Introduction to spectral clustering

Vasileios Zografos
zografos@isy.liu.se

Klas Nordberg
klas@isy.liu.se

What this course is

- Basic introduction into the core ideas of spectral clustering
- Sufficient to get a basic understanding of how the method works
- Application mainly to computer vision
- In the end you should be:
  - Able to implement and tune S.C.
  - Make design choices for particular problems

What this course is not

- Not a course in graph theory
  - Many connections and proofs from spectral graph theory are not here. [Look at F. Chung, Spectral graph theory]
- Not covering advanced features and applications of SC
- Connection to other methods is not covered in detail. [Look at website and papers by Chris Ding]
- Only looking at undirected simple graphs

Course contents

- 3 lectures
  - Lecture 1: Basic concepts, graph cuts, a S.C. algorithm,
  - Lecture 2: The mechanics of S.C., different S.C. algorithms
  - Lecture 3: Applications of S.C., extensions and enhancements, practical issues
- 1 coursework
  - Simple spectral clustering problem (data provided)
  - Our your own problem

Course contents – Part 1

1. Overview of clustering
2. Properties of a cluster
3. Basic graph theory
4. Graph cuts and clustering
5. Introduction to spectral clustering
6. A simple spectral clustering example

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 eigenvectors of matrices derived from the data.
  - Data mapped to a low-dimensional space that are separated and can be easily clustered.

Pros and cons of spectral clustering

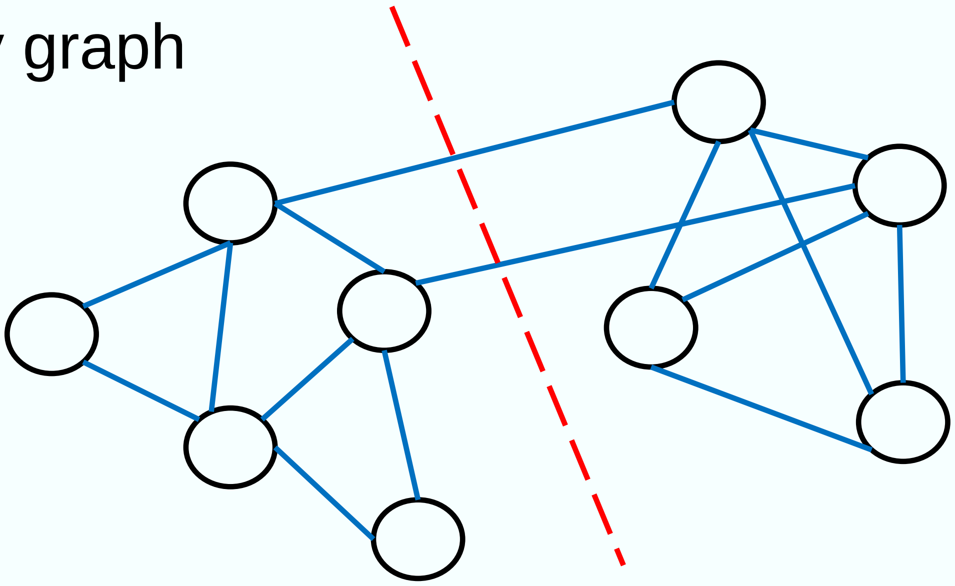
- Advantages:
  - Does not make strong assumptions on the statistics of the clusters
  - Easy to implement.
  - Good clustering results.
  - Reasonably fast for sparse data sets of several thousand elements.

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

Spectral clustering in one slide

Graph theoretic point of view

- Given data points $x_1, \dots x_N$, pairwise affinities $\mathrm{A}_{ij} = A(x_i, x_j)$
- Build similarity graph

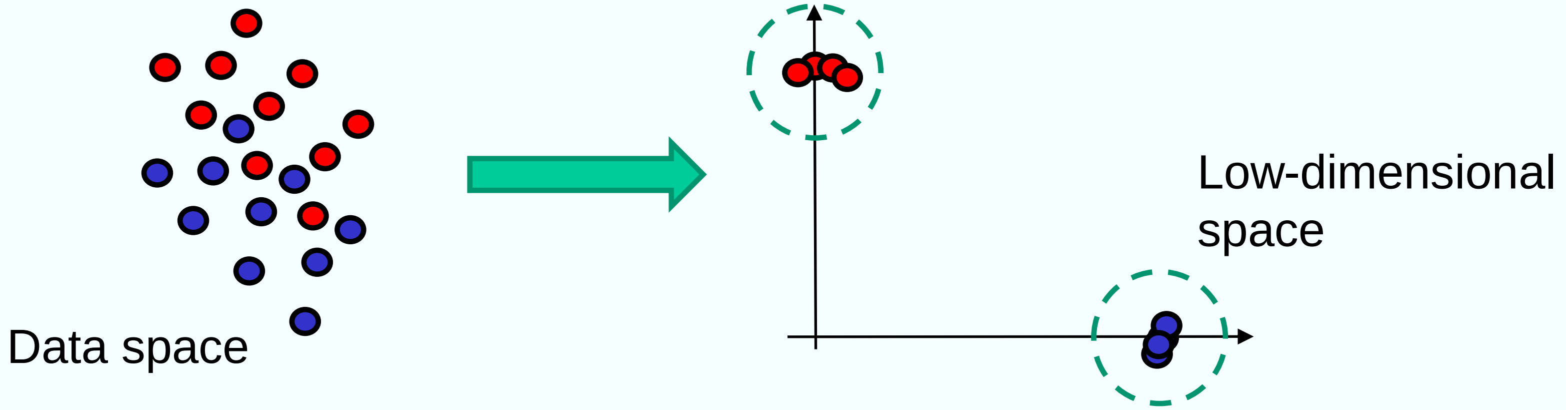


- Clustering = find a cut through the graph
  - Define a cut-type objective function
  - Solve it

Spectral clustering in one slide

Low-dimensional embedding point of view

- Given data points $x_1, \ldots x_N$, pairwise affinities $\mathrm{A}_{ij} = A(x_i, x_j)$
- Find a low-dimensional embedding
- Project data points to new space



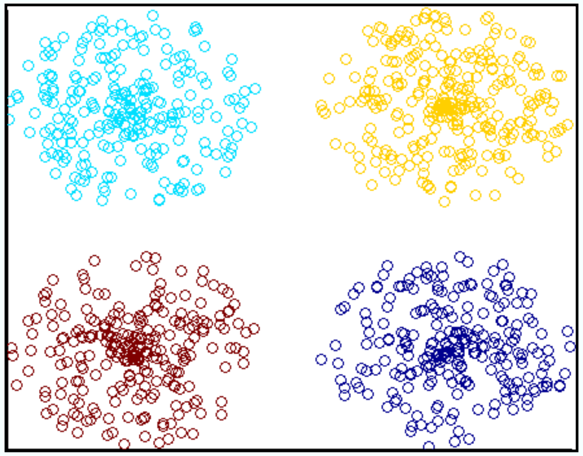
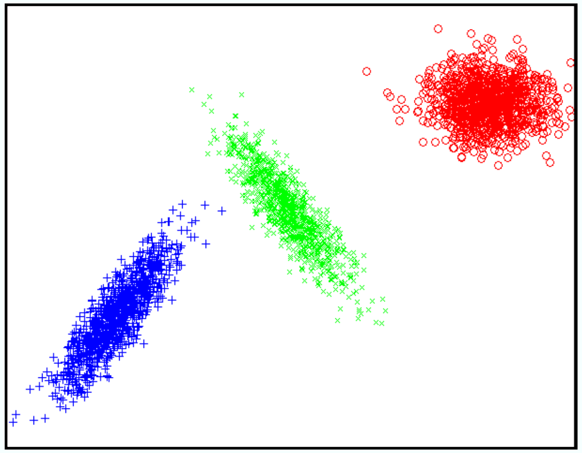
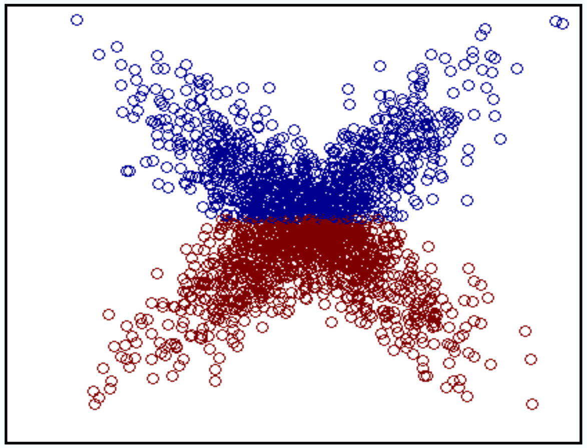
- Cluster using favourite clustering algorithm

Spectral clustering in one slide

- Both points of view 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 k-eigen-vectors
- Projection and clustering equates to graph partition by different min-cut criteria

Overview of clustering

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

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 *disimilar* (inter-cluster)

Overview of clustering

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

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
- …

Some definitions

- **Clustering function** f for some domain X, is a function that takes a distance d over X and outputs a clustering C of X

