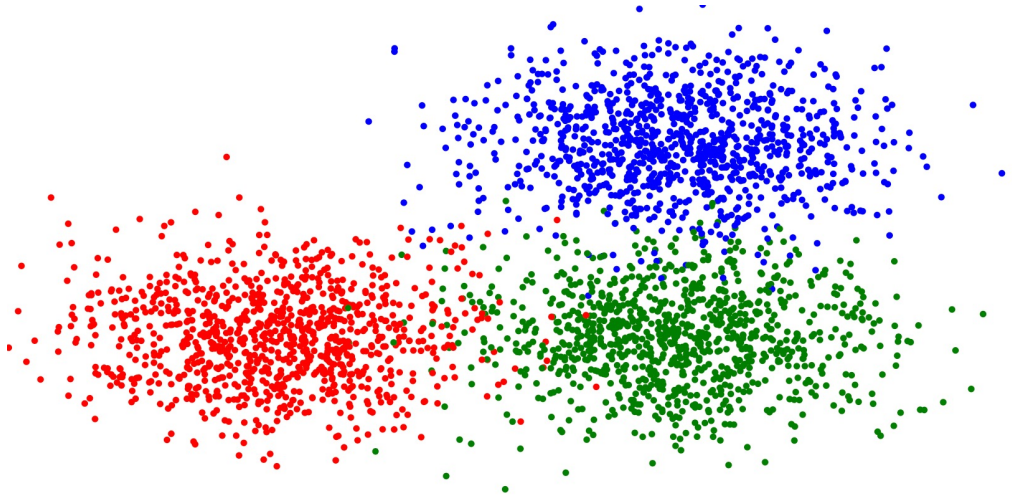
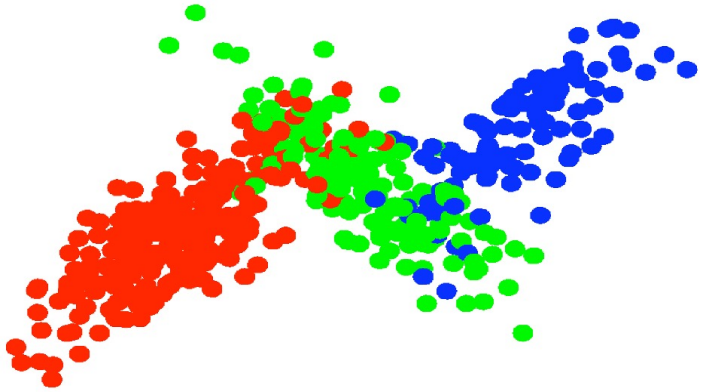
Graph Laplacian & Spectral Clustering

Limitation

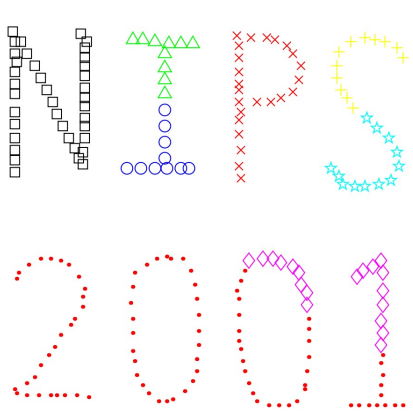
Mixture models and K-means focus on data **compactness**

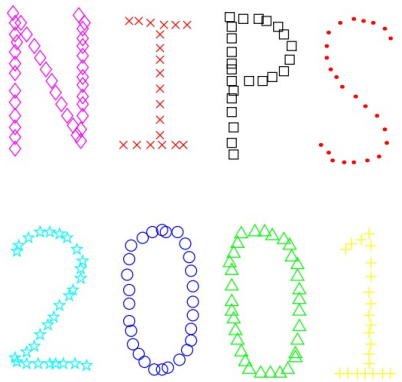


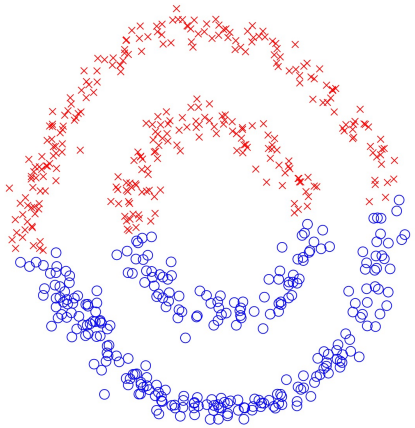


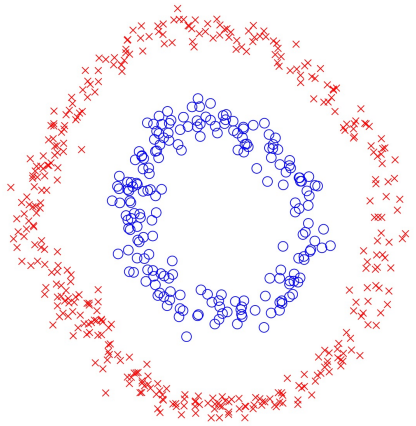
Limitation

What about distributions that have strong data **connectivity**









K-Means

Idea result

Spectral (Graph) Clustering

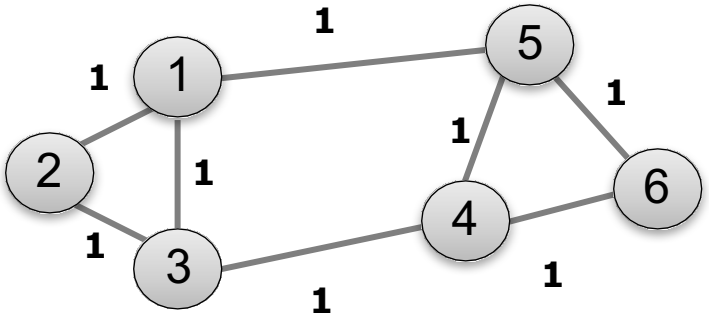
Preliminaries

Eigenvectors and Eigenvalues

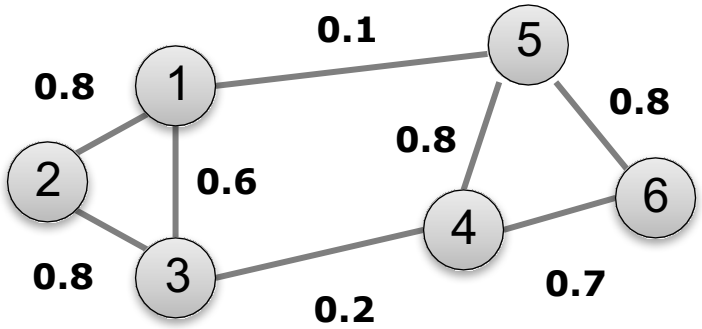
- Let $\boldsymbol{A} \in \mathbb{R}^{N \times N}$
  - $\boldsymbol{v} \in \mathbb{R}^{N}$ is an **eigenvector** of $\boldsymbol{A}$ if $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$
  - If $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$, λ is an **eigenvalue** associated to $\boldsymbol{v}$
    - If k eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ with the same eigenvalue λ, i.e., $\boldsymbol{A}\boldsymbol{v}_i = \lambda \boldsymbol{v}_i$ for all i, then λ has (algebraic) **multiplicity** of k
    - N-by-N matrix has N eigenvectors and N eigenvalues
      - counting the multiplicity
      - eigenvalues can be **complex**
    - If $\boldsymbol{A}$ is **symmetric**
      - eigenvalues are **real-valued**
    - If $\boldsymbol{A}$ is **positive semi-definite**
      - eigenvalues are **non-negative**
      - eigenvectors are **orthogonal** $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$ for any $i \neq j$

Graph

A graph G=(V,E) consists of nodes V and edges G



Unweighted graph

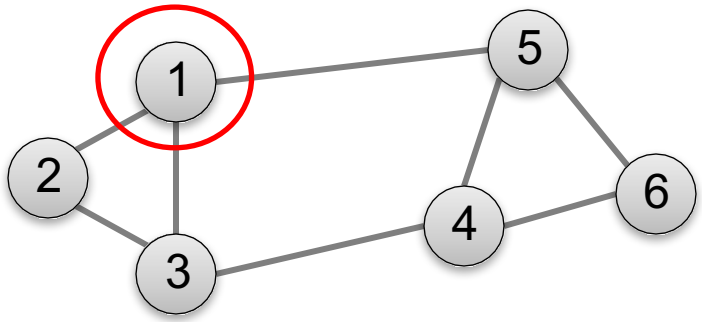


Weighted graph

Matrix Representations

Adjacency matrix (A)

- $N \times N$ matrix
- $A=[a_{ij}]$, $a_{ij}=1$ if an edge between node i and j



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

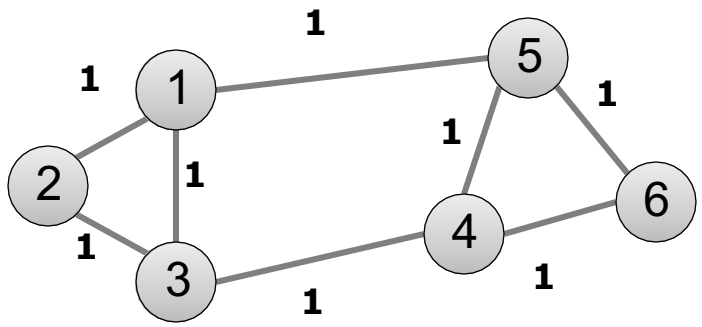
Important properties

- Symmetric matrix
- Eigenvectors are real-valued and orthogonal

Matrix Representations

Degree matrix (D): Unweighted graph

- $N \times N$ diagonal matrix
- $\mathbf{D}=[\mathbf{d_{ii}}]$, $d_{ii} = \sum_j A_{ij}$ = degree of node i

