Data Science and Big Data Analytics: Making Data-Driven Decisions

Module 1 Making Sense of Unstructured Data

Lecture Notes

Overview

Nowadays, nearly everything in our lives can be quantified by data. Whether it involves search engine results, social media usage, weather trackers, cars, or sports, data is always being collected to enhance our quality of life. How do we get from all this raw data to improving the level of performance? The answer is found in this opening module! This module will introduce us to the tools and techniques developed to make sense of unstructured data and discover hidden patterns. Specifically, the main topics that are covered are:

1. Supervised & Unsupervised Learning and the main techniques corresponding to each one (Classification and Clustering, respectively)

2. An in-depth look at the K-Means algorithm

3. Dimensionality Reduction & Spectral Techniques (Clustering in networks, PCA, eigenvectors, embeddings, and other special clustering techniques)

This module is taught by Stefanie Jegelka and Tamara Broderick. Both are professors at MIT with the Department of Electrical Engineering & Computer Science and also with the Center for Statistics at the Institute for Data Systems and Society. Their combined work expertise covers machine learning methods for all communicational data, Bayesian and non-parametric Bayesian-learning, and methods for unsupervised learning.

Goals:

1. Understanding of the many different techniques used to discover patterns in a set of data

2. Knowledge of when to use each of the above techniques based on the given information

3. In-depth understanding of the K-Means algorithm

Objectives:

Students should be able to:

1. Know the difference between supervised learning and unsupervised learning

2. Know how to apply clustering and classification techniques

3. Understand how the K-Means algorithm works and how to implement one

4. Know what Mixed Membership Models and Feature Allocations are

5. Understand what Principal Component Analysis is and how to compute it using eigenvectors

6. Understand what Spectral and Modularity Clusterings are, as well as the term "embeddings"

1.1 What is Unsupervised Learning, and Why is it Challenging?

Learning: find hidden pattern in the data

Hidden patterns:

1) understand some scientific result,

2) improve our user experience

3) help us maximize profit in some investment.

Supervise learning:

Learn from data but we have labels for all the data we've seen so far

Unsupervised learning:

Learn from data but don't have any labels

Every machine learning problem has a data set: a collection of data points that help us learn

Example of supervised machine learning: to predict whether a new email is SPAM or NOT SPAM, that is predict the label of next email.

Example of unsupervised machine learning: group emails by topics, that is group emails without label.

Challenge of unsupervised learning: it's sometimes expensive to get labels, but it's cheap if a machine learning algorithm can label data automatically.

Challenge of unsupervised learning: hard to find pattern that is important and hard to evaluate the performance.

Benefit of supervised learning: easy to evaluate the performance.

1.2 Tools and Applications & How They Fit Together

Example of supervised learning:

1) predict air pollution concentration at a new location

data = collection of information about surroundings: location, weather, date&time

label = air pollution concentration

2) predict new rock's lift power

data = rocket booster's specification

label = simulated lift

Example of unsupervised learning:

1) discover social groups among monks

2) image compression

1.3 What is Clustering?

Unsupervised learning = finding hidden patterns in data without labels

Clustering = group data according to similarity

Example of similarity: close in distance

Cluster: group of similar data points

Clustering: unsupervised problem of assigning each data point to exactly one group/cluster.

Classification: predict new labels using clusters with labels.

Clustering finds hidden grouping in data even we can't run classification, more difficult to undertake.

1.4 When to Use Clustering?

In clustering, we want to find a latent grouping such that each data point belongs to exactly one group called a cluster.

1.5 K-means Preliminaries

Most popular algorithm in clustering and unsupervised learning：

1) simple algorithm
2) take advantage of parallel computing: calculate simultaneously across multiple cores
3) fast in programmer time: straight forward to understand and code, doesn’t require large number of parameter tweaks

Set up and assumptions of K-means algorithm:
Assumption: we can express any data point as a list, or vector, of continuous values.
Assumption: dissimilarity = squared Euclidean distance:

$$dis(X, Y) = (X_1 - Y_1)^2 + (X_2 - Y_2)^2$$

For each index d from 1 to D, add the distance in the d direction:

$$dis(X, Y) = \sum_{d=1}^{D} (X_d - Y_d)^2$$

Assumption: we know what K is.

1.6 The K-means Algorithm

Set up of k-means clustering problem:

1) we assumed that each data point is a vector of continuous values,
2) we're using square Euclidean distance as our dissimilarity measure between two data points.

Output of K-means:
A set of cluster centers: μ_1, μ_2, μ_3

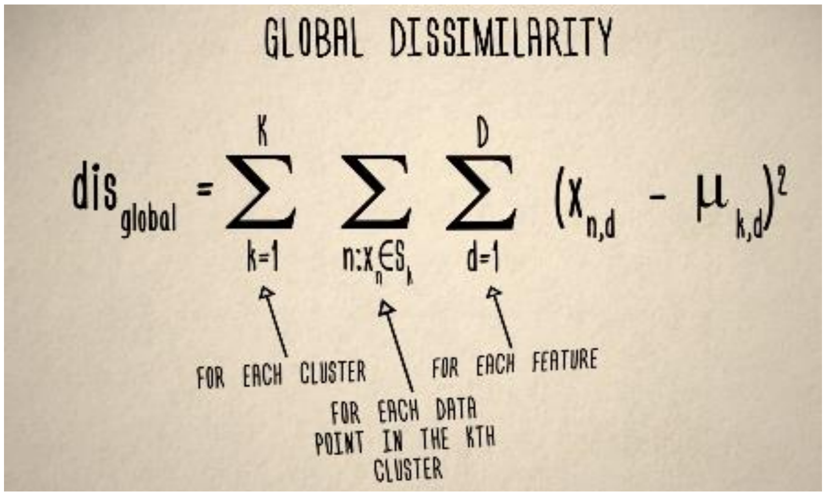
Goal: minimize dissimilarity within each cluster

Global dissimilarity:

The dissimilarity between any data point and a cluster center is the distance from the data

point to the cluster center:

$$\sum_{d=1}^{D}(X_{n,d} - \mu_{k,d})^2$$


GLOBAL DISSIMILARITY
$dis_{global} = \sum_{k=1}^{K} \sum_{n:x_n \in S_k} \sum_{d=1}^{D} (x_{n,d} - \mu_{k,d})^2$
FOR EACH CLUSTER
FOR EACH DATA POINT IN THE KTH CLUSTER
FOR EACH FEATURE

K-means objective function = Global dissimilarity

K-means algorithm (Lloyd's algorithm):

Initialize cluster centers: randomly draw data from existing data points, more sophisticated initialization: K-means++

Repeat 2) and 3) until Global Dissimilarity doesn't change:

Assign each data point to the cluster with the closest center.

Embarrassingly parallel computation: divide up the data points across cores or processors and perform these calculations separately for speed-ups and running time.

Recalculate the cluster center using the result from 2)

Does this algorithm always stop in finite time? Yes

K - mean Assumption:

Data point can be expressed as a list or vector of continuous values

Use squared Euclidean distance as dissimilarity measurement

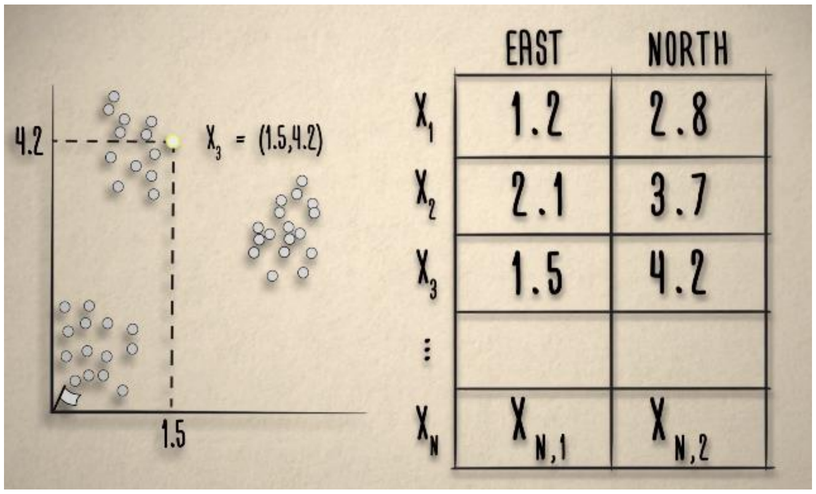
Have K clusters

Each data must be assigned to exactly one cluster

What is our data?

Our data point is defined by two values: distance in the north direction and distance in the east direction.

N: number of data points (X_1, X_2, X_3, …, X_N)

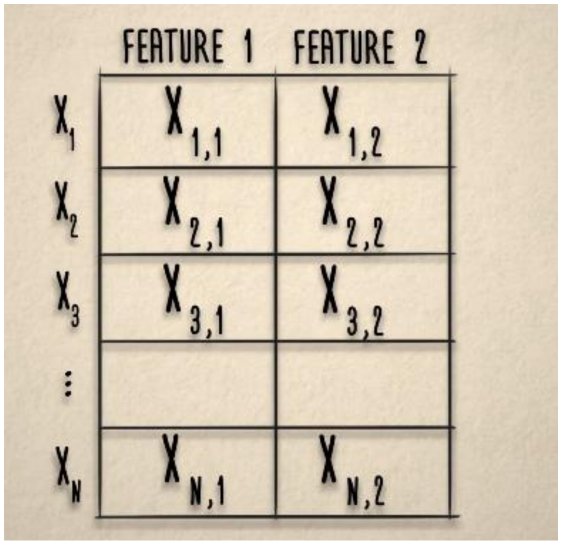


	EAST	NORTH
X_1	1.2	2.8
X_2	2.1	3.7
X_3	1.5	4.2
⋮		
X_N	$X_{N,1}$	$X_{N,2}$

$X_1 = (X_{1,1}, X_{1,2})$

$\dots X_N = \left(X_{N,1}, X_{N,2}\right)$

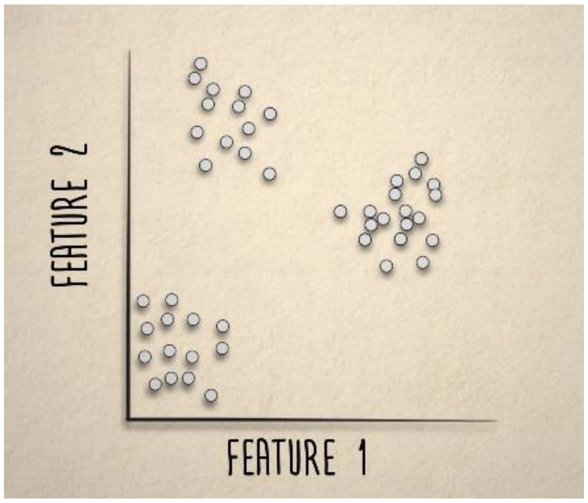
Each element in X_N is called a feature. In this example, we have two features: distance in the north direction and distance in the east direction.



	FEATURE 1	FEATURE 2
X_1	$X_{1,1}$	$X_{1,2}$
X_2	$X_{2,1}$	$X_{2,2}$
X_3	$X_{3,1}$	$X_{3,2}$
⋮		
X_N	$X_{N,1}$	$X_{N,2}$

Generally, we might have any finite number of features.

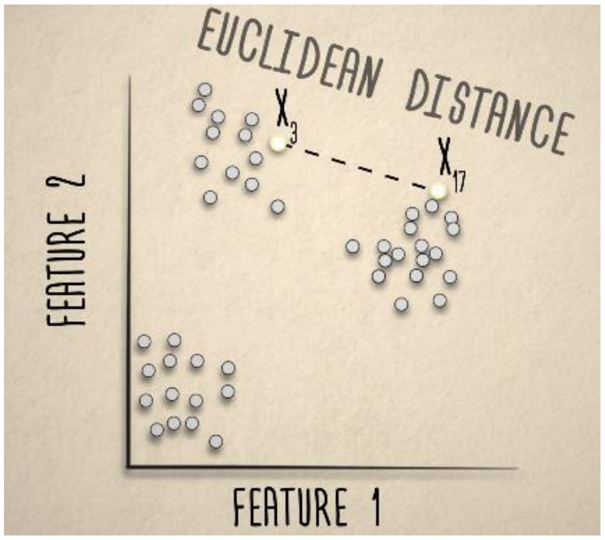
The number of features is denoted as: d which means dimension.



What is our measurement of similarity?