- **Clustering quality measure** is a function m that given a clustering C over (X, d) returns a non-negative real number

What is a good clustering

- Kleinberg's axioms for clustering functions f :
  - **Scale invariance**: The output of a clustering function should be invariant to uniform scaling of the input

$$\boldsymbol{f}(\boldsymbol{d}(\boldsymbol{x},\boldsymbol{y})) = \boldsymbol{f}(\boldsymbol{\lambda d}(\boldsymbol{x},\boldsymbol{y}))$$

  - **Consistency**: If intra-cluster distances are decreased and inter-cluster distances are increased then the clustering output should not change

If $\boldsymbol{f}(\boldsymbol{d}) = \boldsymbol{C}$ and $\boldsymbol{d}'$ is a $\mathbf{C} -$ enhancing transformation of $\boldsymbol{d}$, then $\boldsymbol{f}(\boldsymbol{d}') = \boldsymbol{C}$

$\boldsymbol{d}'$ is a $\boldsymbol{C} -$ enhacning transformation of $\boldsymbol{d}$ if

$$\boldsymbol{d}'(\boldsymbol{x},\boldsymbol{y}) \leq \boldsymbol{d}(\boldsymbol{x},\boldsymbol{y}), \text{for } \boldsymbol{x},\boldsymbol{y} \in \boldsymbol{C} \quad \textbf{and}$$
$$\boldsymbol{d}'(\boldsymbol{x},\boldsymbol{y}) \geq \boldsymbol{d}(\boldsymbol{x},\boldsymbol{y}), \text{for } \boldsymbol{x},\boldsymbol{y} \notin \boldsymbol{C}$$

  - **Richness**: By modifying the distance function, any partition of the underlying data can be obtained

$$\forall \text{ partition } \boldsymbol{C} \text{ of } \boldsymbol{X}, \text{there exists } \boldsymbol{d} \text{ over } \boldsymbol{X}\ s.t.\ \boldsymbol{f}(\boldsymbol{d}) = \boldsymbol{C}$$

What kinds of algorithms satisfy the axioms?

- Single linkage till you get k clusters.
  - satisfies ***scale invariance*** and ***consistency***, but not ***richness***

- Single linkage till distances exceed $\tau \max_{ij} d(x, y)$, where τ is some constant.
  - satisfies ***scale invariance*** and ***richness*** but not ***consistency***

- Single linkage until distances exceed some threshold r.
  - satisfies ***richness*** and ***consistency*** but not ***scale invariance***

What is a good clustering

- Kleinberg’s impossibility theorem

“There exists no clustering function that simultaneously satisfies ***scale invariance****,* ***consistency*** *and* ***richness****”*

- Instead of defining clustering functions, we focus on the *quality* of a given clustering

Properties of a good cluster

- Clustering quality measure $m(C, d) \in R$
  - Scale invariance

    $$\boldsymbol{m}(\boldsymbol{C}, \boldsymbol{d}) = \boldsymbol{m}(\boldsymbol{C}, \boldsymbol{\lambda d}), \forall \boldsymbol{d} \text{ and } \boldsymbol{\lambda} > \mathbf{0}$$

  - Consistency

    $$\text{If } \boldsymbol{d}' \text{ is } a \ \mathbf{C} - \text{enhancing transformation of } \boldsymbol{d}$$
    $$\text{then } \boldsymbol{m}(\boldsymbol{C}, \boldsymbol{d}) \leq \boldsymbol{m}(\boldsymbol{C}, \boldsymbol{d}')$$

  - Richness

    $$\forall \ \boldsymbol{C} \ \exists \ \boldsymbol{d} \ s.t. \boldsymbol{C} = \mathbf{arg\,max}_{C} \, \boldsymbol{m}(\boldsymbol{C}, \boldsymbol{d})$$

  - Isomorphic invariance

    $$\text{If } \boldsymbol{C} \approx {}_{\boldsymbol{d}}\boldsymbol{C}' \text{ then } \boldsymbol{m}(\boldsymbol{C}, \boldsymbol{d}) = \boldsymbol{m}(\boldsymbol{C}', \boldsymbol{d}')$$

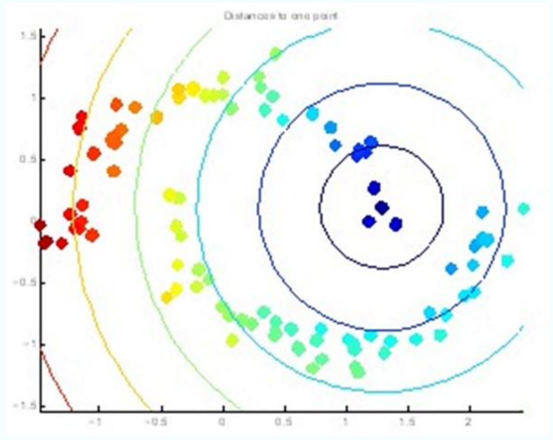
Quality measures

- Relative margin (Ackerman and Ben David, 2008)
- C-index (Dalrymple and Alford, 1970)
- Gamma (Baker and Hubert, 1975)
- D-index (Dalrymple and Alford, 1970)
- Dunn’s index (Dunn, 1973)
- Distortion (Lloyd, 1957)
- Silhouette (Kaufman and Rousseeuw, 1990)
- Davies-Bouldin (Davies and Bouldin, 1979)
- Calinski-Harabasz (Calinski and Harabasz 1974)
- Hartigan (Hartigan, 1975)
- Krzanowski-Lai (Krzanowski and Lai, 1985)
- …

- Quality measures will be revisited in the 3rd lecture

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



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

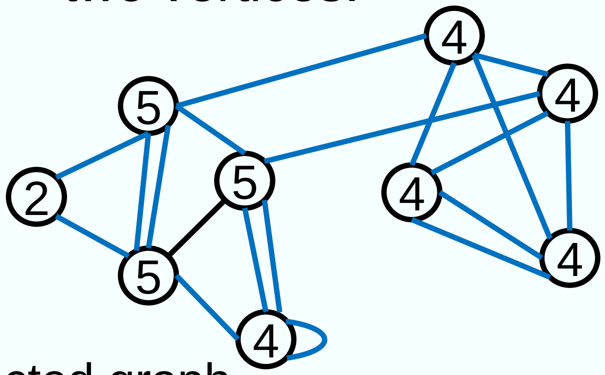
Affinity and distance

- An **affinity score** between two objects x_i, x_j is “high” if the objects are “very similar”
  - E.g. the Gaussian kernel $s(i,j) = \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right)$
- A **distance score** between two objects x, y is “small” if the objects are “close” to each other
  - E.g. the Euclidean distance $d(i,j) = \|x_i - x_j\|$

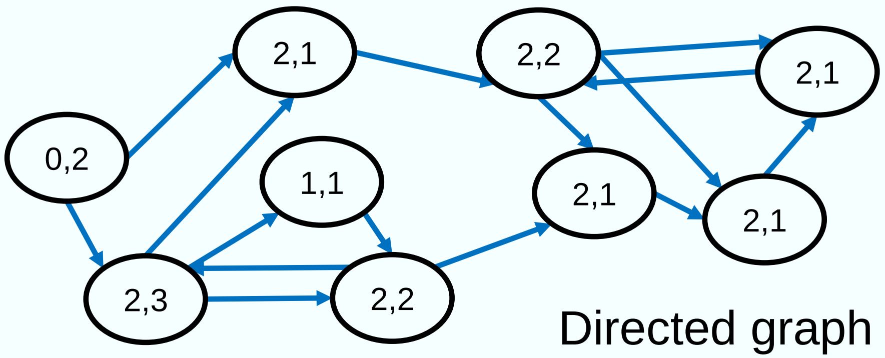
- Distances and affinities have an inverse relationship **high affinity** ↔ **low distance**
- A distance can be turned into an affinity by using an appropriate kernel
- Many choices of kernels. One of the most important choices in spectral clustering

Graph basics

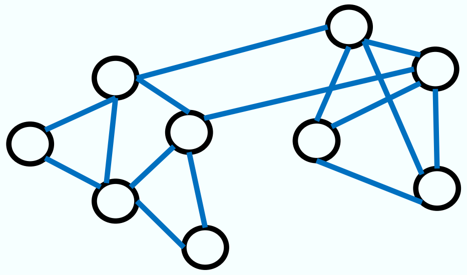
- Definition: A **graph** G is a triple consisting of a **vertex set** V(G), an **edge set** E(G) and a **relation** that associates with each edge two vertices.



