

Clustering and Embedding

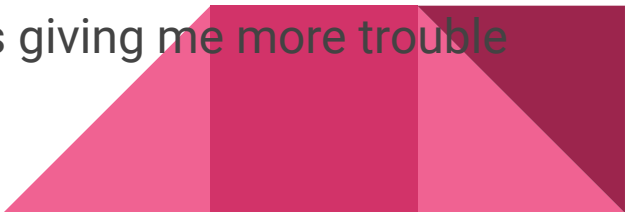
Computational Graph Theory // Day 7

Overview

- Dailies and HW Review
- Image Processing tips for HW6
- Pytorch Review for HW6
- K-Means Clustering
- Agglomerative Clustering
- Laplacian Eigenvalues & Eigenvectors
- Spectral Clustering AKA Laplacian Eigenmaps
- Start HW7



Dailies

- I spent a lot of time today debugging the neural network project, I still can't seem to get it working though I've resolved all the errors.
 - I am getting close to finishing the PageRank assignment, but I have not yet.
 - I was finally able to finish PageRank today and that made me really happy today.
 - Can you shorten the break time so we have some time to go over homework at the end of the class? I think we can do shorter breaks.
 - It was really helpful to have an example of the procedures to build a neural network in class.
 - Lecture made sense but the page rank assignment is giving me more trouble than I thought it would.
- 

Homework Questions?

PyTorch Questions/ Image Loading

Clustering

Problem: labelling data is *expensive*.

Classification (which we discussed yesterday) relies on labeled data.

This is called “supervised” machine learning - we want our classifier to match the labels on a dataset.

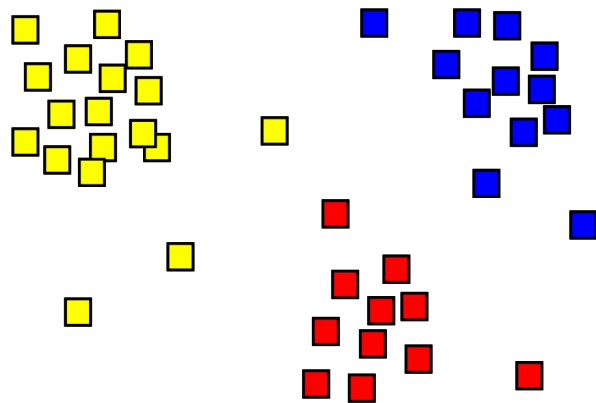
Clustering avoids this problem by ... ignoring it.



Clustering

Clustering is an example of *unsupervised* machine learning:

We want to find structure in the data, but we don't necessarily have any labels to use.



Clustering Algorithms

Generally, all clustering algorithms take in a dataset - a $D \times F$ tensor of data - and a number of clusters, K .

The goal is to produce an assignment that assigns the i th data point to one of the K clusters.

Various clustering approaches do this in different ways!



Distance Calculation

Most clustering algorithms need some way to measure the distance between clusters.

This is going to be different from case to case - our data might be vectors, or sentences, or images!



Distance Calc Example

K-Means Clustering

K-Means Clustering

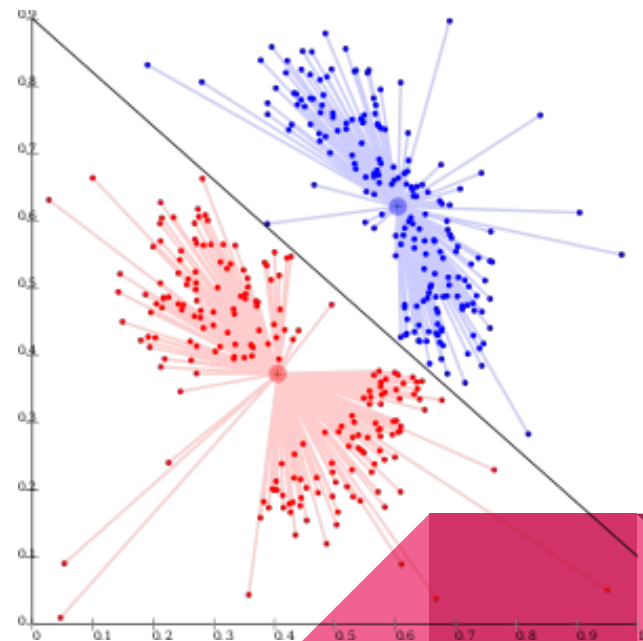
Basic idea: keep track of “cluster centers”. Then, repeatedly do the following:

1. Assign each point to the cluster center that's closest to it.
2. Recalculate the new centers of each cluster.



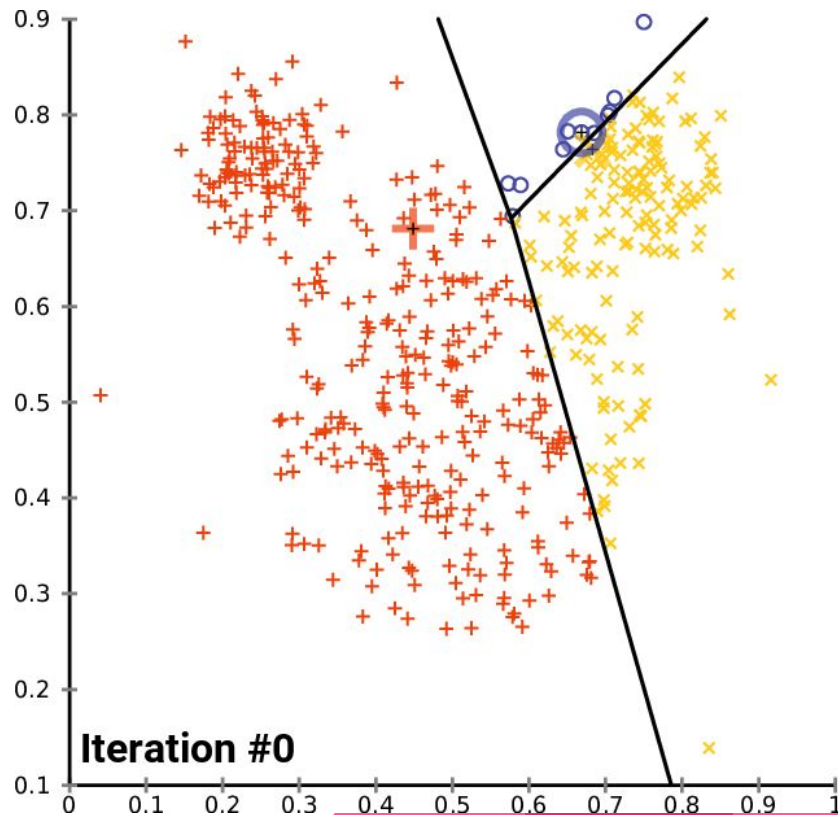
K Means Error

The 'error' that Kmeans is trying to minimize is the total length of lines between each data point and the 'hub' it belongs to.



K-Means Clustering Pseudocode

1. Initialize K random points to be the 'hubs'.
2. Assign each data point to its closest hub.
3. Recalculate the hub positions as the mean of all the points assigned to the hub.
4. If not converged, GOTO 2.



K-Means Convergence

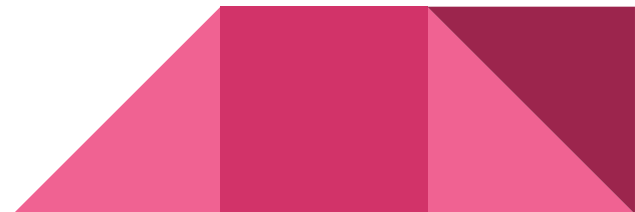
How do we know when the K-means process is “done”?

> when it stops changing.

How do we know this will happen?

Error *has* to decrease in steps 2 and 3.

