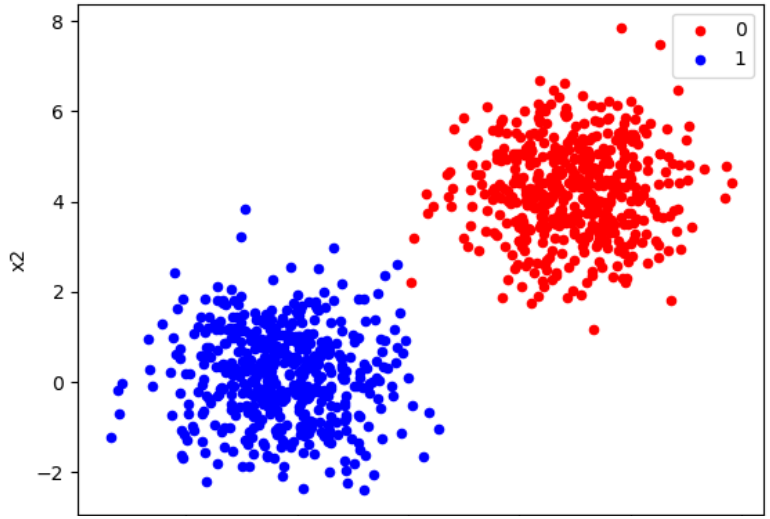
Spectral Clustering

Highlights:

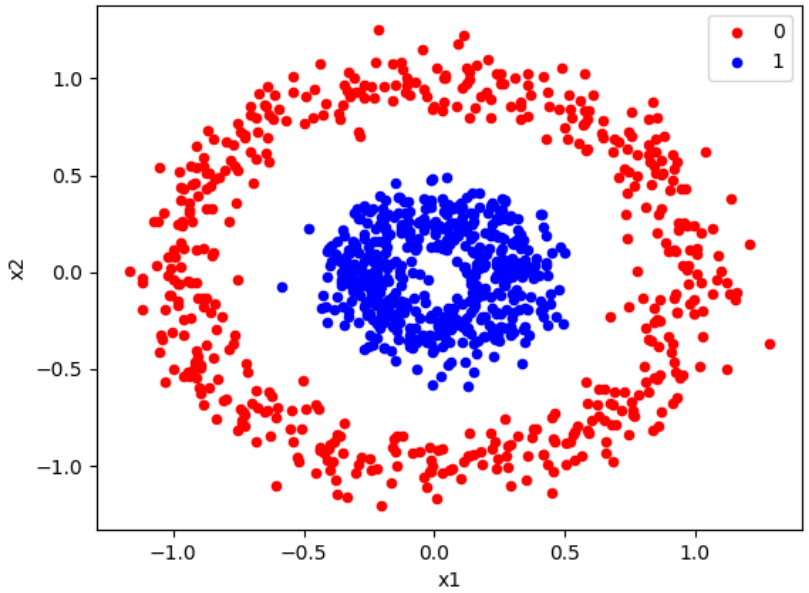
- Clusters may not follow spherical or elliptical shapes
- But may follow other shapes:
  - Spirals, moons, arbitrary curves, lines, etc.

Main idea:

- Perform nonlinear mapping of data onto a new space
- Apply *k*-Means or the like on that space
- Implicit mapping (kernel trick) at work:
  - Use kernels instead of explicit mapping
- Work on frequency domain instead of spatial domain
- Use sparsity:
  - Consider nearest neighbors only

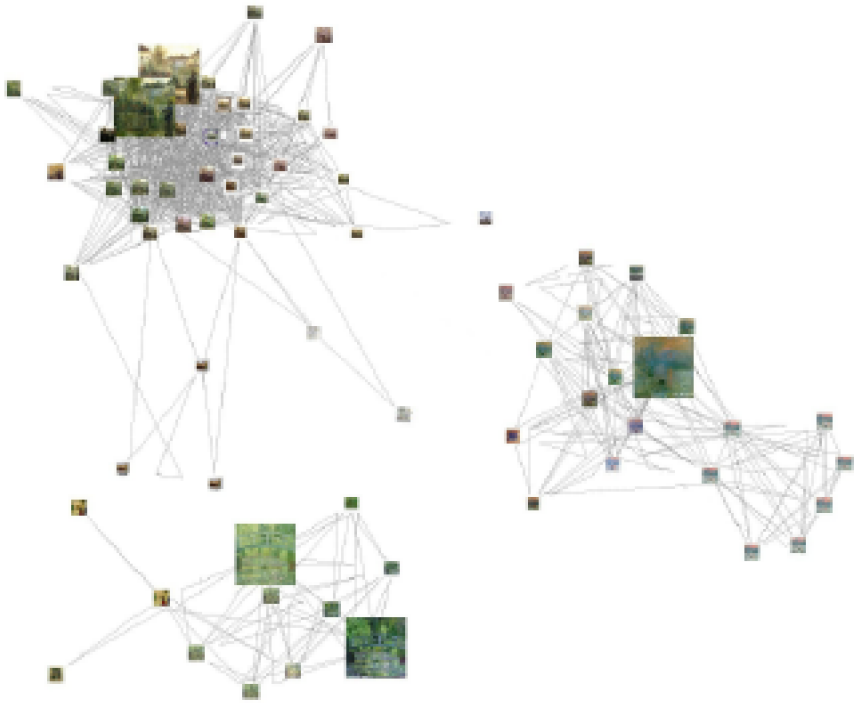


vs

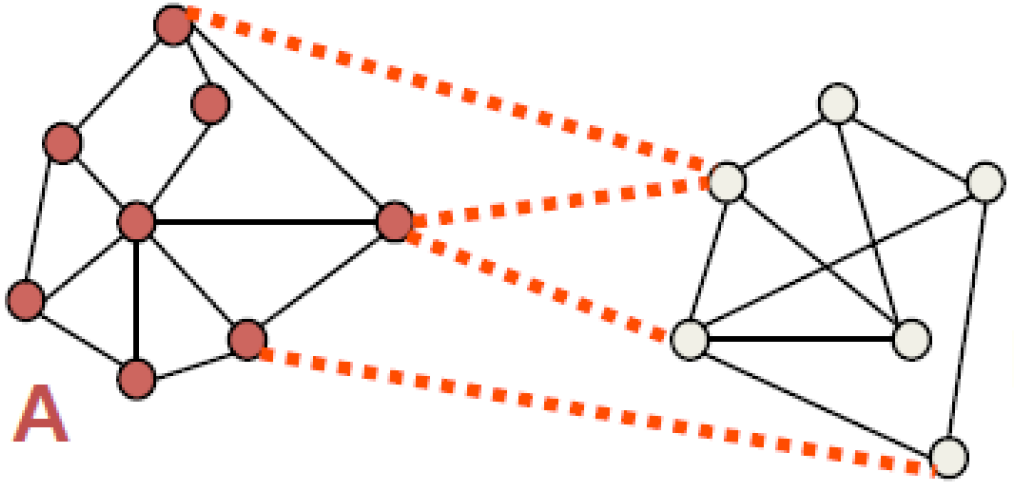


Motivation: Network Data Analysis

- Clustering problem in a network:
  - E.g., social or PPI network
  - Nodes contained in a non-Euclidean (2D) space



- Group points based on links in the graph



Motivation: Graph Partitioning

Consider *G(V,E)*, an undirected weighted graph

W represents the adjacency matrix:

$$W = \{w_{ij}\}$$

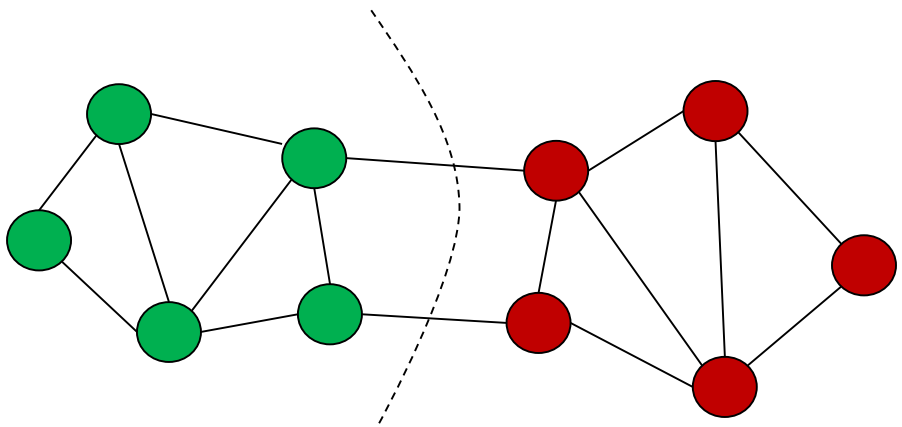
where

- *V* : set of nodes/vertices of G
- *E* : set of edges of graph G
- w_{ij} : weight of edge between vertices v_i and v_j
- *n* : the total number of vertices in *G*

Good clustering:

- Maximize the number of connections within the same cluster
- Minimize the number of connections between different clusters

Intuitively, G can be partitioned as:



Cluster Quality – Cut Score

•Cluster quality can be measured as a function of edge cut of cluster

•Cut is the set of edges that have one node inside a cluster

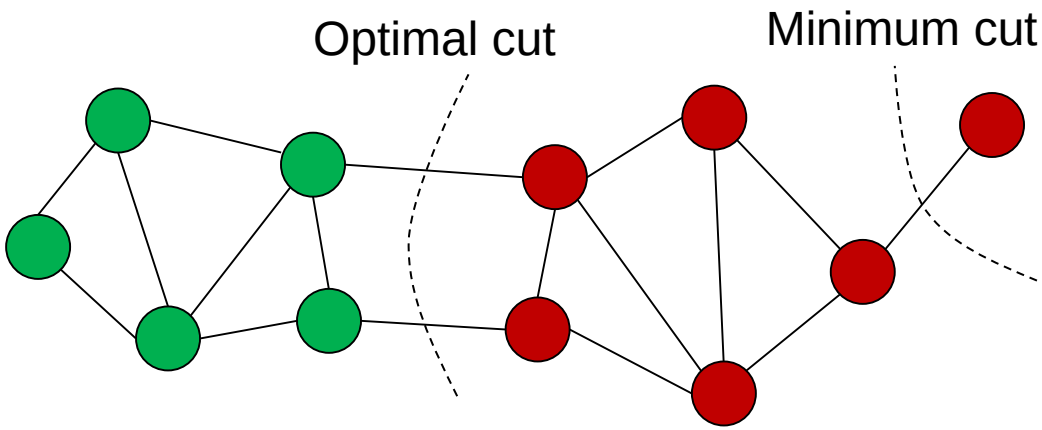
For cluster A in graph G:

$$cut(A) = \sum_{v_i \in A, v_j \notin A} w_{ij} = 2$$

Better clustering means G should be partitioned such that cut score is minimum

Disadvantages of using cut score:

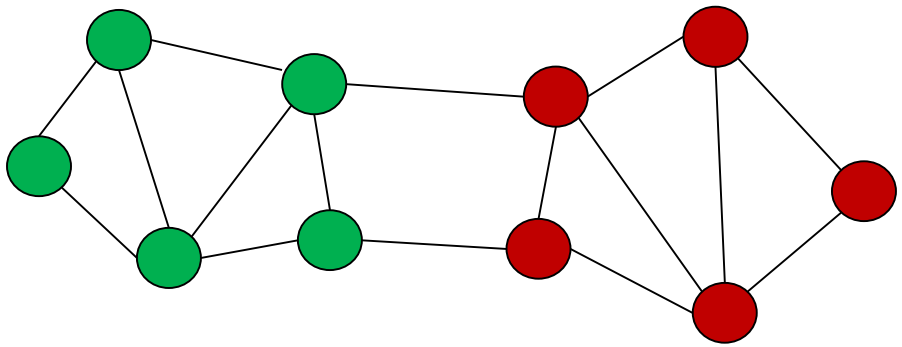
- Partitioning based on cut score focuses only on minimizing the connections between clusters.
- Does not consider maximizing connections within one cluster



•Cut score alone cannot be used for optimal partitioning of the graph

Spectral Clustering

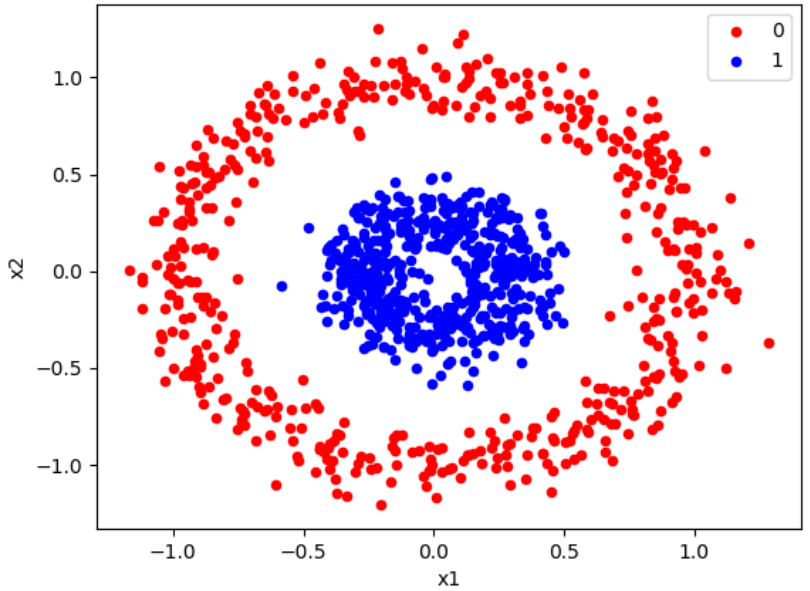
- Input: Graph with vertices and edges with weights



- Use weights $W = \{w_{ij}\}$

vs

- Input: a set of points in an Euclidean space



- Create weights via embedding or kernel and t-nearest neighbors

Spectral Clustering – Motivating Example

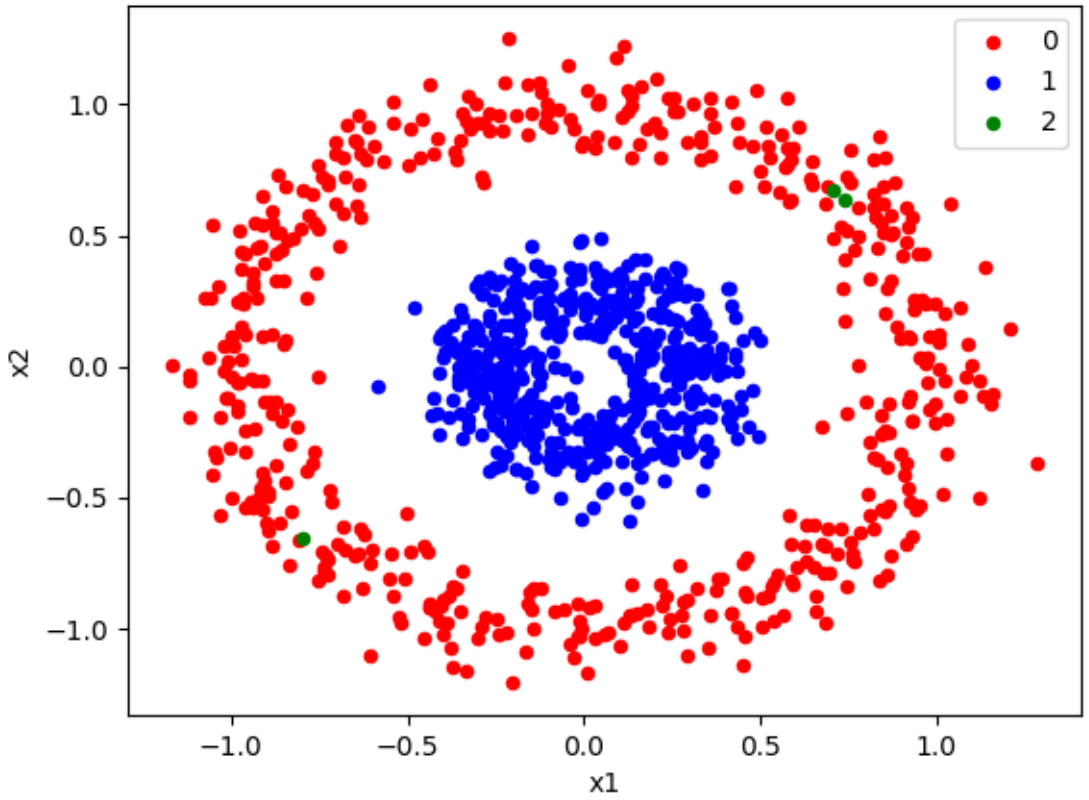
- Let **x** = $[x_1, x_2]^t$ in 2D space mapped to **y** in the 2D space:

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1^2 + x_2^2 \end{bmatrix}$$

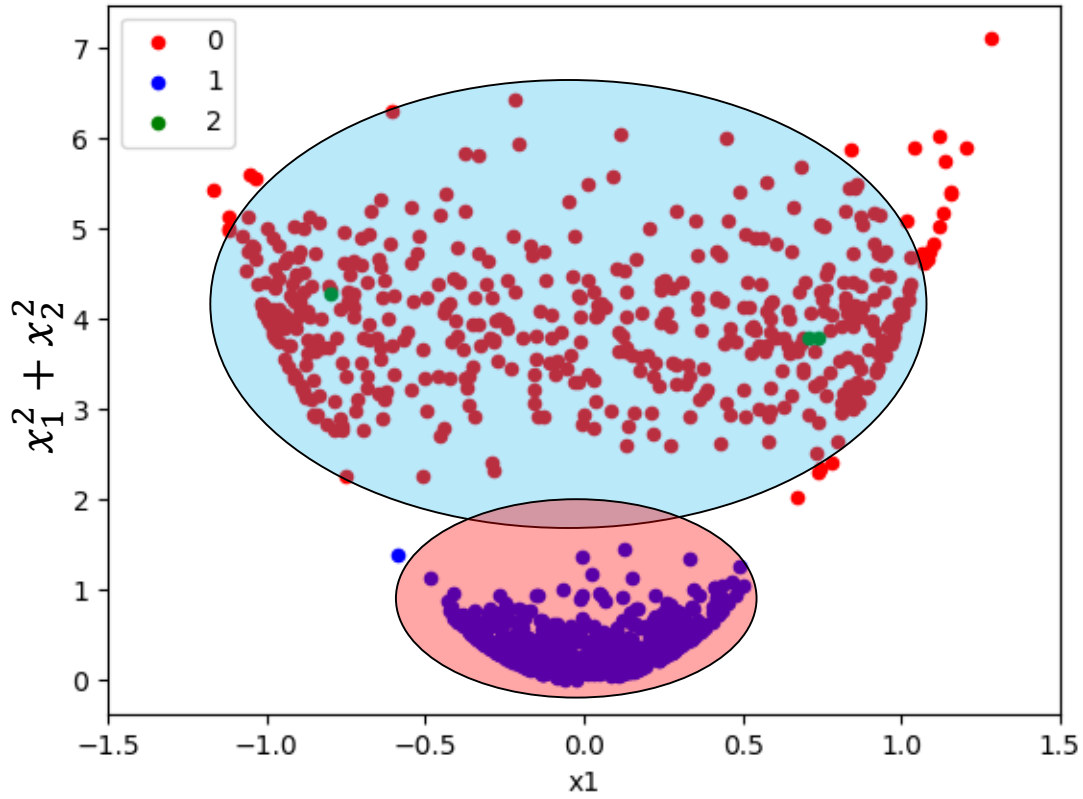
- This transformation places all data in an *arbitrary parabolic* form in 2D (see next page)
- The classifier in the original 2D space has a circular form (circle) to divide the space into two regions.
- It is equivalent to a line in the new 2D space!