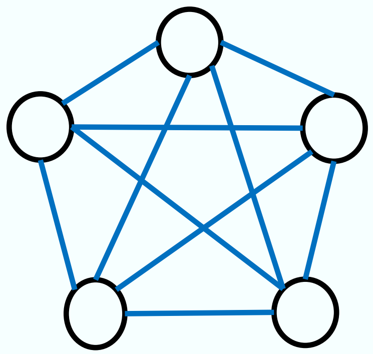
Undirected graph



Directed graph


Simple undirected graph

In spectral clustering we always work with undirected graphs


Complete graph

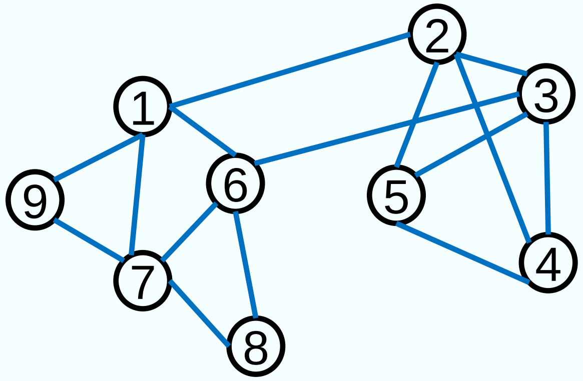
Graph basics

Adjacency matrix W of undirected graph

- NxN 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

Affinity matrix A of undirected graph

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

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

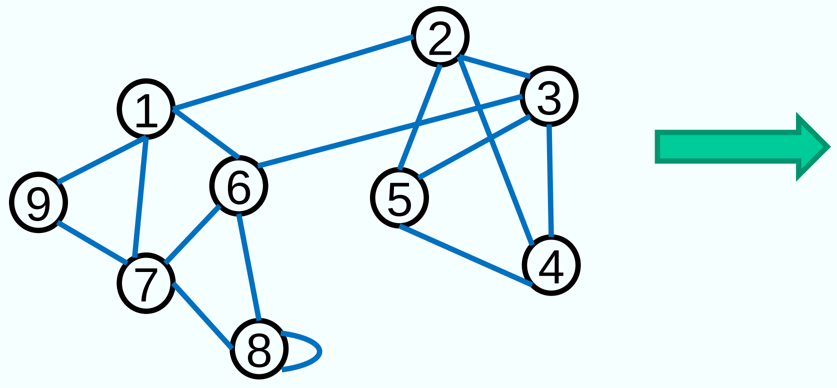
- By adjusting the kernel parameter we can set the affinity of dissimilar vertices to zero and essentially disconnect them

Graph basics

Degree matrix D of undirected graph

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

$$\mathbf{D}(\boldsymbol{i},\boldsymbol{j}) = \mathbf{0} \text{ if } \boldsymbol{i} \neq \boldsymbol{j}$$
$$\mathbf{D}(\boldsymbol{i},\boldsymbol{j}) = \boldsymbol{d}(\boldsymbol{v}_i) \text{ if } \boldsymbol{i} = \boldsymbol{j} \quad \Rightarrow \quad \boldsymbol{D} = \mathbf{diag}(\boldsymbol{d}_1, \dots, \boldsymbol{d}_N)$$



4	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	**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

- $L = D - A$ (Degree – Affinity) *(Unnormalised)*
- L is symmetric and positive semi-definite
- The smallest eigen-value is 0, the corresponding eigen-vector is the constant one $\mathbf{1}$
- 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**.
- Laplacian has a complete set of orthonormal eigen−vectors

Graph construction

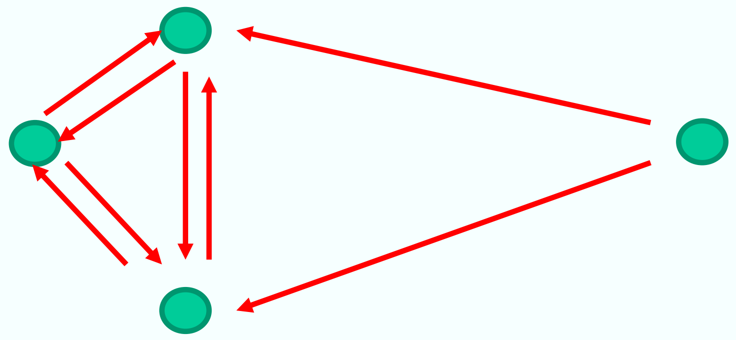
- There are different ways to construct a graph representing the relationships between data points :
  - **Fully connected graph**: All vertices having non-null similarities are connected each other
  - **r-neighbourhood graph**: Each vertex is connected to vertices falling inside a ball of radius r where r is a real value that has to be tuned in order to catch the local structure of data.
  - **k-nearest neighbour graph**: Each vertex is connected to its k-nearest neighbours where k is an integer number which controls the local relationships of data.
- Different graph constructs reprensent different local-structure of the data

Graph construction – Examples

k-nearest neighbour graphs

- Given data points and their pairwise affinities $A(i,j)$
- Connect each point to its k-nearest neighbours
- Weigh the edges by the affinity score

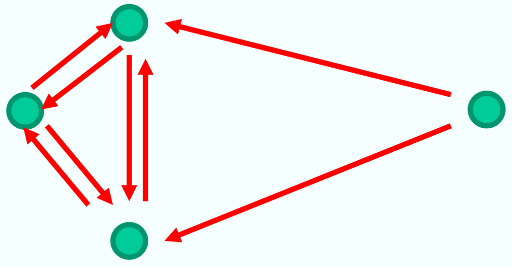
- Generally graph is **directed** and **non-symmetric** (neighbourhood relationship is not symmetric)
- Example 2-nearest neighbours

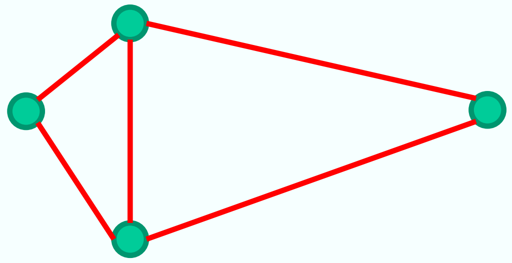


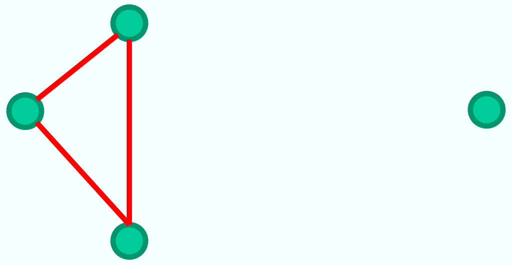
Graph construction – Examples

Undirected k-nearest neighbour graphs

- Make a directed graph to an undirected using ”AND” or ”OR” operations
- The **symmetric** kNN graph connects A with B if $A \to B$ or $B \to A$
- The **mutual** kNN graph connects A with B if $A \to B$ and $B \to A$