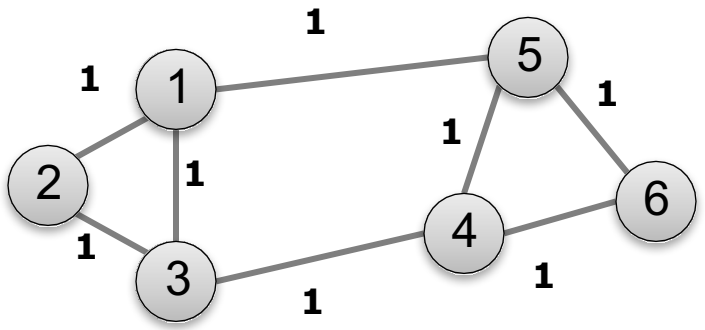


	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

Graph Laplacian matrix (L): Unweighted graph

- $N \times N$ diagonal matrix
- **L=D - A**



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

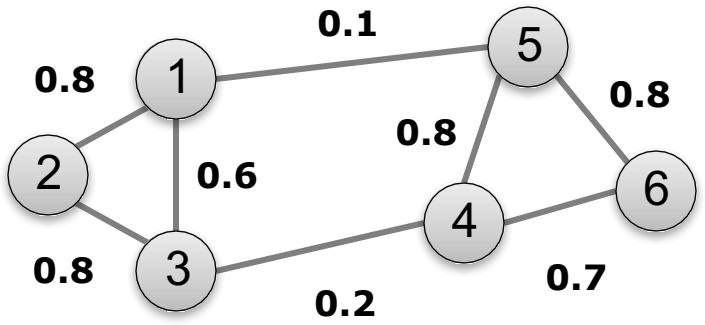
Important properties

- Symmetric matrix
- Eigenvectors are real-valued and orthogonal
- Row sum to 0

Matrix Representations

Weight (Similarity) matrix (W): Weighted graph

- $N \times N$ matrix
- $\mathbf{W}=[W_{ij}], W_{ij}=$ edge weight between i and j



	1	2	3	4	5	6
1	0	0.8	0.6	0	0.1	0
2	0.8	0	0.8	0	0	0
3	0.6	0.8	0	0.2	0	0
4	0	0	0.2	0	0.8	0.7
5	0.1	0	0	0.8	0	0.8
6	0	0	0	0.7	0.8	0

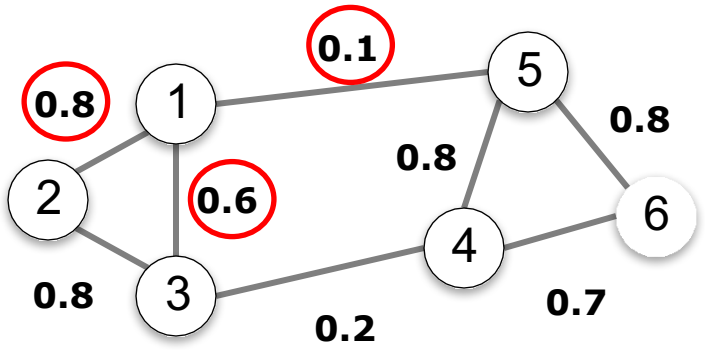
Important properties

- Symmetric matrix
- Eigenvectors are real-valued and orthogonal

Matrix Representations

Weighted degree matrix (D): Weighted graph

- $N \times N$ diagonal matrix
- $\mathbf{D}=[\mathbf{d_{ii}}]$, $d_{ii} = \sum_j W_{ij}$ = weighted degree of node i

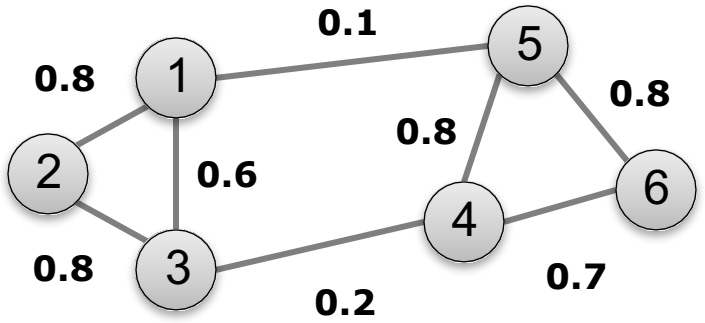


	1	2	3	4	5	6
1	**1.5**	0	0	0	0	0
2	0	**1.6**	0	0	0	0
3	0	0	**1.6**	0	0	0
4	0	0	0	**1.7**	0	0
5	0	0	0	0	**1.7**	0
6	0	0	0	0	0	**1.5**

Matrix Representations

Graph Laplacian matrix (L): Weighted graph

- $N \times N$ diagonal matrix
- **L=D - W**



	1	2	3	4	5	6
1	1.5	-0.8	-0.6	0	-0.1	0
2	-0.8	1.6	-0.8	0	0	0
3	-0.6	-0.8	1.6	-0.2	0	0
4	0	0	-0.2	1.7	-0.8	-0.7
5	-0.1	0	0	-0.8	1.7	-0.7
6	0	0	0	-0.8	-0.7	1.5

Important properties

- Symmetric matrix
- Eigenvectors are real-valued and orthogonal
- Row sum to 0

Graph-based Clustering

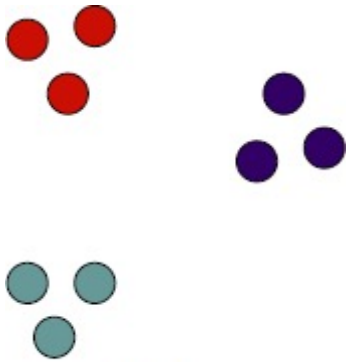
Graph-based Clustering

- **Clustering**: Given data points x_1, …, x_n, partition data into groups so that
  - Data points in a group are similar
  - Data points in different groups are dissimilar

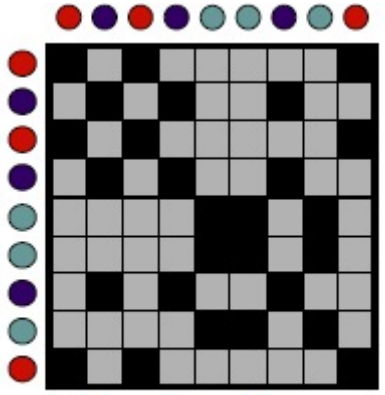
- **Similarity graph**: Represent data points x_1, …, x_n via a similarity graph G(V,E)
  - Each node represents a data point
  - Each edge weight indicates similarity between two data points

- **Graph-based clustering**: Given a similarity graph built by data points x_1, …, x_n, partition the similarity graph into groups so that
  - Edges within a group have large weights
  - Edges across different groups have small weights

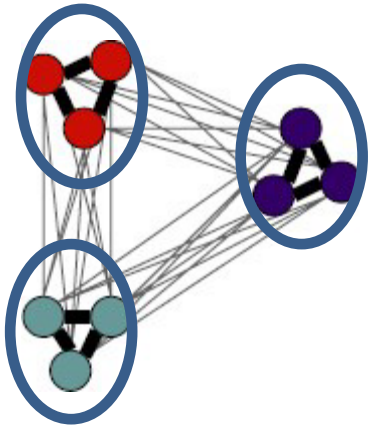
Graph-based Clustering



Data



Similarities

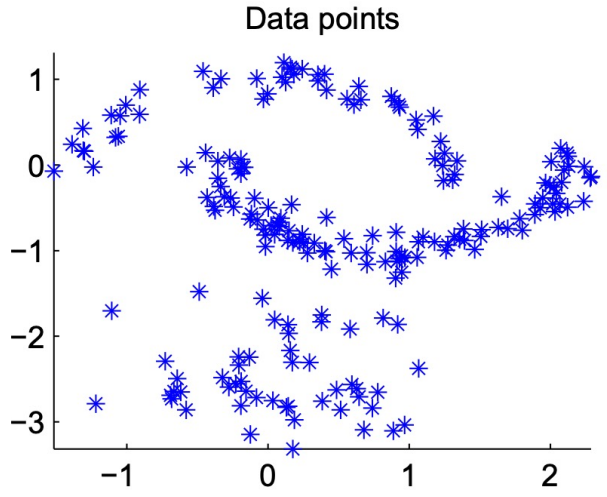


Similarity graph

- **Graph-based clustering**: Given a similarity graph built by data points x_1, …, x_n, partition the similarity graph into groups so that
  - Edges within a group have large weights
  - Edges across different groups have small weights

Similarity Graph Construction

- **ε-neighborhood graph**
  - If a pairwise distance between two nodes x_1 and x_2 is $d(x_1, x_2) \leq \varepsilon$, connect x_1 and x_2
- Similarity: 1- ε



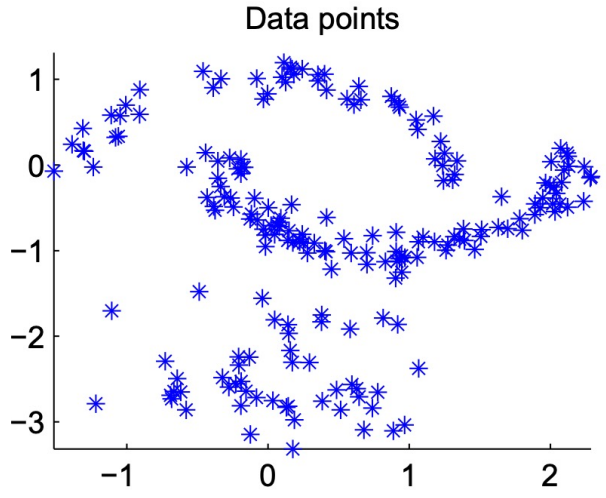
Data points on 2D with three clusters:

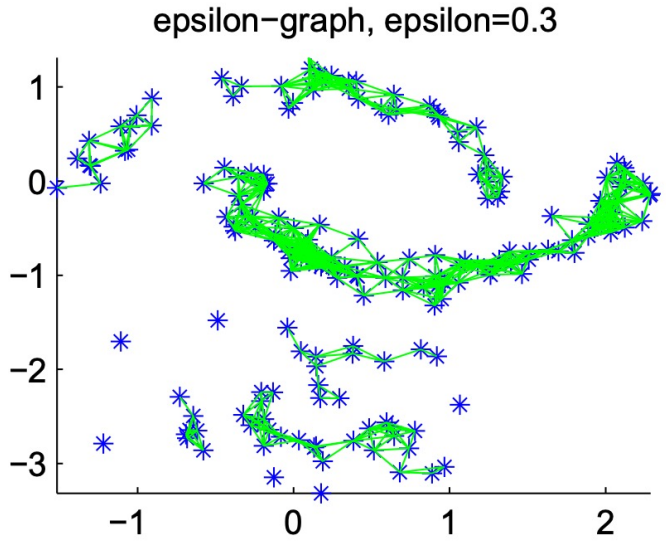
- 2 “moons”
- 1 Gaussian distribution

Density of the bottom moon is larger than the one of the top moon

Similarity Graph Construction

- **ε-neighborhood graph**
  - If a pairwise distance between two nodes x_1 and x_2 is $d(x_1, x_2) \leq \varepsilon$, connect x_1 and x_2
- Similarity: 1- ε