Graphically

Original 2D space:

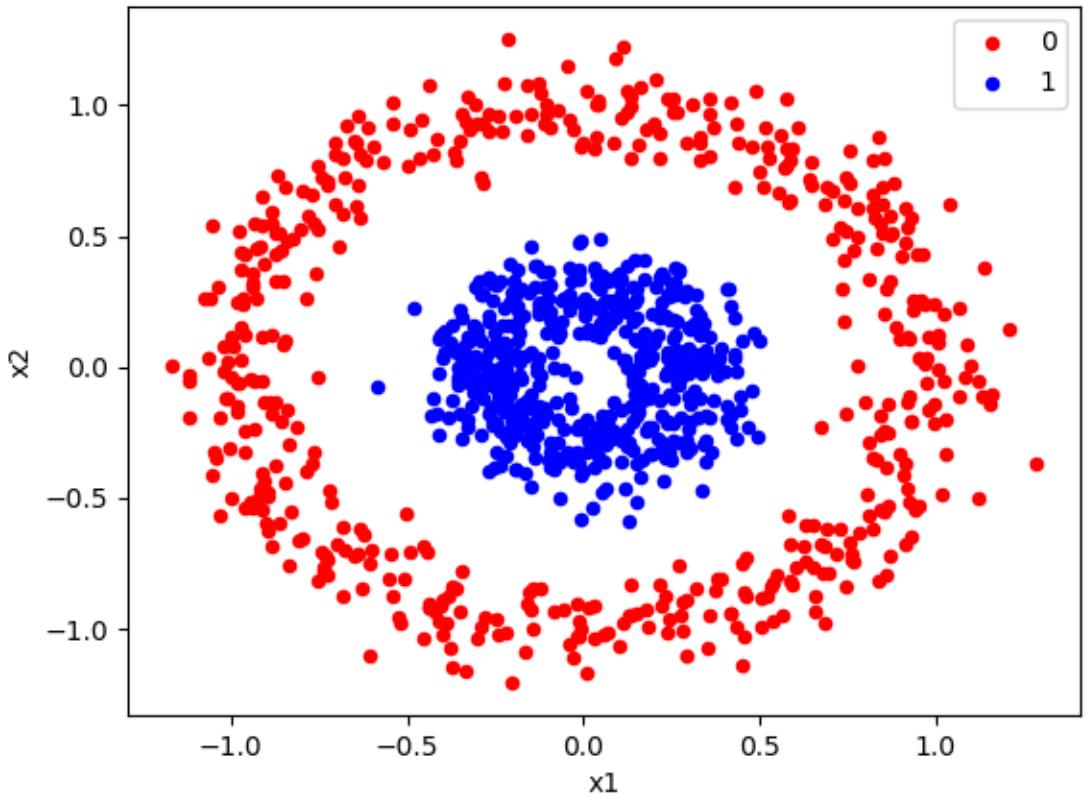


New 2D space: Apply EM or k-means

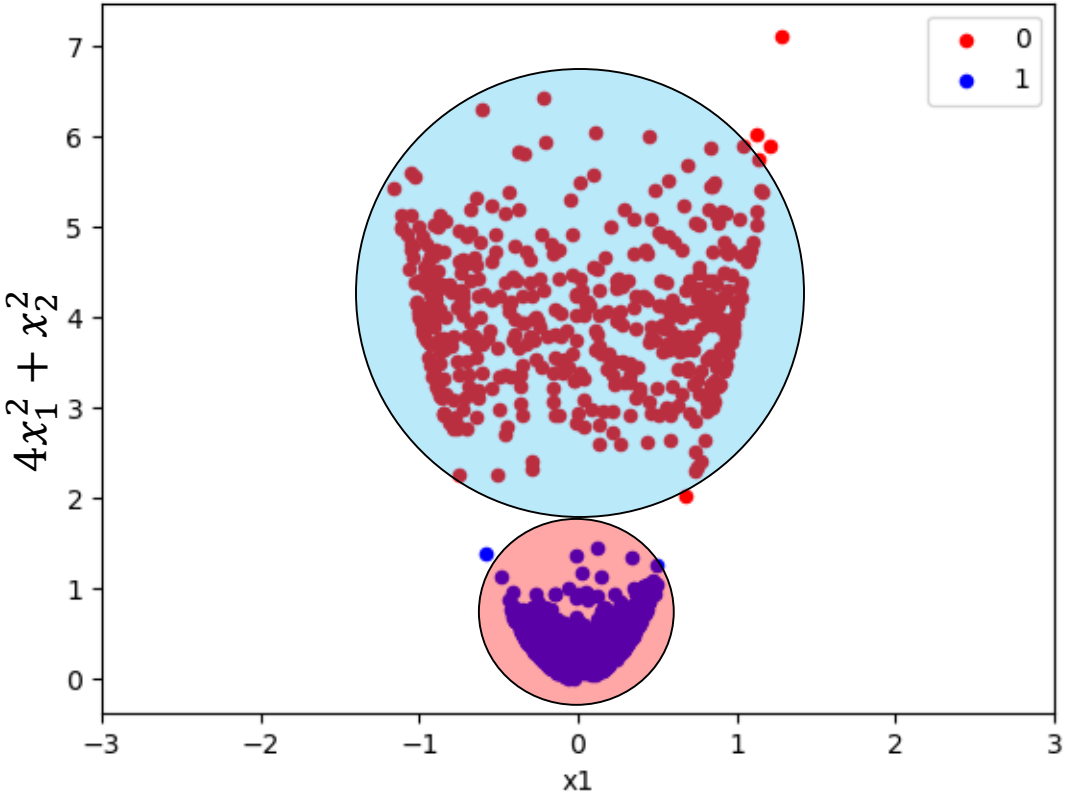


Graphically – cont'd

Original 2D space:



New 2D space: apply EM or k-means!



Spectral Clustering – Graph Partitioning Problem

- In Spectral Clustering, each data point $\mathbf{x}_i$ is considered as a single node of the graph.
- Nodes are the data points
- Edges represent the associations among data points
- Weight of an edge between a pair of data points indicates the strength of association/similarity between them.

- Association or similarity between data points can be measured by similarity function, typically, a kernel and neighborhood (t neighbors)
- Commonly used kernel: RBF

$$w_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$$

- Sparsity of G depends on t
  - Small $t \Rightarrow G$ is very sparse
  - Large $t \Rightarrow G$ is very dense
- Obtaining a graph G=(V,E), where:
  - V is set of nodes/vertices of G
  - E is set of edges of G.
  - W is the weight matrix.

Similarity Matrix

Ways of creating similarity matrix:

- t-NN is a simple approach where
  - a graph is created by connecting each data point to t nearest data points.
  - Weight of edge between points is Euclidean distance.
- Epsilon approach:
  - edges between vertices within ϵ distance assigned one particular value as weights of edges.
  - Those vertices which are beyond ϵ distance are assigned another weight value.

Recall, commonly used kernel: RBF

$$w_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$$

Sparsity of G depends on t

Small $t \Rightarrow G$ is very sparse

Large $t \Rightarrow G$ is very dense

Objective of Spectral Clustering

- Let vector ***f*** represent the cluster membership values for *n* points as:

$$f = \begin{bmatrix} f_1 \\ f_2 \\ . \\ f_N \end{bmatrix} \quad f_i \text{ indicates to which cluster point } i \text{ belongs}$$

For example, for 2 clusters, we can expect ***f*** to look like:

$$f = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{where } 0 \text{ and } 1 \text{ indicate cluster } c_1 \text{ and } c_2$$

- Based on the idea of cut score, the best possible way of partitioning the graph can be obtained by minimizing following expression:

$$\underset{f}{\operatorname{argmin}} \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2$$

which means:

Find such ***f*** that minimizes the above expression

Simplifying the Objective Function

$$\text{argmin}_f \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2$$

$$\text{argmin}_f \sum_{i,j=1}^{N} w_{ij}(f_i^2 - 2f_if_j + f_j^2)$$

$$\text{argmin}_f \underbrace{\sum^{N} w_{ij}.f_i^2}_{\text{Part A}} - 2\underbrace{\sum_{..}^{N} w_{ij}.f_if_j}_{\text{Part B}} + \underbrace{\sum_{..}^{N} w_{ij}.f_j^2}_{\text{Part C}}$$

Simplifying part A:

$$\sum_i^N (\sum_j^N w_{ij}.f_i^{2.})$$

$$\sum_i^N (w_{i1}.f_i^2 + w_{i2}.f_i^2 + ...)$$

$$\sum_i^N (w_{i1} + w_{i2} + ...)f_i^2$$

$$\sum_i^N g_i f_i^2 \quad .....\text{where } g_i \text{ is degree of vertex } i$$

$$g_1.f_1^2 + g_2.f_2^2 + ...g_N.f_N^2$$

$$f^T G f \quad .....\text{G is degree matrix}$$

…continued

Similarly, after simplifying Part B and Part C, we obtain:

$$\text{argmin}_f \sum_{i,j}^{N} w_{ij}.f_i^2 - 2\sum_{i,j}^{N} w_{ij}.f_i f_j + \sum_{i,j}^{N} w_{ij}.f_j^2$$

$$\text{argmin}_f \; f^T G f - 2 f^T W f + f^T G f$$

$$\text{argmin}_f \; 2 f^T (G - W) f$$

$$\text{argmin}_f \; 2 f^T L f$$

where $G - W = L$ is called Laplacian Matrix

The final optimization form is:

$$\text{argmin}_f \; 2 f^T L f$$

It's a quadratic optimization problem.