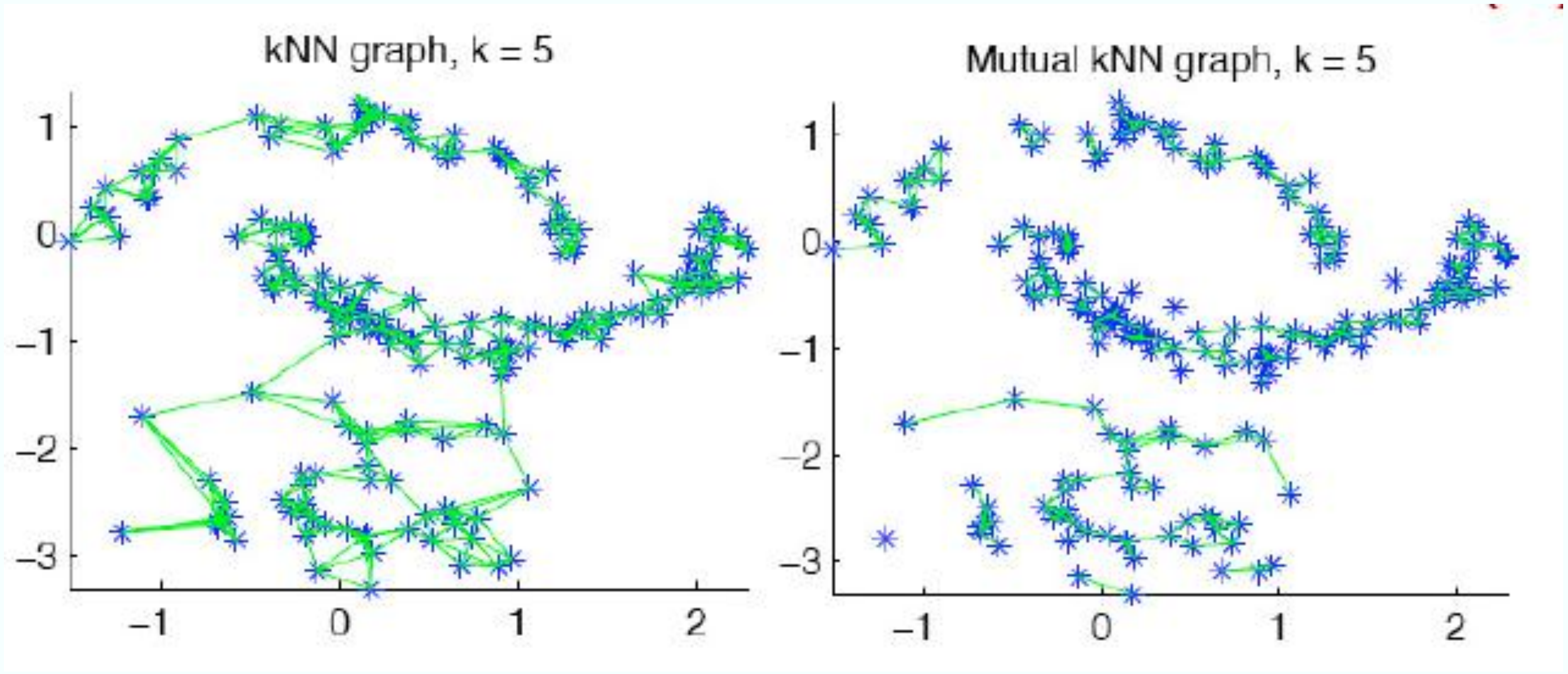

kNN graph


Symmetric kNN graph


Mutual kNN graph

Graph construction – Examples

Undirected k-nearest neighbour graphs



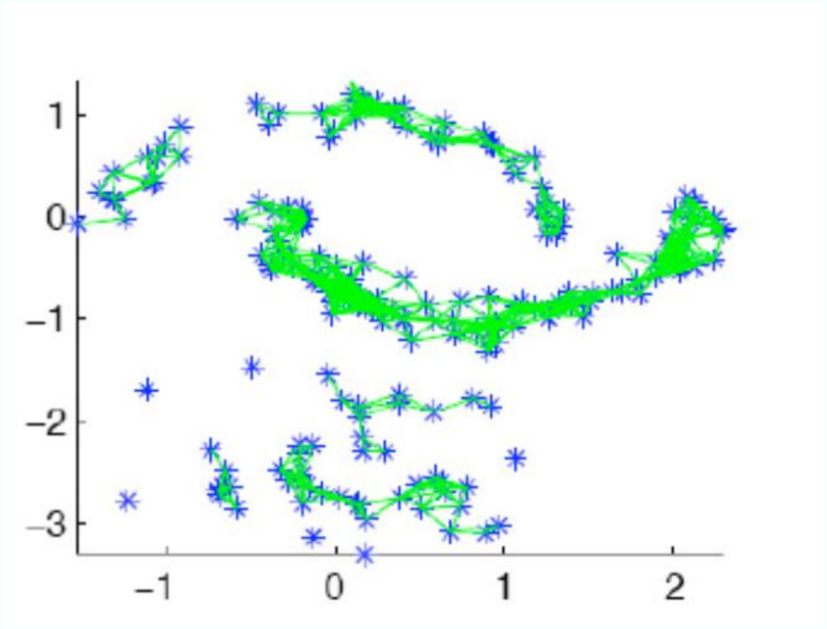
Symmetric kNN Mutual kNN

- The mutual kNN graph is a subset of the symmetric kNN

Graph construction – Examples

r-neighbourhood graph

- Given data points and their pairwise affinities $A(i,j)$
- Connect each point to all other points that have affinity above a threshold r
- Weigh the edges by the affinity score or use unweighted graph



Graph spectrum

- Spectrum is the multiset of the eigen−values of the Laplacian matrix or the graph associated with it

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

where $\lambda_1 \ldots \lambda_t$ is the set of **distinct** eigen−values and $m_1 \ldots mt$ their multiplicities.

- Laplacian matrix depends on the vertex labelling, its spectrum is invariant (i.e. does not depend on the representation)
- Multiplicity of 0 eigen-value is the number of **connected components** k of the graph (i.e. clusters)
- The eigen-space is spanned by the indicator vectors $\mathbf{1}_{V_1}, \ldots, \mathbf{1}_{V_N}$ of those components

Clustering as a graph-theoretic problem

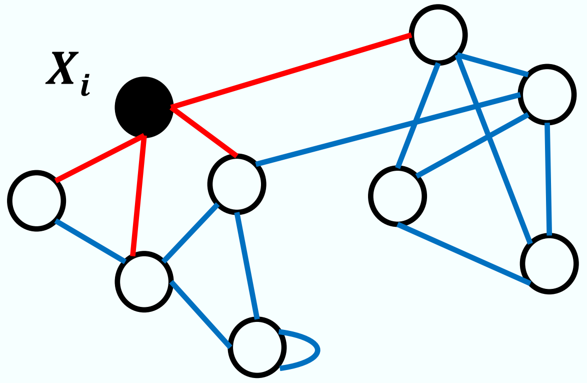
- G a graph with vertex set $\boldsymbol{V} = \{\boldsymbol{v_1}, \dots, \boldsymbol{v_N}\}$
- Subset $\boldsymbol{Z} \subset \boldsymbol{V}$
- $\boldsymbol{A}(\boldsymbol{Z_i}, \boldsymbol{Z_j}) = \sum_{\boldsymbol{i} \in \boldsymbol{Z_i}, \boldsymbol{j} \in \boldsymbol{Z_j},} \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ for $\boldsymbol{Z_i}, \boldsymbol{Z_j} \subset \boldsymbol{V}$
- $|\boldsymbol{Z}|$: number of vertices in Z
- $\boldsymbol{vol}(\boldsymbol{Z}) = \sum_{\boldsymbol{i} \in \boldsymbol{Z}} \boldsymbol{D_i}$: volume of Z
  - i.e. sum of the weights of all edges attached to vertices in Z
- All vertices that can be reached from each other by a path form a ***connected component*** (i.e. no connections between Z and $\bar{Z}$. $\bar{Z}$ is the complement of Z)
- The non-empty sets $Z_1, \dots, Z_k$ form a ***partition*** of the graph if $Z_i \cap Z_j = \emptyset$ and $Z_1 \cup \cdots \cup Z_k = V$