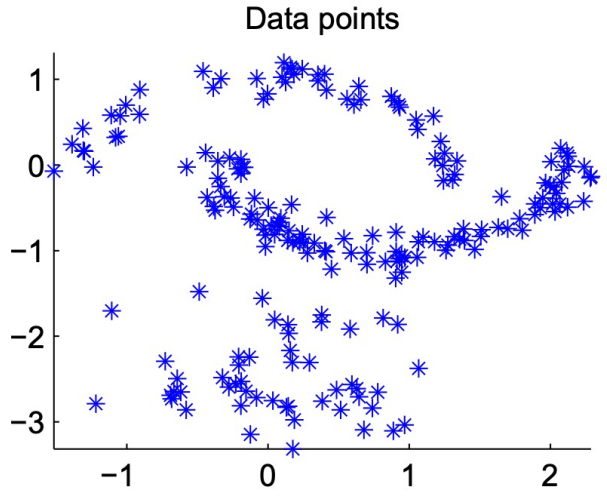
- Data points on the middle moon are tightly connected
- Data points in the Gaussian are barely connected

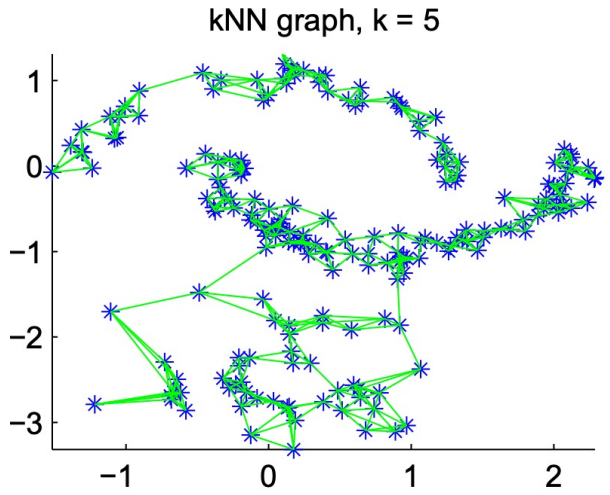
This problem always occurs if we have data on different scales, that is, distances between data points are different in different regions of the space

Similarity Graph Construction

- ***k*-nearest neighbor (k-NN) graph**
  - If x_1 is one of the k nearest neighbors of x_2, connect x_1 and x_2
  - Similarity: Gaussian kernel

$$w_{ij} = \exp(-\| x_i - x_j \|^2 / \sigma^2)$$



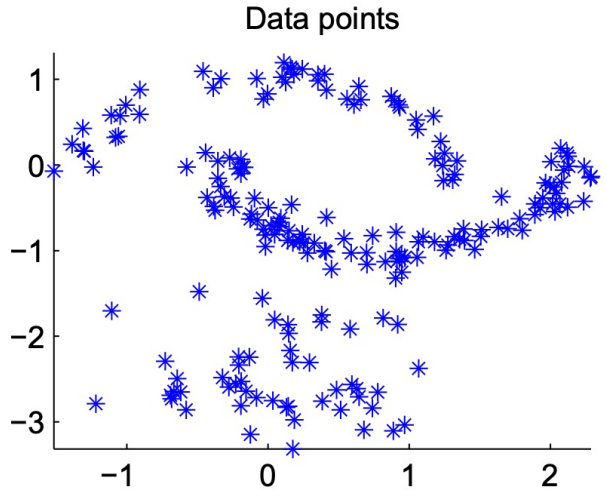


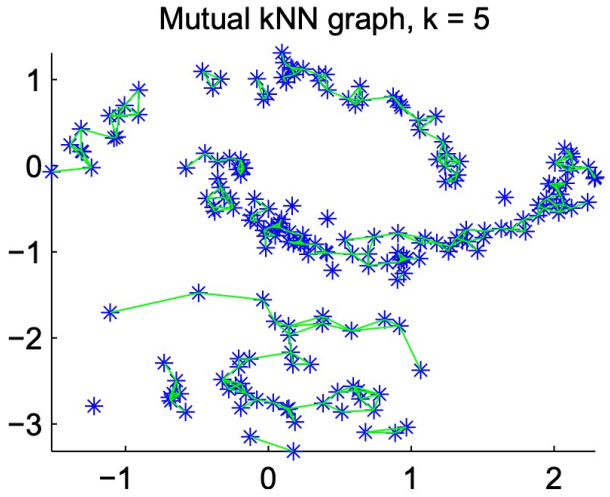
- k-NN graph can connect points "on different scales"
- Points in the low-density Gaussian are connected with points in the high-density moon

*k*NN graph can break into disconnected components if there are high density regions which are reasonably far away from each other

Similarity Graph Construction

- ***Mutual k-NN graph***
  - If x_1 is one of the k nearest neighbors of x_2
  - **AND** x_2 is one of the k nearest neighbors of x_1, connect x_1 and x_2

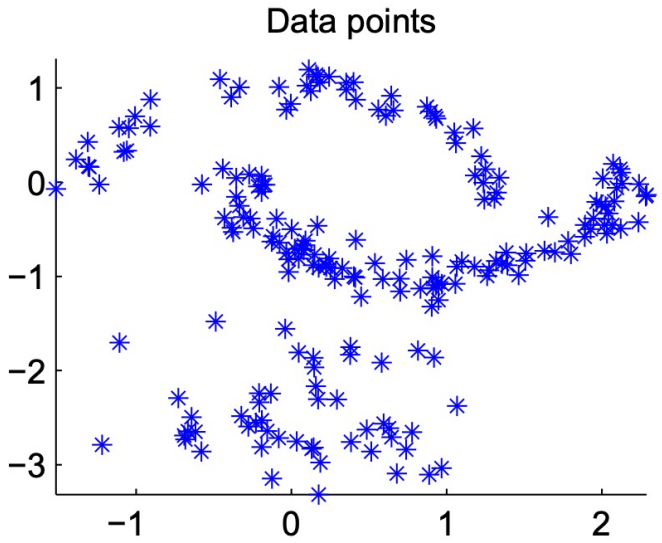




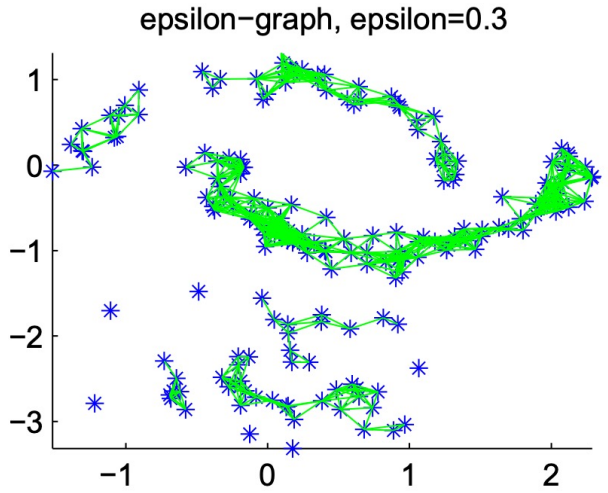
- Connect data points within regions of constant density
- Does not connect regions of different densities

Mutual k-NN graphs being "in between" the ε-neighborhood graph and kNN graph

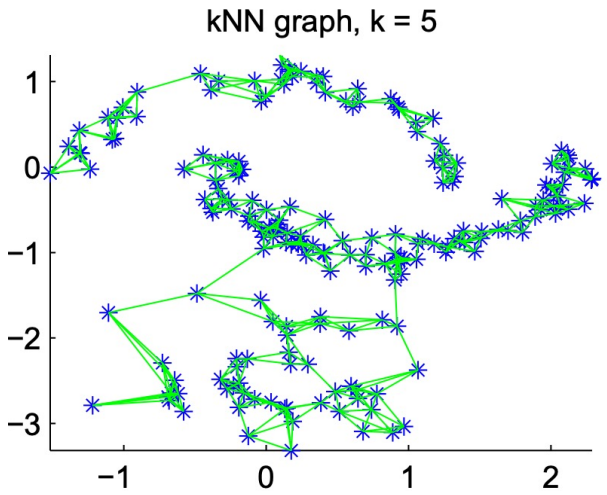
- It is able to act on different scales, but does not mix those scales with each other



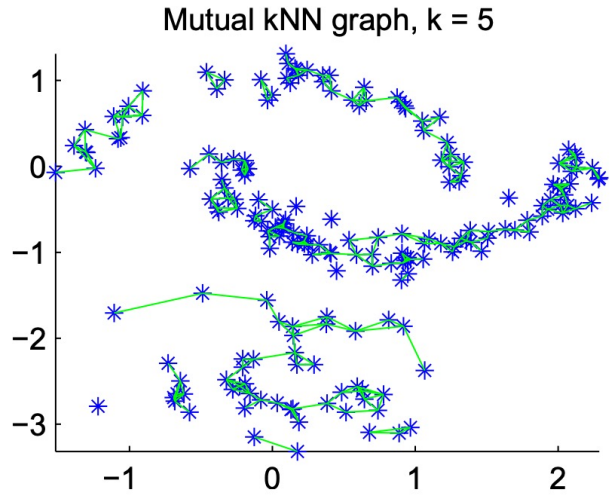
2 moons with different densities + 1 Gaussian



High density region densely connected; Low density region barely connected



Connect data points “on different scales”



Connect data points within regions of constant density

Goal & Objective

- **Goal**: Given a similarity graph G constructed by data points X_1, …, X_n, partition the graph into groups so that
  - Edges within a group have large weights
  - Edges across different groups have small weights

- Two things needed
  - An ***objective function*** to determine the optimal partition
    - I.e., the best way to “cut” the edges of a graph
  - An ***algorithm*** to find the optimal partition