Similarity = dissimilarity

Euclidean distance: the distance between two points



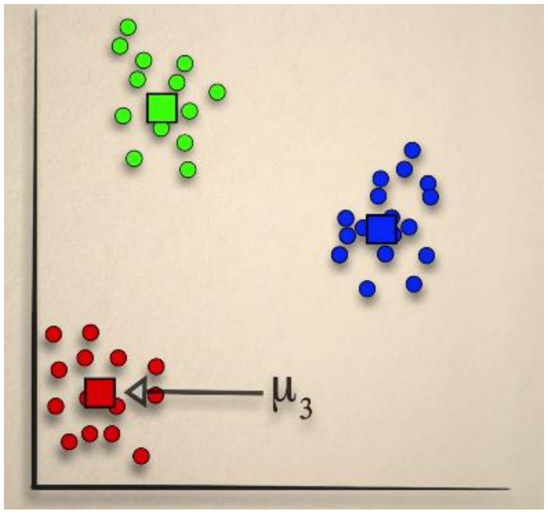
The square of Euclidean distance:

$$dis(X,Y) = (X_1 - Y_1)^2 + (X_2 - Y_2)^2$$

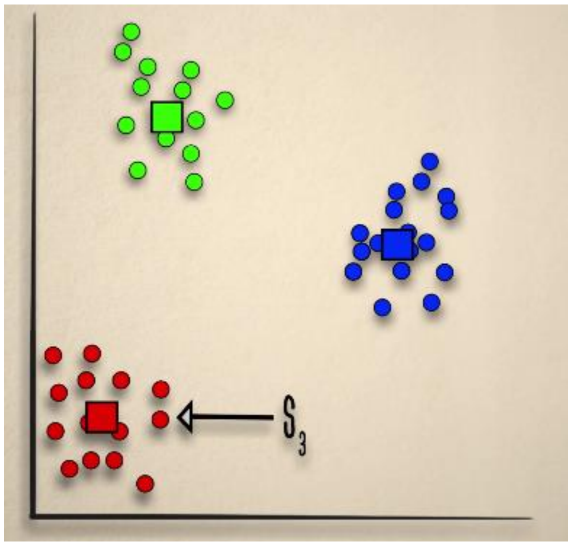
Dissimilarity function:

$$\mathrm{dis}(X,Y) = \sum_{\substack{d=1 \\ \uparrow \\ \text{FOR EACH FEATURE}}}^{D} (X_d - Y_d)^2$$

The cluster center μ_k:

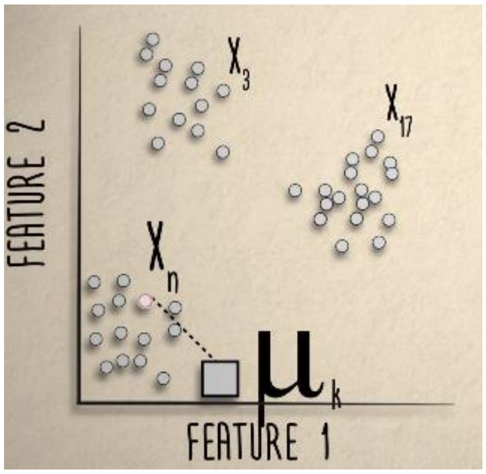


The cluster S_k:



Goal: Minimize dissimilarity with in each cluster

Output: set of cluster center $\mu_1 \dots \mu_k$ and a set of assignments of data points to cluster $S_1 \dots S_k$



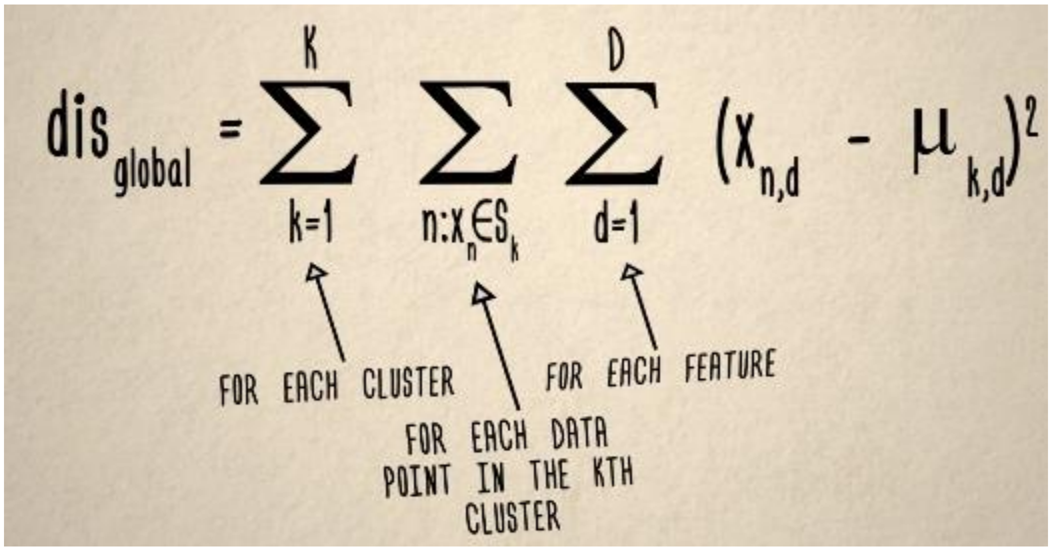
The dissimilarity function:

$$\sum_{d=1}^{D} (x_{n,d} - \mu_{k,d})^2$$

The dissimilarity within a cluster is the sum over the distances between data points in that cluster and the cluster center.

$$\sum_{n:x_n \in S_k} \sum_{d=1}^{D} (x_{n,d} - \mu_{k,d})^2$$

The global dissimilarity is sum of each cluster's dissimilarity within a cluster.



$$dis_{global} = \sum_{k=1}^{K} \sum_{n:x_n \in S_k} \sum_{d=1}^{D} (x_{n,d} - \mu_{k,d})^2$$

The K - mean clustering problem is to minimize this global dissimilarity also known as K - means objective function.

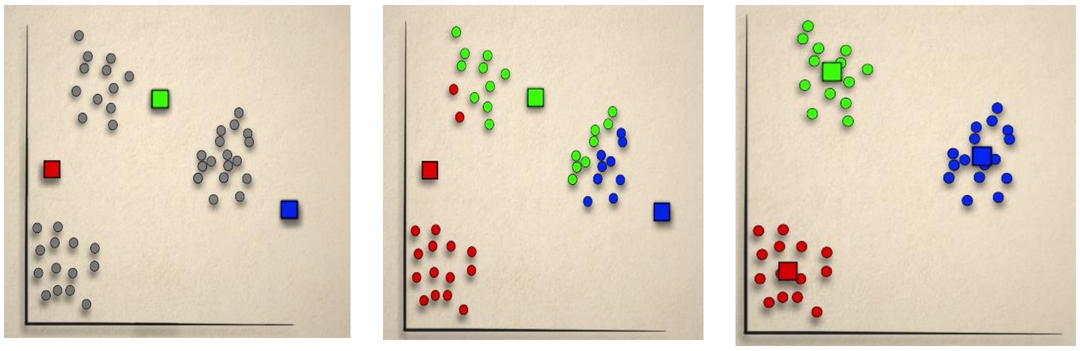
K-mean algorithm or Lloyd's algorithm (assume we know the value of K - the data have K clusters):

Initialize cluster centers

Assign each data point to the closest cluster center

Update cluster center to be the mean of all data points in the cluster

Repeat 2) and 3) until converge



Initialize the cluster center:

Randomly picked from the data

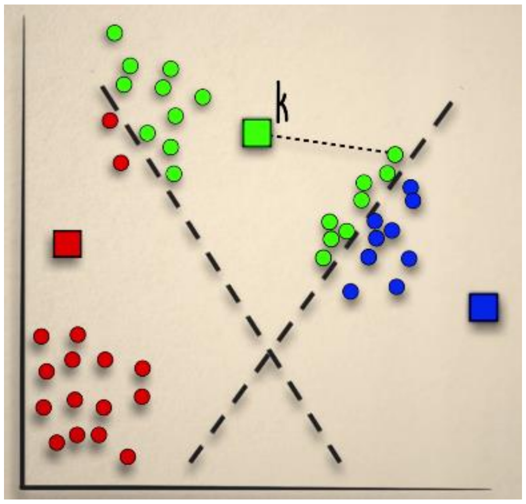
K-means++

Check converge:

The assignments of data points don't change

K-mean objective function (global dissimilarity) doesn't change

Embarrassingly parallel:



Center of cluster: Mean of data points in each cluster

K-mean algorithm will always stop in finite time.

1.7 How to Evaluate Clustering?

For a classification problem, the evaluation is by Training/Test data split.

Measurements like the Rand Index and Adjusted Rand Index help us measure whether the clustering found by our algorithm really captures the information in the ground truth clustering.

They do so by counting the pairs of data points that belong to the same or different clusters according to the algorithm and the same or different clusters according to the ground truth.

Measures like the Rand Index are called external evaluations because they require outside information about a ground truth clustering.

Evaluation is hard!

Visual check: GGOBI tool help us find meaningful visualizations of high dimensional data for evaluating the clustering.

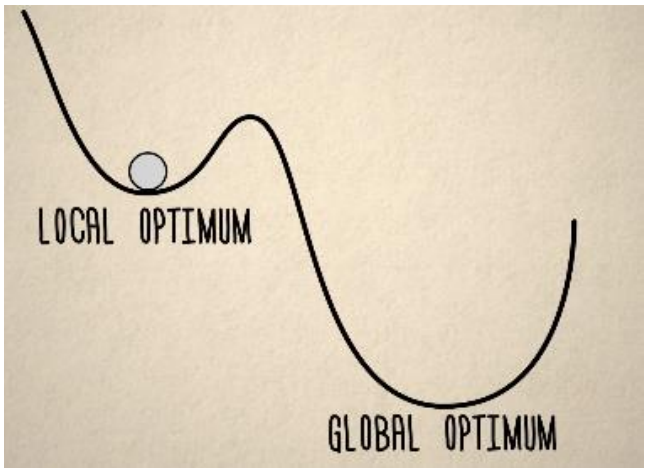
Internal evaluation: the basic idea of these measures is typically to make sure that data within a cluster are relatively close to each other and data in different clusters are relatively far from each other - Silhouette Coefficient

1.8 Beyond K-means: What Really Makes a Cluster?

K-Means algorithm will move to a new clustering only if the new clustering decreases the K-Means objective.

Local optimum VS global optimum:

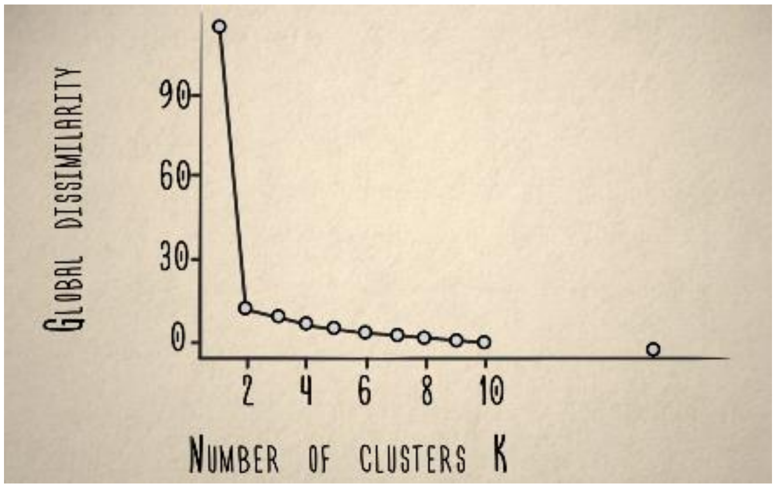
To get closer to the global optimum is to run the K-Means algorithm multiple times for multiple random initializations. We take the configuration of cluster centers and cluster assignments with the lowest objective function value, that is, the least global dissimilarity.



K-means++:
potential to produce total running time.
With theoretical bounds on the running time

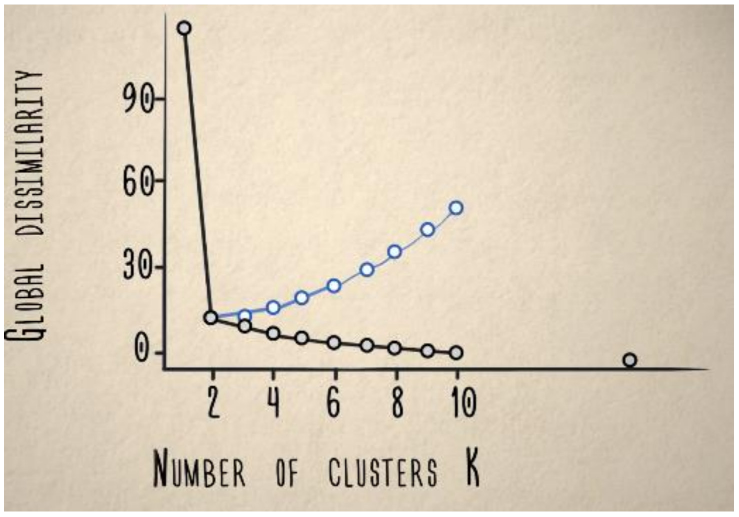
1.9 Beyond K-means: other notations of distance

K-means requires the user to specify the number of cluster K.



K = the elbow point where global dissimilarity suddenly slows down

Gap statistic:
Add penalty to the K-means objective function. Some penalties are given by AIC, BIC…., are called information criteria. After the Penalties are added, the new objective function as function of K Has a non-trivial minimum.



Hierarchical clustering: agglomerative clustering

Hard clustering: each data point belongs to one and only one cluster.

Soft clustering: each data point might have a different degree of membership. For instance, a probability distribution that describes how likely a data point belonging to different cluster.

1. Fuzzy clustering
2. Gaussian mixture model

1.10 Beyond K-means: Grouping Data by Similarity

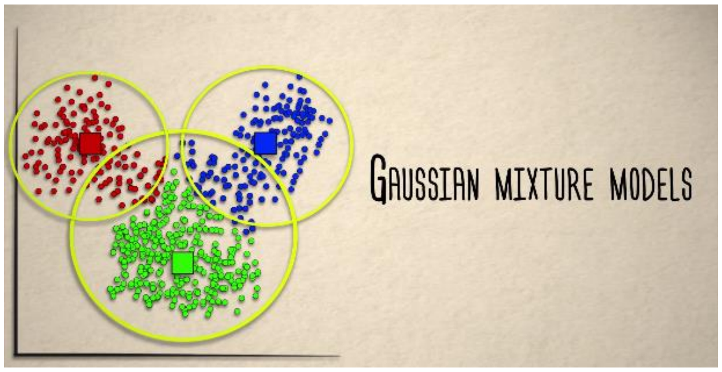
Group by what?

K-means algorithm:

1. group by similarity
2. K spherical equal-sized cluster
3. dissimilarity measured by Euclidean distance

Cluster with different size:

1. Gaussian Mixture Model
2. Expectation-Maximization (EM) algorithm



Outliers:
The data points that are relatively distant from most of the data points in the data set.

K-means clustering is very sensitive to outliers.