Node volume vs Set (cluster) volume

- Node volume

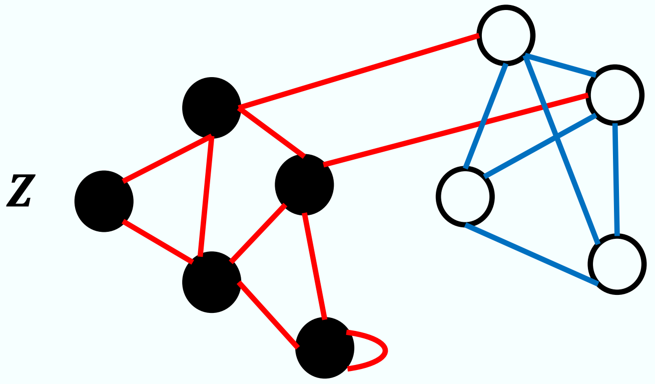
$$D_i = \sum_{j=1}^{N} A(i,j)$$

i.e. weighted sum of all of the edges connected to the node



- Set (cluster) volume

$$vol(Z) = \sum_{i \in Z} D_i$$

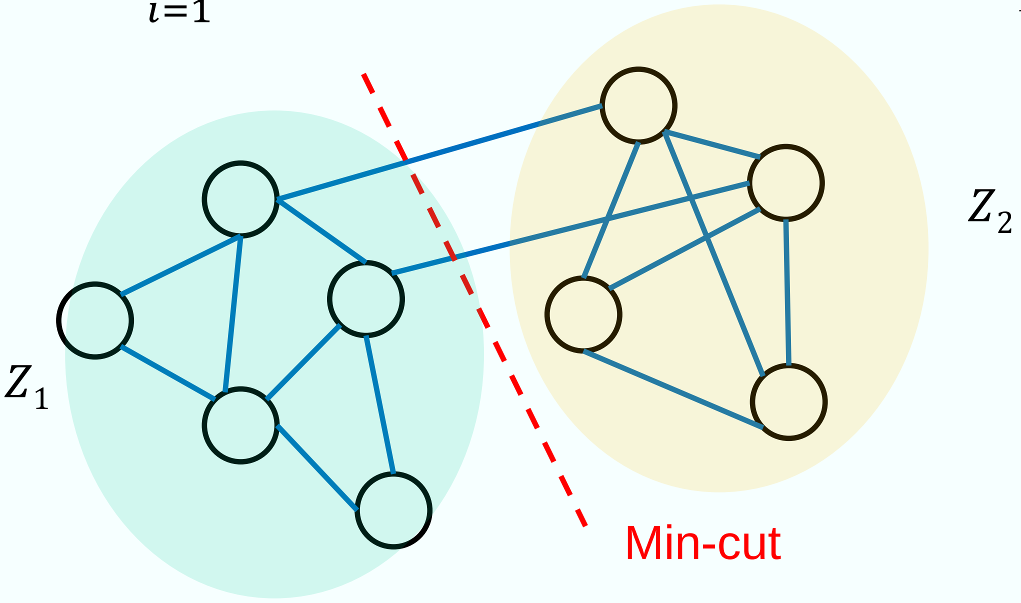


i.e. sum of the weights of all edges attached to vertices in Z

Clustering as a graph-theoretic problem

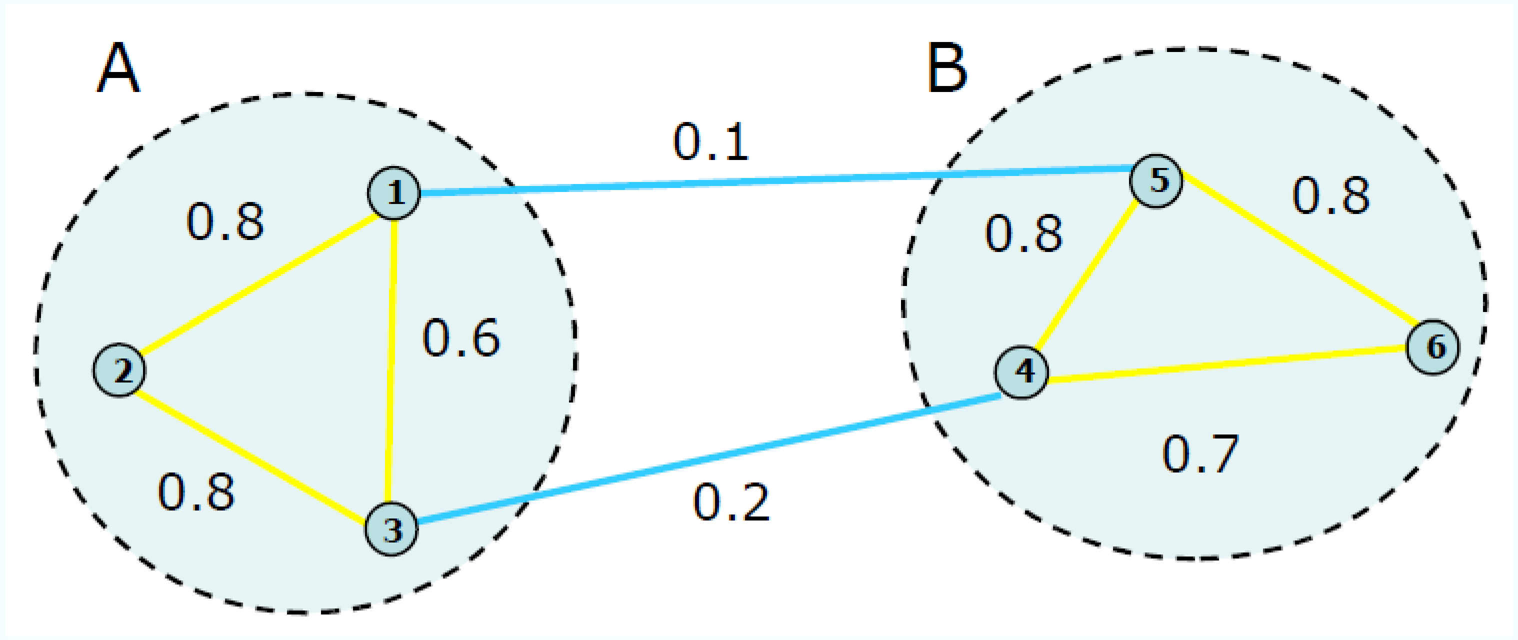
- Given a similarity graph with affinity matrix A the simplest way to construct a partition is to solve the min-cut problem:
  - Choose the partition $Z_1, \dots, Z_k$ that minimises

$$\mathrm{cut}(Z_1, \dots, Z_k) = \frac{1}{2}\sum_{i=1}^{k} A(Z_i, \bar{Z}_i) \quad \text{where } A(Z_1, Z_2) = \sum_{i \in Z_1, j \in Z_2} A(i, j)$$



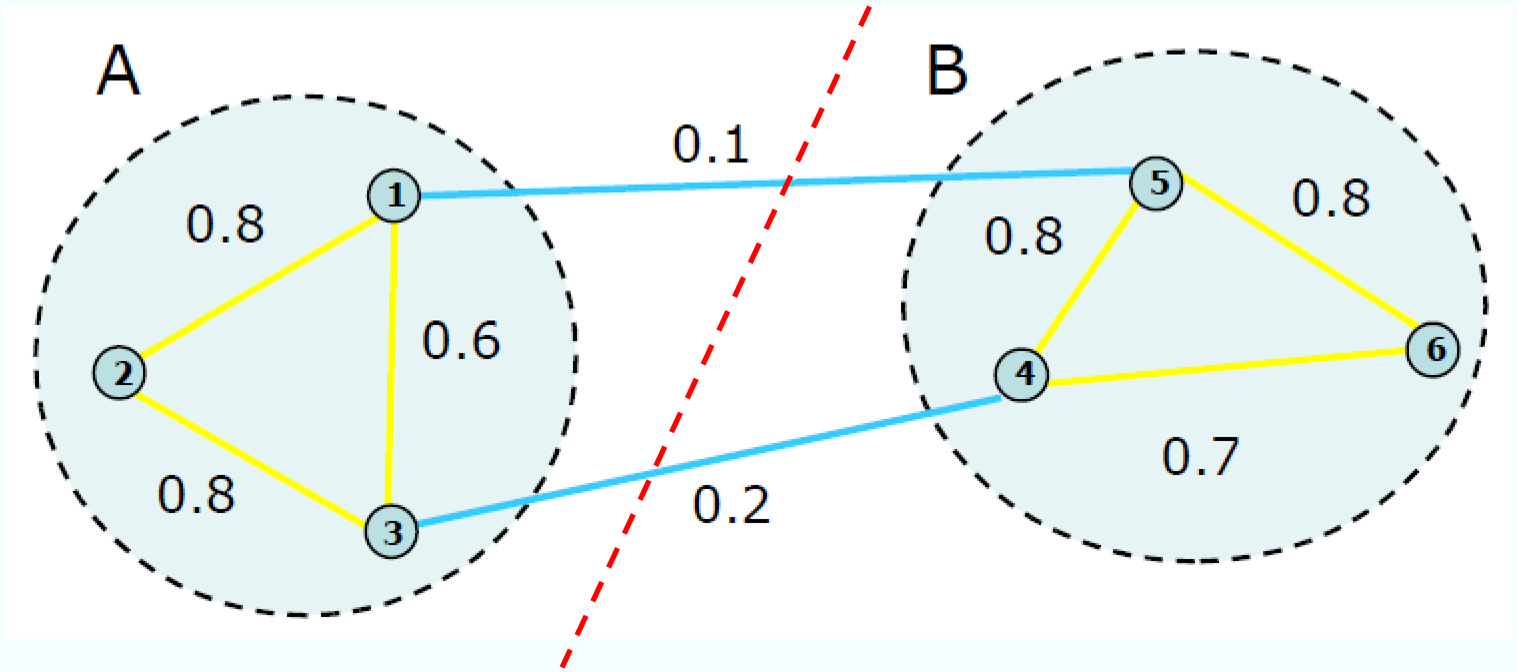
Clustering as a graph-theoretic problem – An example

- We require 2 clusters
- It is obvious we need to cut at least 2 edges



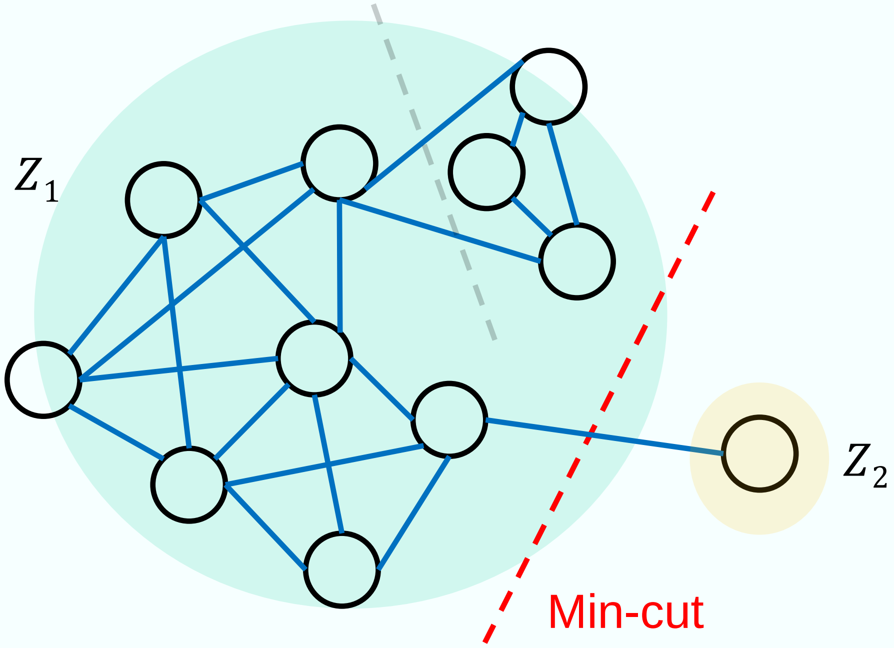
Clustering as a graph-theoretic problem – An example