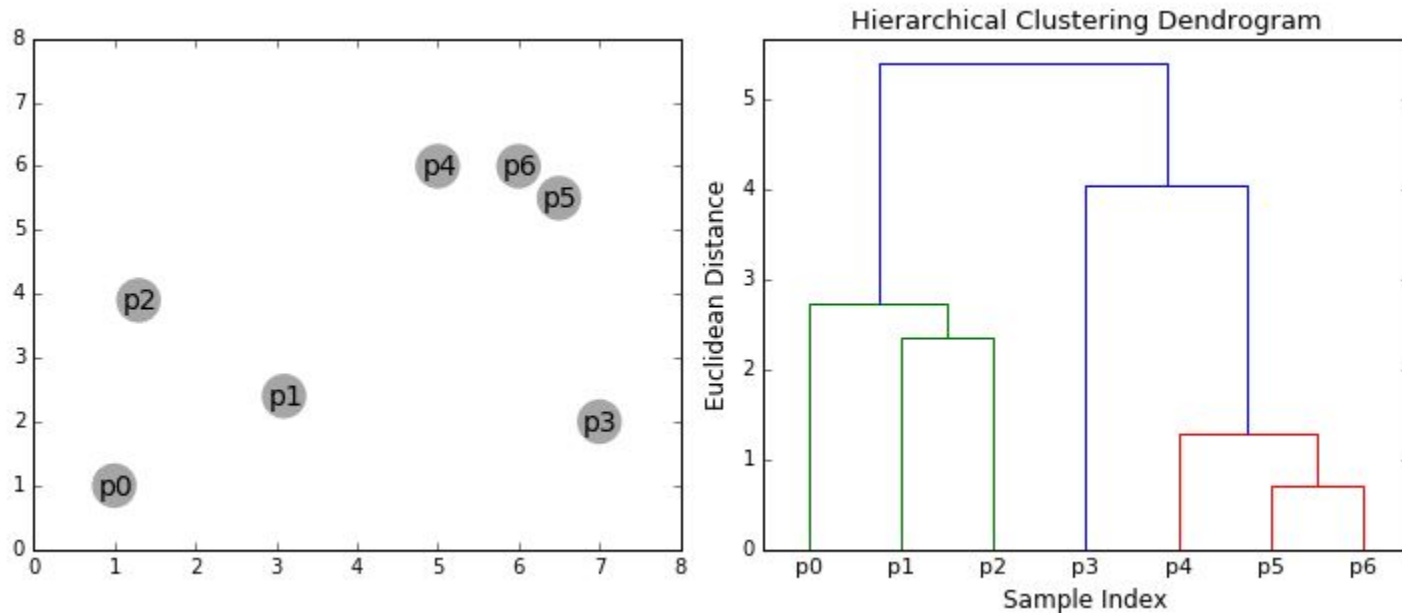
Small groups: come up with an argument for why this is the case, for each of steps 2 and 3.