Graph Cut & Normalized (Ratio) Cut

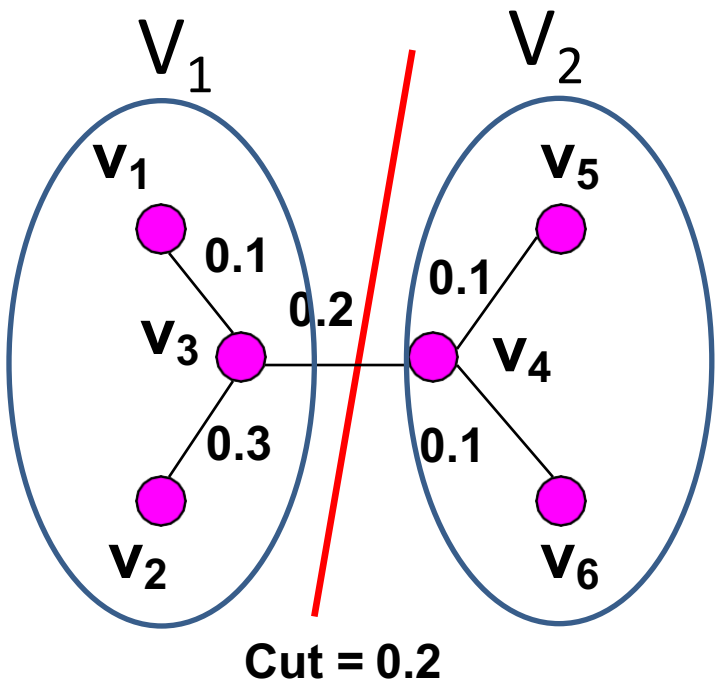
Graph Cut

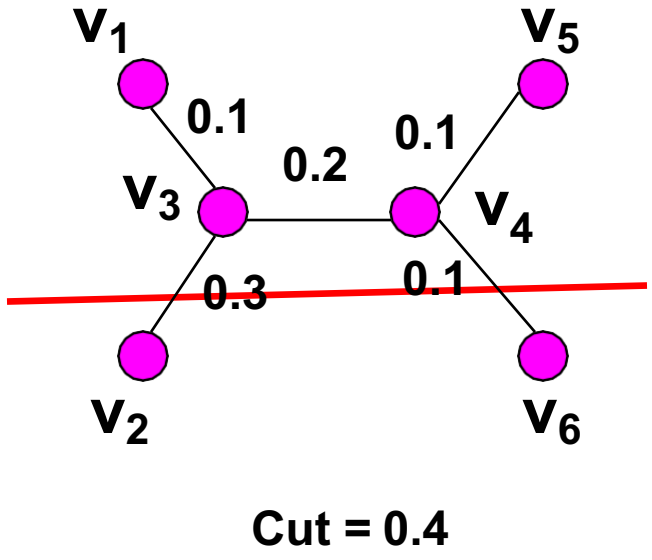
Goal: Partition the set of nodes V into two groups: V_1 and V_2

Graph Cut: sum of weights of edges across different groups

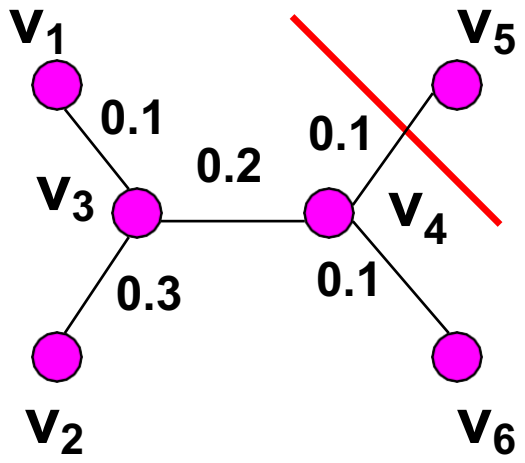
Objective function: minimize graph cut

$$\min \mathrm{Cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} W_{ij}$$





Limitation



Cut = 0.1

Optimal solution splits up a single node from the rest of the graph

- Not a desirable solution

Reason: Do not consider size/density of different cuts

Ratio Cut (RCut) & Normalized Cut (NCut)

Not only minimize the graph cut, but also look for “balanced” clusters

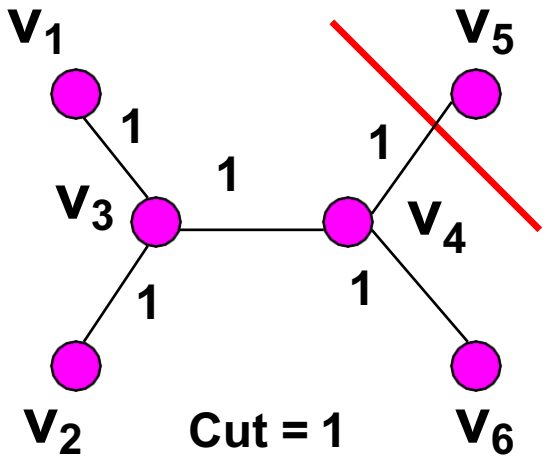
$$\mathrm{RCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{Cut}(V_1, V_2)}{|V_2|}$$

$$|V_i| : \text{ number of nodes in } V_i$$

$$\mathrm{NCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{\mathrm{Vol}(V_1)} + \frac{\mathrm{Cut}(V_1, V_2)}{\mathrm{Vol}(V_2)}$$

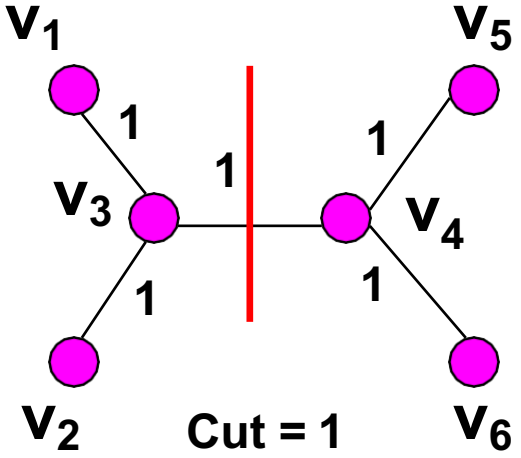
$$\mathrm{Vol}(V_k) = \sum_{i \in V_k} d_i \qquad d_i = \sum_j W_{ij}$$

Example: Unweighted Graph



$$\text{RCut}(V_1, V_2) = \frac{\text{Cut}(V_1, V_2)}{|V_1|} + \frac{\text{Cut}(V_1, V_2)}{|V_2|}$$

$$\text{NCut}(V_1, V_2) = \frac{\text{Cut}(V_1, V_2)}{\text{Vol}(V_1)} + \frac{\text{Cut}(V_1, V_2)}{\text{Vol}(V_2)}$$



Rcut = ?, Ncut = ?

$$|V_1| = |\{v_5\}| = 1$$
$$|V_2| = |\{v_1, v_3, v_2, v_4, v_6\}| = 5$$

RCut = 1/1 + 1/5 = 1.2

$$\text{Vol}(V_1) = d_5 = 1$$
$$\text{Vol}(V_2) = d_1 + d_3 + d_2 + d_4 + d_6 = 1 + 3 + 1 + 3 + 1 = 9$$

NCut = 1/1 + 1/9 = 1.11

>

>

Rcut = ?, Ncut = ?

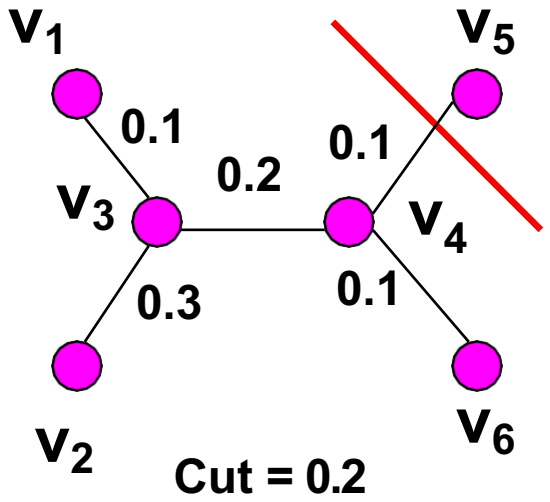
$$|V_1| = |\{v_5, v_4, v_6\}| = 3$$
$$|V_2| = |\{v_1, v_3, v_2\}| = 3$$

RCut = 1/3 + 1/3 = 0.67

$$\text{Vol}(V_1) = d_5 + d_4 + d_6 = 1 + 3 + 1 = 5$$
$$\text{Vol}(V_2) = d_1 + d_3 + d_2 = 1 + 3 + 1 = 5$$

NCut = 1/5 + 1/5 = 0.2

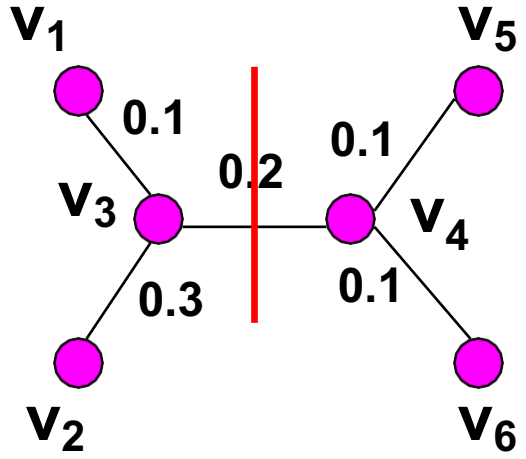
Example: Weighted Graph



Cut = 0.2

$$\mathrm{RCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{Cut}(V_1, V_2)}{|V_2|}$$

$$\mathrm{NCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{\mathrm{Vol}(V_1)} + \frac{\mathrm{Cut}(V_1, V_2)}{\mathrm{Vol}(V_2)}$$



Cut = 0.2

Rcut = ?, Ncut = ?

Rcut = ?, Ncut = ?

RCut = 0.1/1 + 0.1/5 = 0.12 < RCut = 0.2/3 + 0.2/3 = 0.13

Ncut = 0.1/0.1 + 0.1/1.5 = 1.07 > NCut = 0.2/1 + 0.2/0.6 = 0.53

Prefer Ncut in Practice!

Normalized Cut: Problem Formulation

$$\mathrm{NCut}(A,B) = \mathrm{Cut}(A,B)\left(\frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)}\right)$$

Let $f = [f_1\ f_2\ \ldots f_N]^T$ with $f_i = \begin{cases} \frac{1}{\mathrm{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\mathrm{vol}(B)} & \text{if } i \in B \end{cases}$

Graph Laplacian

$L = D - W$

$D_{ii} = \sum_j W_{ij}$

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{ij} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 = \sum_{i\in A, j\in B} w_{ij}\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)^2$$

$$\mathbf{f}^T\mathbf{D}\mathbf{f} = \sum_j d_i \mathbf{f}_i^2 = \sum_{i\in A} \frac{d_i}{\mathrm{vol}(A)^2} + \sum_{j\in B} \frac{d_i}{\mathrm{vol}(B)^2} = \frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}$$

$$\mathrm{NCut}(A,B) = \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}}$$

Normalized Cut: NP Hard

$$\min_{A,B} \mathrm{NCut}(A,B) = \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}} \qquad f_i = \begin{cases} \frac{1}{\mathrm{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\mathrm{vol}(B)} & \text{if } i \in B \end{cases}$$

Equivalent form:

$\min f^T L f \quad \text{s.t. } f^T D f = 1,$ f discrete NP hard!