- We require 2 clusters
- It is obvious we need to cut 2 edges
- $\mathbf{cut}(\boldsymbol{A},\boldsymbol{B}) = \frac{1}{2}\sum_{i\in A, j\in B} \mathbf{Affinity}(\boldsymbol{A},\boldsymbol{B}) = \mathbf{0.3}$



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 $Z_1, \dots, Z_k$ 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 $|Z|$

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

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

$$NCut(Z_1, \dots, Z_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{A(Z_i, \bar{Z}_i)}{\mathrm{vol}(Z_i)} = \sum_{i=1}^{k} \frac{\mathrm{cut}(Z_i, \bar{Z}_i)}{\mathrm{vol}(Z_i)}$$

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

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

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** $\mathrm{Min{-}Max{-}Cut}$ **lead to normalised spectral clustering. Relaxing** $\mathrm{RatioCut}$ **leads to unormalised spectral clustering** [von Luxburg 2007]

- Relaxed **RatioCut** solution: eigenvectors

$$\boldsymbol{X} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_k)\ \ s.t.\ (\boldsymbol{D}-\boldsymbol{W})\boldsymbol{v}_k = \boldsymbol{\lambda}_k\boldsymbol{v}_k \ \ \text{where}\ \boldsymbol{L} = \boldsymbol{D}-\boldsymbol{A}$$

- Relaxed **Ncut** solution: eigenvectors

$$\boldsymbol{Y} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_k)\ \ s.t.\ \ (\boldsymbol{I} - \boldsymbol{L}_{\text{sym}})\boldsymbol{u}_k = \boldsymbol{\lambda}_k\boldsymbol{u}_k \ \ \text{where}\quad \boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-0.5}\boldsymbol{A}\boldsymbol{D}^{-0.5}$$

- Relaxed Min-Max-cut solution: eigenvectors

$$\boldsymbol{Y} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_k)\ \ s.t.\quad \boldsymbol{L}_{\text{sym}}\boldsymbol{u}_k = \boldsymbol{\lambda}_k\boldsymbol{u}_k \ \ \text{where}\quad \boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-0.5}\boldsymbol{A}\boldsymbol{D}^{-0.5}$$

- Quality of solution of 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 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 $\boldsymbol{P} \in \mathbb{R}^{\boldsymbol{N}\times\boldsymbol{F}}$ ($\boldsymbol{N}$ =data points, $\boldsymbol{F}$ = dimensions), $\boldsymbol{k}$ number of clusters

- Construct **pairwise** affinity matrix $\boldsymbol{A}(\boldsymbol{i},\boldsymbol{j}) = \boldsymbol{exp}\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{2\boldsymbol{\sigma}^2}\right)$
- Construct degree **matrix** $\boldsymbol{D} = \mathbf{diag}(\boldsymbol{d}_1, \dots, \boldsymbol{d}_N)$
- Compute Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ (unormalised)
- Compute the first k eigen-vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_k$ of $\boldsymbol{L}$
- Let $\boldsymbol{U} \in \mathbb{R}^{\boldsymbol{N}\times\boldsymbol{k}}$ contain the vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_k$ as columns
- Let $\boldsymbol{y}_i \in \mathbb{R}^{\boldsymbol{k}}$ be the vector corresponding to the $\boldsymbol{i}$-th row of $\boldsymbol{U}$
- Cluster the points $(\boldsymbol{y}_i)$ $_{i\,=\,1,\dots,N}$ into $\boldsymbol{k}$ clusters $\boldsymbol{h}_1, \dots, \boldsymbol{h}_k$ with K-means

Output: Clusters $\boldsymbol{Z}_1, \dots \boldsymbol{Z}_k$ with $\boldsymbol{Z}_i = \{\boldsymbol{i} | \boldsymbol{y}_i \in \boldsymbol{h}_i\}$

K-means

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

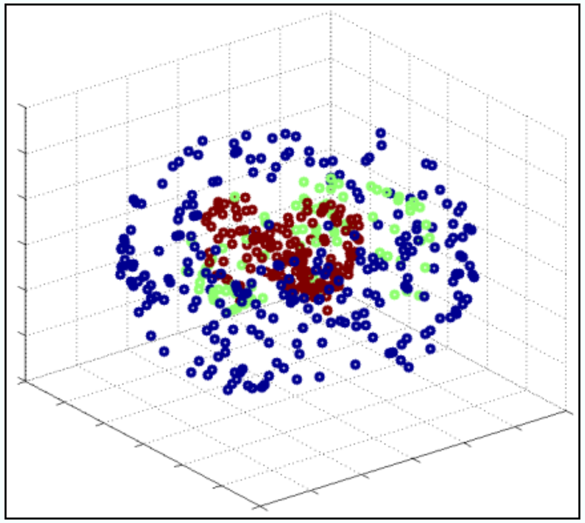
$$\arg\min \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

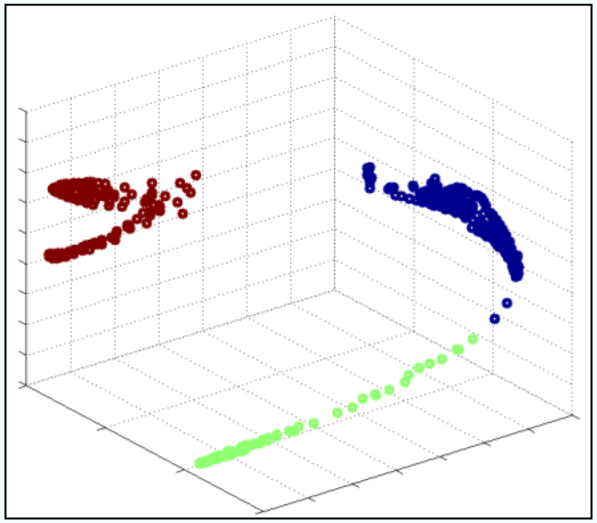
- NP-hard. Iterative algorithm available
  1. Initialise k clusters
  2. Calculate cluster means μ_i
  3. Calculate distances of each point x_j to each cluster mean μ_i
  4. Assign point to nearest cluster
  5. Goto 2 until convergence

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

Why not just use K-means?

- One could use K-means directly on the affinity matrix (or some other clustering approach such as mean shift)
- S.C. separates data while projecting in the low-dimensional space
- Allows clustering of non-convex data


Before spectral clustering

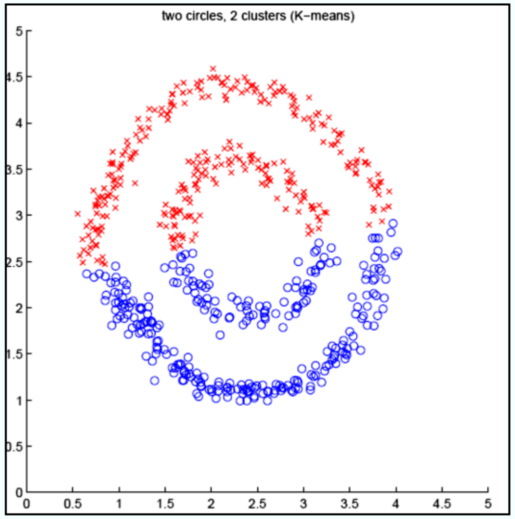

After spectral clustering

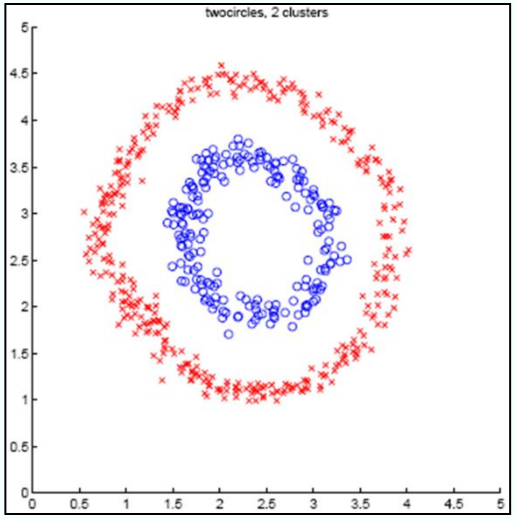
Why not just use K-means?

