Visual Representations for Machine Learning

Spectral Clustering and Channel Representations

Lecture 1

Spectral Clustering: introduction and confusion

Michael Felsberg Klas Nordberg

- The Spectral Clustering part is to a large extent the same as the 2012 course
- Then planned and presented by
  - **Vasileios Zografos** & Klas Nordberg

What this course is

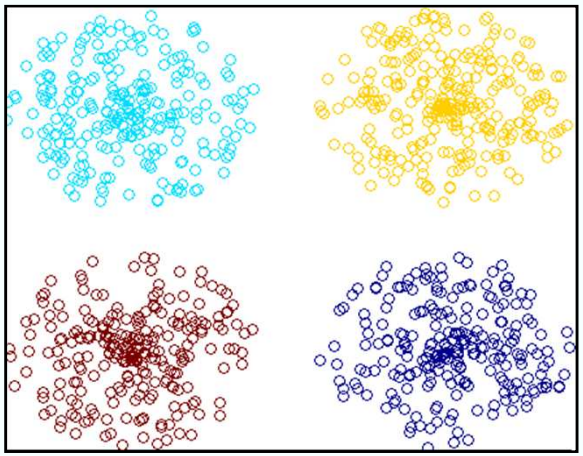
- Basic introduction into the core ideas of spectral clustering and channel representations
- Sufficient to get a basic understanding of how the methods work
- Application mainly to computer vision

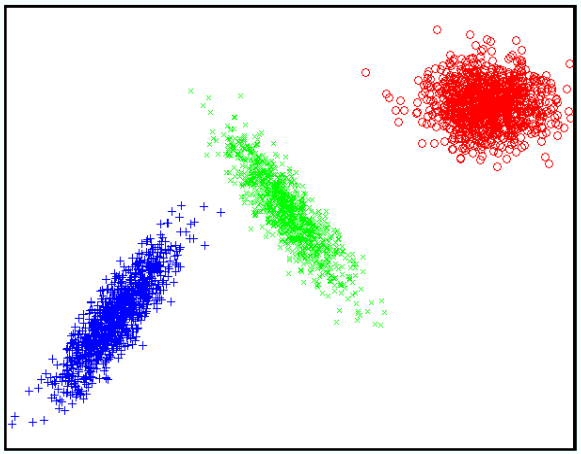
Course contents

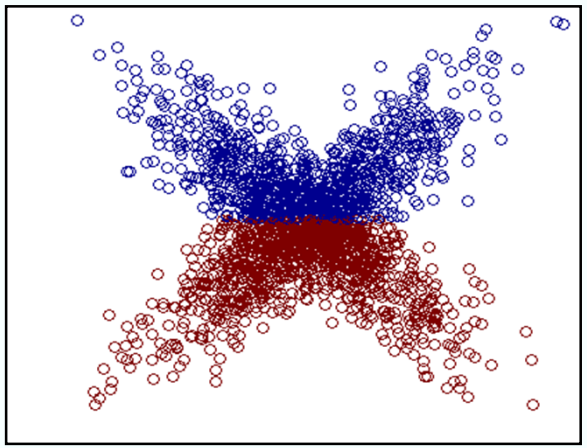
- 4 lectures
  - Lecture 1: Spectral clustering: Introduction and confusion, KN
  - Lecture 2: Spectral clustering: From confusion to clarity, KN
  - Lecture 3: Channel Representations: encoding, MF
  - Lecture 4: Channel Representations: decoding, MF
- 2 courseworks (seminars)
  - Article seminar on spectral clustering
  - Article seminar on channel representations

Overview of clustering

- **What is clustering?**
  - Given some data and a notion of *similarity*
  - Partition the input data into maximally *homogeneous* groups (i.e. *clusters*)







Overview of clustering

- Applications
  - Image processing and computer vision
  - Computational biology
  - Data mining and information retrieval
  - Statistical data analysis
  - Machine learning and pattern recognition
  - …

Overview of clustering

- **What is a cluster?**
  - Homogeneous group
  - No universally accepted definition of *homogeneity*

- In general a cluster should satisfy **two** criteria:
  - **Internal:** All data inside a cluster should be highly *similar* (intra-cluster)
  - **External:** Data between clusters should be highly *dissimilar* (inter-cluster)

A clustering of clustering

Connectivity based
- Hierarchical clustering
- …

Mode seeking
- Mean / Median shift
- Medoid shift
- …

Distribution based
- E-M algorithm
- KDE clustering
- …

Centroid based
- K-Means
- …

Graph theoretic
- Graph cuts
- Spectral clustering
- …

K-means

- Basic clustering algorithm. Given a set of observations $x_1, \ldots x_N$, partition them into k clusters with means μ_i s.t. the within cluster sum of squares (distortion) is minimised

$$\arg\min \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

- NP-hard. Iterative algorithm available
  1. Initialise k clusters
  2. Calculate cluster means μ_i
  3. Calculate distances of each point x_j to each cluster mean μ_i
  4. Assign point to nearest cluster
  5. Goto 2 until convergence

- Number of clusters k need to be known. Gives convex clusters

What is spectral clustering

- In relation to spectral clustering
  - Similarity is quantified by **affinity**
  - Affinity *A* between two points x, y:
    - In general: $0 \leq A \leq 1$ and

$$A(x,y) = \begin{cases} \approx 1, & \text{when } x \text{ and } y \text{ are similar,} \\ \approx 0, & \text{when } x \text{ and } y \text{ are dissimilar.} \end{cases}$$

What is spectral clustering

- Clustering algorithm:
  - Treats clustering as a **graph partitioning** problem without making specific assumptions on the form of the clusters.
  - Cluster points using **eigensystem** of matrices derived from the data.
  - Data **projected** to a low-dimensional space that are separated and can be easily clustered.

Pros and cons of spectral clustering

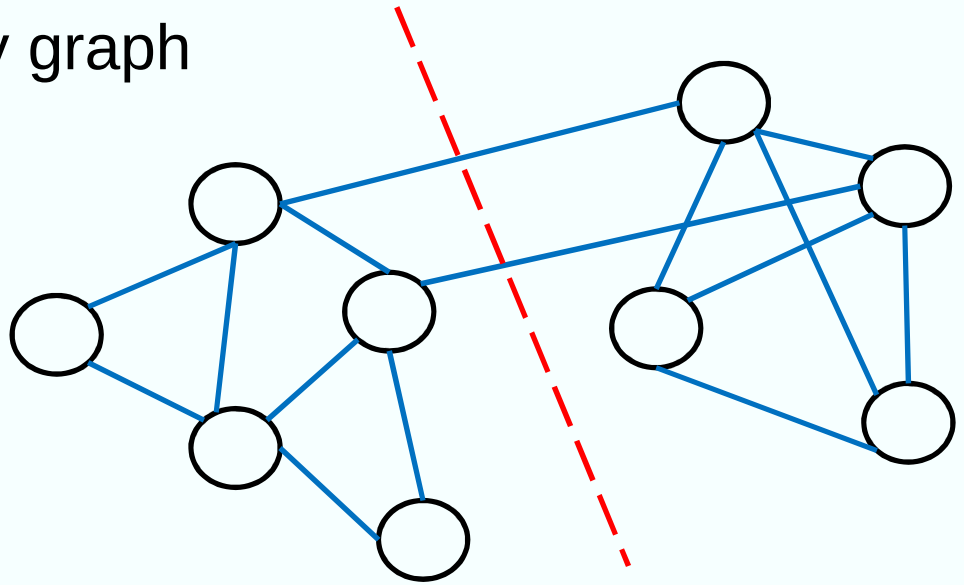
- Advantages:
  - Does not make strong assumptions on the statistics or shape of the clusters
  - Easy to implement.
  - Good clustering results.
  - Reasonably fast for sparse data sets of several thousand elements.