Continuous relaxation:

$\min f^T L f \quad \text{s.t. } f^T D f = 1,$ f continuous

Solution f: second eigenvector of generalized eigenvalue problem

$$\mathbf{L}\mathbf{f} = \lambda\mathbf{D}\mathbf{f}$$ Spectral clustering

Graph Laplacian & Spectral Clustering (Andrew Ng, NIPS'01)

Spectral Clustering: Overview

- Build a similarity graph (with similarity matrix W) using unlabeled data
- Define a diagonal weighted degree matrix D

$$D_{ij} = \begin{cases} \sum_{k=1}^{N} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

- Obtain (normalized) graph Laplacian L = D - W (or L' = D^{-1} L)
- Study spectral properties of (normalized) graph Laplacian
  - Map data into the eigenvalue-eigenvector domain of graph Laplacian
- Spectral clustering: use eigenvectors to perform clustering

Properties of Graph Laplacian

- L is symmetric, i.e., ($L^T = L$)
  - Eigenvectors are real-valued and orthogonal

- L is a positive semi-definite (PSD) matrix
  - For all real-valued vectors x: $x^T Lx \geq 0$

$$x^T Lx = x^T (D - W)x = x^T Dx - x^T Wx$$

$$= \sum_i d_i x_i^2 - \sum_{i,j} w_{ij} x_i x_j \qquad (\text{where } d_i = \textstyle\sum_j w_{ij})$$

$$= \frac{1}{2}\left(\sum_{i,j} w_{ij} x_i^2 - 2\sum_{i,j} w_{ij} x_i x_j + \sum_{i,j} w_{ij} x_i^2 \right)$$

$$= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(x_i - x_j \right)^2 \geq 0$$

  - All eigenvalues of L are ≥ 0

Properties of (Connected) Graph Laplacian

- Eigenvalue 0 of L is with the respective eigenvector e = [1 1 …1]^T

$$e = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} \qquad We = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1d} \\ w_{21} & w_{22} & \dots & w_{2d} \\ \dots & \dots & \dots & \dots \\ w_{d1} & w_{d2} & \dots & w_{dd} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} w_{1j} \\ \sum_{j=1}^{d} w_{2j} \\ \dots \\ \sum_{j=1}^{d} w_{dj} \end{bmatrix} = \begin{bmatrix} D_{11} \\ D_{22} \\ \dots \\ D_{dd} \end{bmatrix}$$

$$De = \begin{bmatrix} D_{11} & 0 & \dots & 0 \\ 0 & D_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & D_{dd} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} = \begin{bmatrix} D_{11} \\ D_{22} \\ \dots \\ D_{dd} \end{bmatrix}$$

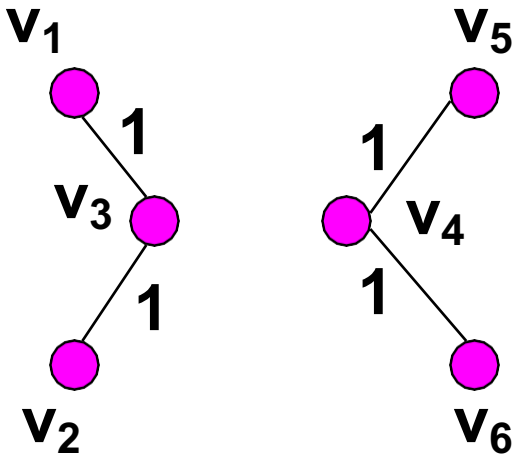
$$De = We \Rightarrow Le = 0$$

Eigenvalue equation

$$Le = \lambda e$$

Since e ≠[0..0]^T, thus λ= 0; Furthermore, since L is PSD, 0 is the smallest eigenvalue

Matrix Representation (2 Connected Components)



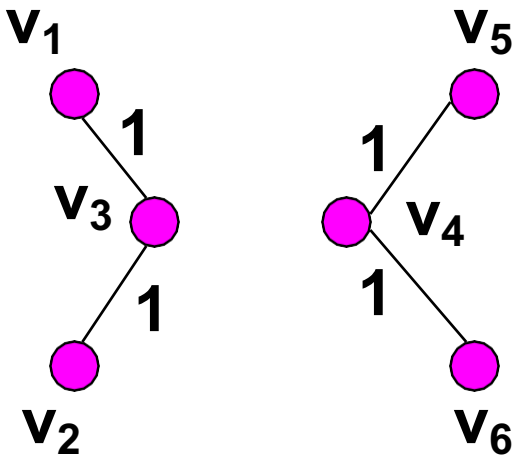
$$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Two block-diagonal matrices

$$D_{ij} = \begin{cases} \sum_{k=1}^{n} w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Graph Laplacian (2 Connected Components)



$$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

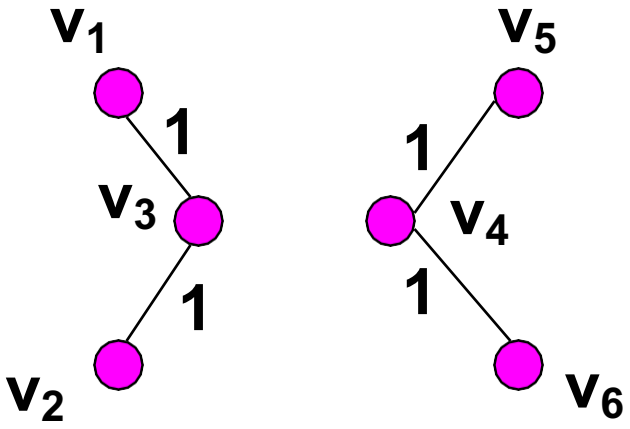
Two block-diagonal matrices

Graph Laplacian,
L = D − W

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

Graph Laplacian also has a block structure

Graph Laplacian (2 Connected Components)



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

Graph Laplacian also has a block structure

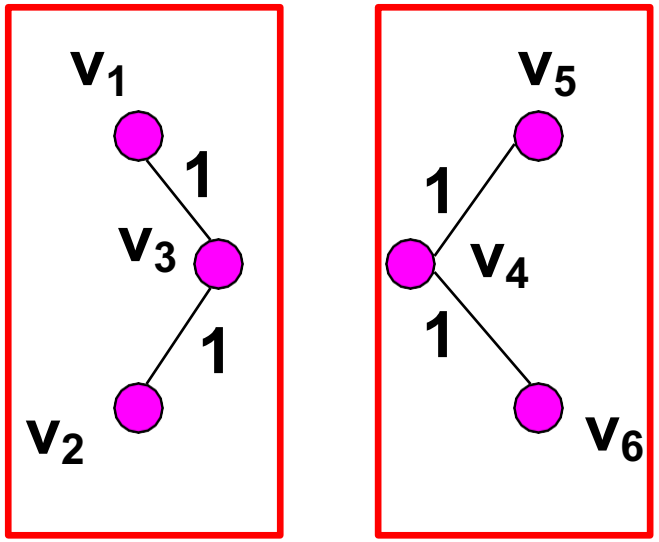
Eigenvalues of L

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Eigenvectors of L

$$V = \begin{bmatrix} 0.58 & 0 & 0.71 & 0 & 0 & -0.41 \\ 0.58 & 0 & -0.71 & 0 & 0 & -0.41 \\ 0.58 & 0 & 0 & 0 & 0 & 0.82 \\ 0 & -0.58 & 0 & 0 & -0.82 & 0 \\ 0 & -0.58 & 0 & -0.71 & 0.41 & 0 \\ 0 & -0.58 & 0 & 0.71 & 0.41 & 30 \end{bmatrix}$$

Graph Laplacian (2 Connected Components)



If we cluster the data using only the first 2 eigenvectors, we get two desired clusters

Eigenvalues of L

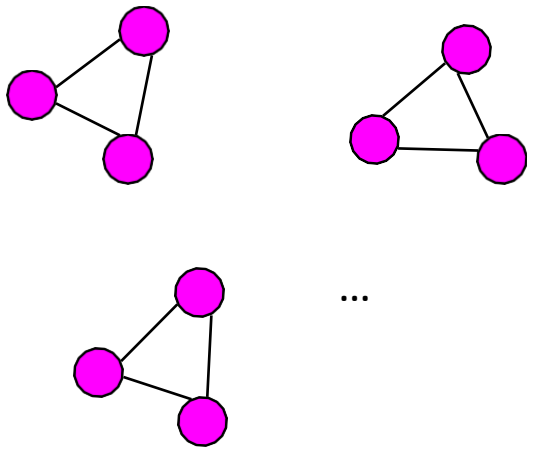
$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Eigenvectors of L

$$V = \begin{bmatrix} 0.58 & 0 & 0.71 & 0 & 0 & -0.41 \\ 0.58 & 0 & -0.71 & 0 & 0 & -0.41 \\ 0.58 & 0 & 0 & 0 & 0 & 0.82 \\ 0 & -0.58 & 0 & 0 & -0.82 & 0 \\ 0 & -0.58 & 0 & -0.71 & 0.41 & 0 \\ 0 & -0.58 & 0 & 0.71 & 0.41 & 0 \end{bmatrix}$$

Graph Laplacian (k Connected Components)

More generally, the similarly graph has k connected components, then L is a k-block diagonal matrix



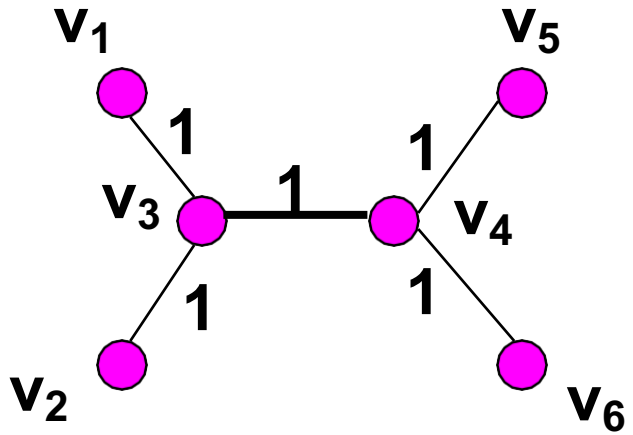
$$L = \begin{bmatrix} L_1 & 0 & 0 & 0 \\ 0 & L_2 & 0 & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & L_k \end{bmatrix}$$