How to deal with outliers?

Use alternative measurement of distance which is less sensitive to outliers

L2 distance: sum of squared Euclidean distance

$$dis(x_3, x_{17}) = \sum_{d=1}^{D} (x_{3,d} - x_{17,d})^2$$

L^2 DISTANCE

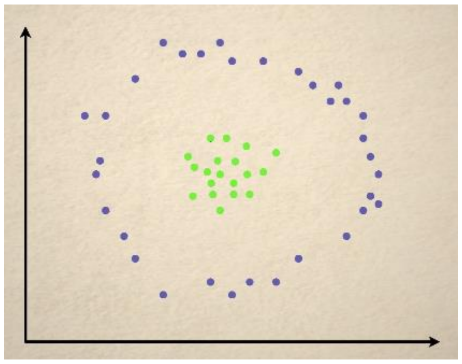
Alternative L1 distance: sum of absolute values of the differences

$$dis(x_3, x_{17}) = \sum_{d=1}^{D} |x_{3,d} - x_{17,d}|$$

L^1 DISTANCE

For K-means using L1 distance is called L-medoids.

What about cluster shape other than spherical?



1. Radial similarity
2. Polar coordinates
3. Kernel methods
4. Agglomerative clustering

K-means requires continuous and numerical features!

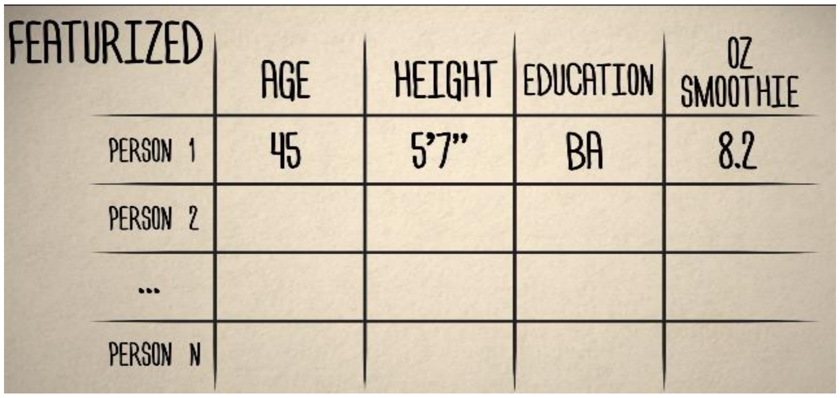
Data need to be transformed for yes-no data or text data.

1.11 Beyond K-means: Data and Pre-processing

Before running K-means, check your data:

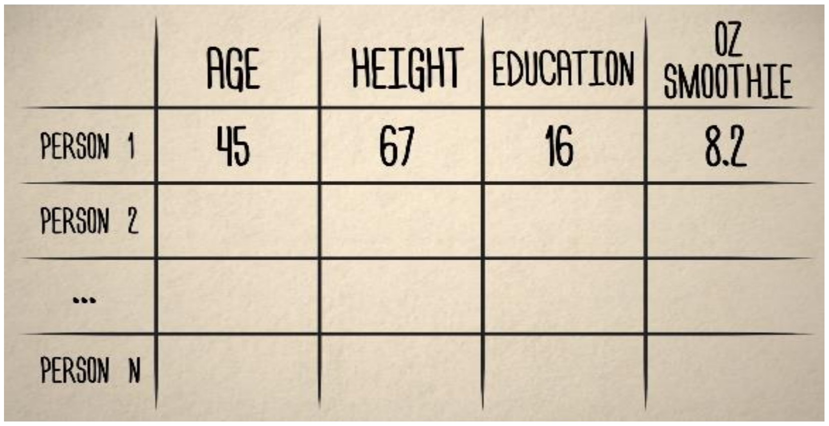
1. Is your data featurized?

   Data need to be vector of features



FEATURIZED	AGE	HEIGHT	EDUCATION	OZ SMOOTHIE
PERSON 1	45	5'7"	BA	8.2
PERSON 2				
...				
PERSON N				

2. Is your data numerical?



	AGE	HEIGHT	EDUCATION	OZ SMOOTHIE
PERSON 1	45	67	16	8.2
PERSON 2				
...				
PERSON N				

3. Is your data on the same scale?

   Normalize your data so that each feature has a standard deviation of 1

4. Are there too many features?

   Reduce the dimension of your feature by Principal Component Analysis (PCA). Run PCA before K-means to keep only useful features.

5. Can you use domain knowledge?
   Use your domain knowledge to choose best features.

1.12 Beyond K-means: Big Data and Nonparametric Bayes

What if K is not fixed?
K-means clustering = fixed K clustering

Streaming data: new data is added all the time

Size of the data grows = number of cluster grows

For a growing size of cluster: Non-parametric Bayesian method

Mad-Bayes:
Turn non-parametric Bayesian methods into K-means like problems, and algorithms for clustering in particular, and unsupervised learning in general.

1.13 Beyond Clustering

We might want our data points, the pictures, to belong to multiple groups simultaneously or no groups or just one group. Any of these options should be allowed. In this case, we call the groups features instead of clusters, and the underlying structure is a feature allocation instead of a clustering.

A similar idea is to say that the data points exhibit mixed membership. They can belong to multiple groups at the same time unlike clustering.

Mixture model: clustering – each data point come from only one of many groups
Admixture model: each data point can belong to multiple groups at the same time

Data points belong to multiple groups simultaneously:
1. Feature allocation
2. Mixed membership
3. Admixture

The most popular of mixed membership structure in data algorithm is:
Latent Dirichlet Allocation or LDA.

Other K-means like algorithms for this feature allocation problem are Mad-Bayes based like in Tumor study: DP-means and BP-means.

1.14 Networks and Complex Data

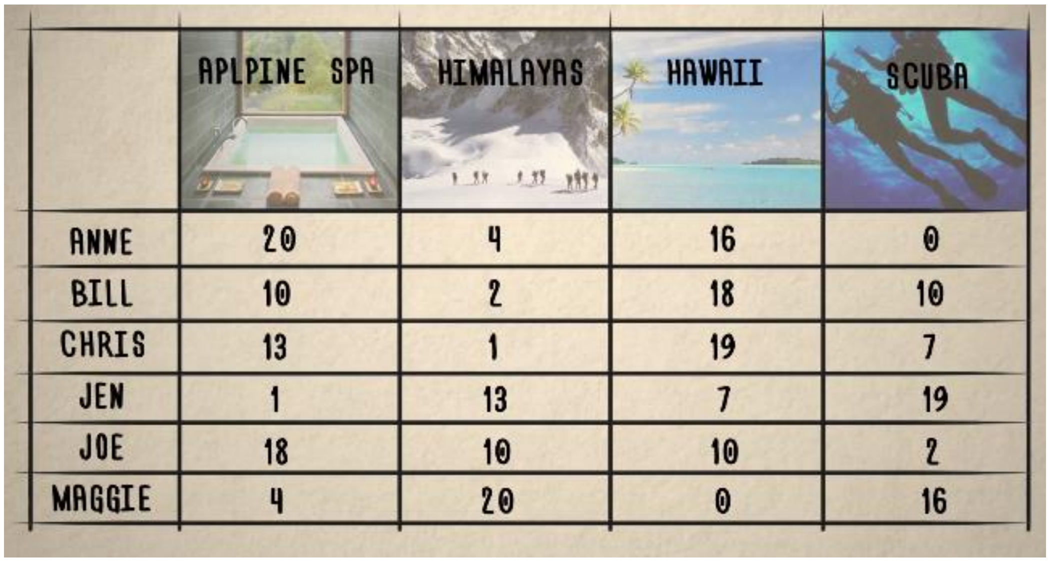
For clustering and many other tasks, our data comes in form of data points. Each data point is a feature vector.

1.15 Finding the Principal Components in Data and Applications

PCA (Principal Component Analysis):
It is used most often when each data point contains a lot measurement, and not all of those may be meaningful, or there's a lot of covariance in the measurements.

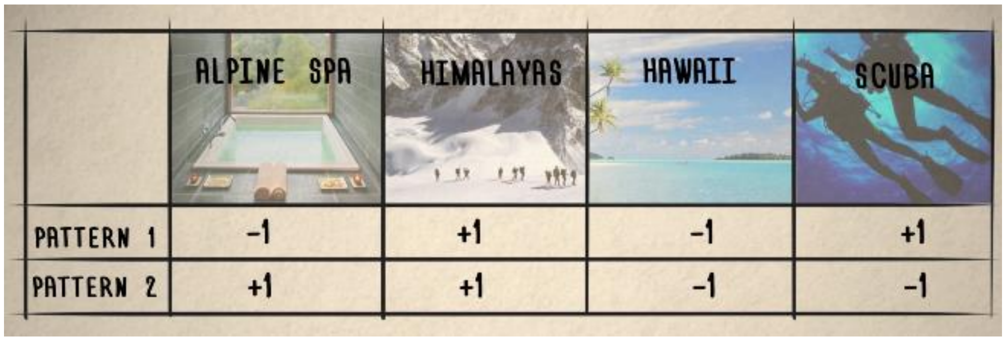
Principal component = typical pattern

Holiday destination score:



	APLPINE SPA	HIMALAYAS	HAWAII	SCUBA
ANNE	20	4	16	0
BILL	10	2	18	10
CHRIS	13	1	19	7
JEN	1	13	7	19
JOE	18	10	10	2
MAGGIE	4	20	0	16

We want to find patterns such that each person's rating can be explained as a combination of patterns.
These patterns are the principal components. Each principal component is a vector.

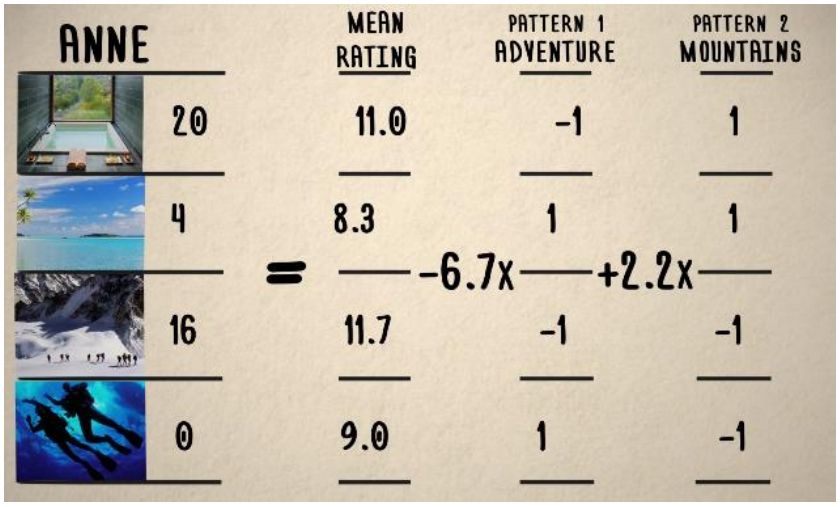


	ALPINE SPA	HIMALAYAS	HAWAII	SCUBA
PATTERN 1	-1	+1	-1	+1
PATTERN 2	+1	+1	-1	-1

Component 1: [-1, +1, -1, +1]
Component 2: [+1, +1, -1, -1]

Pattern one corresponds to the amount of adventure for each place.
Pattern two express a liking for mountains.

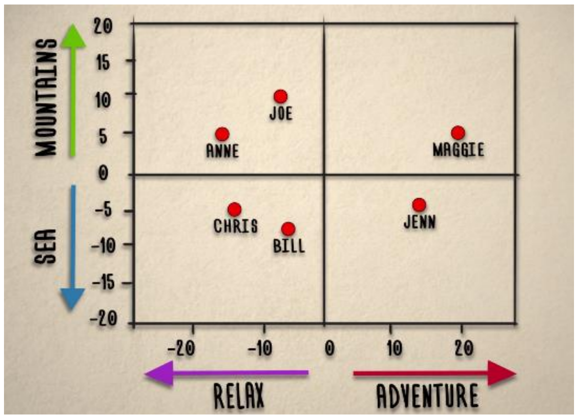
Now, each person's ratings can be expressed by two patterns and a mean rating. And ratings,



ANNE

$$\begin{array}{c} 20 \\ 4 \\ 16 \\ 0 \end{array} = \underset{\text{MEAN RATING}}{\begin{array}{c} 11.0 \\ 8.3 \\ 11.7 \\ 9.0 \end{array}} - 6.7 \times \underset{\text{PATTERN 1 ADVENTURE}}{\begin{array}{c} -1 \\ 1 \\ -1 \\ 1 \end{array}} + 2.2 \times \underset{\text{PATTERN 2 MOUNTAINS}}{\begin{array}{c} 1 \\ 1 \\ -1 \\ -1 \end{array}}$$

We can now also visualize the data. For each person, we compute the two coefficients, one for each pattern, and plot each person as a data point on these two axis.



Instead of four features, we now have only two features.
The major axis in our data = patterns

Each data point can be expressed as a linear combination of these patterns or components.

The principal components capture major patterns, and each data point can be expressed by these components.