- Disadvantages:
  - May be sensitive to choice of parameters
  - Computationaly expensive for large datasets

Graph partitioning

Graph cut point of view

- Given data points $x_1, \ldots xN$, pairwise affinities $\mathrm{A}_{ij} = A(xi, xj)$
- Build similarity graph

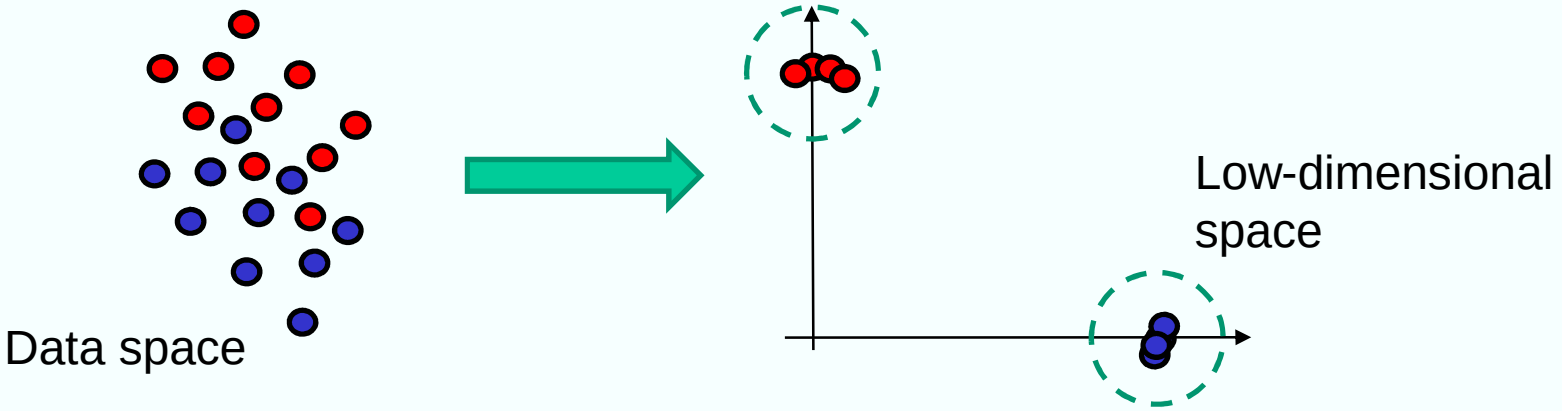


- Clustering = find a cut through the graph
  - Define a cost function, a function over different partitions (cuts)
  - Solve it = find cut of **minimal cost**

Spectral clustering

Low-dimensional embedding point of view

- Given data points $x_1, \ldots xN$, pairwise affinities $A_{ij} = A(xi, xj)$
- Find a low-dimensional embedding (not same as PCA!)
- Project data points to new space



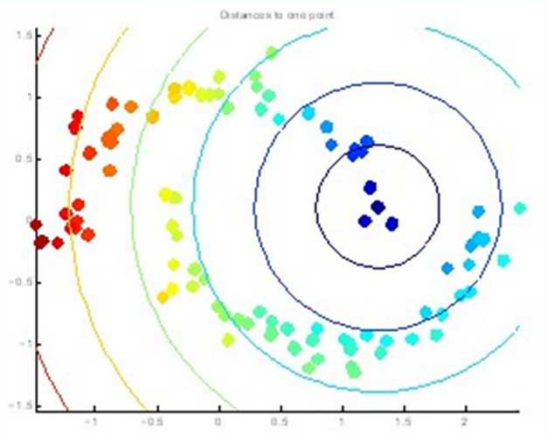
- Cluster using favourite clustering algorithm (e.g. *k*-means)

Graph cut vs spectral clustering

- The two points of views are related
- The low-dimensional space is determined by the data
- Spectral clustering makes use of the ***spectrum*** of the graph for dimensionality reduction
  - Embed data points in the subpace of the “largest” eigenvectors
- Projection and clustering equates to graph partition by different min-cut criteria

Graphs

- Graphs are an important component of spectral clustering
- Many datasets have natural graph structure
  - Web pages and links
  - Protein structures
  - Citation graphs
  - …
- Other datasets can be transformed simply into **similarity (or affinity) graphs**
  - Affinity can encode **local-structure** in the data
  - Global structure induced by a distance function is often misleading



- Suited for representing data based on pairwise relationships (e.g. affinities, local distances)
- A positive symmetric matrix can be represented as a graph

Affinity and distance

- An **affinity score** between two objects x_i, xj is "high" if the objects are "very similar"
  - E.g. the Gaussian kernel $s(i,j) = \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right)$

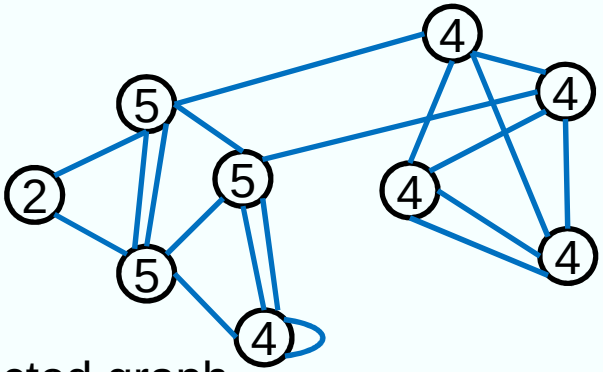
σ is a parameter!

- A **distance score** between two objects x, y is "small" if the objects are "close" to each other
  - E.g. the Euclidean distance $d(i,j) = \|x_i - xj\|$

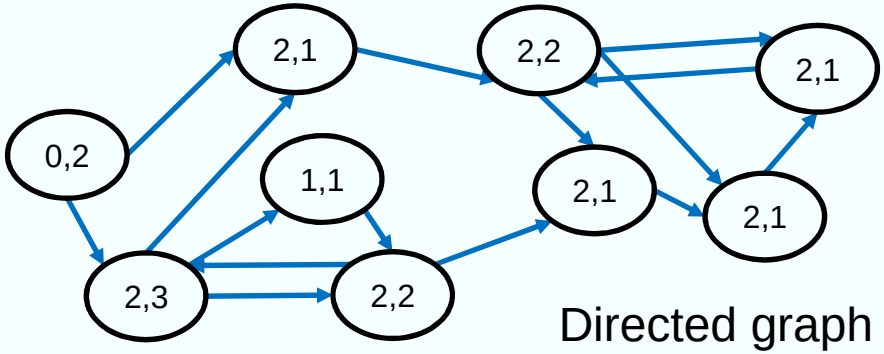
- Distances and affinities have an inverse relationship **high affinity ↔ small distance**
- A distance can be turned into an affinity by using an appropriate kernel
- Many choices of kernels. One of the most important choices in spectral clustering

Graph basics

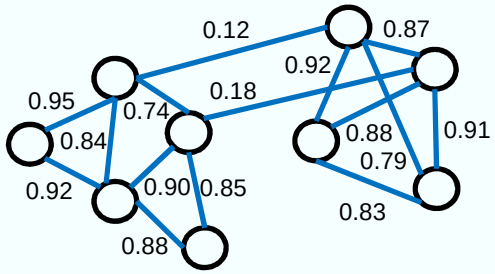
- Definition: A **graph** G is a triple consisting of a **vertex set** V(G), an **edge set** E(G) and a **relation** that associates with each edge two vertices.