Assume L is symmetric

$$\text{Solution of } \text{argmin}_x x^T A x \text{ is :}$$

$$Ax = \lambda x$$

Solved by eigen decomposition of A

…continued

$Ae = \lambda e$ is similar to the form of eigenvectors

where:

e is an eigenvector of A

λ is the corresponding eigenvalue of e

Solution of $\text{argmin}_f \, f^T L f$ is:

$Lf = \lambda f$

Then, the eigenvector corresponding to the smallest eigenvalue will give vector f such that $\text{argmin}_f \, f^T L f$ is minimum.

Important properties:

- L is a symmetric matrix
- Therefore, $L = L^T$

- $(f_1, \lambda_1), (f_2, \lambda_2), \ldots . (f_N, \lambda_N)$ are n pairs of orthogonal eigenvectors of L and their corresponding real valued, non-negative eigenvalues

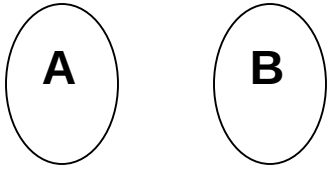
$$f^T f = f f^T = I$$

- where I is the identity matrix

Interpretation of Eigenvectors

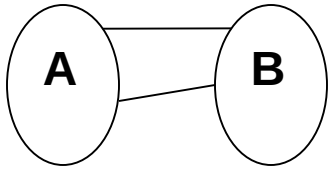
- Eigenvectors of *L* sorted in increasing order corresponding eigenvalues called **spectrum of graph**.
- The number of eigenvectors whose corresponding eigenvalues are 0 indicate the number of connected components in graph.
- Eigenvector corresponding to second smallest eigenvalue is called **Fiedler vector**.
- Third, fourth, etc., eigenvectors may be chosen

Intuitively,



$$\lambda_1 = \lambda_2 = 0$$

The graph is **disconnected (clustered)** into 2 components when the first two eigenvalues are 0



$$\lambda_1 \approx \lambda_2$$

The graph becomes **more connected (less clustered)** as the second eigenvalue more distant from 0 (first eigenvalue)

Spectral Clustering Steps

Example:

4 Data Points: (2,1), (2,3), (3,1), (3,3)

Step 1:

- Create a graph by considering data points as vertices and connections between them as edges.
- In this example, we use simple *t*-NN approach where $t = 3$.
- The, for each point, create an edge between that point and its 2 nearest data points
- Use Euclidean distance

- For simplicity, let the weights of all edges be 1.

Therefore, we obtain the adjacency and degree matrices as:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

W = adjacency matrix G = degree matrix

Spectral Clustering Steps

Step 2:

➢Next step: compute the Laplacian matrix *L* as:

$L = G - W$

Obtain the following Laplacian matrix:

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

L is symmetric, that is

$$L = L^T$$

Step 3:

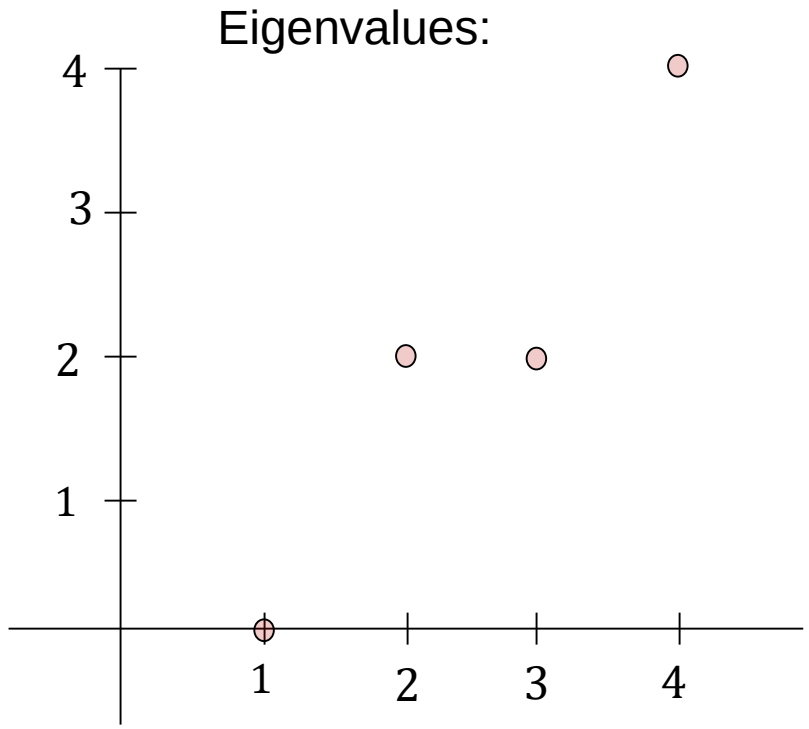
➢Compute the eigenvectors and eigenvalues of L

➢There are 4 data points and so we obtain 4 pairs of eigenvectors and eigenvalues

➢Each eigenvector will have 4 values which represent the class membership of 4 data points.

Spectral Clustering Steps

Eigenvalues	Eigenvectors
$-2.22e^{-16}$	[0.5 0.5 0.5 0.5]
2.0	[0.40 -0.57 0.57 -0.40]
2.0	[0.70 $4.80e^{-16}$ $1.77e^{-16}$ -0.70]
4.0	[-0.5 0.5 0.5 –0.5]



Important Observations:

- Since L is symmetric, its eigenvectors are orthogonal
  - dot product of any two eigenvectors is 0
- The first eigenvalue is approximately 0
  - the graph is one connected component.
  - There are no disconnected components in graph.
- The second eigenvalue is greater than 0 and whose corresponding eigenvector is the Fielder vector which partitions the graph into two clusters

Spectral Clustering Steps

Step 4:

- We expect the class membership values as 0 or 1 for each data point. This indicates whether data point belongs to one cluster or another.

- However, values of the Fiedler vector are not strictly 0 or 1

- To separate the data points, we use conventional clustering
  - Example: *k*-means can be applied on the values of the Fiedler vector

Step 5:

- Output of k-means is a vector with values 0 or 1 indicating the class membership of data points

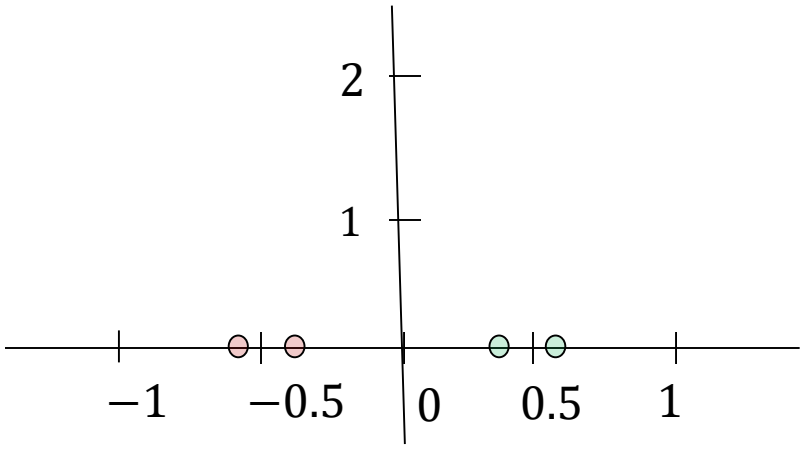
Another view:

- Spectral Clustering performs **dimensionality reduction** and then uses conventional clustering in lower dimension

- For *n* data points, space is transformed from ***n x n to N x k*** where k is the number of clusters.
- *k*-means is then performed in ***k-*** dimensional space