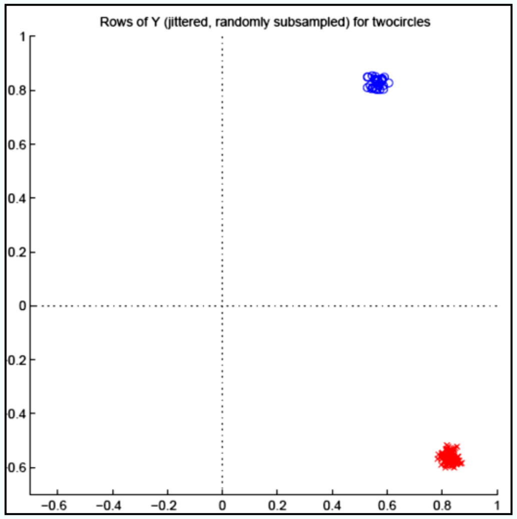
We do K-means here instead

K-means

Spectral clustering

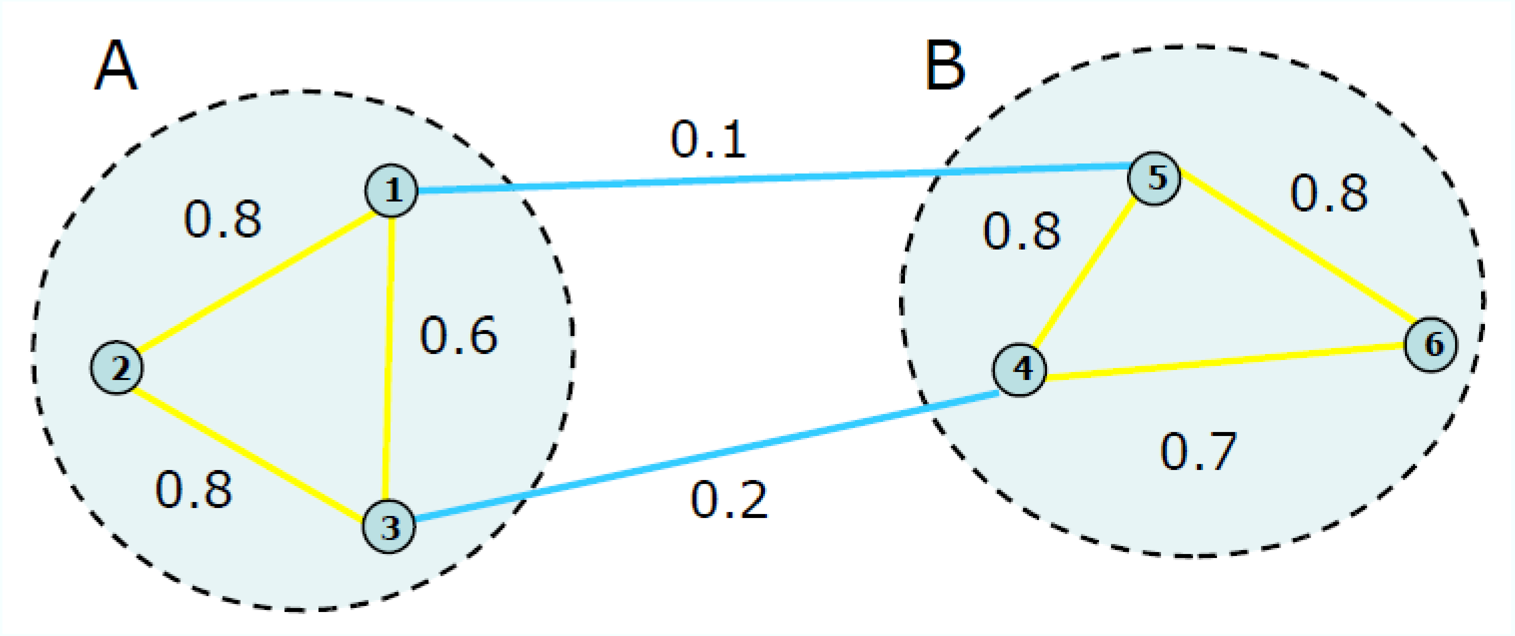




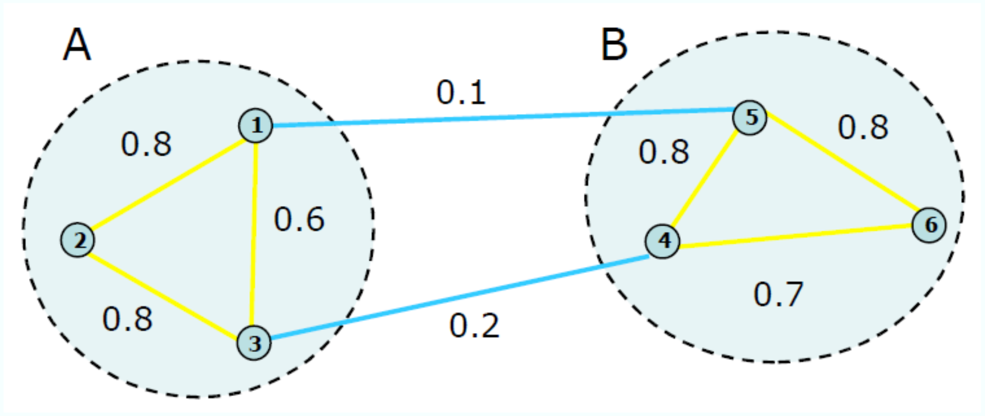


Simple example revisited

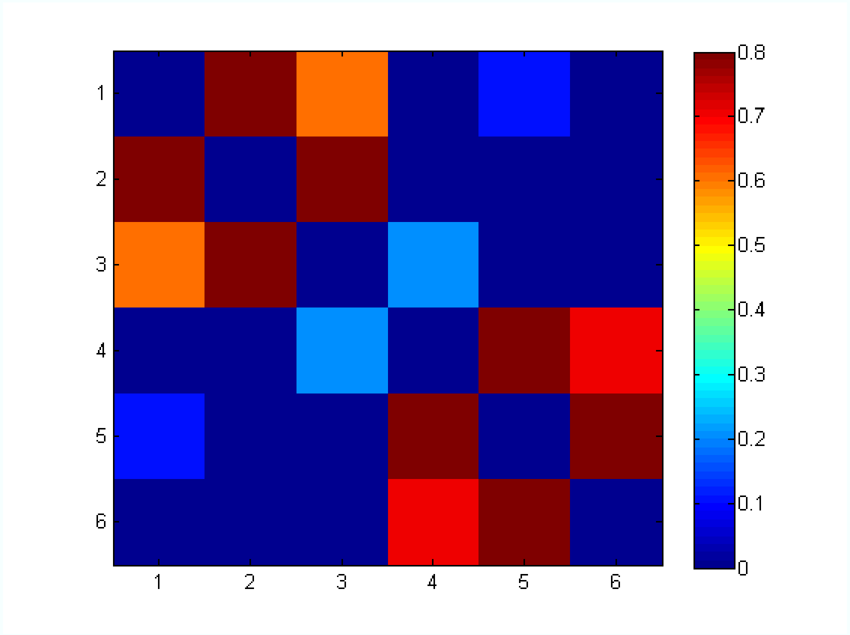
- Now we will use spectral clustering instead