Undirected graph

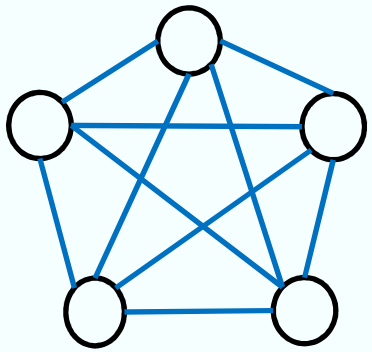


Directed graph



Weighted undirected graph

In spectral clustering we always work with undirected graphs, weighted or not



Complete graph

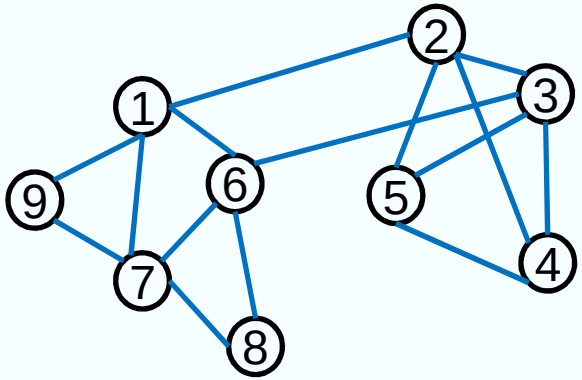
Graph basics

The Adjacency matrix W of an undirected graph

- $N \times N$ symmetric binary matrix
- rows and columns represent the vertices and entries represent the edges of the graph.
- Simple graph = **zero diagonal**

$\boldsymbol{W(i,j) = 0}$ **if** $\boldsymbol{i,j}$ **are not connected**

$\boldsymbol{W(i,j) = 1}$ **if** $\boldsymbol{i,j}$ **are connected**



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

Graph basics

The Affinity matrix A of an undirected graph

- Weighted adjacency matrix
- Each edge is weighted by pairwise vertex affinity

$$A(i,j) = 0 \text{ if } i,j \text{ are not connected}$$
$$A(i,j) = s(i,j) \text{ if } i,j \text{ are connected}$$

$s(i,j)$ is the previous kernel function

- By adjusting the kernel parameter we can set the affinity of dissimilar vertices to zero and essentially **disconnect them**
- ***A* is similar to *W***, but allows “non-binary” relations

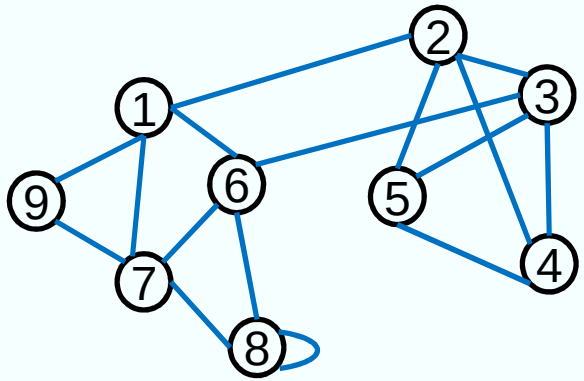
Graph basics

The Degree matrix D of an undirected graph

- $N \times N$ diagonal matrix that contains information about the degree of each vertex
- Degree $\boldsymbol{d(v_i)}$ of a vertex $\boldsymbol{v_i}$ of a graph is the number of edges incident to the vertex. Loops are counted twice

$\mathbf{D}(\boldsymbol{i},\boldsymbol{j}) = \mathbf{0}$ **if** $\boldsymbol{i} \neq \boldsymbol{j}$

$\mathbf{D}(\boldsymbol{i},\boldsymbol{j}) = \boldsymbol{d(vi)}$ **if** $\boldsymbol{i} = \boldsymbol{j} \quad \Rightarrow \quad \boldsymbol{D} = \mathbf{diag}(\boldsymbol{d_1}, \ldots, \boldsymbol{dN})$



4	0	0	0	0	0	0	0	0
0	**4**	0	0	0	0	0	0	0
0	0	**4**	0	0	0	0	0	0
0	0	0	**3**	0	0	0	0	0
0	0	0	0	**3**	0	0	0	0
0	0	0	0	0	**4**	0	0	0
0	0	0	0	0	0	**4**	0	0
0	0	0	0	0	0	0	**4**	0
0	0	0	0	0	0	0	0	**2**

Graph basics

Laplacian matrix of simple undirected graph

- $L = D - W$ (Degree – Adjacency), or
- $L = D - A$ (Degree – Affinity)
- L is symmetric and positive semi-definite

Vertex labeling

- All these matrices are symmetric
  - ON-basis of eigenvectors in R^N exists
- All these matrices depend on the labeling of the graph vertices
- Re-labeling of the vertices = permutation of the matrix rows and columns
  - Same permutation of both rows and columns!

Laplacian matrix

- The smallest eigenvalue is 0, the corresponding eigenvector is the constant one $\mathbf{1}$ (when $L = D - W$)
- *N* non-negative real-valued eigen-values

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$$

- The smallest non-zero eigenvalue of **L** is called the **spectral gap**.

The gap can be seen as a quality mesure of the clustering

Graph spectrum

- The **spectrum** os a graph *G* is the multiset of the eigenvalues of the Laplacian matrix or the graph associated with it

$$\mathrm{Spec}(G) = \begin{pmatrix} \lambda_1 \dots \lambda_t \\ m_1 \dots mt \end{pmatrix}$$

where $\lambda_1 \dots \lambda_t$ is the set of **distinct** eigenvalues and $m_1 \dots mt$ their multiplicities.

Graph spectrum

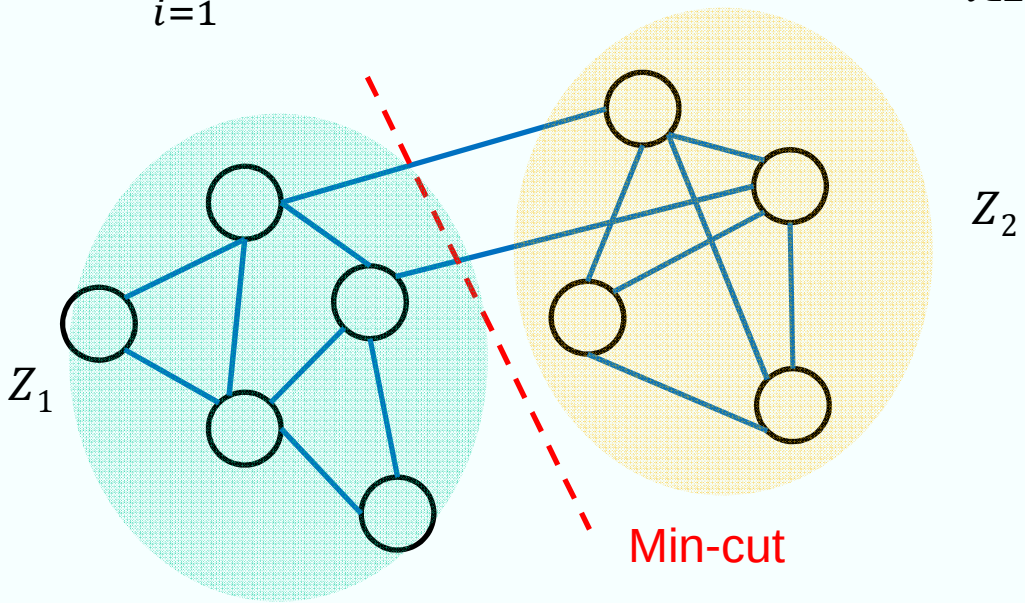
- The Laplacian matrix depends on the vertex labeling,
  - Re-labeling = row & column permutation
  - but its spectrum is **invariant**, it does not depend on the labeling
- Multiplicity of 0 eigenvalue is the number of **connected components** k of the graph (i.e. clusters)
- The corresponding eigenvectors are the **indicator vectors** $\mathbf{1}_{V_1}, \ldots, \mathbf{1}_{V_N}$ of those components

Number of clusters need not be known!?