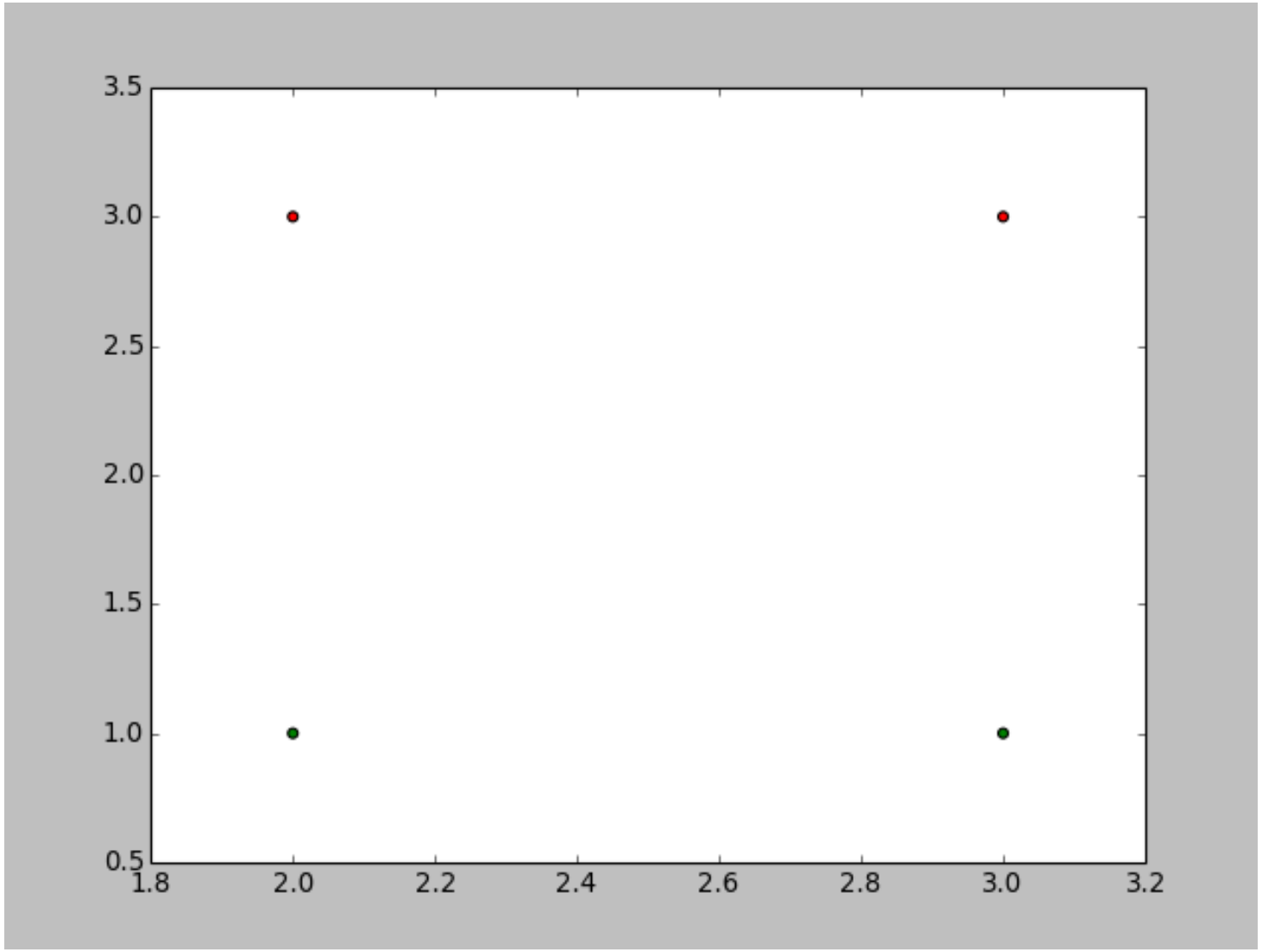
Spectral Clustering Steps

Eigenvalues	Eigenvectors
$-2.22e^{-16}$	[0.5 0.5 0.5 0.5]
2.0	[0.40 -0.57 0.57 -0.40]
2.0	[0.70 $4.80e^{-16}$ $1.77e^{-16}$ -0.70]
4.0	[-0.5 0.5 0.5 −0.5]



- 4 points are the 4 values of second eigenvector
- k-means is then performed on these points to "*group together*" similar class membership values
- In this example, green colored values represent one group (0) and red colored represent another group (1)

Spectral Clustering - Clusters



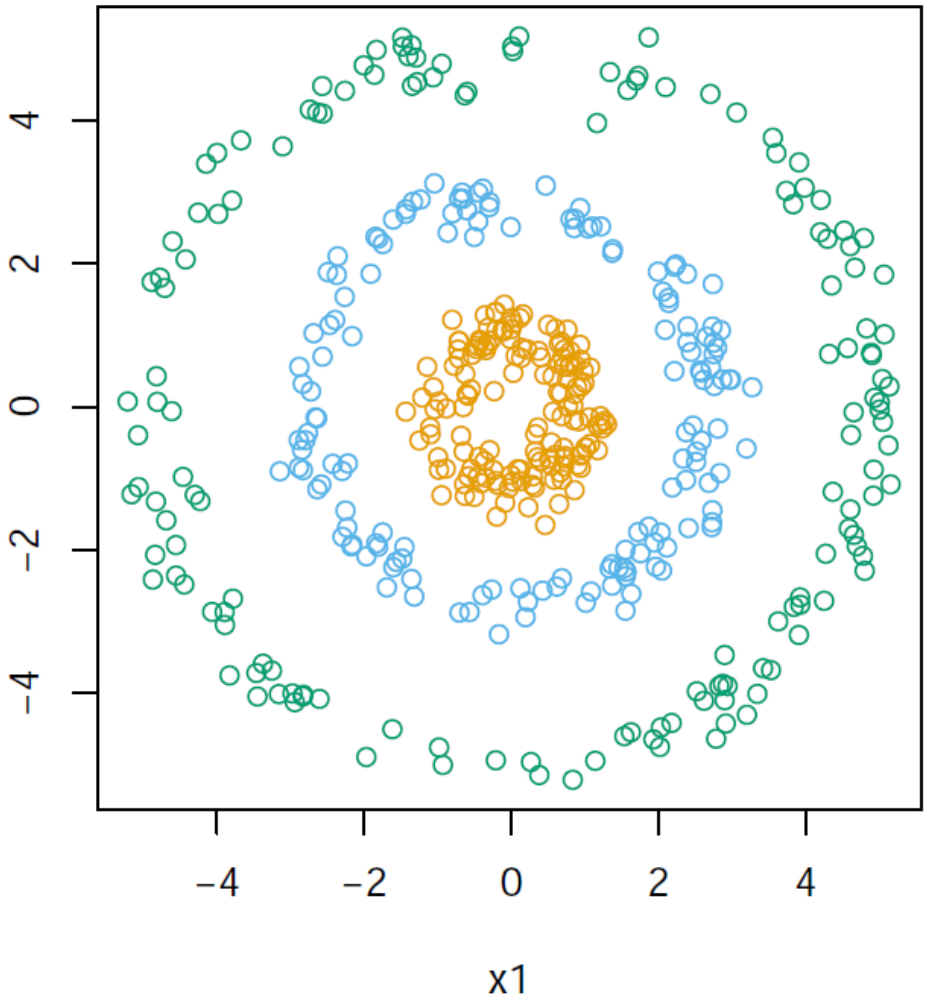
Clustered Data points:

- (2,3) and (3,3) is one cluster (red colored).
- (2,1) and (3,1) is second cluster(green colored)

