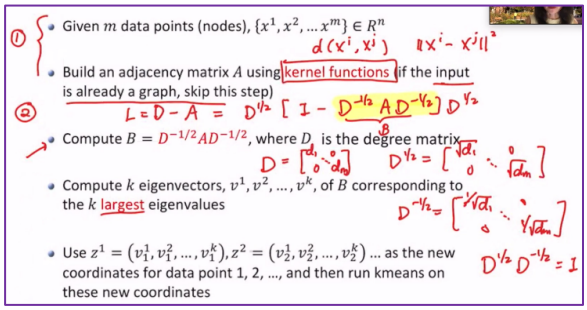


There's Epsilon neighborhood and there are other ways to do this, for example, you can find k nearest neighbor, and this is based on the idea of saying, instead of setting a fixed radius, you going to say, I am going to find at least k, say 10, for example, k = 10. I'm going to find the 10 years neighbor and put an edge between myself and these 10 years neighbors. So, this way, if you do 10 years neighbor, you are guaranteed that these nodes will have at least this many neighbors in the definition of the network of the matrix.

What happens by adding more data points?: So, if you add some additional data points, and sometimes this will make the problem extra challenging, and you have imperfectly separated components. Before, if you don't have this, these are two completely separate components. And I want to go to demonstrate that you would waste this additional noise added here that confuses you are within the spectral clustering based on nearest neighbor graph is still able to recover the grand components. By the way, all the demos in the course are posted online, you're very welcome to run this code yourself, either in MATLAB or Python, play with the code and change the parameters that will help you to understand the properties of algorithm better.



Variants of spectral clustering: Finally, as promised, I will talk about a variant of spectral clustering. So, these first two steps could be if you're given vectorial data, these are the features for the data points, then the first step you have to find the nearest neighbor graph and you can find the nearest neighbor graph by your choice of the kernel function that's representing the d, x^i, x^j. In the previous slide, I have chosen this to be the L2 distance between the two data points, but it could be also other things of your choice, you can choose any kernel function that measures the distance between the data points. And you can also, if the data given to you is already a graph, then you can stick this step. So, either you're given a graph to start with, or you're given these factorial data points and you have to construct a nearest neighbor graph. Then this step is different from before, remember in the original version of the graph, Laplacian is defined as d minus a, so we can first, we written this graph Laplacian in different form which is, I'm going to write it first and explain $D^{1/2}[i - D^{-1/2}AD^{-1/2}]D^{1/2}$. So, what is D^{-} and $D^{+1/2}$? And remember what is D? D is the degree matrix, it's that diagonal matrix that only have non zeros on the diagonal. $AD^{1/2}$ is a square root of a diagonal matrix, where each entry on the diagonal is just square root of your degree, that's it. And similarly, $D^{-1/2}$, is going to be the 1 over square root of the degrees on the diagonals. And you can verify even 1/2 times $D^{-1/2}$ is equal to identity matrix, so they're inverse matrix each other. So, because of that, you can verify what I wrote here actually holds because there's this cancellation happens each other. These are this decomposition, this way of writing it, you will realize there's another way of performing spectral clustering. In the original version, I will try to find the smallest eigenvalue of my graph Laplacian, but if I write it this way, you will see finding the smallest eigenvalue because I have i minus this thing. Finding the smallest eigenvalue will be corresponding to finding the largest eigenvalue of this matrix in yellow. So, I'll try to find the largest eigenvalue of this block matrix B. I define this as B, find the largest eigenvalue, and the rest of the stuff is the same. Why doing this? Because sometimes this gives you numerically more stable solution, you will, especially when degrees of the nodes are very different, sometimes this gives you a more stable solution than the other approach.



So, this wrap us the lecture for today, we talk about spectral clustering, the difference between spectral clustering and the k-means before is that the k-means is based on Euclidean distance and the spectral clustering is based on connectivity of your data. So, you're even given a graph to start with, or you have to construct a similarity graph based on the data given to you. So, the capture is very different geometry in your data, and spectral clustering, as you realize in one of the steps also will utilize k-means as a sub step to implement. So, you will practice using this and hopefully you realize the difference between the two k-means and spectral clustering will have your different purposes when you analyze data. All right, that's it for today. Thank you.