If each component describes a feature, we have reduced the number of features.
Dimensionality reduction = reduce the number of features

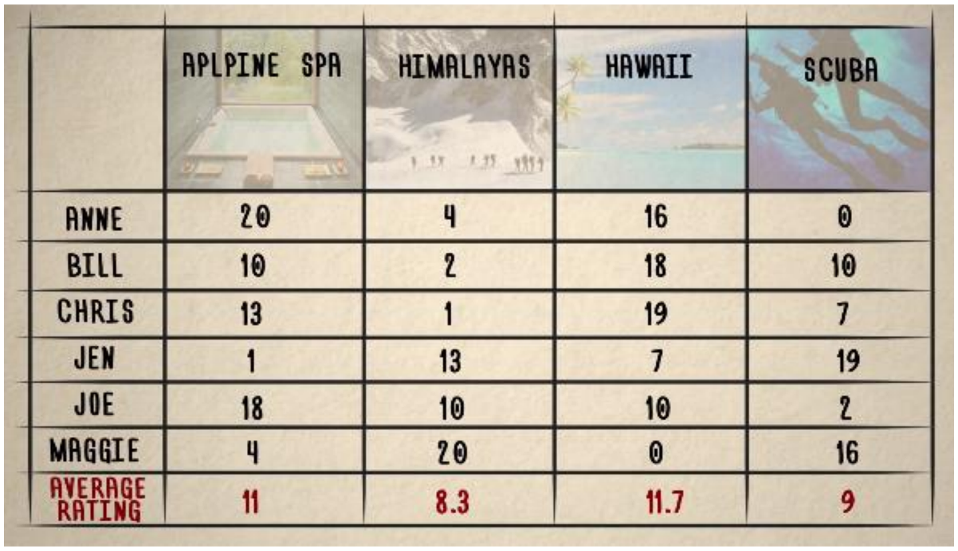
1.16 The Magic of Eigenvectors I

How to compute PCA? Eigenvectors!

In PCA, we want to find principal patterns. Typically, these patterns do not affect a single feature, or in the holiday example, a single rating. They affect many ratings together. Ratings for several holiday locations go up and down together, governed by some latent factor like adventure. This going up and down together is the covariance of the ratings. They co-vary from the mean.

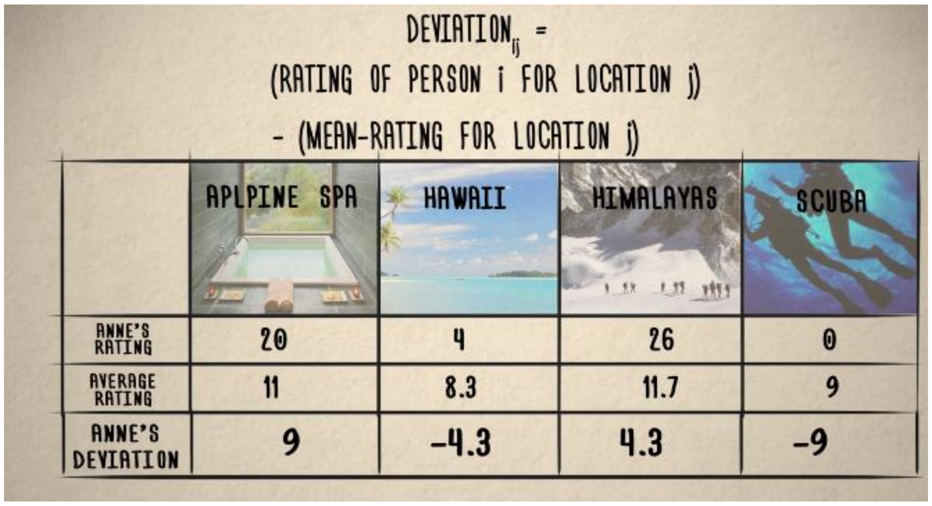
We want to find the major directions of covariance. To find those, a helpful tool is the covariance matrix of the data.

Average rating for each destination:



	APLPINE SPA	HIMALAYAS	HAWAII	SCUBA
ANNE	20	4	16	0
BILL	10	2	18	10
CHRIS	13	1	19	7
JEN	1	13	7	19
JOE	18	10	10	2
MAGGIE	4	20	0	16
AVERAGE RATING	11	8.3	11.7	9

Anne’s rating deviation:



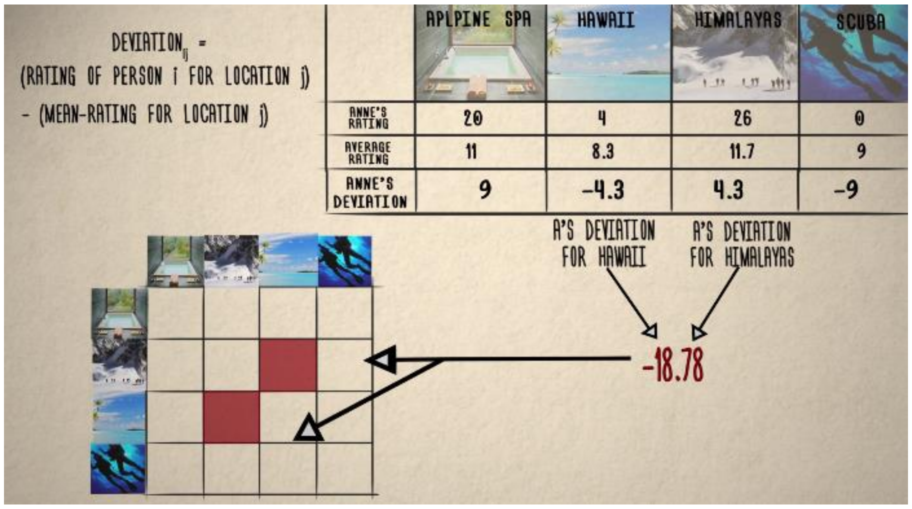
DEVIATION_{ij} =
(RATING OF PERSON i FOR LOCATION j)
- (MEAN-RATING FOR LOCATION j)

	APLPINE SPA	HAWAII	HIMALAYAS	SCUBA
ANNE'S RATING	20	4	26	0
AVERAGE RATING	11	8.3	11.7	9
ANNE'S DEVIATION	9	-4.3	4.3	-9

The covariance of two locations, Hawaii and the Himalayas, measures how much the ratings both vary in the same direction.

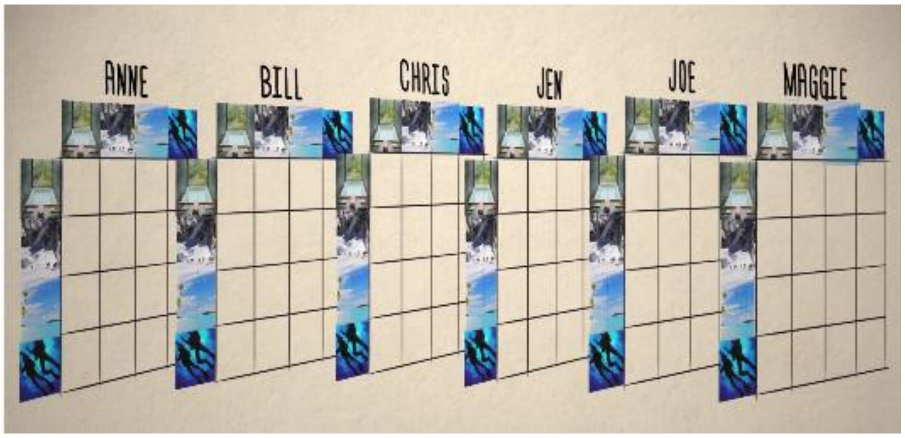
For each person, we multiply the deviation for Hawaii and for Himalayas. If both ratings are larger than their mean, then the product is positive. If they vary from the mean in different directions, the product is negative.

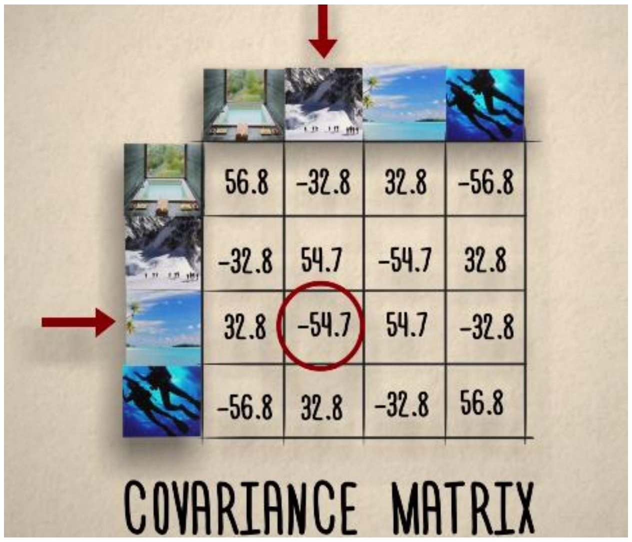
We do that for all people. The covariance of Hawaii and the Himalayas is then the sum of those.



DEVIATION$_{ij}$ =
(RATING OF PERSON i FOR LOCATION j)
- (MEAN-RATING FOR LOCATION j)

	APLPINE SPA	HAWAII	HIMALAYAS	SCUBA
ANNE'S RATING	20	4	26	0
AVERAGE RATING	11	8.3	11.7	9
ANNE'S DEVIATION	9	-4.3	4.3	-9





56.8	-32.8	32.8	-56.8
-32.8	54.7	-54.7	32.8
32.8	-54.7	54.7	-32.8
-56.8	32.8	-32.8	56.8

COVARIANCE MATRIX

Covariance matrix for n feature is: n by n.

The principal components that we are looking for are the eigenvectors of this coherence matrix.

What are eigenvectors?
We can define them by our matrix and vector products.

$$\underset{\text{MATRIX}}{\begin{bmatrix} 6 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix}} \times \underset{\text{VECTOR}}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} = \begin{bmatrix} 11 \\ 3 \\ 6 \end{bmatrix}$$

$$\underset{\text{MATRIX}}{\begin{bmatrix} 6 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix}} \times \underset{\text{EIGENVECTOR}}{\begin{bmatrix} .85 \\ .26 \\ .45 \end{bmatrix}} = \underset{\text{EIGENVALUE X EIGENVECTOR}}{8.2\text{x} \begin{bmatrix} .85 \\ .26 \\ .45 \end{bmatrix}}$$

When we multiply a vector it by a matrix, the vector gets rotated and scaled. If the vector is an eigenvector, its direction stays the same. It only gets scaled. The amount by which it gets scaled is the corresponding eigenvalue.
Each eigenvector has an eigenvalue. In some sense, the eigenvectors capture the major directions that are inherent in the matrix, and the larger the eigenvalue, the more important is the vector.

$$\underbrace{A}_{\text{MATRIX}} \; \underbrace{x}_{\text{EIGENVECTOR}} = \underbrace{\lambda}_{\text{EIGENVALUE}} \; \underbrace{x}_{\text{EIGENVECTOR}}$$

The eigenvectors with the largest eigenvalues are the principal components.
PCA can only capture linear relationship in the data.
PCA: patterns are linear combination of features.

1.17 Clustering in Graphs and Networks

Data point described by a set of features? Or not…such as network or graph.

Our data is given as a network or a graph. Each data point, for example, a person in a social network, is a node in the graph. The connections are edges.

Social network
Internet
Biological Network
Computer Network
Data Represented as Networks

Graph: nodes and edges

Community structure: Dense groups?

Cluster analysis:

Social network: how epidemics spread
Dynamic social network: information of friends and opinions
Biology: cluster of protein can be indicative of functionality
Science: coauthor networks indicate research topics

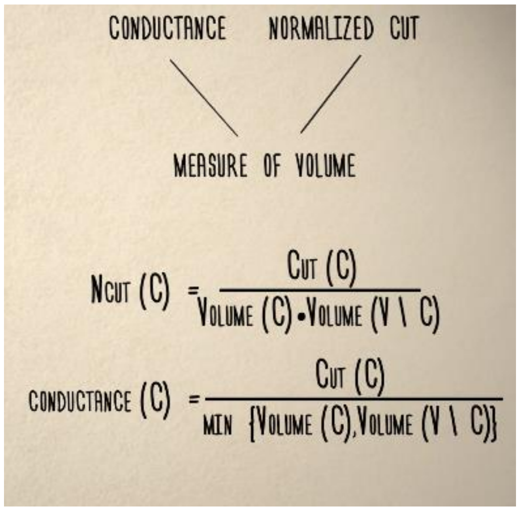
What is cluster in a graph?
A cluster consists of points that are well connected with each other.

Volume: Number of edges in cluster
Density: Volume per node

Cluster in a graph:

1) large density of node.
2) between different clusters there should not be many edges.

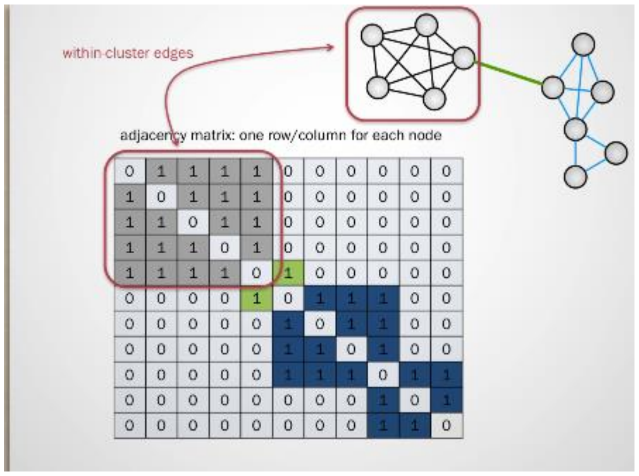
The separation of cluster is measured by the cut value
Low cut value = good criteria?
No. Instead combining cut and volume, such as conductance and normalized cut.



$$\mathrm{Ncut}(C) = \frac{\mathrm{Cut}(C)}{\mathrm{Volume}(C) \cdot \mathrm{Volume}(V \setminus C)}$$

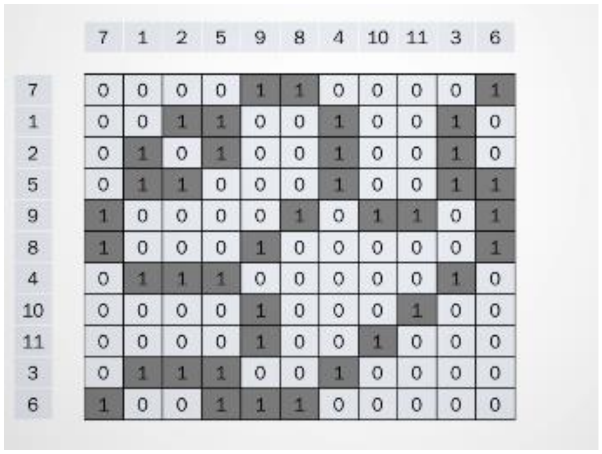
$$\mathrm{Conductance}(C) = \frac{\mathrm{Cut}(C)}{\min\{\mathrm{Volume}(C), \mathrm{Volume}(V \setminus C)\}}$$

By this measure, good clusters are not too small, internally well connected and separated from the rest of the nodes.
Clusters can also be seen in the so-called adjacency matrix.
Adjacency matrix (has an edge = 1):



adjacency matrix: one row/column for each node

0	1	1	1	1	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0
0	0	0	0	1	0	1	1	1	0	0
0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	1	1	0	1	0	0
0	0	0	0	0	1	1	1	0	1	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	1	1	0



	7	1	2	5	9	8	4	10	11	3	6
7	0	0	0	0	1	1	0	0	0	0	1
1	0	0	1	1	0	0	1	0	0	1	0
2	0	1	0	1	0	0	1	0	0	1	0
5	0	1	1	0	0	0	1	0	0	1	1
9	1	0	0	0	0	1	0	1	1	0	1
8	1	0	0	0	1	0	0	0	0	0	1
4	0	1	1	1	0	0	0	0	0	1	0
10	0	0	0	0	1	0	0	0	1	0	0
11	0	0	0	0	1	0	0	1	0	0	0
3	0	1	1	1	0	0	1	0	0	0	0
6	1	0	0	1	1	1	0	0	0	0	0

Modularity measure indicates how high the density of edges is within groups, compared to a baseline graph where edges occur randomly.