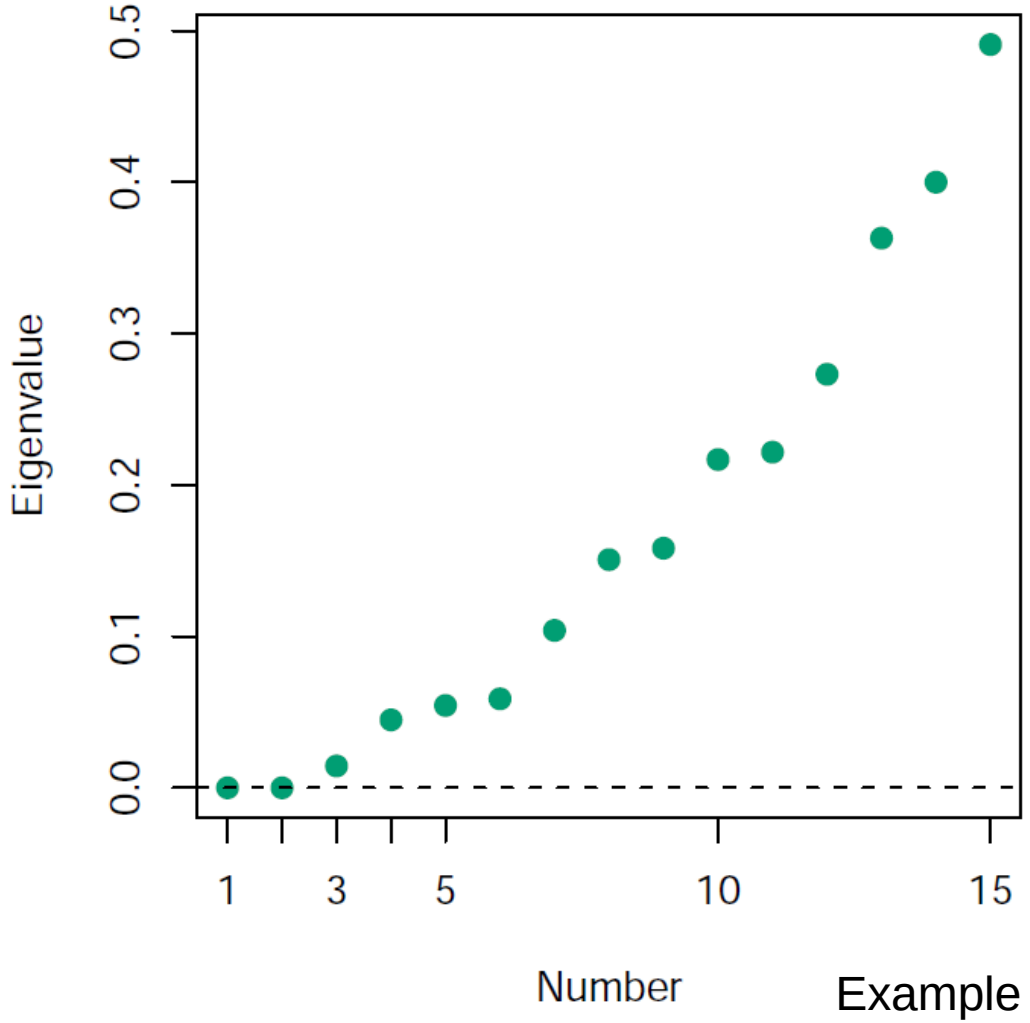
Example: 3 clusters

t-NN applied, where t = 10

Original data – 3 Clusters



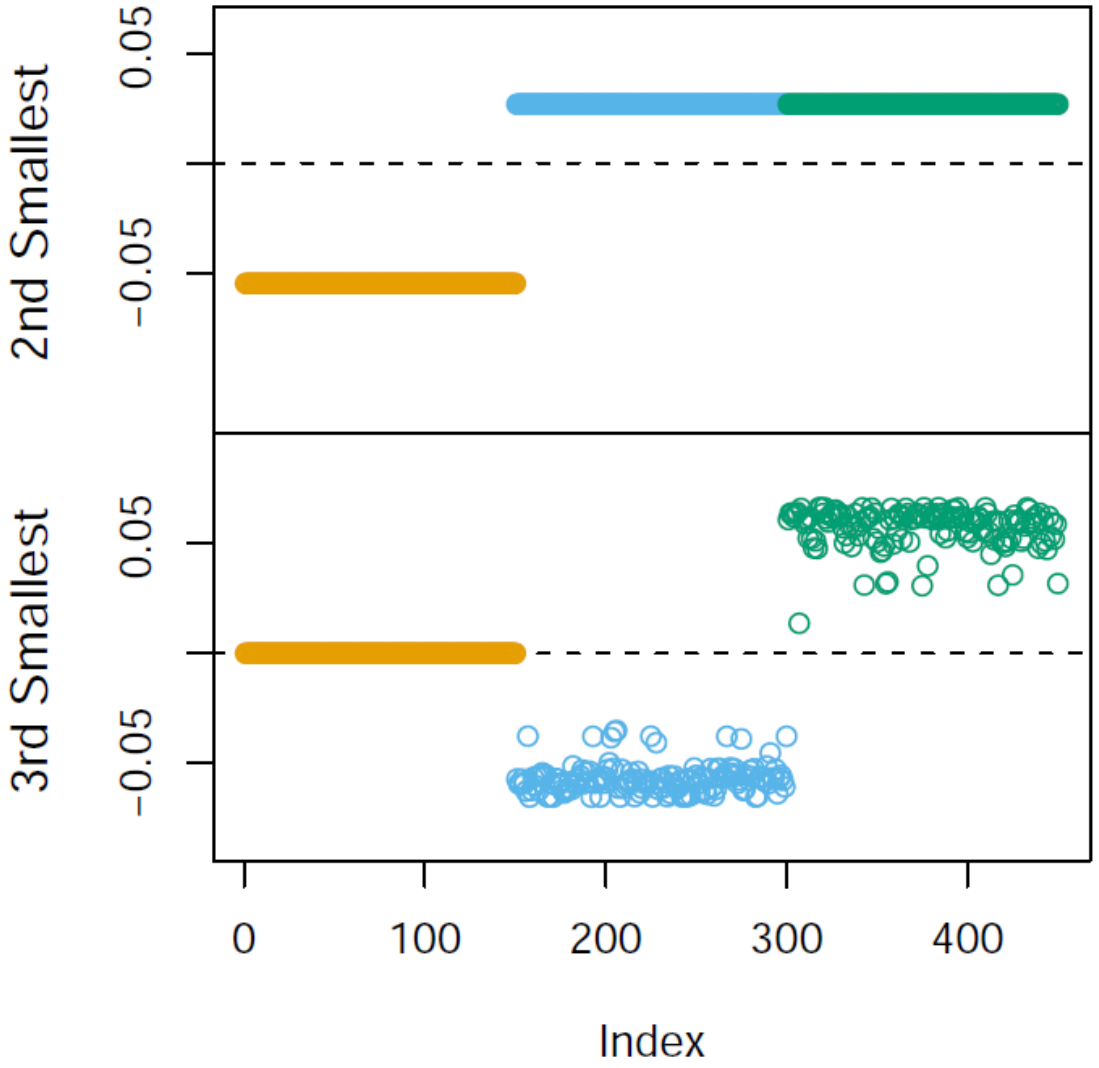
Eigenvalues in ascending order

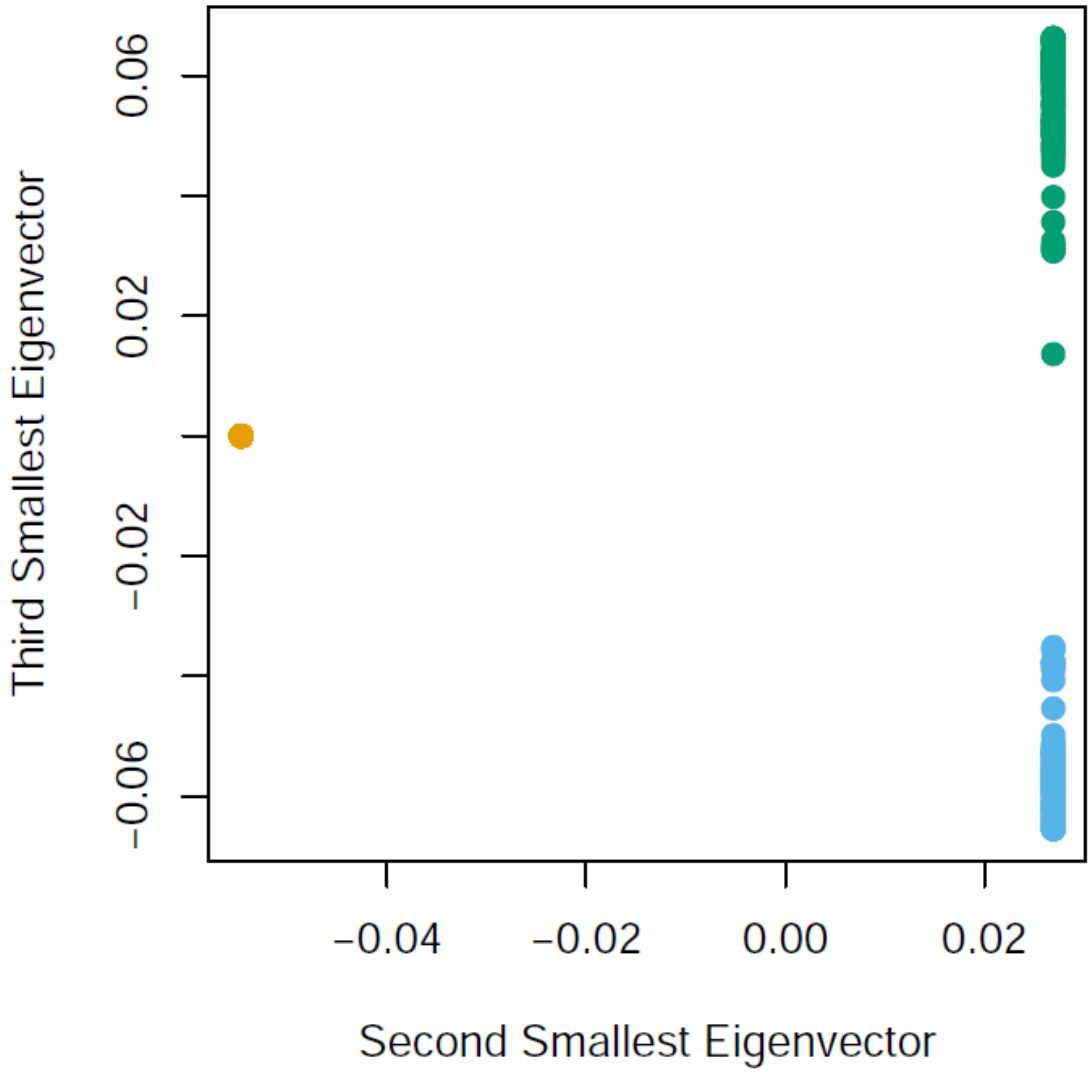


Example from ref [3]