Agglomerative Clustering

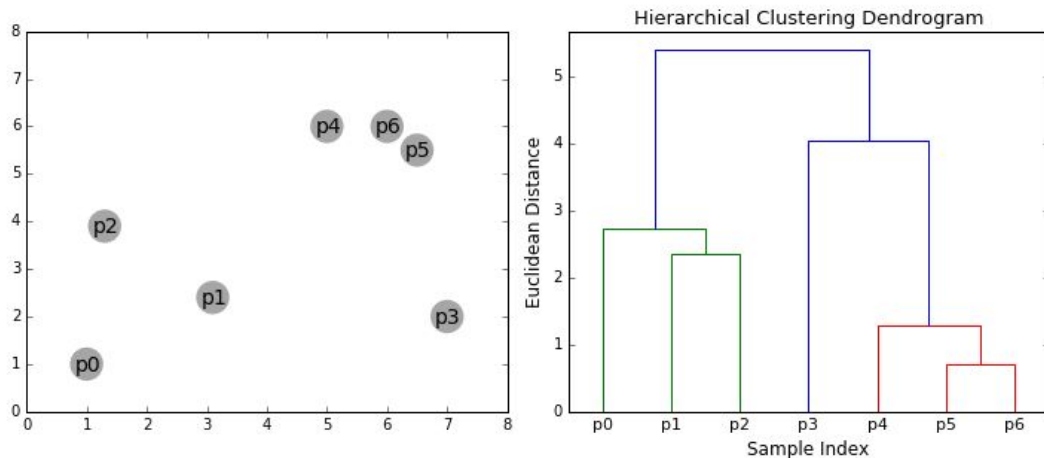
Agglomerative Clustering

Basic idea: slowly build up a hierarchy of clusters.



Agglomerative Clustering Pseudocode

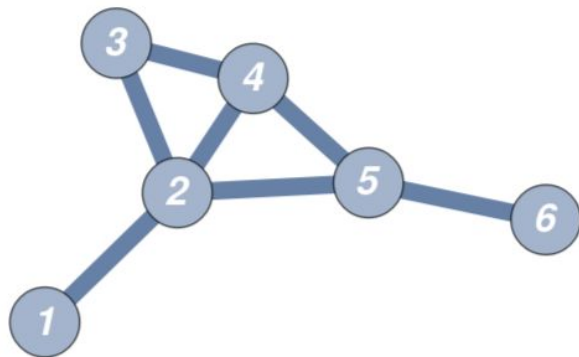
1. Start with each point in its own cluster.
2. While we have more than K clusters:
 - a. Compute the distance between each pair of clusters;
 - b. Merge the two closest clusters.



Spectral Clustering

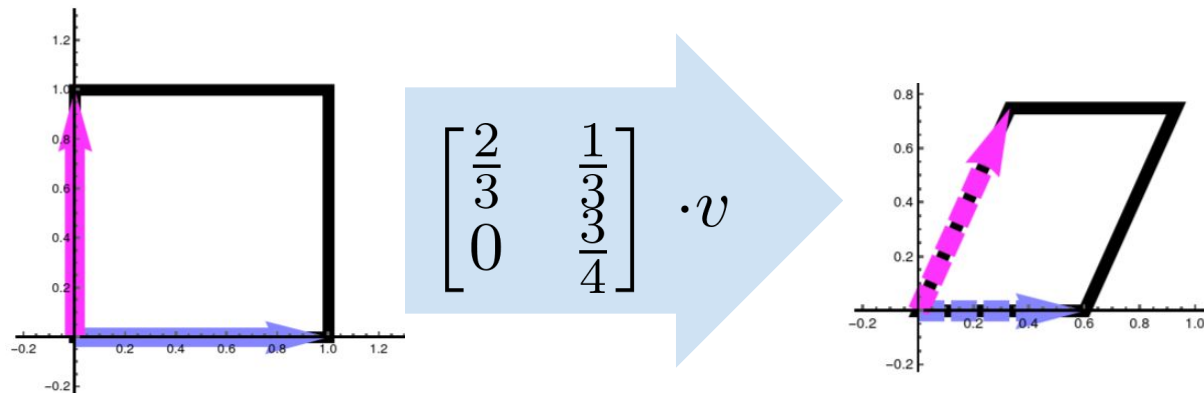
Laplacian Matrix

$$L(G) = [l_{ij}] = \begin{cases} 1 & v_i \sim v_j \\ -d(v) & i = j \\ 0 & \text{otherwise.} \end{cases}$$