Clustering as a graph-theoretic problem

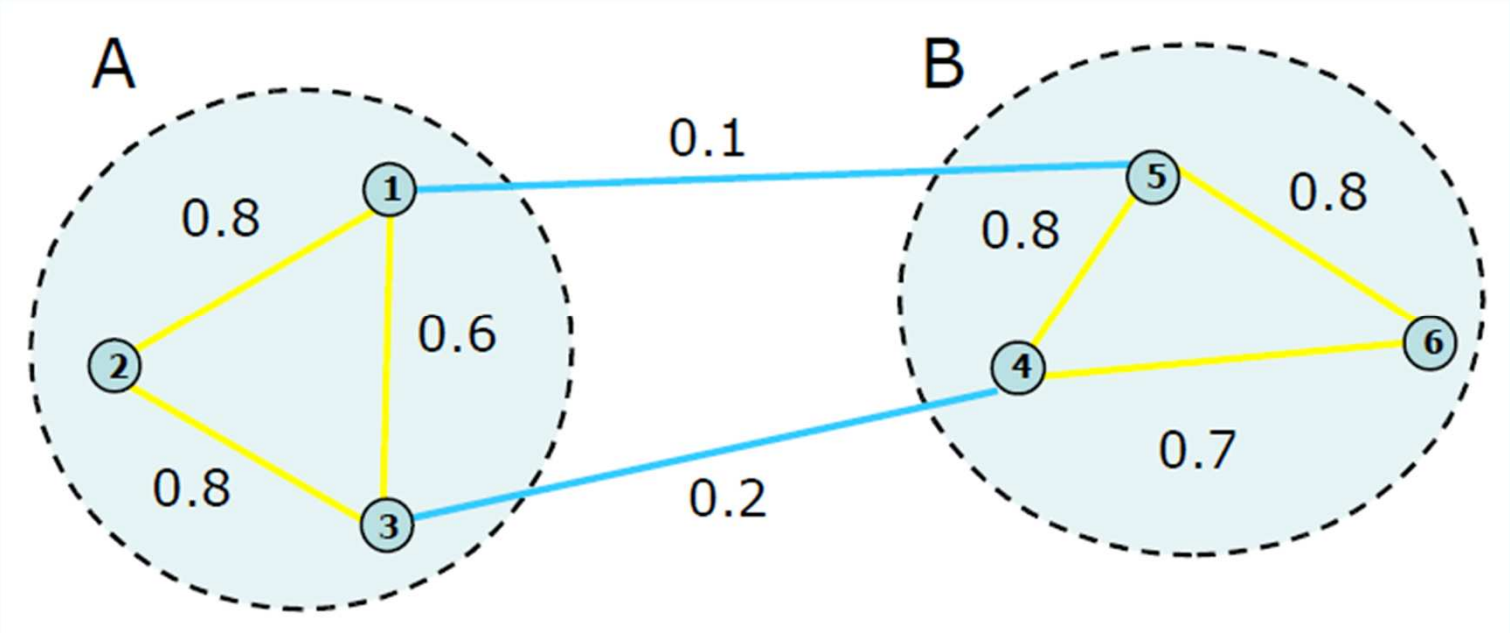
- Given a similarity graph with affinity matrix A the simplest way to construct a partition is to solve the min-cut problem:
  - Choose the partition $Z_1, \ldots, Zk$ that minimises

$$\text{cut}(Z_1, \ldots, Zk) = \frac{1}{2}\sum_{i=1}^{k} A(Zi, \overline{Z}_i) \quad \text{where } A(Z_1, Z_2) = \sum_{i \in Z_1, j \in Z_2} A(i, j)$$



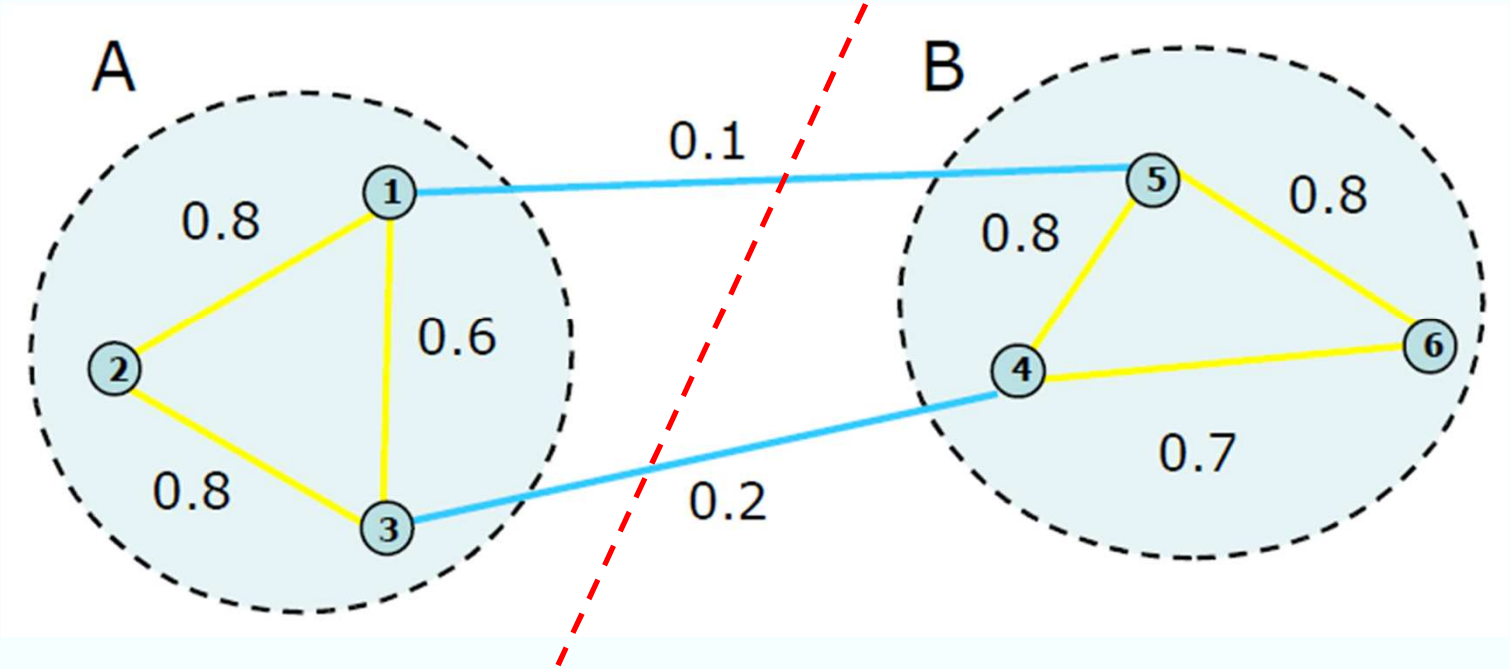
Clustering as a graph-theoretic problem – An example

- We require 2 clusters



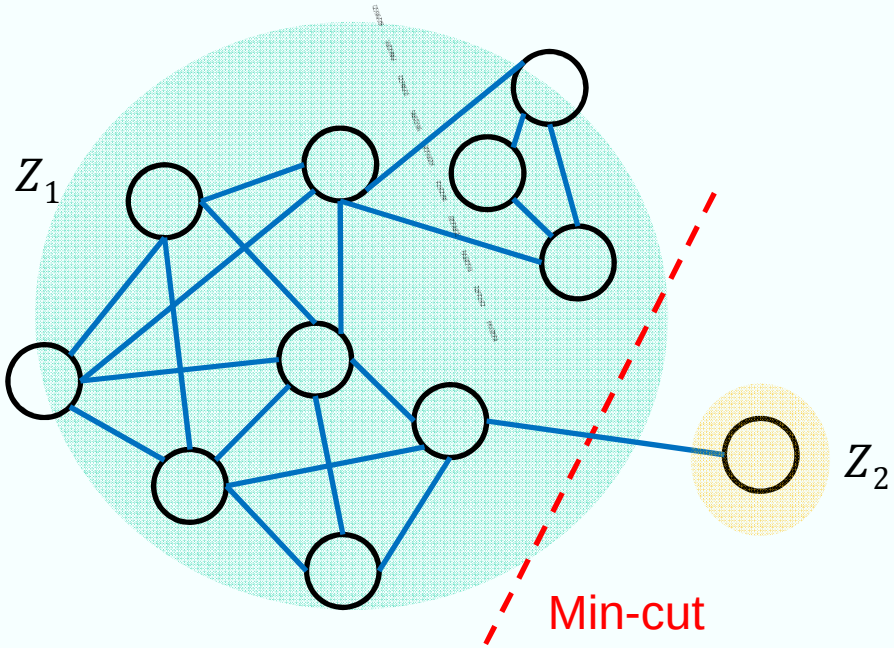
Clustering as a graph-theoretic problem – An example

- $\text{cut}(A, B) = \frac{1}{2}\sum_{i \in A, j \in B} \text{Affinity}(A, B) = 0.3$