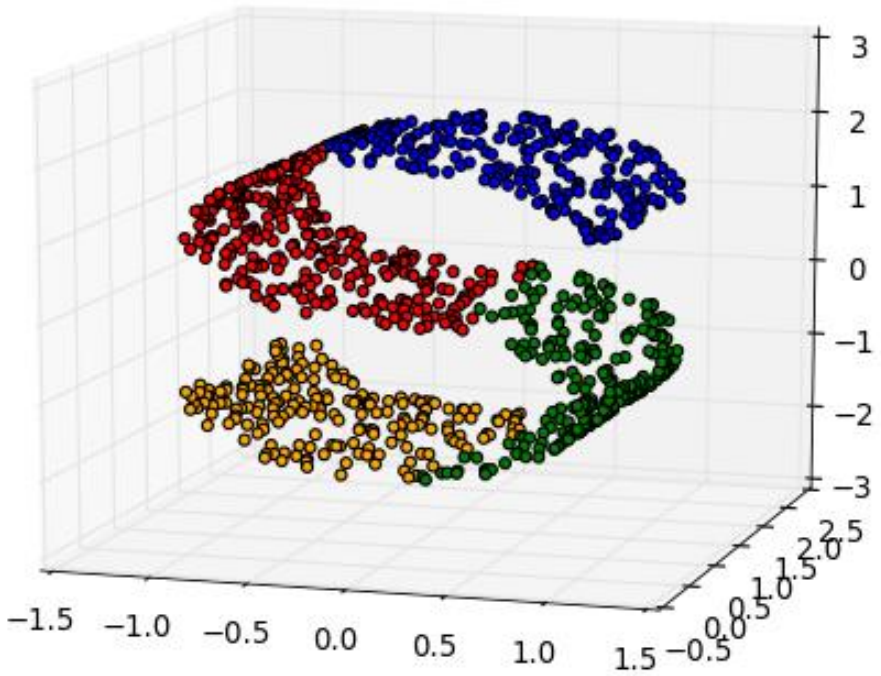
Example: 3 clusters

t-NN applied, where t = 10

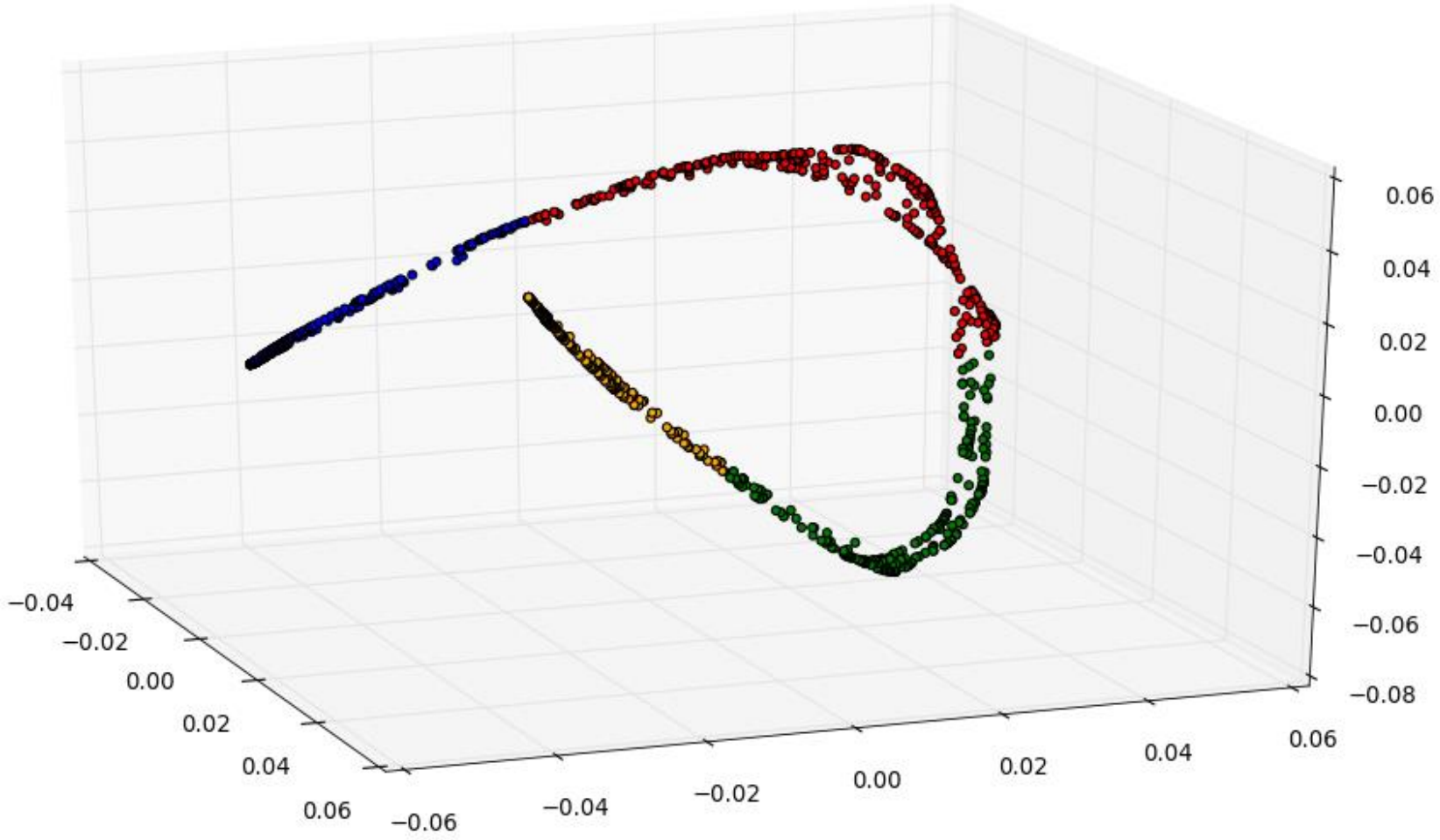




3-Dimensional ‘S’ shape



Clustered Data – 4 clusters



- Plot of 3 smallest eigenvectors whose eigenvalues > 0
- Colors represent Output of *k*-means applied on 3 eigenvectors

Graph Clustering

Consider *G(V,E)*, an undirected weighted graph

W represents the adjacency matrix:

$$W = \{w_{ij}\}$$

where

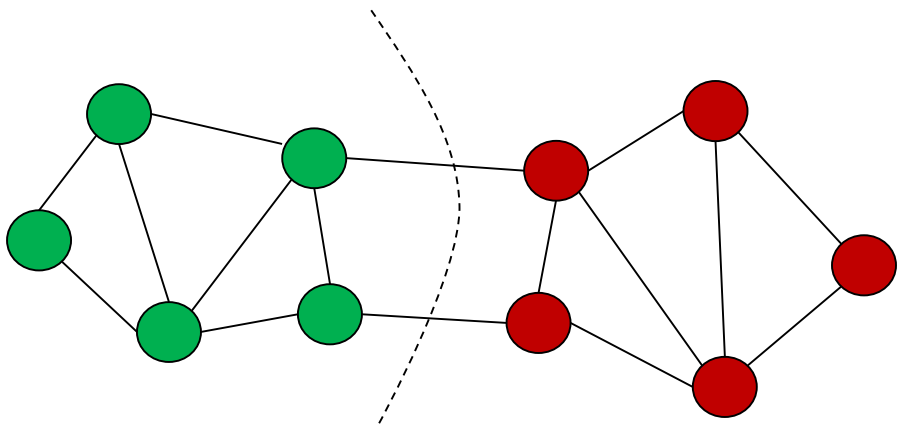
- *V* : set of nodes/vertices of G
- *E* : set of edges of graph G
- w_{ij} : weight of edge between vertices v_i and v_j
- *n* : the total number of vertices in *G*

• Optimal clustering for *k* clusters is NP-Complete

Good clustering:

- Maximize the number of connections within the same cluster
- Minimize the number of connections between different clusters

Intuitively, G can be partitioned as:



Greedy Algorithm: MST Divisive

Algorithm **MST-Divisive**

Input: MST T created from $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
using arbitrary distance (e.g., Euclidean)

begin

$max_w \leftarrow \infty$

for each edge e_i of T

Create MSTs T_1 and T_2 by excluding e_i

if $w(T_1) + w(T_2) < max_w$ and
$|T_1| > 1, |T_2| > 1$

$\max_w \leftarrow w(T_1) + w(T_2)$

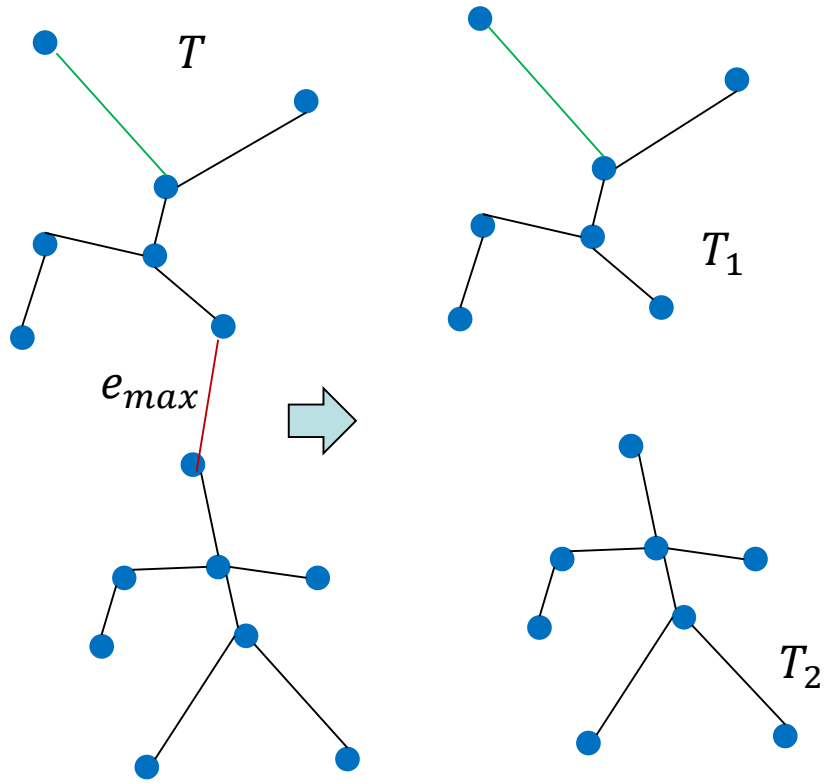
$e_{max} \leftarrow e_i$

Remove e_{max} from T creating T_1 and T_2

if # of clusters < desired

MST-Divisive(T_1)