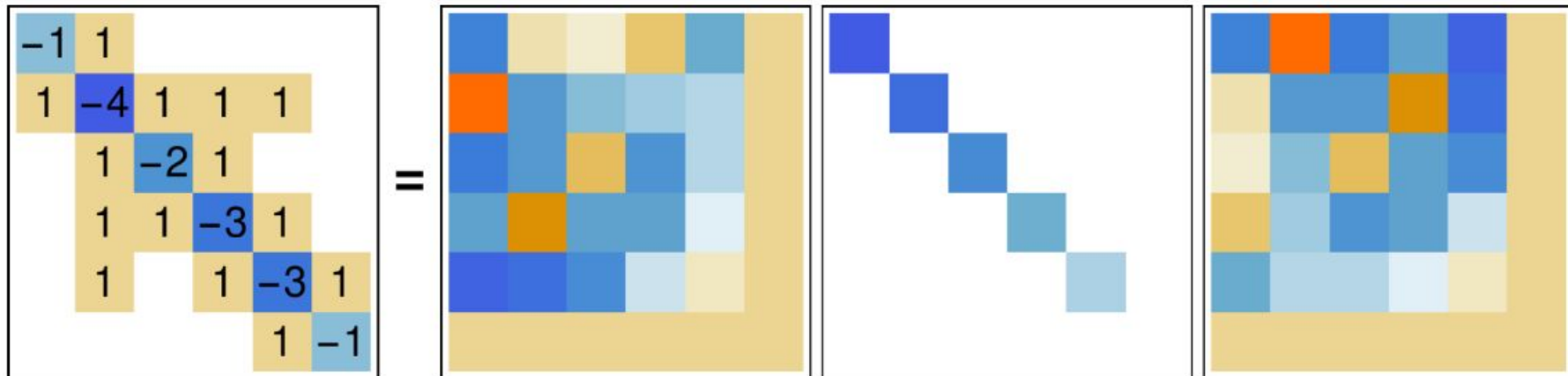
-1	1				
1	-4	1	1	1	
	1	-2	1		
	1	1	-3	1	
	1		1	-3	1
				1	-1