Clustering as a graph-theoretic problem

- Min-cut can be solved efficiently especially for $k = 2$
- Does not always lead to reasonable results if the connected components are not balanced



- **Workaround:** Ensure that the partitions $Z_1, \ldots, Zk$ are sufficiently “large”
- This should lead to more balanced partitions

Clustering as a graph-theoretic problem

- Ratio-cut [Hagen and Kahng, 1992]: The size of a subset Z is measured by its number of vertices $|Z|$

$$\boldsymbol{RatioCut}(\boldsymbol{Z_1}, \dots, \boldsymbol{Z_k}) = \frac{1}{2}\sum_{i=1}^{k} \frac{\boldsymbol{A}(\boldsymbol{Zi}, \overline{\boldsymbol{Z}}_i)}{|\boldsymbol{Z_i}|} = \sum_{i=1}^{k} \frac{\mathrm{cut}(\boldsymbol{Z_i}, \overline{\boldsymbol{Z}}_i)}{|\boldsymbol{Z_i}|}$$

- Normalised cut [Shi and Malik, 2000]: The size of a subset Z is measured by the weights of its edges $\mathrm{vol}(Z)$

$$\boldsymbol{NCut}(\boldsymbol{Z_1}, \dots, \boldsymbol{Zk}) = \frac{1}{2}\sum_{i=1}^{k} \frac{\boldsymbol{A}(\boldsymbol{Zi}, \overline{\boldsymbol{Z}}_i)}{\mathrm{vol}(\boldsymbol{Zi})} = \sum_{i=1}^{k} \frac{\mathrm{cut}(\boldsymbol{Zi}, \overline{\boldsymbol{Z}}_i)}{\mathrm{vol}(\boldsymbol{Zi})}$$

- Min-max cut [Ding et al. 2001]:

$$\boldsymbol{Min - Max - Cut}(\boldsymbol{Z_1}, \dots, \boldsymbol{Zk}) = \frac{1}{2}\sum_{i=1}^{k} \frac{\boldsymbol{A}(\boldsymbol{Zi}, \overline{\boldsymbol{Z}}_i)}{\boldsymbol{A}(\boldsymbol{Z_i}, \boldsymbol{Z_i})} = \sum_{i=1}^{k} \frac{\mathrm{cut}(\boldsymbol{Zi}, \overline{\boldsymbol{Z}}_i)}{\boldsymbol{A}(\boldsymbol{Z_i}, \boldsymbol{Zi})}$$

Min similarity between

Max similarity within

Clustering as a graph-theoretic problem

- Due to the normalisations introduced the solution becomes NP-hard
- **Relaxing Ncut and Min−Max−Cut lead to normalised spectral clustering. Relaxing RatioCut leads to unormalised spectral clustering** [von Luxburg 2007]

- Relaxed **RatioCut** solution: eigenvectors

$$\boldsymbol{X} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_k}) \;\; s.t. \;\; \boldsymbol{L v_k} = \boldsymbol{\lambda_k v_k} \;\; \text{where } \boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

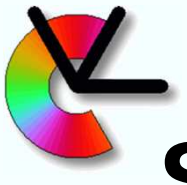
- Relaxed **Ncut** solution: eigenvectors

$$\boldsymbol{Y} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_k}) \;\; s.t. \;\; (\boldsymbol{I} - \boldsymbol{L}_{\text{sym}})\boldsymbol{u_k} = \boldsymbol{\lambda_k u_k} \;\; \text{where} \quad \boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-0.5}\boldsymbol{A}\boldsymbol{D}^{-0.5}$$

- Relaxed Min-Max-cut solution: eigenvectors

$$\boldsymbol{Y} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{uk}) \;\; s.t. \quad \boldsymbol{L}_{\text{sym}}\boldsymbol{u_k} = \boldsymbol{\lambda_k u_k} \;\; \text{where} \quad \boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-0.5}\boldsymbol{A}\boldsymbol{D}^{-0.5}$$

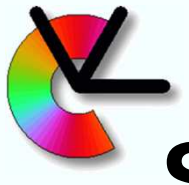
- Quality of solution with relaxation is not guaranteed compared to exact solution



Spectral clustering Method #1

[Perona and Freeman 1999]

- Partition using only one eigenvector at a time
- Use procedure recursively
  - Uses 2nd (smallest) eigenvector to define optimal cut
  - Recursively generates two clusters with each cut



Spectral clustering Method #2

[Shi and Malik 2000, Scott and Longuet-Higgins, Ng et al. 2002]

- Use the *k* smallest eigenvectors
- Directly compute *k*-way partitioning
- Usually performs better

- We will be using this approach from now on

A spectral clustering algorithm

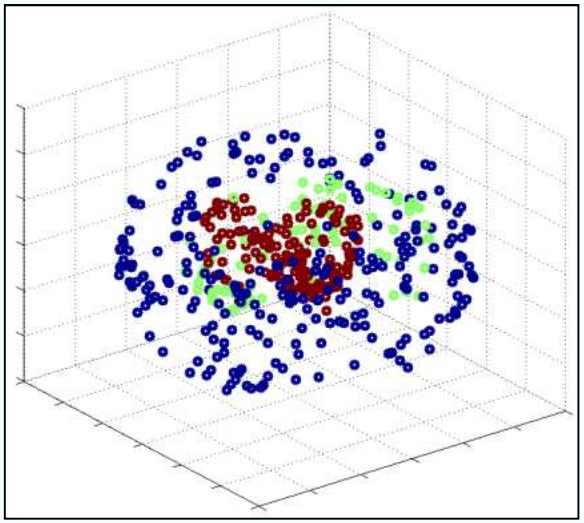
Input: Data matrix $\boldsymbol{P} \in \mathbb{R}^{N \times F}$ ($\boldsymbol{N}$ =data points, $\boldsymbol{F}$ = dimensions), $\boldsymbol{k}$ number of clusters

- Construct **pairwise** affinity matrix $\boldsymbol{A(i,j) = exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right)}$ For example!
- Construct degree **matrix** $\boldsymbol{D} = \text{diag}(\boldsymbol{d_1, \ldots, dN})$
- Compute Laplacian $\boldsymbol{L = D - A}$ $A \approx W$
- Compute the k smallest eigenvectors $\boldsymbol{u_1, \ldots, uk}$ of $\boldsymbol{L}$ *k* known !?
- Let $\boldsymbol{U} \in \mathbb{R}^{N \times k}$ contain the vectors $\boldsymbol{u_1, \ldots, u_k}$ as columns
- Let $\boldsymbol{y_i} \in \mathbb{R}^{k}$ be the vector corresponding to the ***i***-th row of $\boldsymbol{U}$ One $\mathbf{y}_i$ per point
- Cluster the points $(\boldsymbol{y_i})_{\,i = 1, \ldots, N}$ into $\boldsymbol{k}$ clusters $\boldsymbol{h_1, \ldots, hk}$ with *k*-means

Output: Clusters $\boldsymbol{Z_1, \ldots Zk}$ with $\boldsymbol{Z_i = \{j | yj \in h_i\}}$

Why not just use *k*-means?