Then, there are k eigenvalues of L are 0

- The corresponding eigenvectors

$$\begin{bmatrix} e \\ 0 \\ \dots \\ \dots \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ e \\ \dots \\ \dots \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ \dots \\ \dots \\ 0 \\ e \end{bmatrix}$$

Connected Graph Laplacian (2 Clusters)



Clusters are NO perfectly separated

$$L = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

Graph Laplacian DOES NOT have a block structure

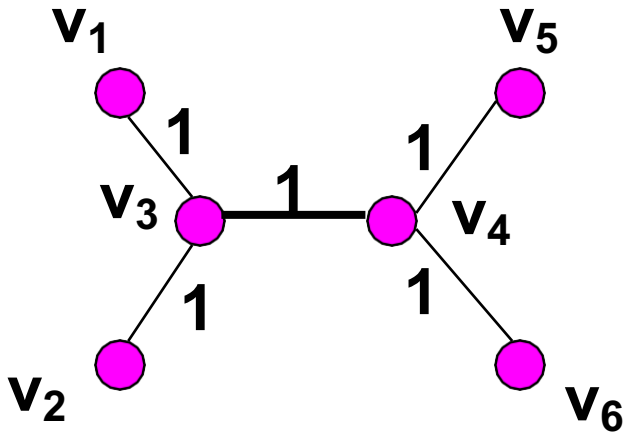
Eigenvalues of L

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4.56 \end{bmatrix}$$

Eigenvectors of L

$$V = \begin{bmatrix} 0.41 & 0.46 & 0.65 & -0.28 & 0.29 & -0.18 \\ 0.41 & 0.46 & -0.65 & 0.28 & 0.29 & -0.18 \\ 0.41 & 0.26 & 0 & 0 & -0.58 & 0.66 \\ 0.41 & -0.26 & 0 & 0 & -0.58 & -0.66 \\ 0.41 & -0.46 & 0.28 & 0.65 & 0.29 & 0.18 \\ 0.41 & -0.46 & -0.28 & -0.65 & 0.29 & 0.18 \end{bmatrix}$$

Connected Graph Laplacian (2 Clusters)



If we cluster the data using only the first 2 eigenvectors, we **STILL** get two desired clusters

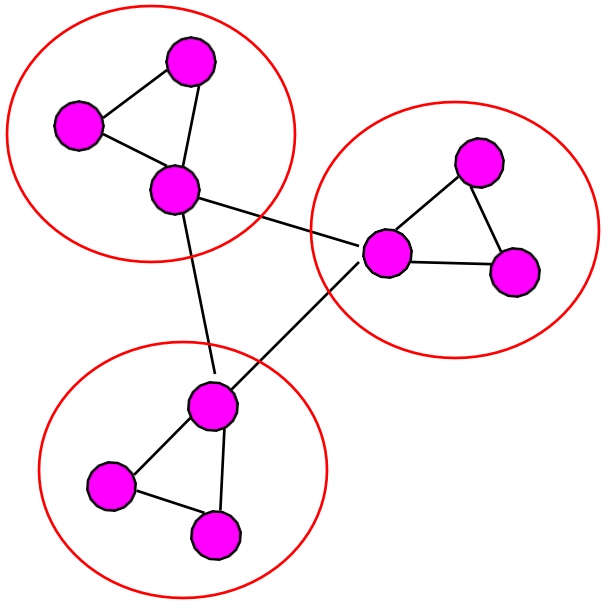
Eigenvalues of L

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4.56 \end{bmatrix}$$

Eigenvectors of L

$$V = \begin{bmatrix} 0.41 & 0.46 & 0.65 & -0.28 & 0.29 & -0.18 \\ 0.41 & 0.46 & -0.65 & 0.28 & 0.29 & -0.18 \\ 0.41 & 0.26 & 0 & 0 & -0.58 & 0.66 \\ 0.41 & -0.26 & 0 & 0 & -0.58 & -0.66 \\ 0.41 & -0.46 & 0.28 & 0.65 & 0.29 & 0.18 \\ 0.41 & -0.46 & -0.28 & -0.65 & 0.29 & 0.18 \end{bmatrix}$$

Connected Graph Laplacian (>2 Clusters)



Eigenvalues of the graph Laplacian:
0, 0.5505, 0.5505, 3, 3, 3, 3, 5.4495, 5.4495

1st eigenvector values: 3+, 3+, 3+

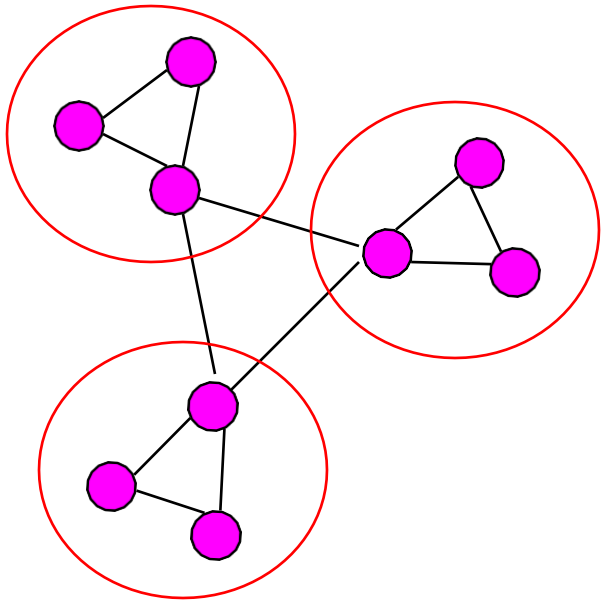
2nd eigenvector values: 3+, 3-, 3+

3rd eigenvector values: 3+, 3+, 3-

Eigenvectors of graph Laplacian

$$V = \begin{bmatrix} 0.33 & 0.32 & 0.45 & 0.71 & -0.18 & \cdots \\ 0.33 & 0.32 & 0.45 & 0.71 & -0.18 & \cdots \\ 0.33 & 0.14 & 0.20 & 0 & 0.35 & \cdots \\ 0.33 & -0.25 & 0.02 & 0 & 0.35 & \cdots \\ 0.33 & -0.55 & 0.05 & 0 & 0.26 & \cdots \\ 0.33 & -0.55 & 0.05 & 0 & -0.61 & \cdots \\ 0.33 & 0.10 & -0.23 & 0 & 0.35 & \cdots \\ 0.33 & 0.23 & -0.50 & 0 & -0.35 & \cdots \\ 0.33 & 0.23 & -0.50 & 0 & 0 & \cdots \end{bmatrix}$$

Connected Graph Laplacian (>2 Clusters)



Eigenvalues of the graph Laplacian:
0, 0.5505, 0.5505, 3, 3, 3, 3, 5.4495, 5.4495

Can be used by k-means to obtain 3 clusters

Eigenvectors of graph Laplacian

$$V = \begin{bmatrix} 0.33 & 0.32 & 0.45 & 0.71 & -0.18 & \cdots \\ 0.33 & 0.32 & 0.45 & 0.71 & -0.18 & \cdots \\ 0.33 & 0.14 & 0.20 & 0 & 0.35 & \cdots \\ 0.33 & -0.25 & 0.02 & 0 & 0.35 & \cdots \\ 0.33 & -0.55 & 0.05 & 0 & 0.26 & \cdots \\ 0.33 & -0.55 & 0.05 & 0 & -0.61 & \cdots \\ 0.33 & 0.10 & -0.23 & 0 & 0.35 & \cdots \\ 0.33 & 0.23 & -0.50 & 0 & -0.35 & \cdots \\ 0.33 & 0.23 & -0.50 & 0 & 0 & \cdots \end{bmatrix}$$

Spectral Clustering Algorithm

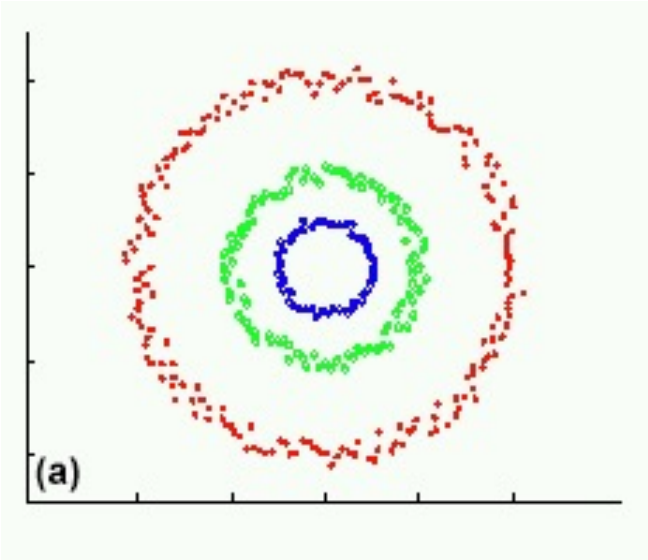
Given an unlabeled data set with N data points

- Construct the ***similarity*** graph with a similarity matrix W
  - ε-neighborhood graph, (mutual) KNN graph, etc.
- Compute graph Laplacian L = D – W or normalized Laplacian L' = D^{-1} L
- Compute the k eigenvectors v_1, v_2, .., v_k of the (normalized) graph Laplacian associated with the k ***smallest*** eigenvalues
  - Create a matrix V ∈ $\mathbb{R}^{N \times k}$ containing eigenvectors v_1, v_2, .., v_k as columns
- Cluster rows in V (as new data points in $\mathbb{R}^k$) into k clusters using k-means

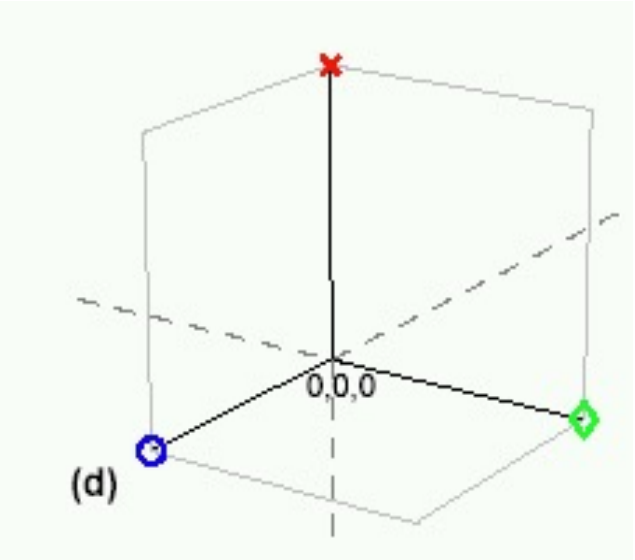
Understanding Spectral Clustering

Data are projected into the spectral/eigenvector domain, where they are easily separable, say using k-means.