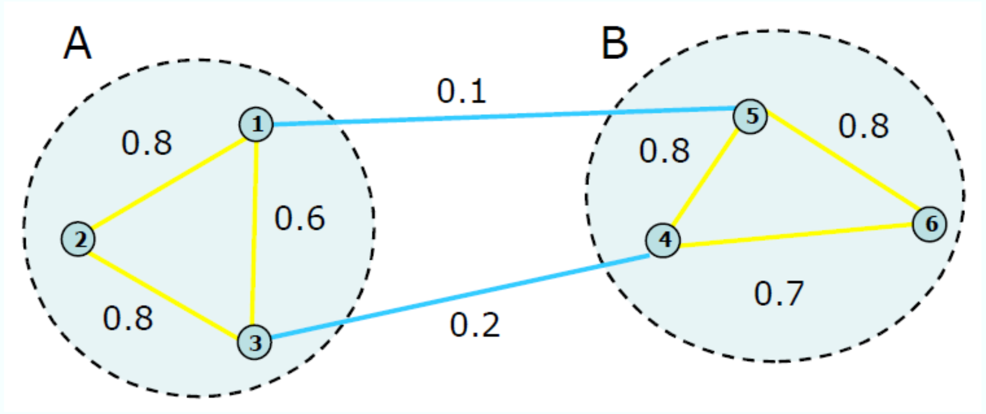
Step 1: Pairwise affinity matrix



	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

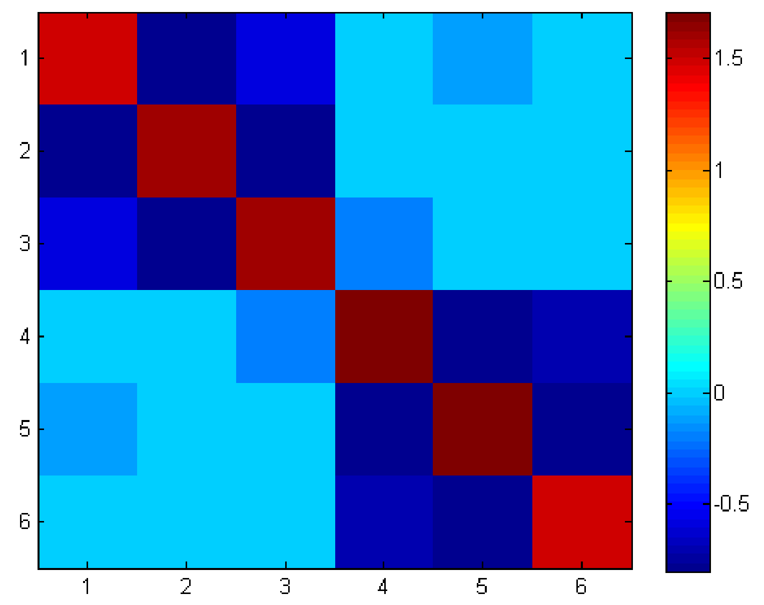


Step 2: Laplacian matrix



$$L = D - A$$

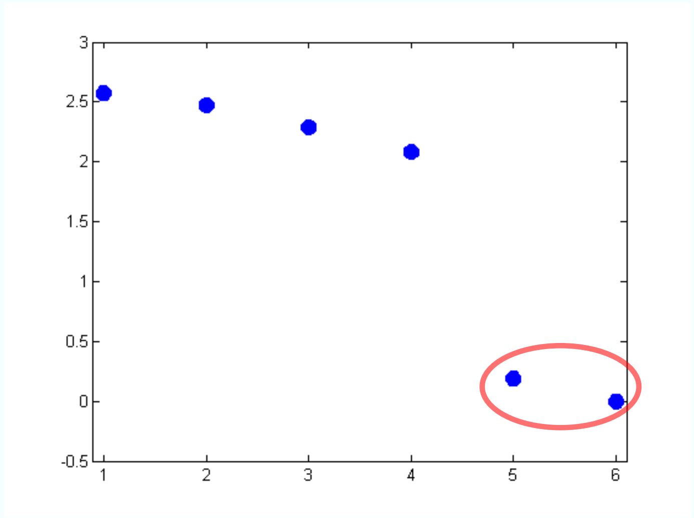
	X_1	X_2	X_3	X_4	X_5	X_6
X_1	1.5	-0.8	-0.6	0	-0.1	0
X_2	-0.8	1.6	-0.8	0	0	0
X_3	-0.6	-0.8	1.6	-0.2	0	0
X_4	0	0	-0.2	1.7	-0.8	-0.7
X_5	-0.1	0	0	-0.8	1.7	-0.8
X_6	0	0	0	-0.7	-0.8	1.5



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

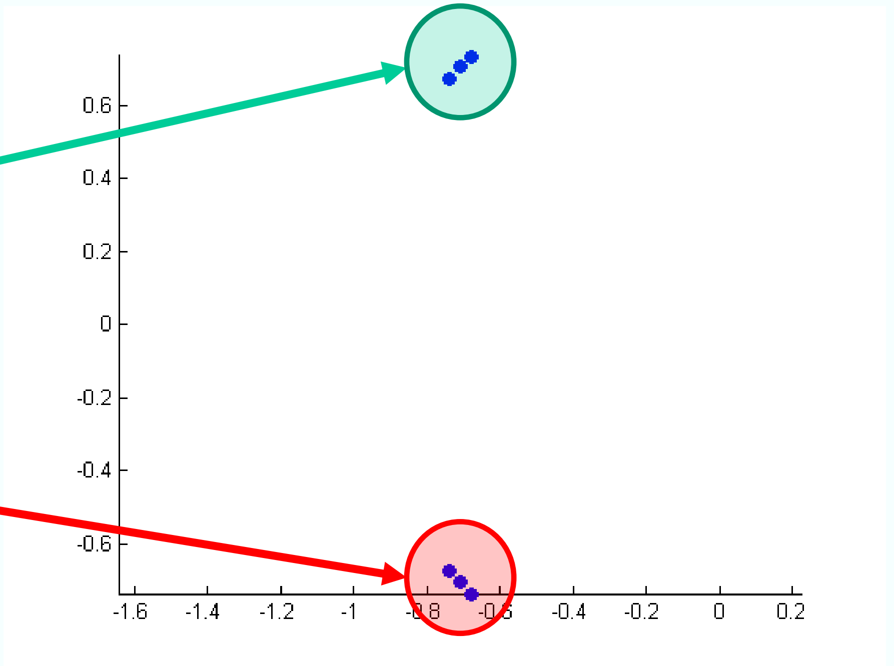
- $U =$

-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 eigen-space

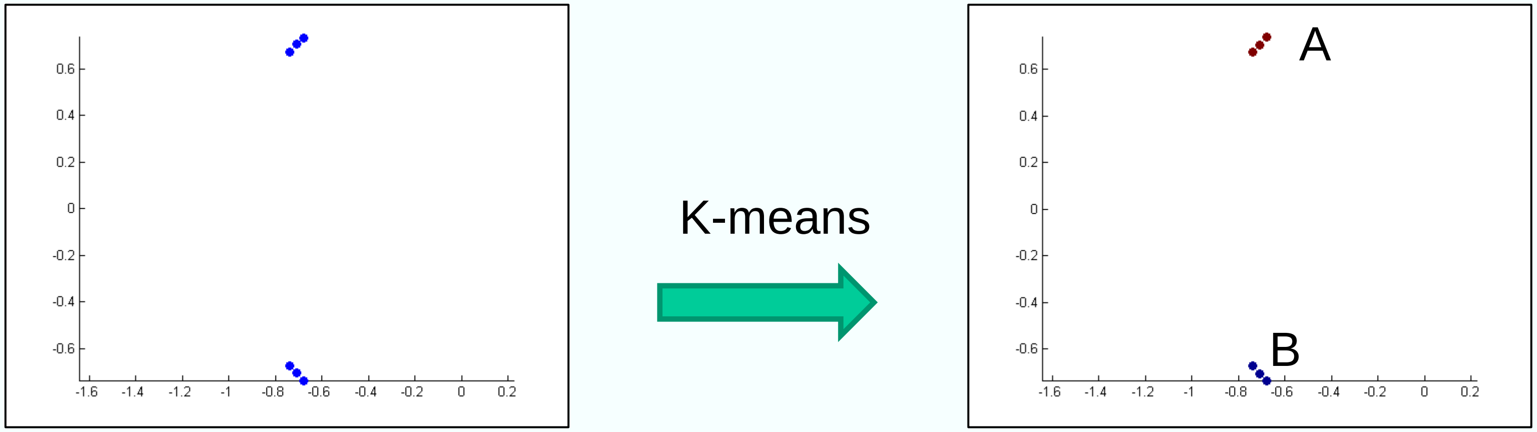
- $Y = \text{row\_normalise}(U)$

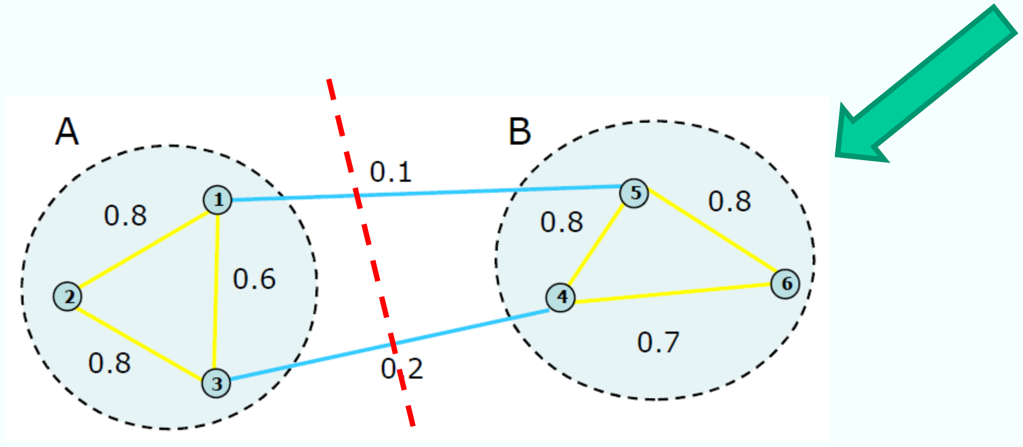
-0.7070	0.7072
-0.6786	0.7345
-0.7398	0.6729
-0.7398	-0.6729
-0.7099	-0.7043
-0.6759	-0.7370



Step 5: Clustering

- K-means clustering with 2 clusters
- Easy, convex clustering problem





Simplex spectral embedding

- Compute k eigen-vectors of the Laplacian.
- Embed objects in the *k*-dim eigen-space

- In the embedded space, objects aggregate to k distinct centroids:
  - Centroids locate on *k* corners of a simplex
  - Simplex consists *k* basis vectors + coordinate origin
  - Simplex is rotated by an orthogonal transformation matrix $T = (\boldsymbol{t}_1, \dots, \boldsymbol{t}_k)$
  - Columns of T are eigenvectors of a *k* × *k* embedding matrix Γ with $\Gamma\boldsymbol{t}_k = \lambda_k \boldsymbol{t}_k$
  - Eigenvalues of Γ = eigenvalues of $L = D - A$

K-means Clustering in Eigen-space

- Simplex spectral embedding theorem provides theoretical basis for K-means clustering in the embedded eigenspace
  - Cluster centroids are well separated (corners of the simplex)
  - *K*-means clustering is invariant under (i) coordinate
  - rotation $x \to Tx$, and (ii) shift $x \to x + a$
  - Thus orthogonal transform T in simplex embedding is irrelevant

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

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