1.18 The Magic of Eigenvectors II

Graph clustering can be useful to understand community structure in a large network.

What is important here is the global connectivity structure of the graph. To capture that connectivity structure, we will see that, once more, eigenvectors can be useful.

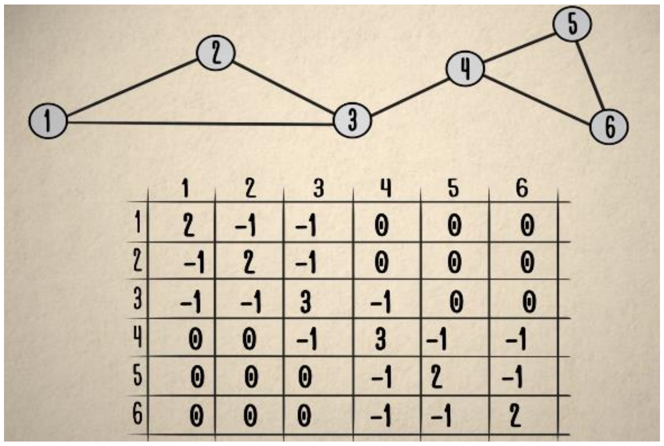
They encode an impressive amount of information about the structure of the graph.

The resulting methods are called spectral methods, and more precisely, spectral clustering.

The spectrum of a matrix is the set of its eigenvalues.

We've seen eigenvectors for PCA, but what is the matrix for a graph that gives us the magic eigenvectors?
The matrix is the so-called Laplacian of the graph.

Matrix of a graph: Laplacian of the graph



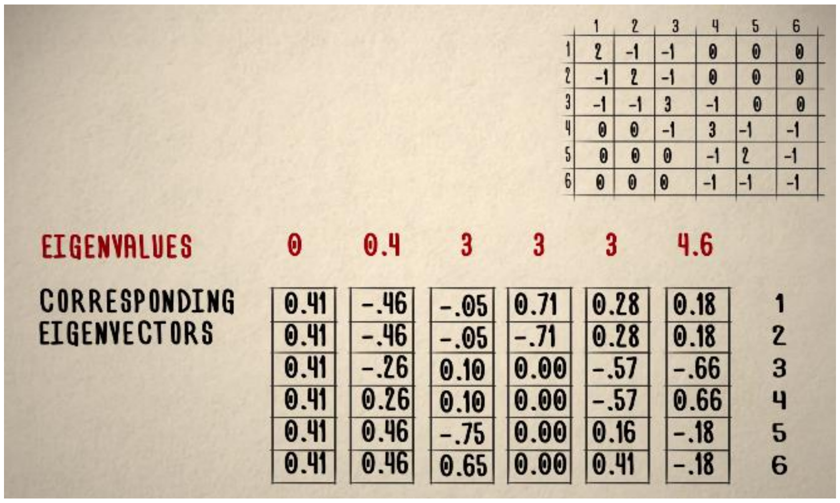
	1	2	3	4	5	6
1	2	-1	-1	0	0	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	0	0	0	-1	2	-1
6	0	0	0	-1	-1	2

Off-diagonal elements:
-1: if there's edge
0: if there's no edge

Diagonal elements: the degrees of the nodes, the number of edge it meets

This matrix is often normalized by dividing by degrees or the square root of the degrees

Eigenvalue and eigenvector of Laplacian matrix:

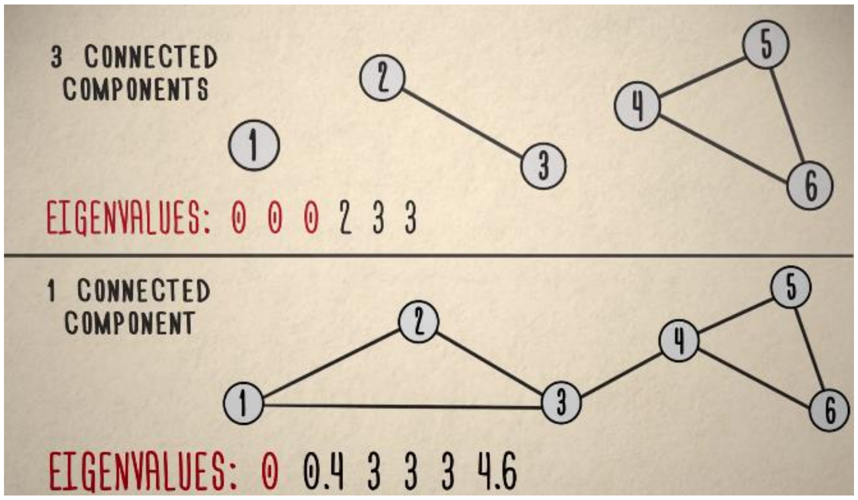


	1	2	3	4	5	6
1	2	-1	-1	0	0	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	0	0	0	-1	2	-1
6	0	0	0	-1	-1	-1

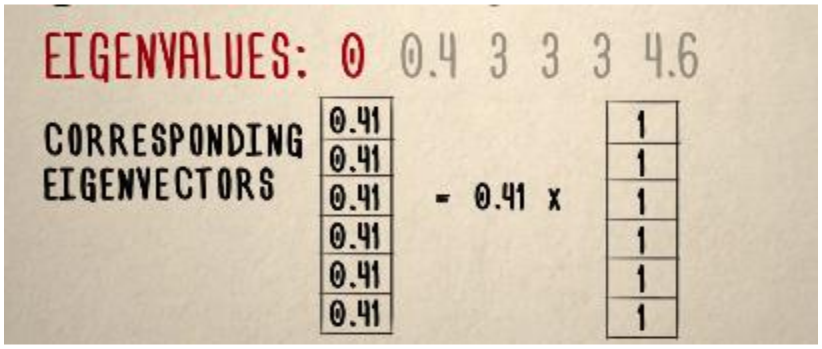
EIGENVALUES	0	0.4	3	3	3	4.6	
CORRESPONDING EIGENVECTORS	0.41	-.46	-.05	0.71	0.28	0.18	1
	0.41	-.46	-.05	-.71	0.28	0.18	2
	0.41	-.26	0.10	0.00	-.57	-.66	3
	0.41	0.26	0.10	0.00	-.57	0.66	4
	0.41	0.46	-.75	0.00	0.16	-.18	5
	0.41	0.46	0.65	0.00	0.41	-.18	6

The number of 0 eigenvalues is exactly the number of disconnected parts of the graph.

These parts are called connected components.

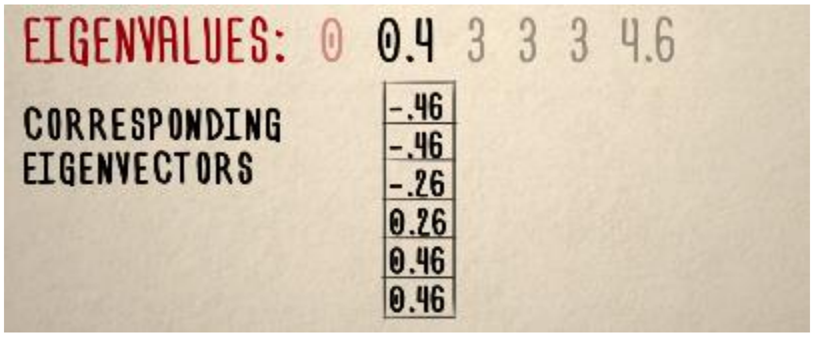


The eigenvector for eigenvalue 0 is the all once vector. That's the same for all graphs, and hence pretty uninteresting.


EIGENVALUES: 0 0.4 3 3 3 4.6

CORRESPONDING EIGENVECTORS

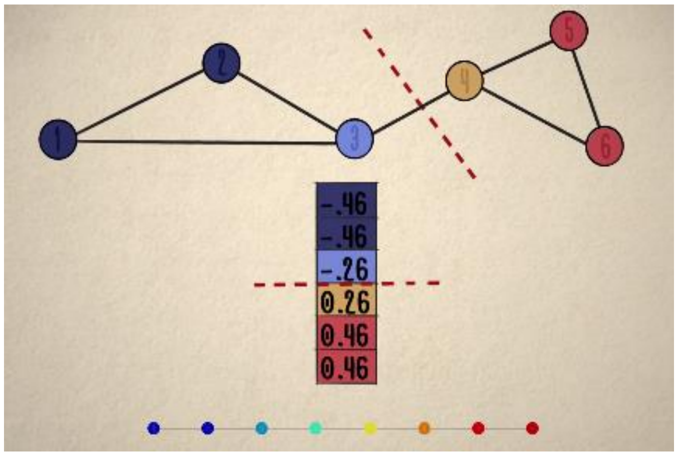
$$\begin{bmatrix} 0.41 \\ 0.41 \\ 0.41 \\ 0.41 \\ 0.41 \\ 0.41 \end{bmatrix} = 0.41 \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

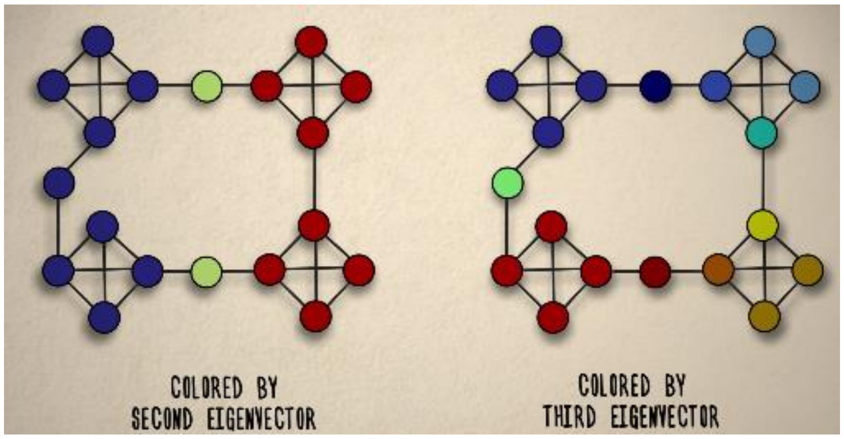

EIGENVALUES: 0 0.4 3 3 3 4.6

CORRESPONDING EIGENVECTORS

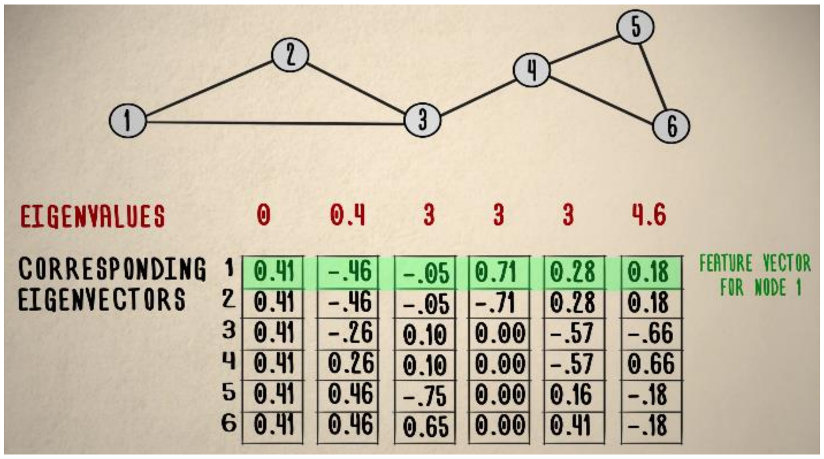
$$\begin{bmatrix} -.46 \\ -.46 \\ -.26 \\ 0.26 \\ 0.46 \\ 0.46 \end{bmatrix}$$

Each item vector has n entries, one for each node in the graph. Here are two graphs, and we color code the notes by the value in the item vector. Now this is interesting. The values in the eigenvector reflect the graph's vector. Close by nodes have similar colors. In fact, we could find a threshold to partition the nodes. All nodes with values above the threshold go in one group and all others in the other group. If we do this here, we actually find a meaningful clustering in the graph.