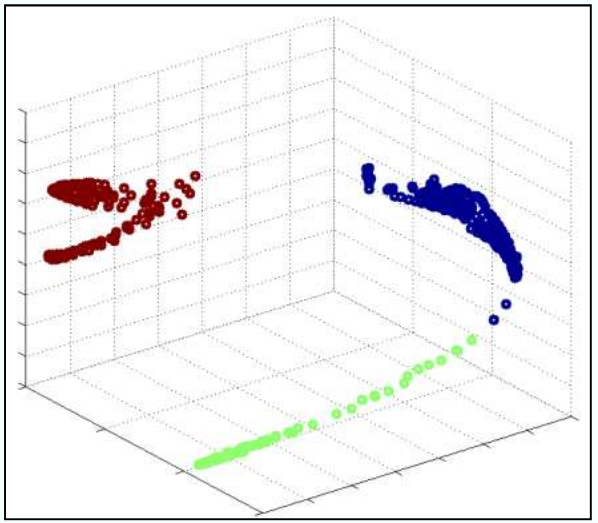
- One could use *k*-means directly in the data space (or some other clustering approach such as mean shift)
- S.C. separates data (based on affinity) into projecting in the low-dimensional eigenspace (rows of **U**)
- Allows clustering of non-convex data



Before spectral clustering

Original data space

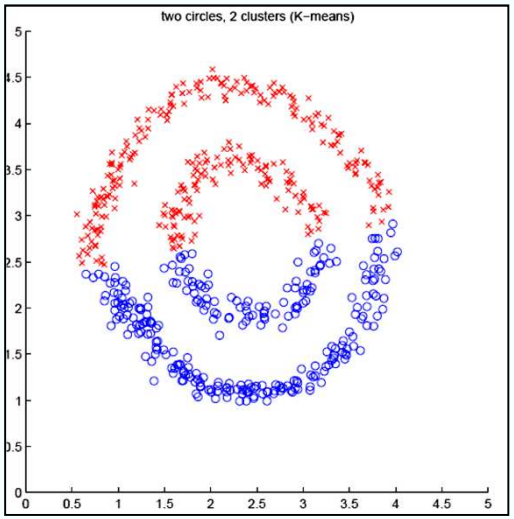
Space spanned by the columns of **U**



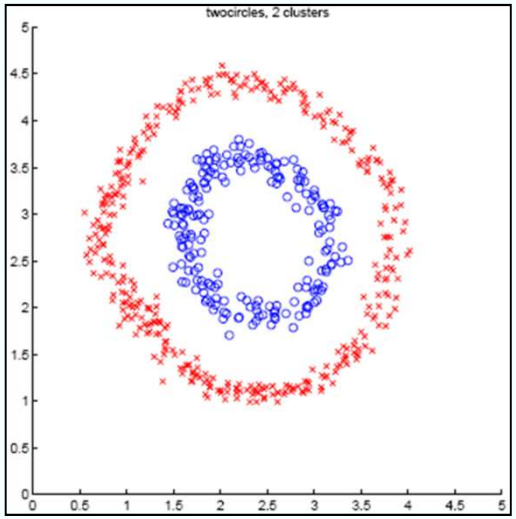
After spectral clustering

Why not just use K-means?

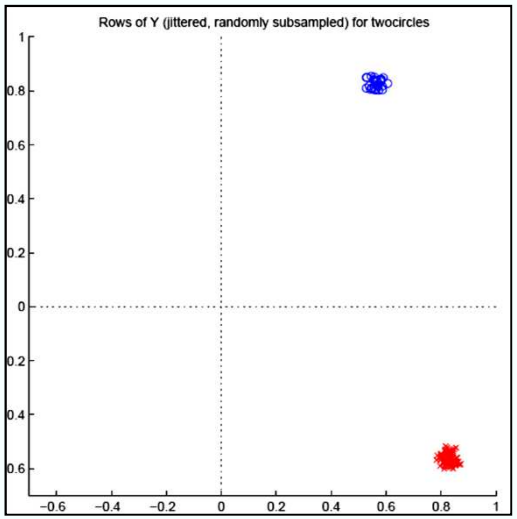
K-means



Spectral clustering

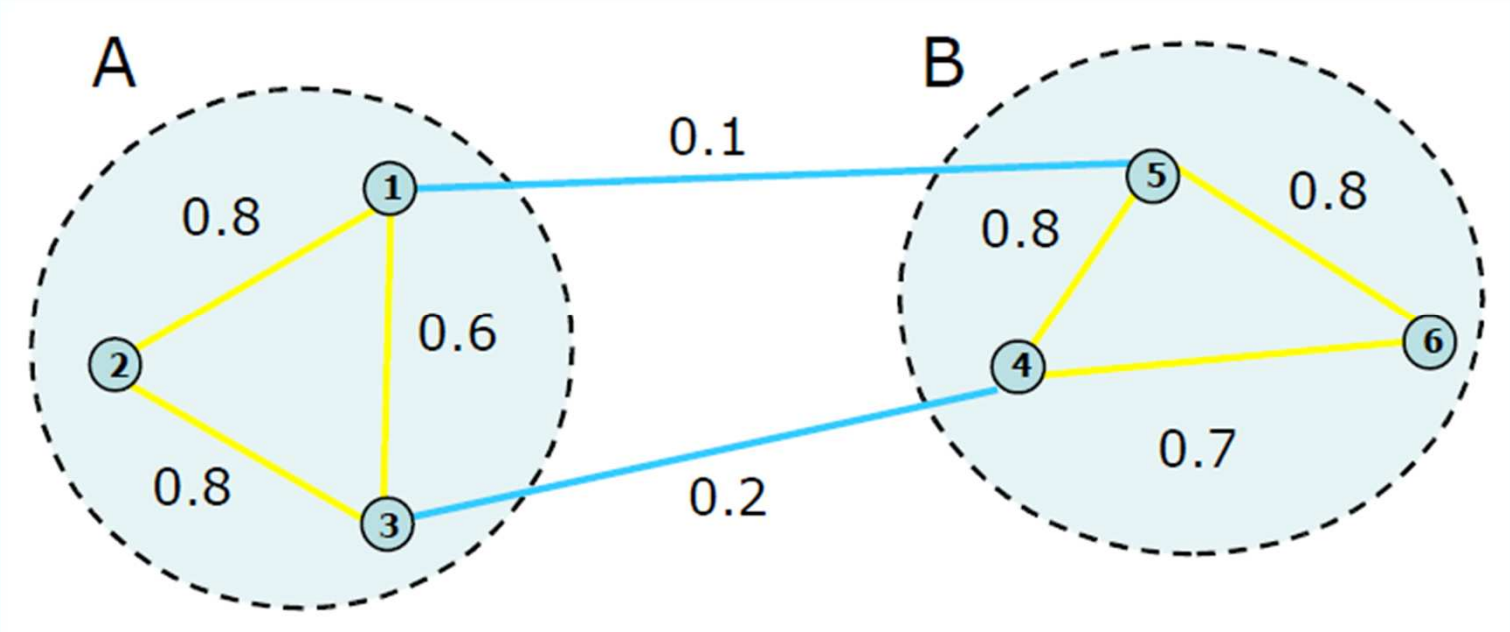


We do K-means here instead

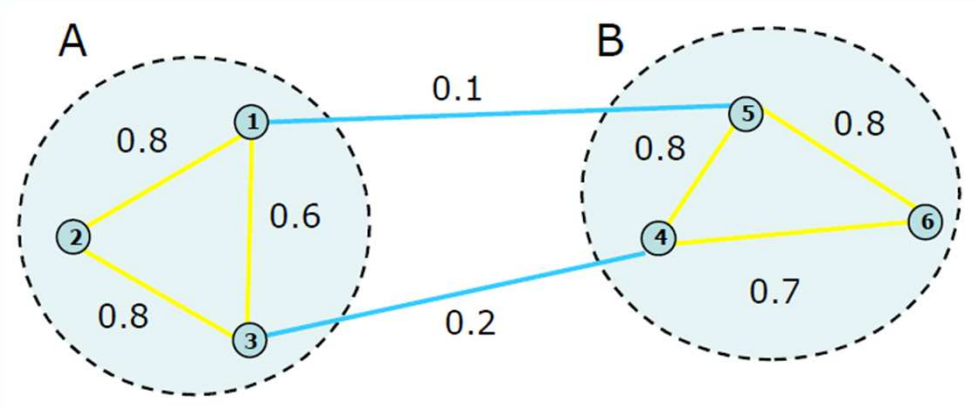


Simple example revisited

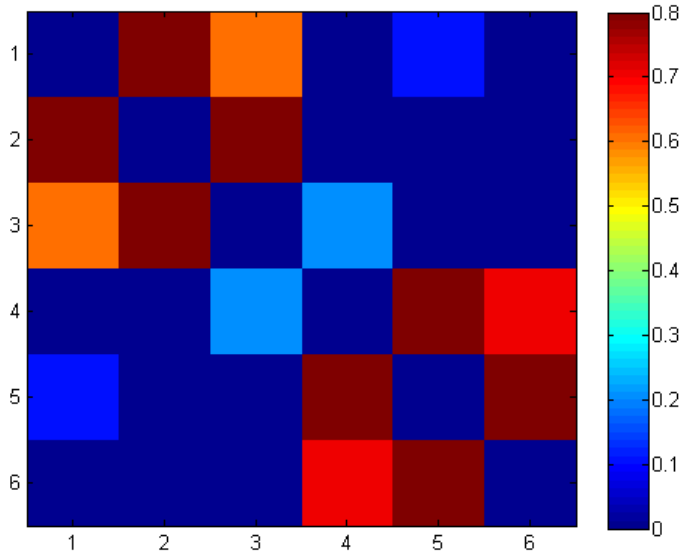
- Now we will use spectral clustering instead