Equation for Eigvals and Eigvects



$$M \cdot v = \lambda v$$

Matrix Decomposition of Laplacian

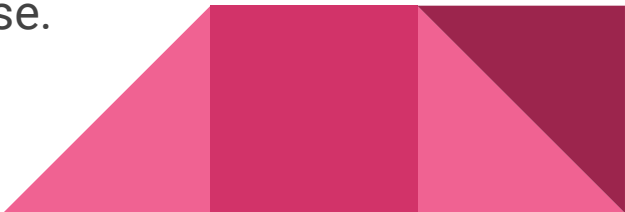


Eigendecomposition Example

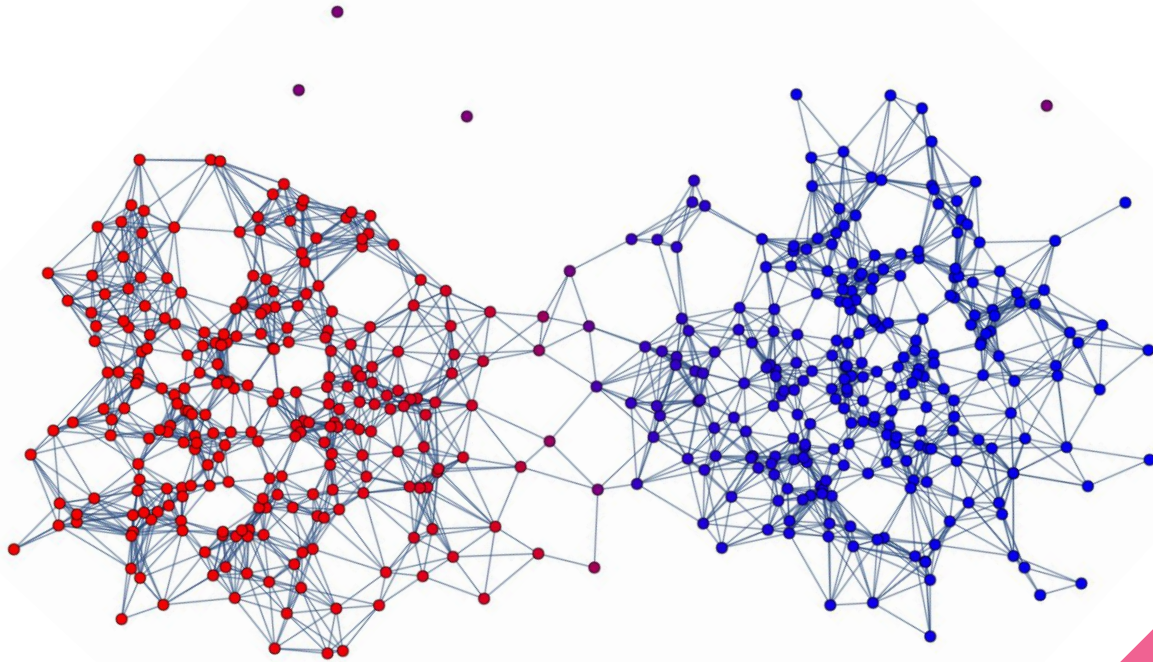
Example on blackboard

$$\begin{bmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_k \\ | & | & & | \end{bmatrix}$$

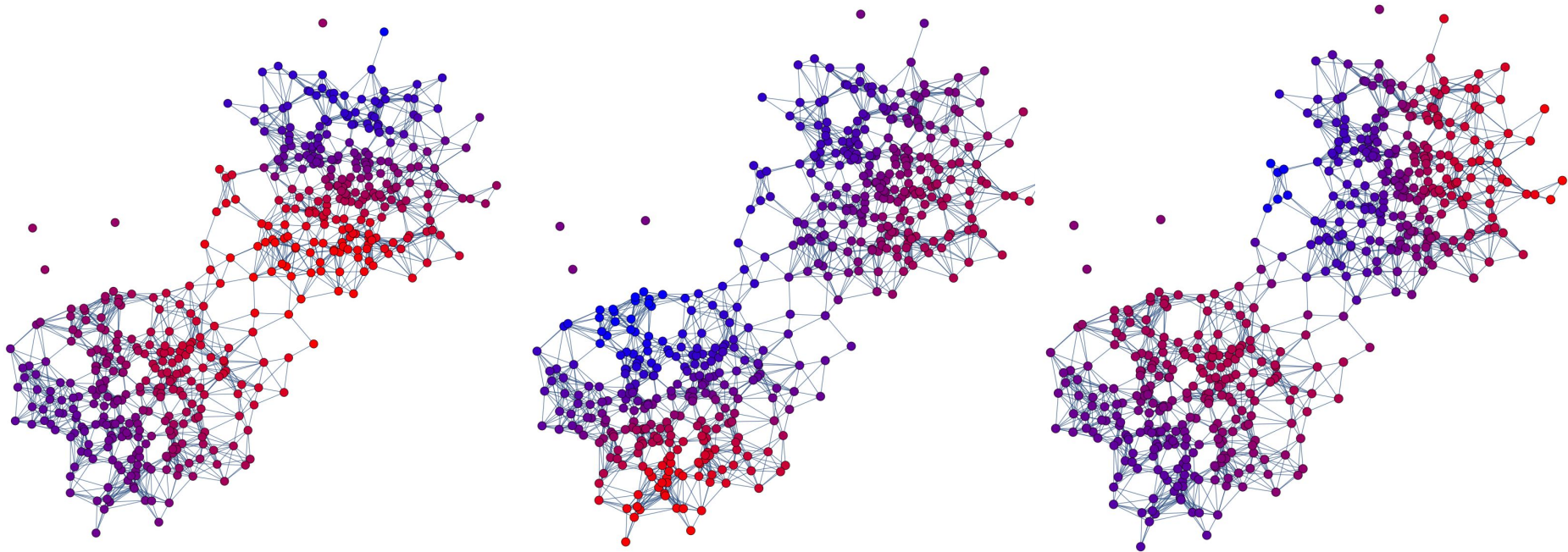
This leads to another interpretation: the eigenvectors are ingredients to rebuild the matrix, and the eigenvalues tell us how much of each to use.