Original data

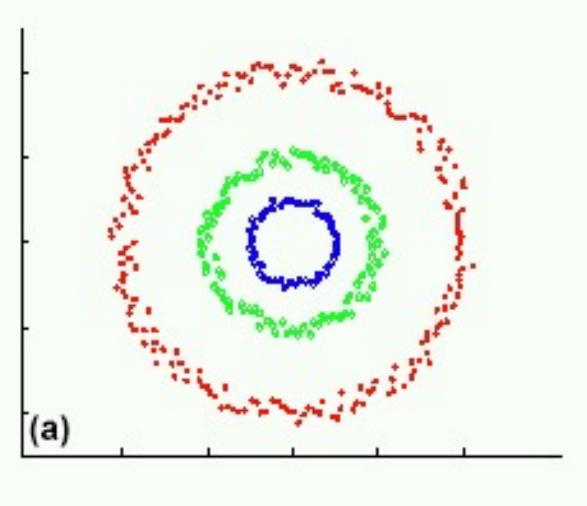


Projected data



Understanding Spectral Clustering

If graph is disconnected (k connected components), first k-eigenvalues are 0 and graph Laplacian is block diagonal



$$L = \begin{bmatrix} L_1 & & 0 \\ & L_2 & \\ 0 & & L_3 \end{bmatrix}$$

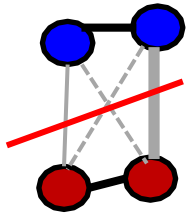
$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

First three eigenvectors

Understanding Spectral Clustering

If graph is connected (off-block diagonal elements not zero), then

1st Laplacian eigenvector are constant, but 2nd eigenvector gets the cut



1	1	.2	0
1	1	0	.1
.2	0	1	1
0	.1	1	1

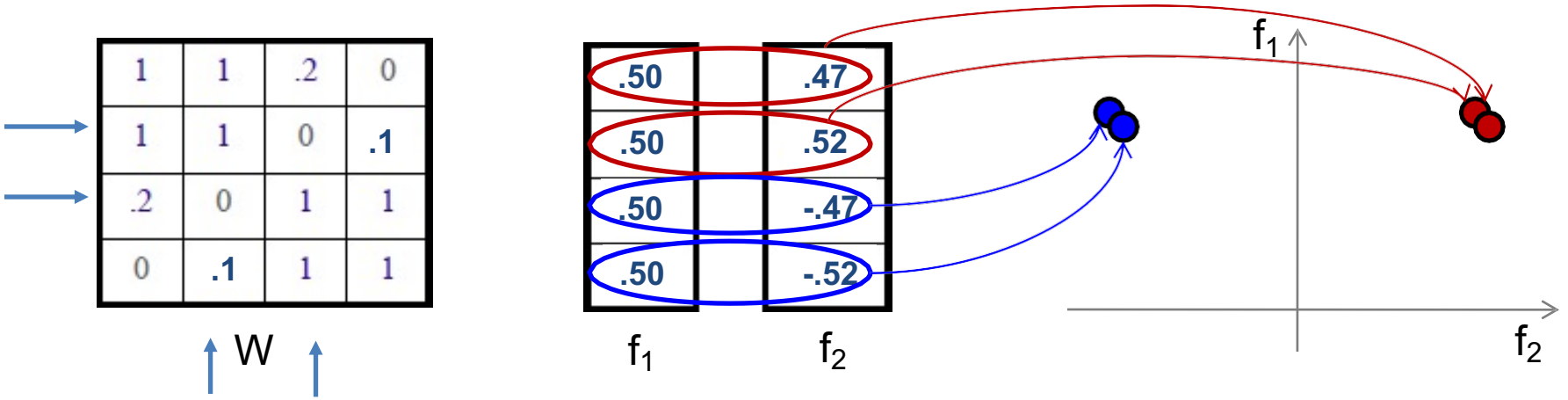
.50
.50
.50
.50

.47
.52
-.47
-.52

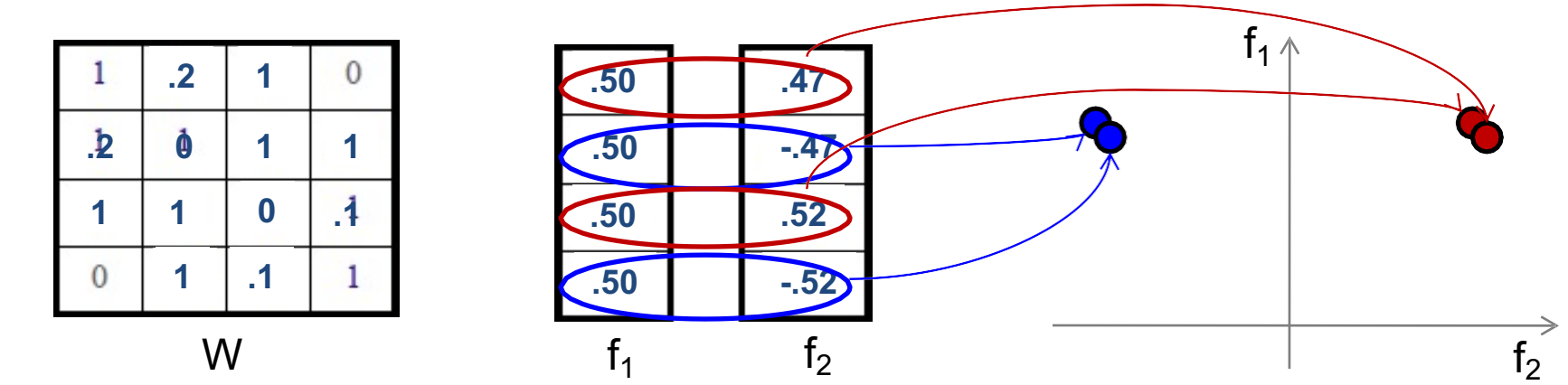
Sign of 2nd eigenvector indicates blocks

Understanding Spectral Clustering

Put data points into blocks using eigenvectors

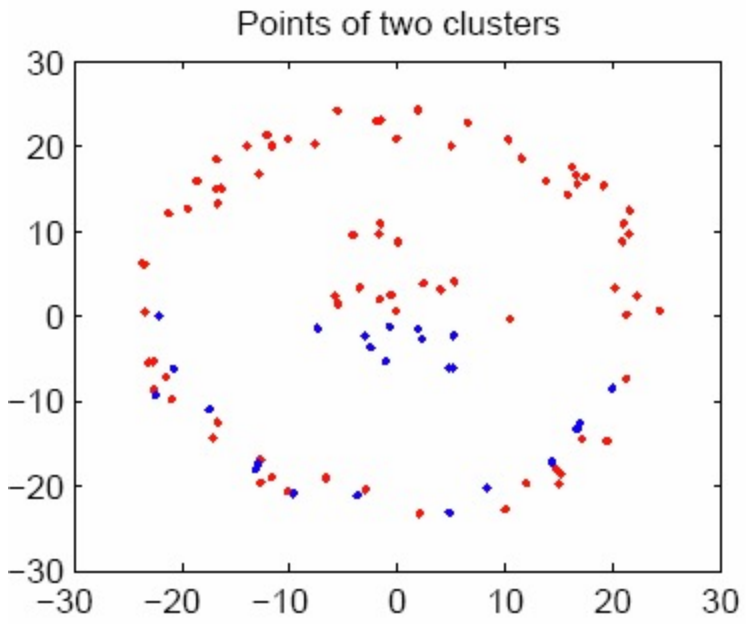


Embedding is same regardless of data ordering

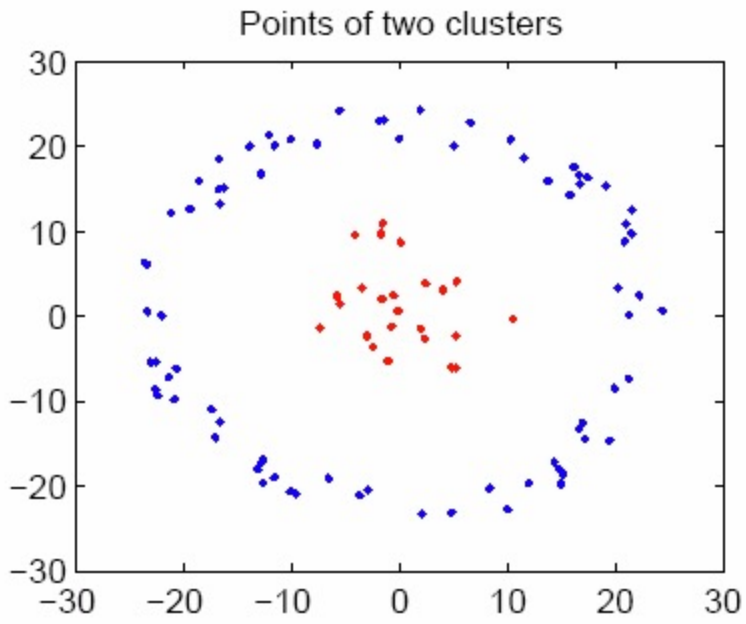


K-Means vs Spectral Clustering

Applying k-means to Laplacian eigenvectors allows us to find cluster with non-convex boundaries