Step 1: Pairwise affinity matrix

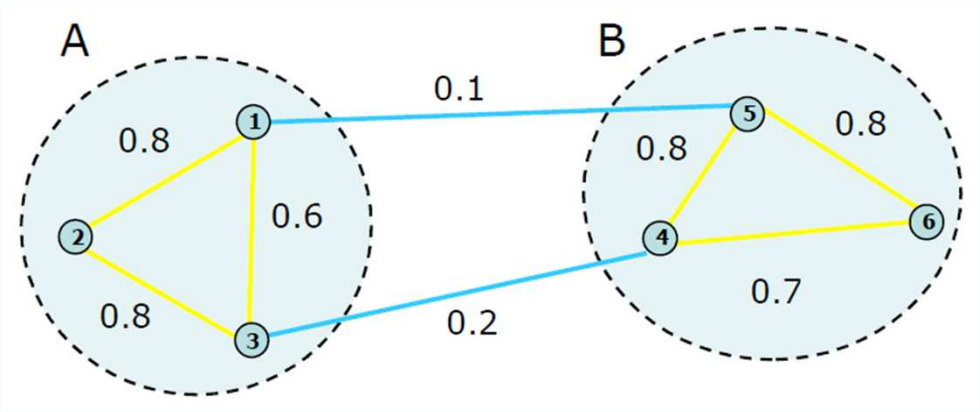


	X_1	X_2	X_3	X_4	X_5	X_6
X_1	0	0.8	0.6	0	0.1	0
X_2	0.8	0	0.8	0	0	0
X_3	0.6	0.8	0	0.2	0	0
X_4	0	0	0.2	0	0.8	0.7
X_5	0.1	0	0	0.8	0	0.8
X_6	0	0	0	0.7	0.8	0

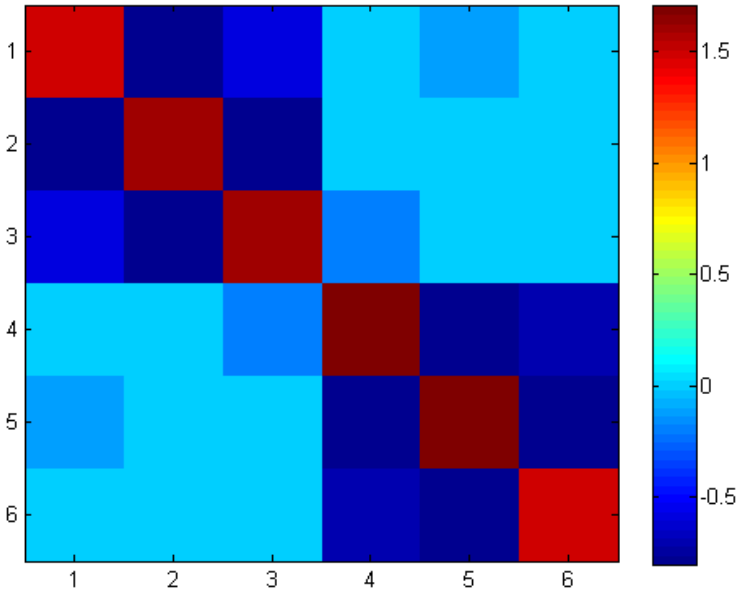


Step 2: Laplacian matrix

$$L = D - A$$



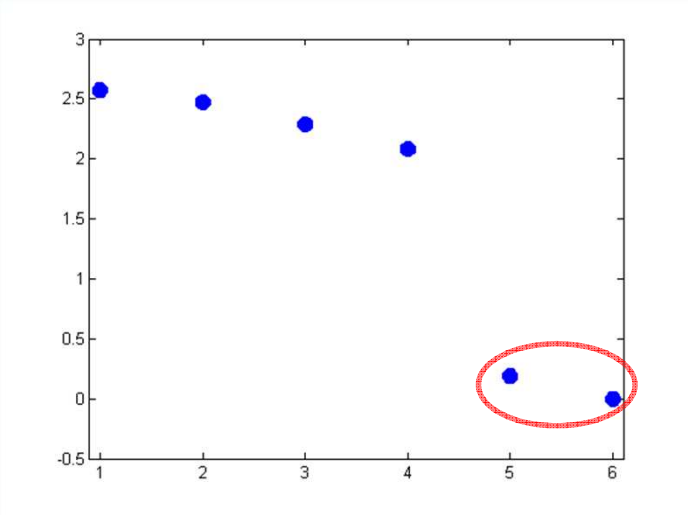
	X_1	X_2	X_3	X_4	X_5	X_6
X_1	1.5	-0.8	-0.6	0	-0.1	0
X_2	-0.8	1.6	-0.8	0	0	0
X_3	-0.6	-0.8	1.6	-0.2	0	0
X_4	0	0	-0.2	1.7	-0.8	-0.7
X_5	-0.1	0	0	-0.8	1.7	-0.8
X_6	0	0	0	-0.7	-0.8	1.5



Step 3: Eigen-decomposition

- Eigen-values λ=

0
0.18
2.08
2.28
2.46
2.57



- Eigen-vectors v =

U

$N \times k$

-0.4082	0.4084	...
-0.4082	0.4418	...
-0.4082	0.3713	...
-0.4082	-0.3713	...
-0.4082	-0.4050	...
-0.4082	-0.4452	...

Step 4: Embedding

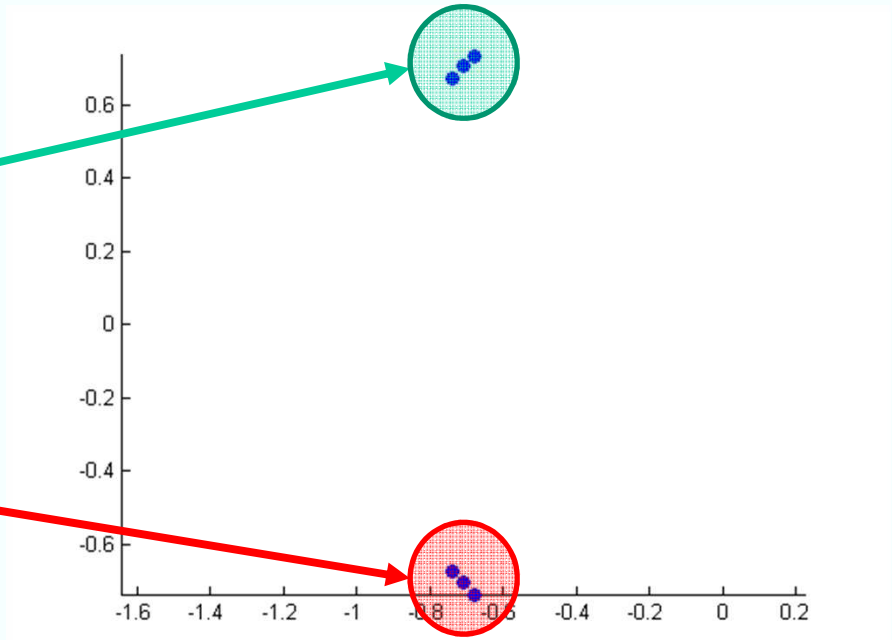
- $U=$

-0.4082	0.4084
-0.4082	0.4418
-0.4082	0.3713
-0.4082	-0.3713
-0.4082	-0.4050
-0.4082	-0.4452