MST-Divisive(T_2)



Note: e_{max} is not necessarily the longest edge

Cluster Quality – Cut Score

•Cluster quality can be measured as a function of edge cut of cluster

•Cut is the set of edges that have one node inside a cluster

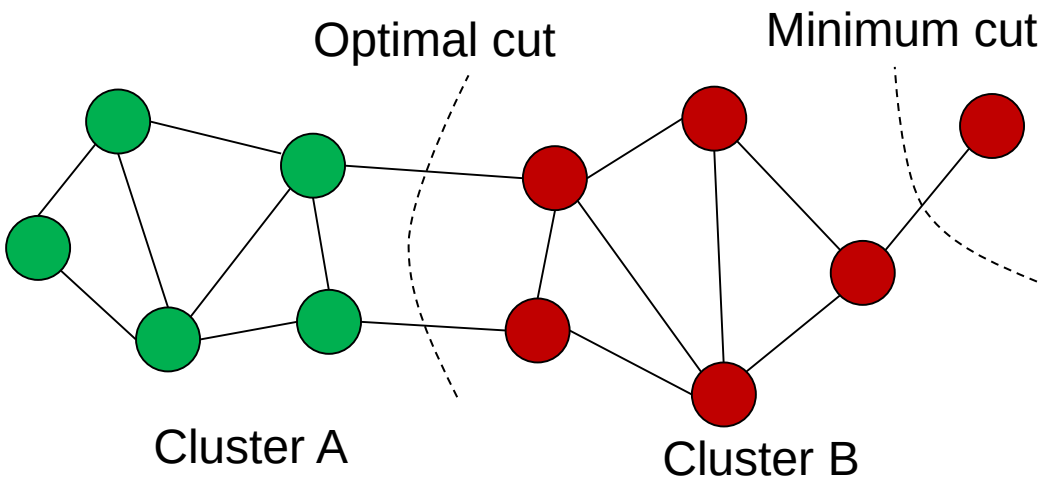
For cluster A in graph G:

$$cut(A) = \sum_{v_i \in A, v_j \notin A} w_{ij} = 2$$

Better clustering means G should be partitioned such that cut score is minimum

Disadvantages of using cut score:

- Partitioning based on cut score focuses only on minimizing the connections between clusters
- Does not consider maximizing connections within one cluster



•Cut score alone cannot be used for optimal partitioning of the graph

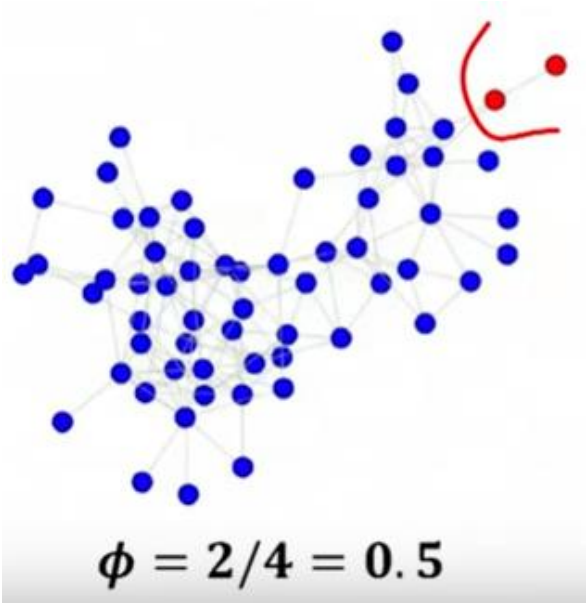
Cluster Quality - Conductance

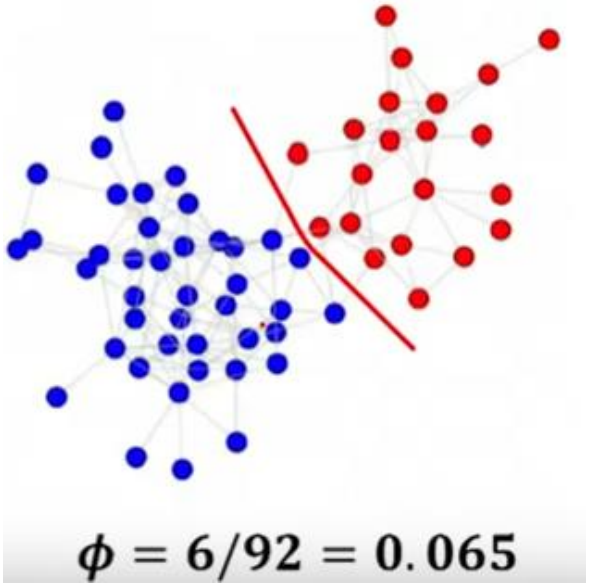
- Conductance measures the connectivity of a cluster wrt another cluster, relative to the density of cluster.
- $\phi(A, \bar{A}) = \frac{cut(A)}{\min\{vol(A), 2m - vol(A)\}}$

where:

$vol(A)$: Total weight of edges with at least one endpoint in cluster A

m : total number of edges in G





Min-Max Cut

- Min-max cut is another approach to partition the graph instead of using cut score [4]

- The disadvantage of cut score is that it only focuses on minimizing connections between two sub graphs

- Min-max cut approach minimizes similarity or association between two sub-graphs and maximizes the association between all pairs of vertices within one sub graph.

Let A and B be two partitioned subgraphs of graph G.

Similarity between A and B is the cutsize :

$$cut(A,B) = W(A,B)$$

where

$$W(A,B) = \sum_{u \in A, v \in B} W_{uv}, \qquad W(A) \equiv W(A,A)$$

Thus, objective function :

$$M_{cut} = \frac{cut(A,B)}{W(A)} + \frac{cut(A,B)}{W(B)}$$

Min-Max Cut

Objective function can be written as :

$$M_{cut} = \frac{x^T (D-W)x}{x^T W x} + \frac{y^T (D-W)y}{y^T W y}$$

where

$$W = \begin{matrix} W_A & W_{A,B} \\ W_{B,A} & W_B \end{matrix}$$

x and y are vectors conformally partitioned with A and B such that

$$x = (1...1,0...0)^T$$

$$y = (0...0,1...1)^T$$

$$x^T D y = 0 \text{ and } x^T W x > 0; y^T W y > 0$$

Therefore, the objective function can be further relaxed as :

$$\min \frac{\hat{x}^T (I - \hat{W})\hat{x}}{\hat{x}^T \hat{W} \hat{x}} + \frac{\hat{y}^T (I - \hat{W})\hat{y}}{\hat{y}^T \hat{W} \hat{y}}$$

subject to :

$$\|\hat{x}\|_2 = \|\hat{y}\|_2 = 1; \hat{x}^T \hat{y} = 0; \hat{x}^T \hat{W} \hat{x} > 0; \hat{y}^T \hat{W} \hat{y} > 0$$

where

$$\hat{W} = D^{-1/2} W D^{-1/2}$$

$$\hat{x} = \frac{D^{1/2} x}{|D|^{1/2}}; \hat{y} = \frac{D^{1/2} y}{|D|^{1/2}}$$

Mcut Algorithm

Algorithm for partitioning a graph into two subgraphs:

- Compute the Fiedler vector
  - The eigenvector corresponding to the second smallest eigenvalue
- Sort nodal values to obtain the Fiedler order
- Search for the optimal cut point corresponding to the lowest Mcut based on the Fiedler order
- Do linkage-based refinements
  - Identify nodes near the cut
  - Equivalent to linkage in hierarchical clustering

Complexity

- Fiedler vector can be quickly done following Lanczos method [6]
- Each iteration runs in O(E + V)
- Sub-optimal solution
- Optimal Mcut for *k* clusters:
  - NP-complete

References - Acknowledgments

The material presented in this chapter has been taken from the following sources (among others):

1. E. Rubinstein. Spectral Clustering. TAU Big Data Processing Seminar, 2014. http://www.cs.tau.ac.il/~amir1/SEMINAR/LECTURES/spectral_clustering.pdf
2. U. von Luxburg. A tutorial on Spectral Clustering. Max Planck Institute for Biological Cybernetics. 2007. https://arxiv.org/pdf/0711.0189.pdf
3. T. Hastie et al. The Elements of Statistical Learning. Second Edition. Springer, 2008.
4. C. Ding et al. A min-max cut algorithm for graph partitioning and data clustering. IEEE International Conference on Data Mining, 2001.
5. R. Horaud. A short tutorial on graphs Laplacians, Laplacian embedding and spectral clustering. INRIA Grenoble Rhone-Alpes, France. https://csustan.csustan.edu/~tom/Lecture-Notes/Clustering/GraphLaplacian-tutorial.pdf
6. B. N. Parlett, The Symmetric Eigenvalue Problem, SIAM Press, 1998.