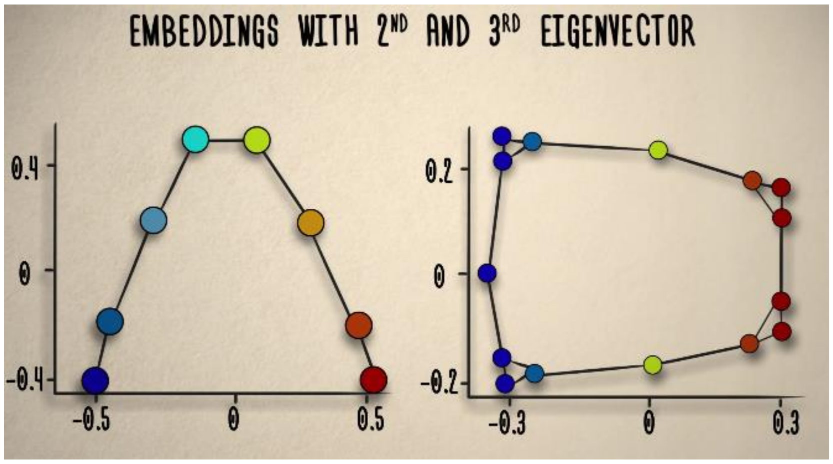


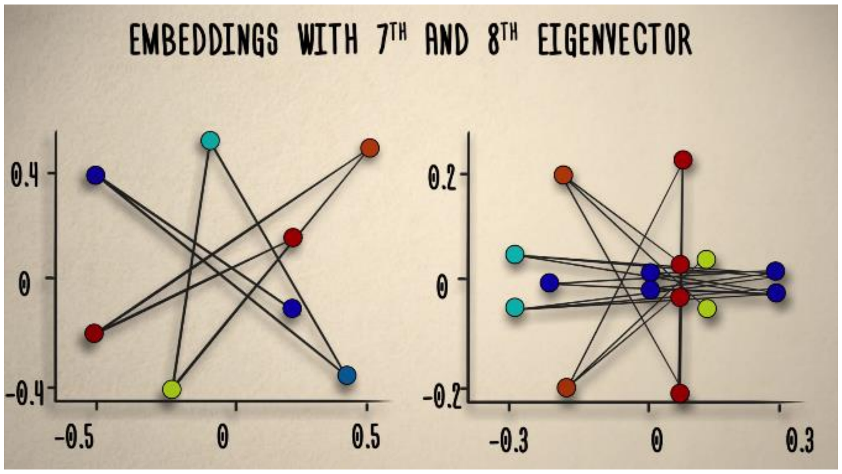
Remember, our nodes do not have feature vectors. Now, we can give each node one feature value from each eigenvector and we suddenly have vectors. These new features are called an embedding.
Embedding: Give each node one feature value from each eigenvector


1
2
3
4
5
6
FEATURE VECTOR FOR NODE 1

EIGENVALUES	0	0.4	3	3	3	4.6
CORRESPONDING EIGENVECTORS 1	0.41	-.46	-.05	0.71	0.28	0.18
2	0.41	-.46	-.05	-.71	0.28	0.18
3	0.41	-.26	0.10	0.00	-.57	-.66
4	0.41	0.26	0.10	0.00	-.57	0.66
5	0.41	0.46	-.75	0.00	0.16	-.18
6	0.41	0.46	0.65	0.00	0.41	-.18

PCA also gave an embedding. For PCA, we use the vectors with the largest eigenvalues. Here, the smallest ones will be important.



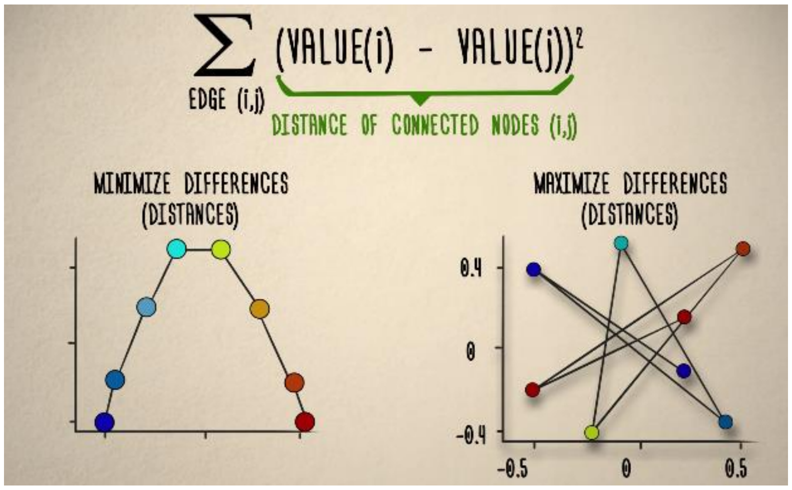


What do these features mean?

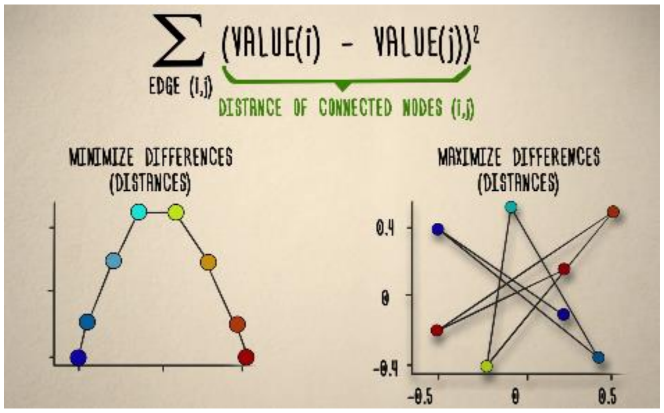
Just like we plotted the fetus from PCA, let's take the second and third smallest eigenvectors and use them as coordinates for the nodes. Here's what this looks like. Nodes that are connected are positioned close by on the plane. Our embedding here wants to minimize the stretch of the edges. And clusters in the graph become clusters in 2D.

So far, we've used the eigenvectors with lowest eigenvalues. What about the ones with the largest eigenvalues? Here is what happens if we use these as an embedding. They have the opposite effect. They maximize the stretch of the edges. Nodes that are connected are placed far away on the plane. This is the same graph but looks so different.

Is there an explanation? In fact, the eigenvectors minimize or maximize a different score.


$\sum_{\text{EDGE }(i,j)} \underbrace{(\text{VALUE}(i) - \text{VALUE}(j))^2}_{\text{DISTANCE OF CONNECTED NODES }(i,j)}$
MINIMIZE DIFFERENCES (DISTANCES)
MAXIMIZE DIFFERENCES (DISTANCES)
0.4
0
-0.4
-0.5
0
0.5

Imbedding minimize the stretch of the edges, the eigenvectors minimize or maximize a different score


$\sum_{EDGE\ (i,j)} \underbrace{(VALUE(i) - VALUE(j))^2}_{DISTANCE\ OF\ CONNECTED\ NODES\ (i,j)}$
MINIMIZE DIFFERENCES (DISTANCES)
MAXIMIZE DIFFERENCES (DISTANCES)
0.4
0
-0.4
-0.5
0
0.5

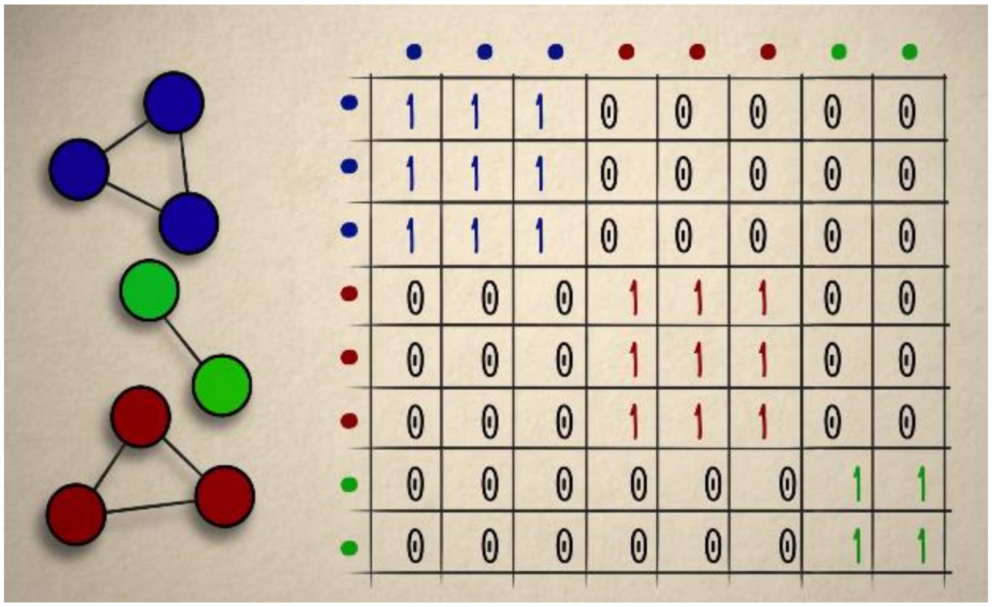
For each edge, we take the difference of the adjacent node values and square that.
We do this for all edges and sum it up.
The eigenvectors with small eigenvalues minimize these differences.
They give similar values to connected nodes.
And the difference can be viewed as the stretch of the edge.
The eigenvectors with large eigenvalues maximize the difference, they get very different values to connect the nodes.

Remember, each node gets a value. For each edge, we take the difference of the adjacent node values and square that. We do this for all edges and sum it up. The eigenvectors with small eigenvalues minimize these differences. They give similar values to connected nodes. And the difference can be viewed as the stretch of the edge. The eigenvectors with large eigenvalues maximize the difference, they get very different values to connect the nodes.

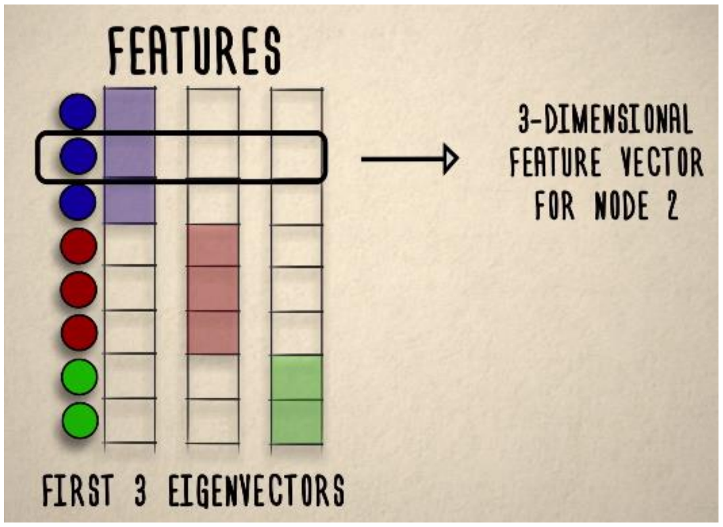
Overall, we see that the eigenvectors and eigenvalues contain a lot of valuable information about the graph structure, as we may suspect now, they are very useful for clustering. In fact, they contain even more information about the number of paths between any two nodes or about the importance of nodes and edges.

1.19 Spectral Clustering

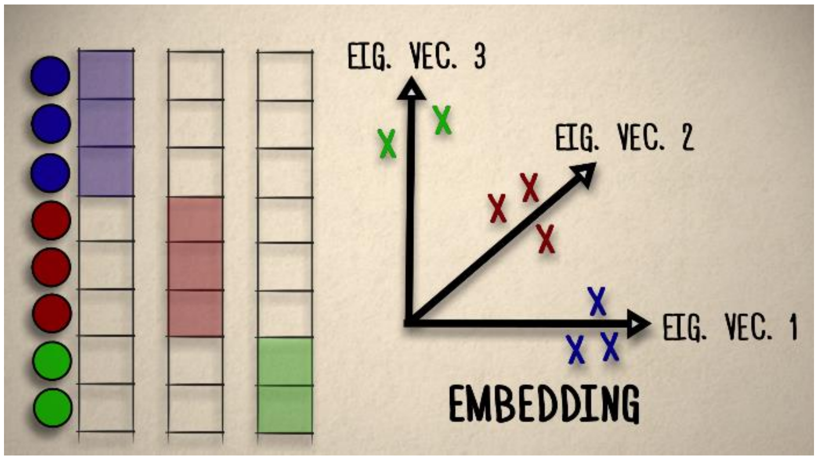
A graph where there are no connections between communities, and build this so-called adjacency matrix



	●	●	●	●	●	●	●	●
●	1	1	1	0	0	0	0	0
●	1	1	1	0	0	0	0	0
●	1	1	1	0	0	0	0	0
●	0	0	0	1	1	1	0	0
●	0	0	0	1	1	1	0	0
●	0	0	0	1	1	1	0	0
●	0	0	0	0	0	0	1	1
●	0	0	0	0	0	0	1	1

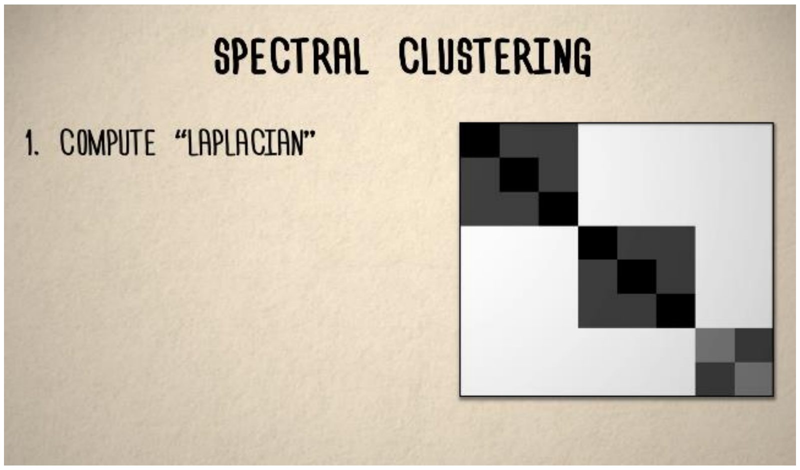


New feature as embedding:



Spectral clustering:

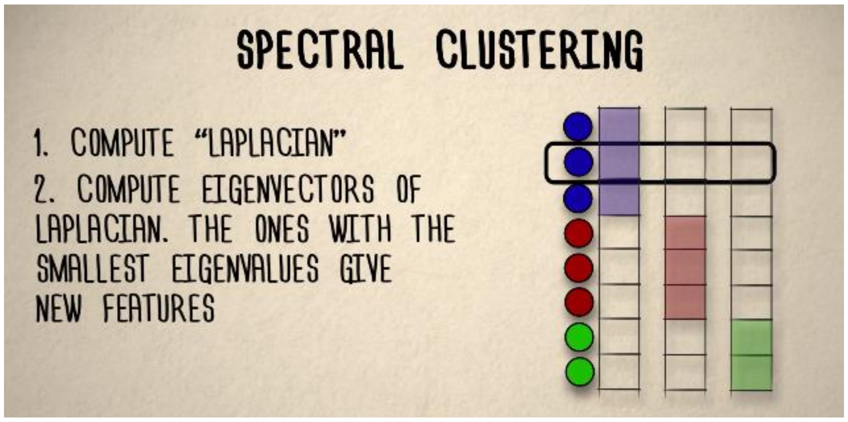
1. Write down the Laplacian matrix



SPECTRAL CLUSTERING

1. COMPUTE "LAPLACIAN"

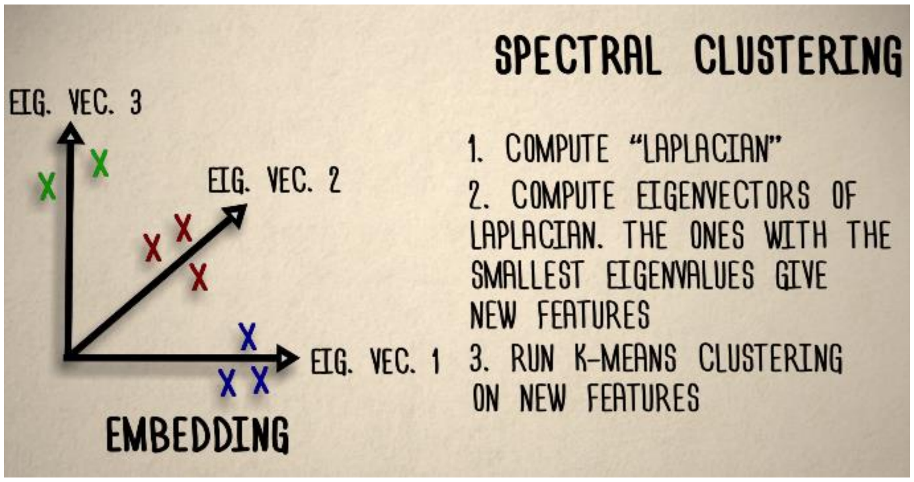
2. Compute the eigenvectors which give new features



SPECTRAL CLUSTERING

1. COMPUTE "LAPLACIAN"
2. COMPUTE EIGENVECTORS OF LAPLACIAN. THE ONES WITH THE SMALLEST EIGENVALUES GIVE NEW FEATURES

3. Use K-means clustering on those new features

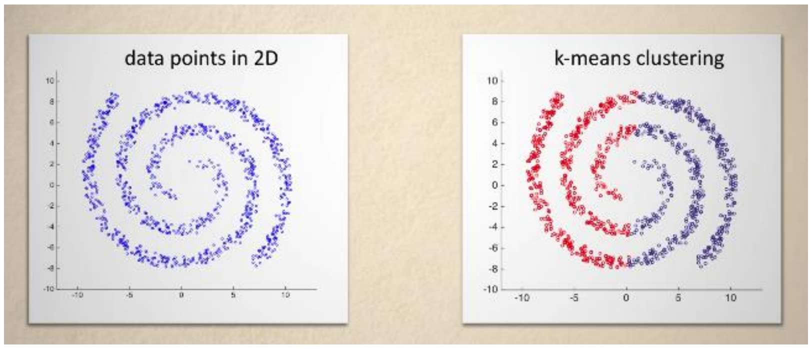


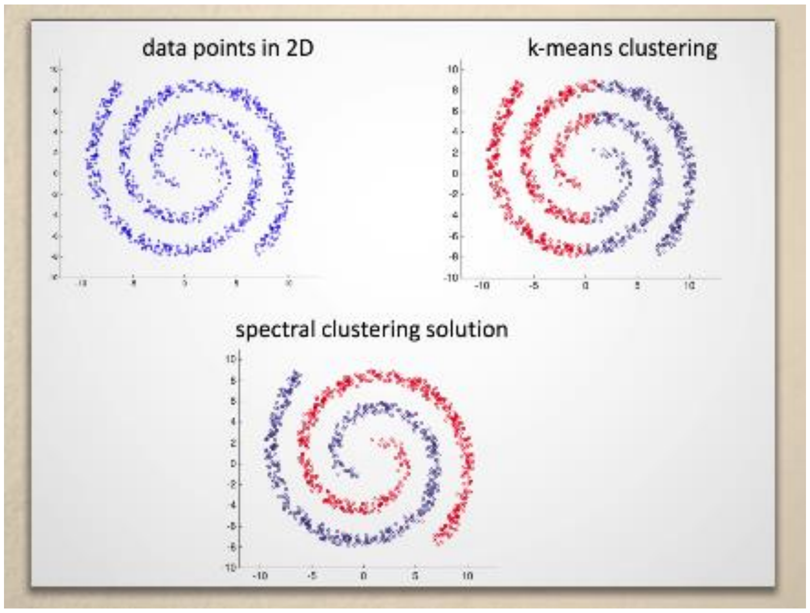
SPECTRAL CLUSTERING

1. COMPUTE "LAPLACIAN"
2. COMPUTE EIGENVECTORS OF LAPLACIAN. THE ONES WITH THE SMALLEST EIGENVALUES GIVE NEW FEATURES
3. RUN K-MEANS CLUSTERING ON NEW FEATURES

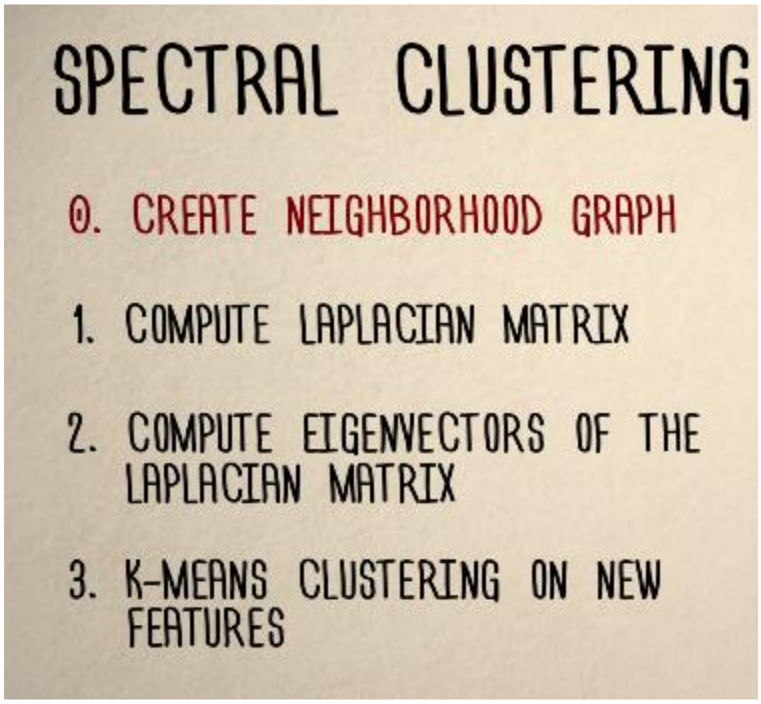
This works because the new features magically separate the communities.
To find K clusters, one typically uses K-eigenvectors.

If the graph is completely connected, there's a boring eigenvector all at ones that we discard right away.





In summary, to use spectral clustering on other data, we just add one step to our procedure. Create the so-called neighborhood graph.



SPECTRAL CLUSTERING

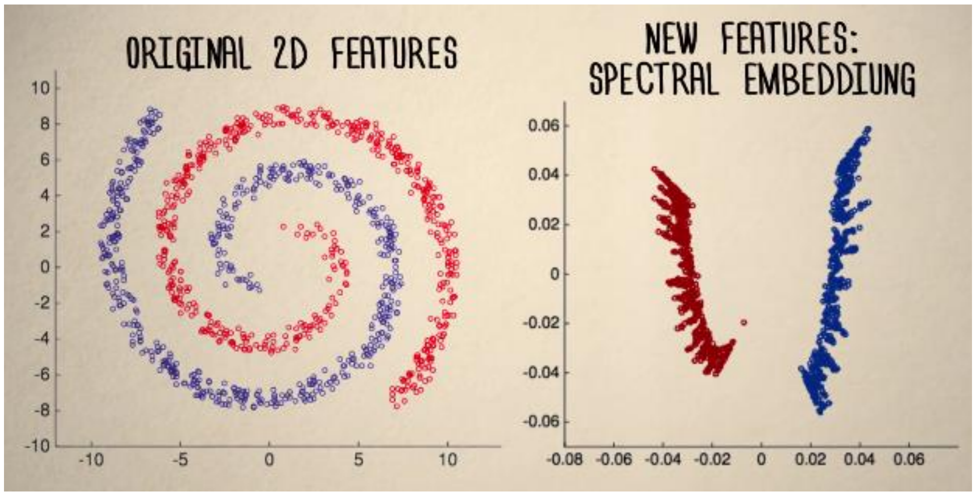
0. CREATE NEIGHBORHOOD GRAPH
1. COMPUTE LAPLACIAN MATRIX
2. COMPUTE EIGENVECTORS OF THE LAPLACIAN MATRIX
3. K-MEANS CLUSTERING ON NEW FEATURES

An edge between two points is weighted by the similarity of the points.

A common similarity function is the Gaussian similarity that decays exponentially with a distance.

What is the difference on spectral clustering and K-means here?
The inter-point distances used by K-means don't capture the complicated shape of the data.

Spectral Clustering also uses K-means but with an important difference.
We created new feature factors and use K-Means with those new features.



Then the clusters follow the inherent tape of the data. In other words, we found them embedding for our data points that captures delayed in structure much better.

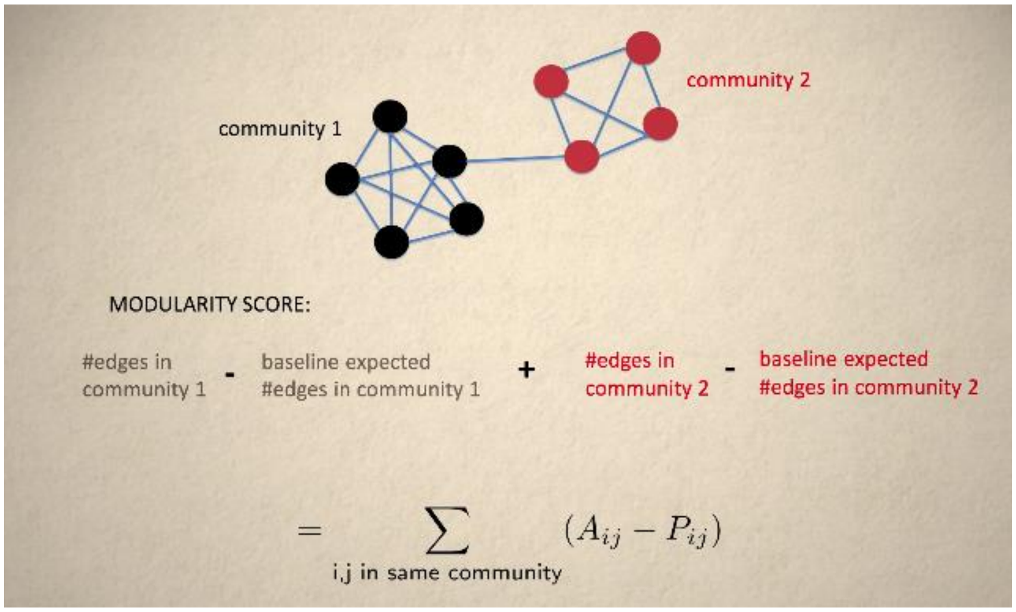
We've seen embedding with PCA, but this one is different.
In PCA the new features are some combination of the existing features, even though we didn't explicitly write it that way.
With a graph, there's no such relatively simple relation. Here we have a nonlinear embedding.
In summary, by putting everything together we saw how to use the eigenvectors of the Laplacian matrix to find meaningful clusters that respect hidden structure in the data.

1.20 Modularity Clustering

Specify the number of communities automatically

Define and maximize the modularity score

The characteristic idea of modularity is to compare the communities to a random baseline graph that share some properties with our actual graph.
Such as the number of edges, and roughly, the degree of the nodes.