- Each row of Y is a point in ”eigenspace”

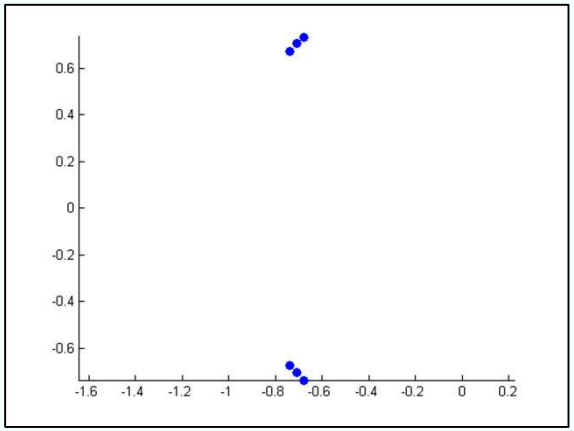
- $Y = \text{row\_normalise}(U)$

-0.7070	0.7072
-0.6786	0.7345
-0.7398	0.6729
-0.7398	-0.6729
-0.7099	-0.7043
-0.6759	-0.7370

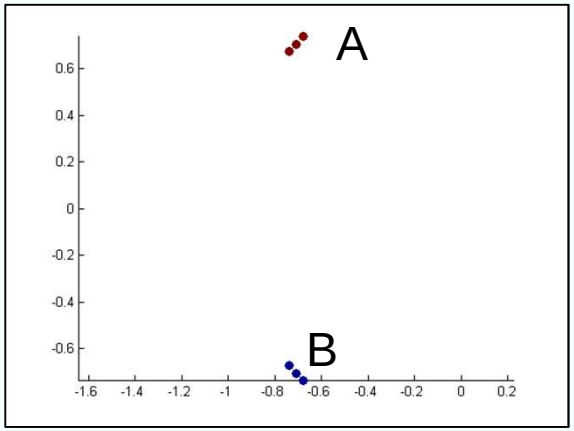


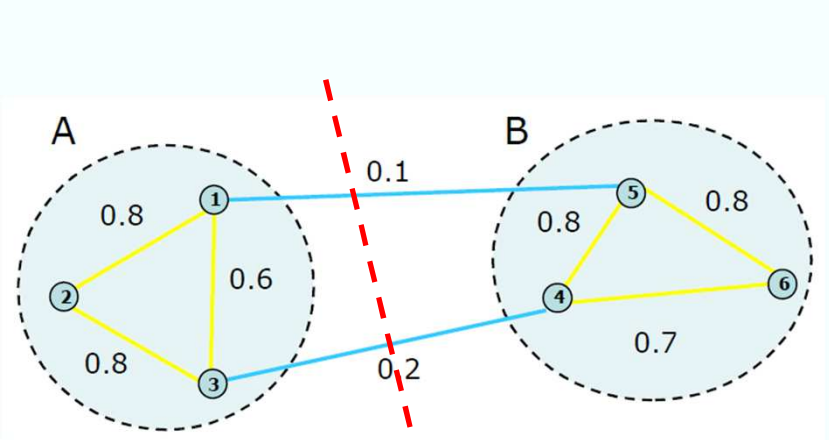
Step 5: Clustering

- *k*-means clustering with 2 clusters
- Easy, convex clustering problem



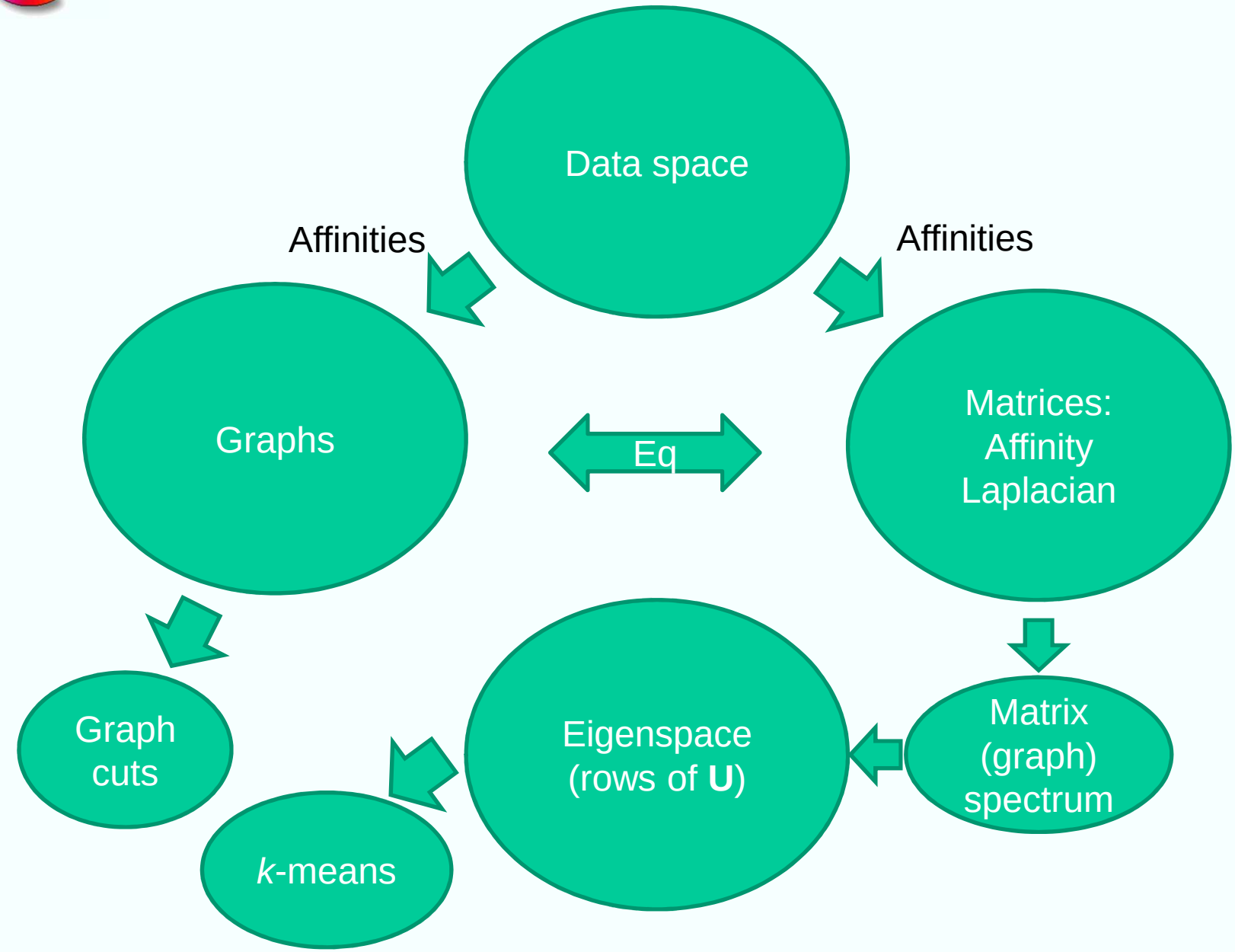
k-means





Choices choices…

- Affinity matrix construction (distance and kernel)
- Choice of kernel parameter σ (scaling factor)
  - Practically, search over σ and pick value that gives the tightest clusters
- Choice of k, the number of clusters
- Choice of clustering method


Data space
Affinities
Affinities
Graphs
Eq
Matrices:
Affinity
Laplacian
Graph
cuts
Eigenspace
(rows of **U**)
Matrix
(graph)
spectrum
k-means

Summary

- **We have seen so far**
  - Basic definitions of cluster, clustering and cluster quality
  - Graph basics, affinity, graph construction, graph spectrum
  - Graph cuts
  - Spectral clustering and graph cuts
  - A spectral clustering algorithm and a simple example
  - *k*-means and spectral clustering

- **For the next lecture**
  - Intuitive explanation of different S.C. algorithms