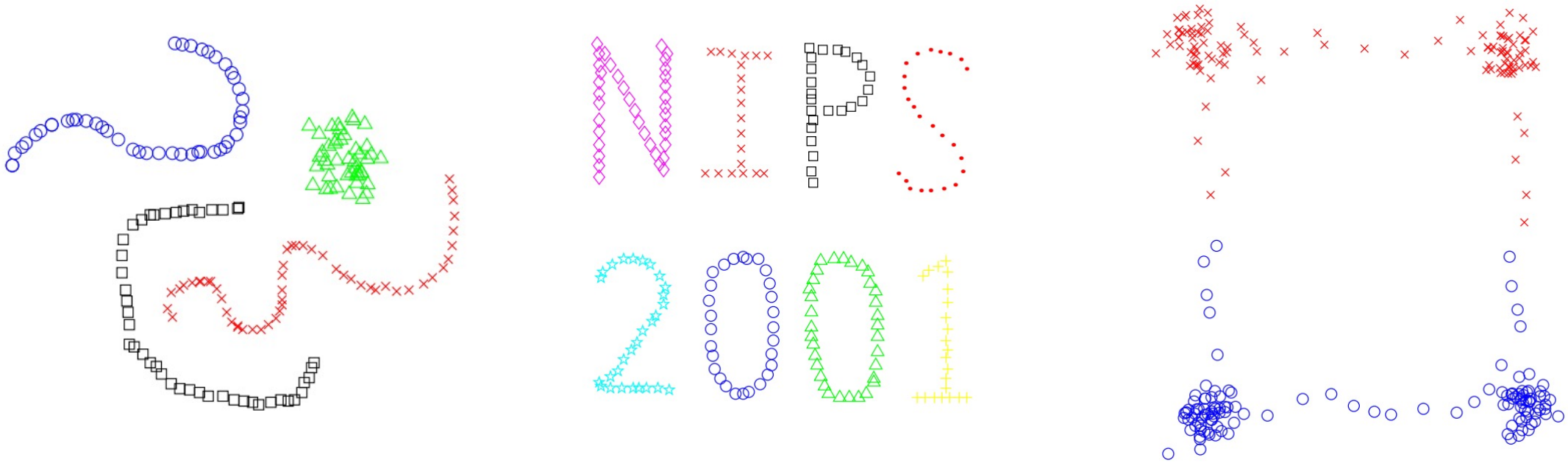
k-means output



Spectral clustering output

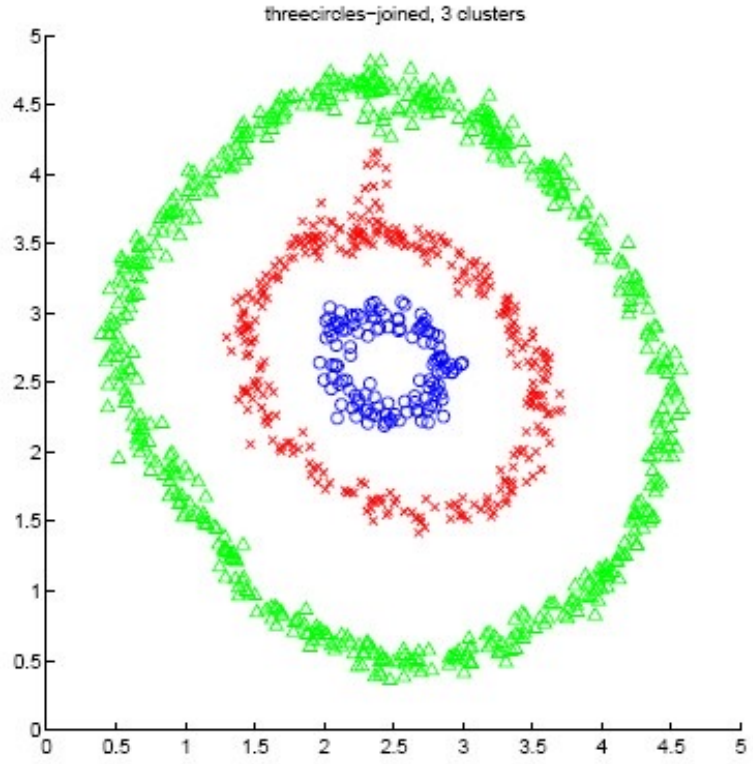
More Examples

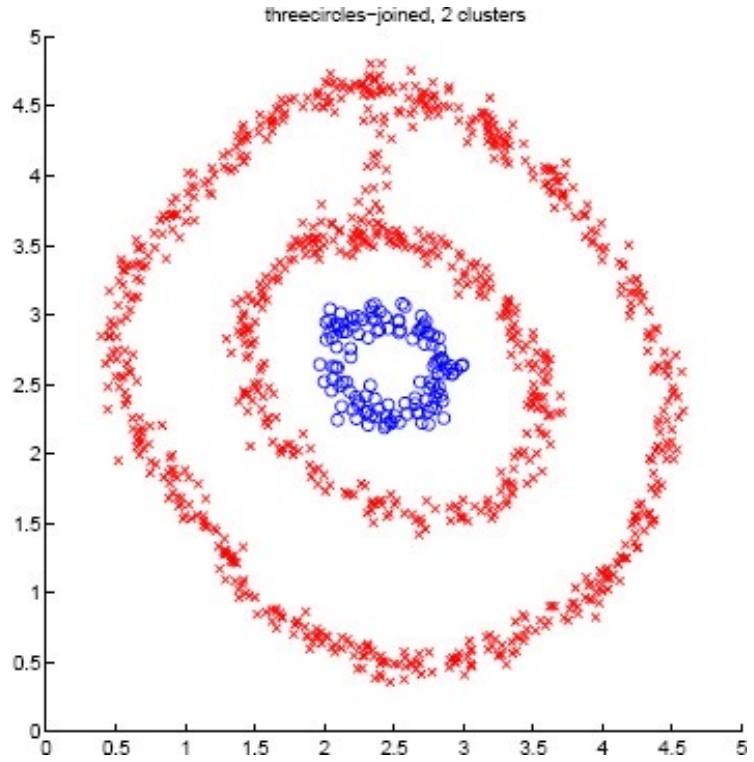
Applying k-means to Laplacian eigenvectors allows us to find cluster with non-convex boundaries



Strong data **connectivity**

Issue: Choice of k



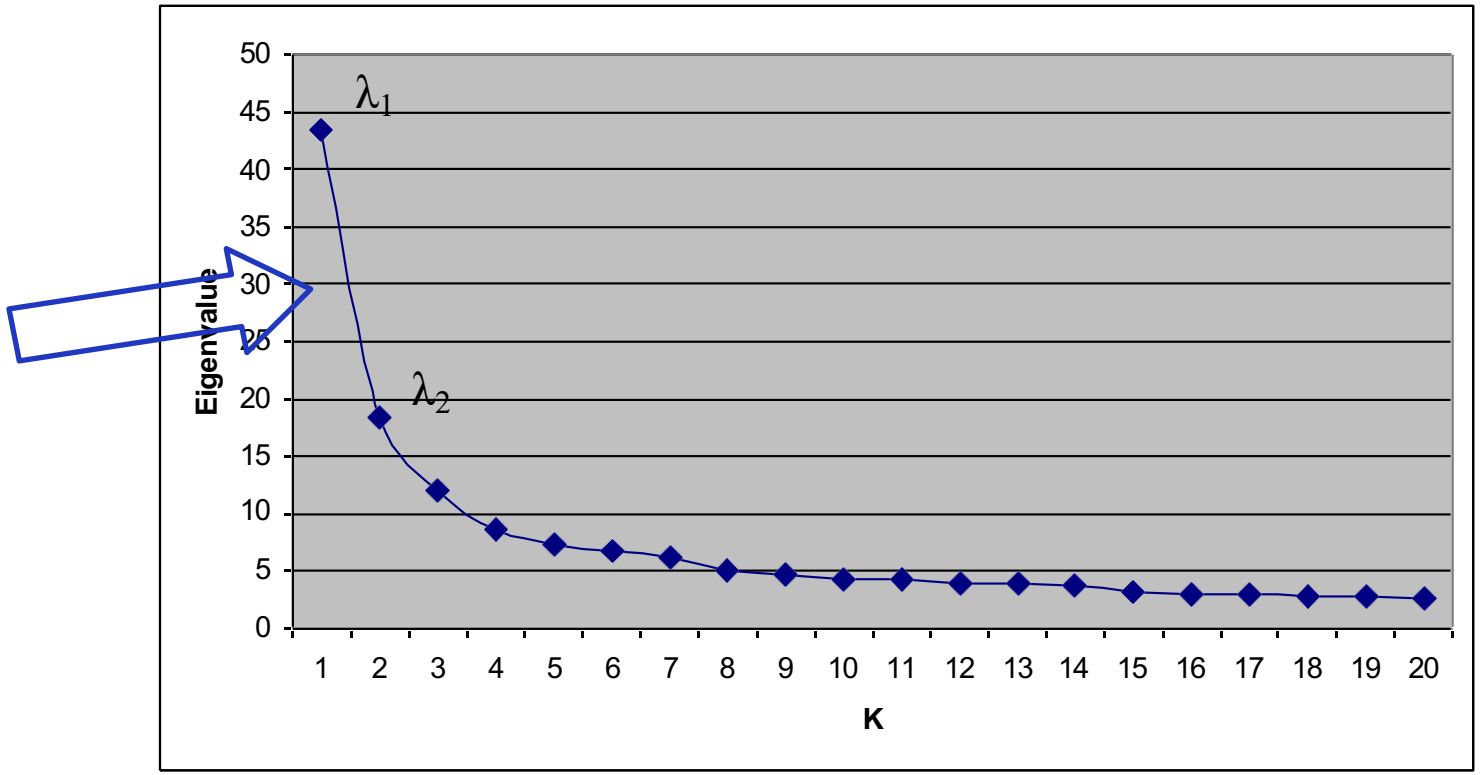


How to Select K?

- **Eigengap**: the difference between two consecutive eigenvalues
- Most stable clustering is given by k that maximizes the eigengap

$$\Delta_k = \left|\lambda_k - \lambda_{k-1}\right|$$

Choose k=2



Summary

- Clustering as graph partition
  - Graph cut: sensitive to outliers
  - Ratio cut and normalized cut (NP hard)
  - Approximate normalized cut via spectral clustering
- Graph Laplacian matrix
  - Symmetric, positive semi-definite
  - Connected graph: (1^{st} eig-value, 1^{st} eig-vector) = (0, all 1s)
  - Disconnected graph: #zero eig-values = #connected components
- Spectral clustering
  - Cluster data points using graph Laplacian matrix
  - Obtain data representation in k-dim space that can be easily clustered
  - Useful in hard non-convex clustering problems

Acknowledgement

Some slides are from

Arti Singh (CMU)
https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_2.pdf
Anil K. Jain (MSU)
https://www.cse.msu.edu/~cse802/S17/slides/Lec_20_21_22_Clustering.pdf