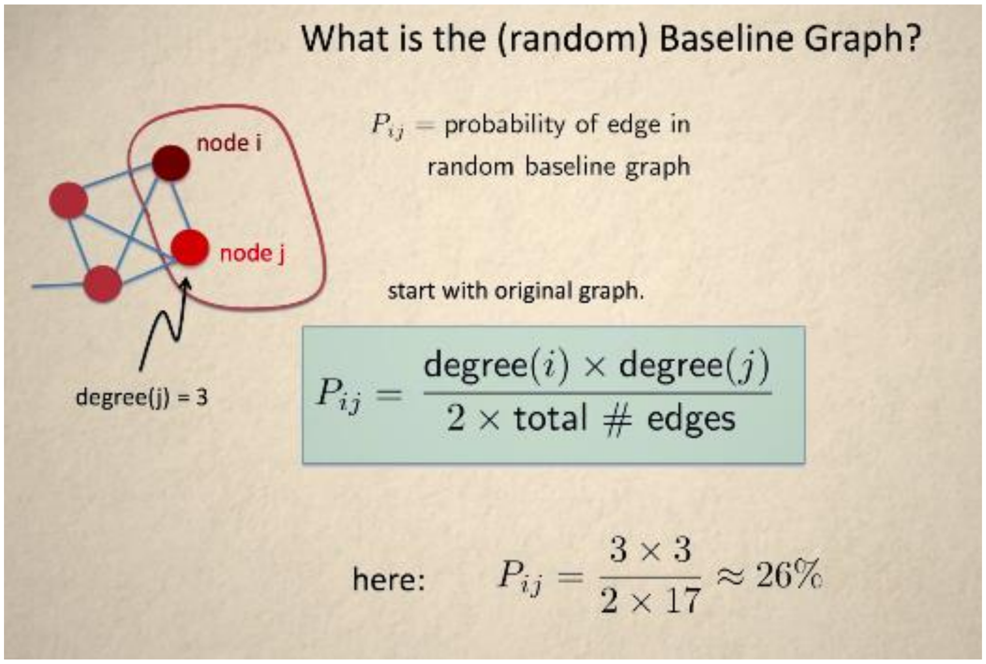
MODULARITY SCORE:

#edges in community 1 - baseline expected #edges in community 1 + #edges in community 2 - baseline expected #edges in community 2

$$= \sum_{i,j \text{ in same community}} (A_{ij} - P_{ij})$$

Modularity is high = community is dense

expected baseline number of edges:



What is the (random) Baseline Graph?

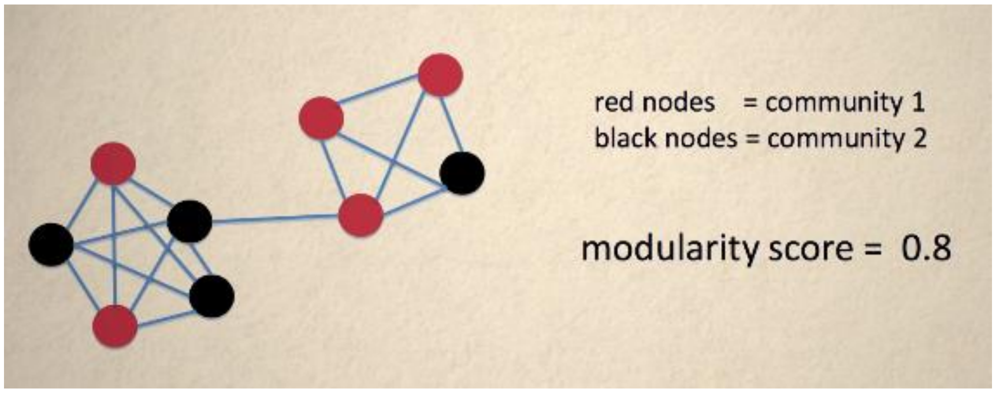
P_{ij} = probability of edge in random baseline graph

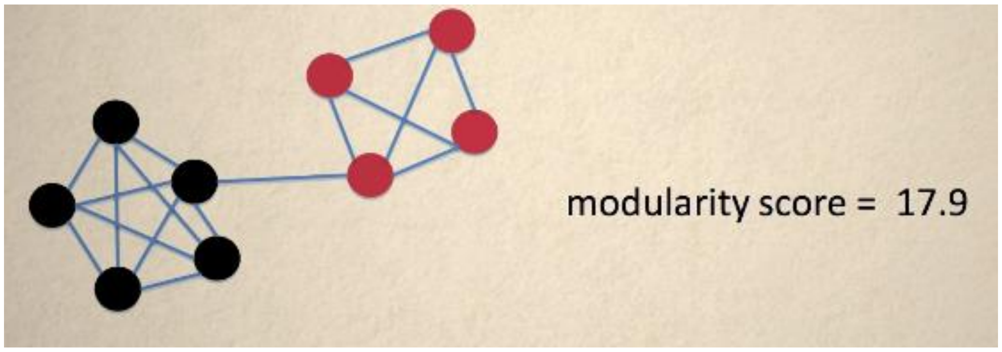
start with original graph.

$$\boxed{P_{ij} = \frac{\text{degree}(i) \times \text{degree}(j)}{2 \times \text{total } \# \text{ edges}}}$$

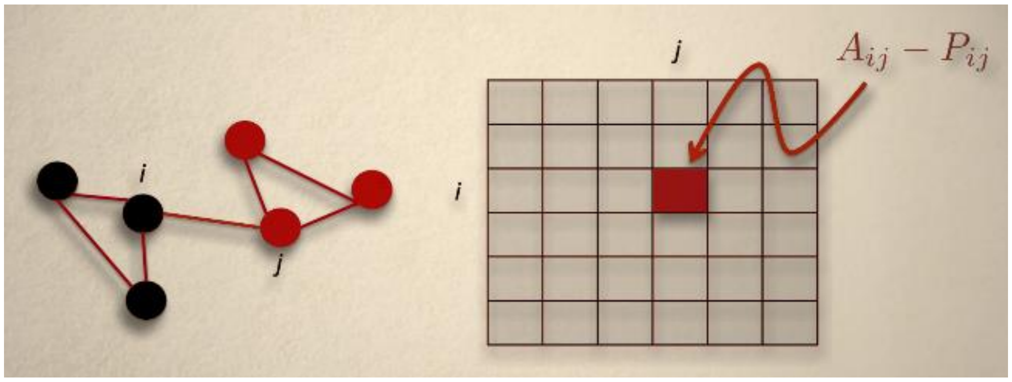
here: $P_{ij} = \dfrac{3 \times 3}{2 \times 17} \approx 26\%$

Nodes with edges in our graph are expected to have many edges in the baseline tool.

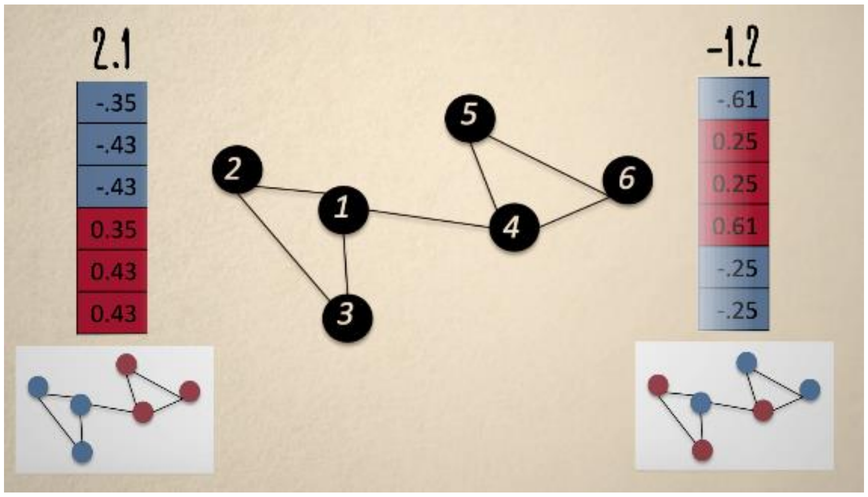




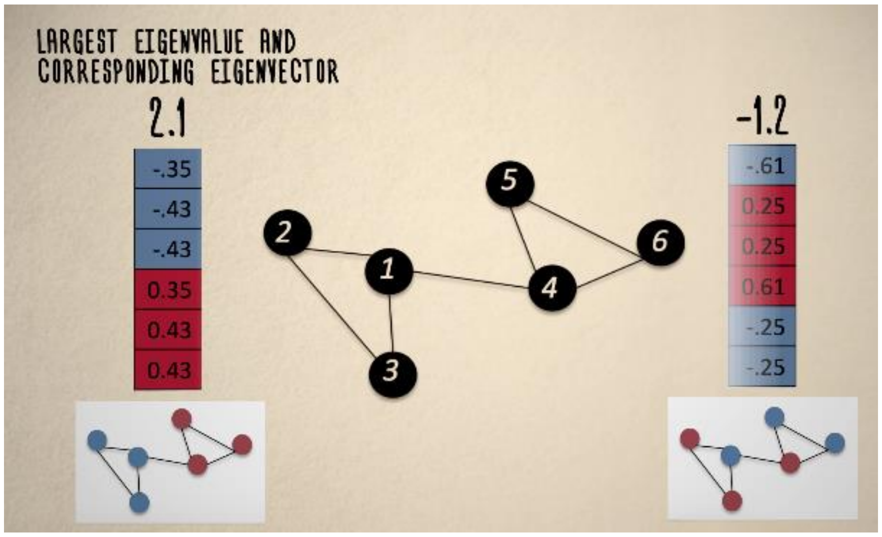
Define matrix:



Compute eigenvector of this matrix



Each eigenvector assigns a value to each node.
For two communities, we use the eigenvector with the largest eigenvalue.



One important difference to our previous vector clustering method:
The eigenvalues here can be negative.

A negative eigenvalue means that the associated eigenvector gives the node value that leads to low modularity.

1.21 Embeddings and Components

Embedding: find new feature vectors for the data points

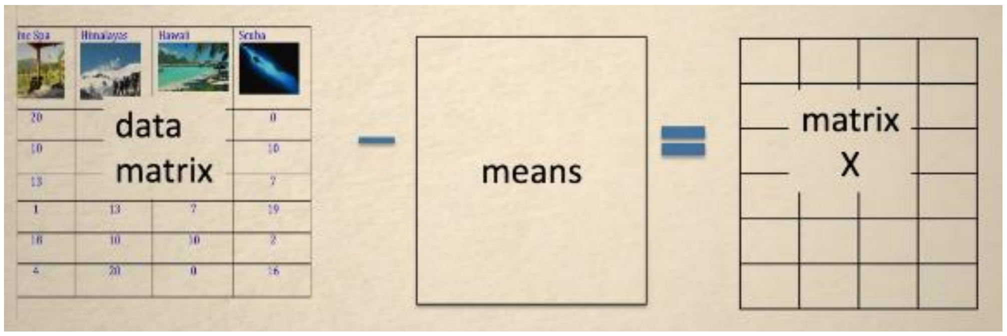
Use embedding to:
Dimensionality reduction
Discover hidden clusters in the data
Help to understand data via major patterns, components or themes.

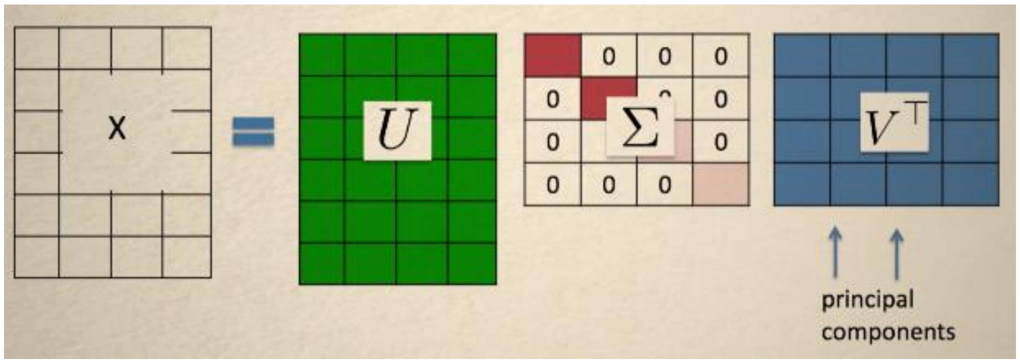
Metric Learning is a method that takes outside information
to guide the embedding.

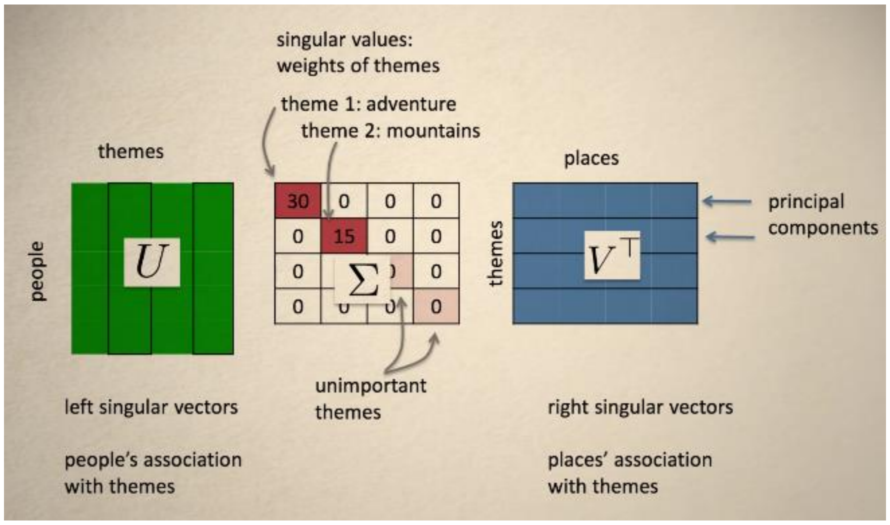
Independent component analysis or ICA recovers components that are statistically independent.
That is, they have no information about of each other.

Dictionary Learning poses constraints on the data point associations.
Each data point maybe associated with only very few components.

Singular Value Decomposition or SVD






singular values:
weights of themes
theme 1: adventure
theme 2: mountains
themes
people
U
30
15
0
Σ
places
themes
$V^\top$
principal
components
unimportant
themes
left singular vectors
people's association
with themes
right singular vectors
places' association
with themes