Second eigenvector separates data into 2 clusters.

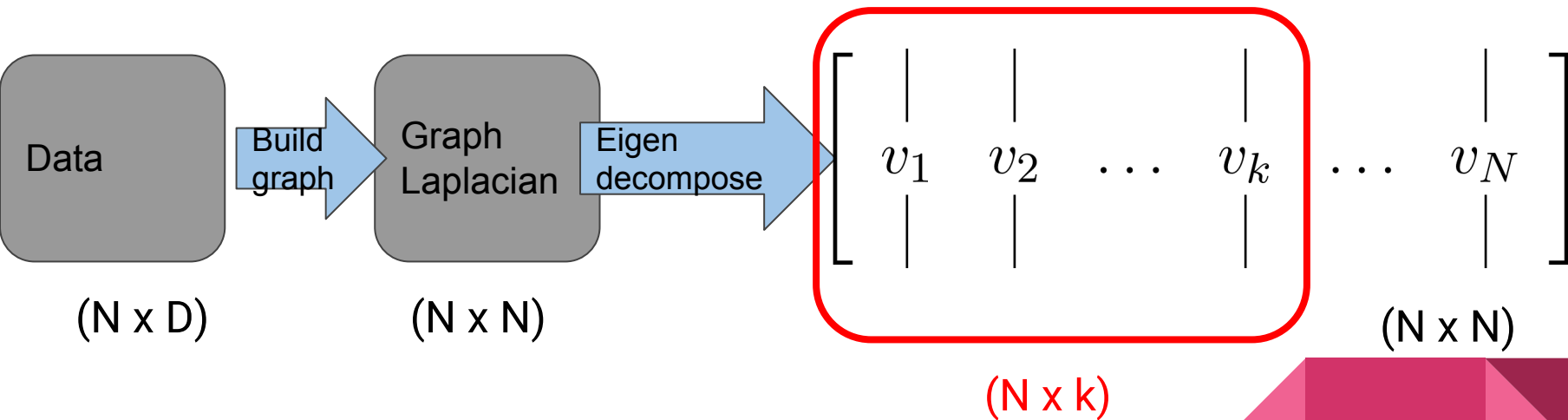


Other eigenvectors extract more structure!

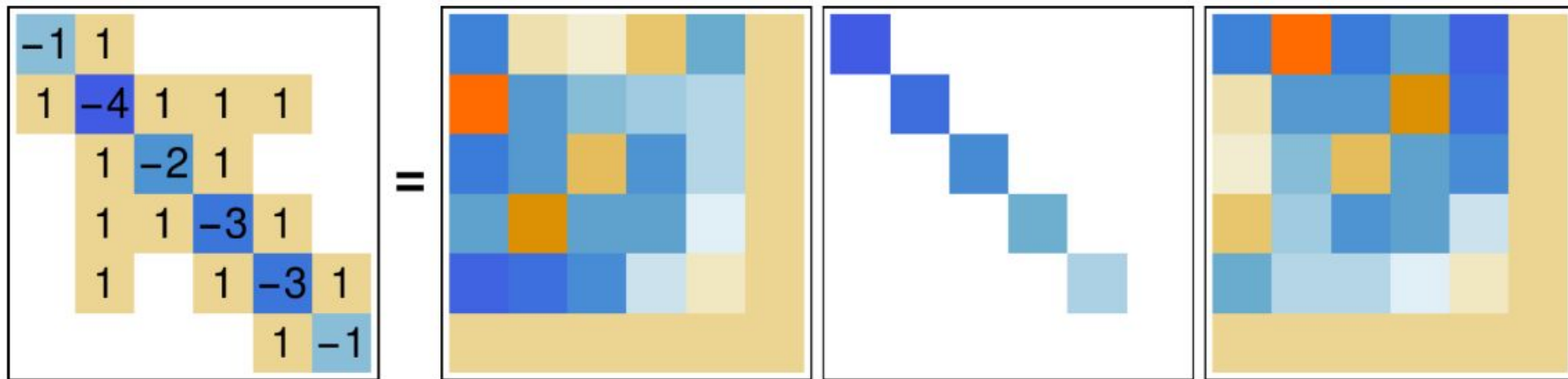


Spectral Coordinates

For spectral embedding, all we need is the following:



Matrix Decomposition of Laplacian



Spectral Clustering

1. Take in initial data, and build a graph from it (N-neighbors, radius-neighbors).
2. Pick K, how many clusters we want to divide our graph into.
3. Get the d first eigenvectors, and build new data vectors from them:

The i th data point is represented by the new vector $\langle v_{1i}, v_{2i}, v_{3i}, \dots, v_{di} \rangle$.

4. Cluster these new data points using the clustering algorithm of your choice.
 - a. (like K-means!)



Motivation for Laplacian Eigenmaps

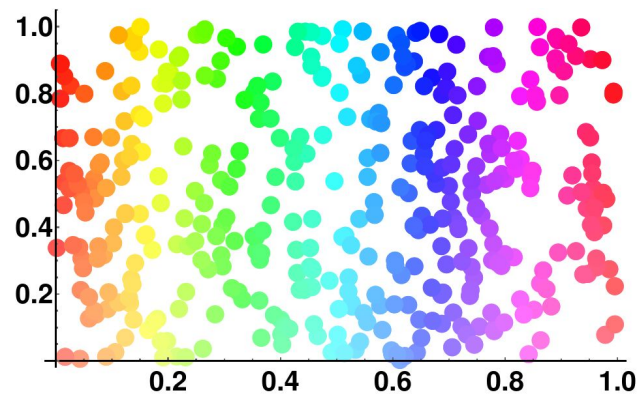
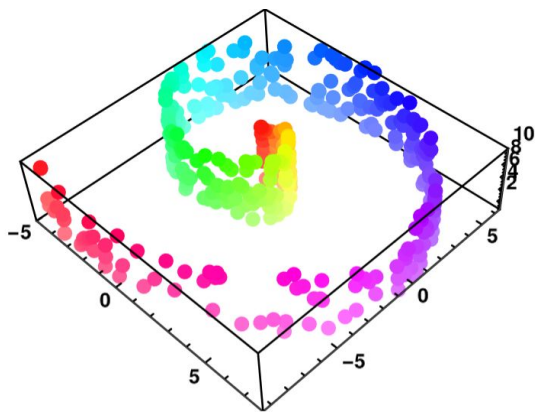
The basic motivation of this method is that the first few eigenvectors (somehow) encode the general *structure* of the dataset.

Most data is actually fairly low dimensional. LEs keep the “local” structure the same but unwrap the global structure to be simpler.

One way of thinking of this: this method transforms the data to use “local” coordinates.



Embedding Complicated Data



Embedding Complicated Data



Edge Weights

Both K-Means and Agglomerative Clustering used some notion of nearness or distance.

We need some way to turn distances into edge weights, and we want weights to be *stronger* for closer nodes - so weight is stronger when distance is small.

One option:

$$w_{ij} = \frac{1}{d_{ij}}$$

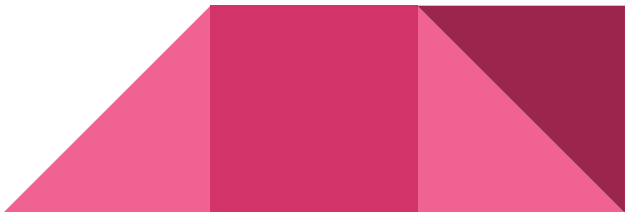
What's wrong with this way of weighting the edges in our graph?



Gaussian Edge Weights

$$w_{ij} = e^{\frac{-d_{ij}^2}{2\sigma^2}}$$

This value is:

- Always between 0 and 1;
 - Adjustable to our specific data by modifying sigma;
 - Decreases very fast as distance increases.
- 

Homework 7