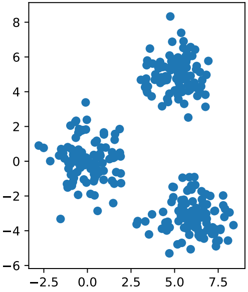
CS-E4075 - Special Course in Machine Learning, Data Science and Artificial Intelligence D: Signed graphs: spectral theory and applications

Spectral clustering

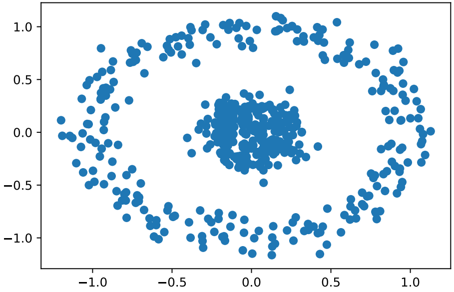
Bruno Ordozgoiti

Aalto University 2021

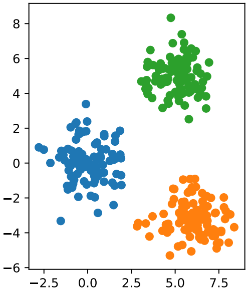
Algorithms for *k*-means will do well on these data.

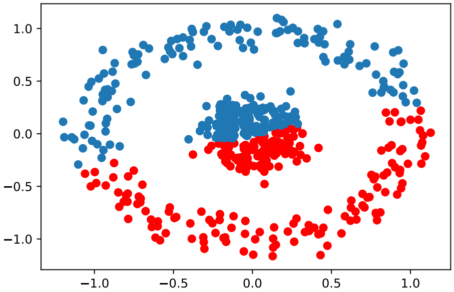


But how about this?

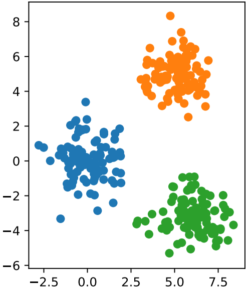


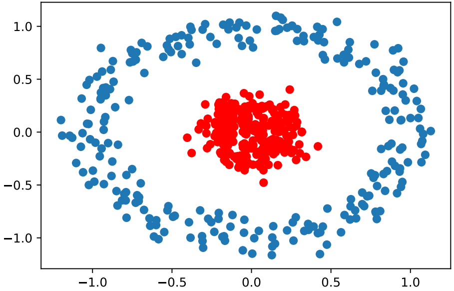
The results:





Let's talk about spectral clustering.

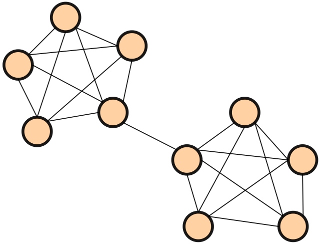




Derivation

Let us try to cluster the graph on the right. There are two "obvious" clusters, but can we use a clustering algorithm to discover them?

We will try to represent the graph vertices in a way that is suitable for an algorithm such as k-means.

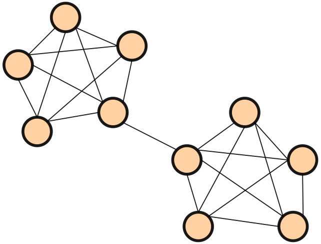


Let us try to cluster the graph on the right. There are two "obvious" clusters, but can we use a clustering algorithm to discover them?

We will try to represent the graph vertices in a way that is suitable for an algorithm such as k-means.

We will consider the adjacency matrix:

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



w_{ij} is the weight of the edge connecting v_i and v_j.

One way to make sure we can use k-means is by:

- being able to compute distances between vertices: $d(v_i, v_j)$,
- being able to compute the mean of a cluster of vertices: μ_j.

One way to make sure we can use k-means is by:

- being able to compute distances between vertices: $d(v_i, v_j)$,
- being able to compute the mean of a cluster of vertices: μ_j.

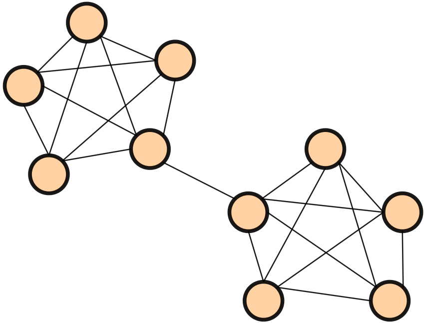
One way to accomplish this is to assign a real number y_i to each vertex v_i, so that

- $d(v_i, v_j) = |y_i - y_j|$ and
- $\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} y_j$.

Thus, our goal is to find a mapping of each vertex $v_i \mapsto y_i \in \mathbb{R}$ so that (intuitively) similar vertices are in the same cluster and different vertices are in different clusters.

In order to ensure that connected vertices are close, we can try to choose the y_i's so that the following is small:

$$cost(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(y_i - y_j)^2.$$



$$cost(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (y_i - y_j)^2 = 2 \sum_{i=1}^{n} y_i^2 \sum_{j=1}^{n} w_{ij} - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} y_i y_j.$$

$$cost(y) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i - y_j)^2 = 2\sum_{i=1}^{n} y_i^2 \sum_{j=1}^{n} w_{ij} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j.$$

Note: $\sum_{j=1}^{n} w_{ij} = d_i$ is the degree of vertex v_i, and $D_{ii} = \sum_{j=1}^{n} w_{ij}$.

$$cost(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(y_i - y_j)^2 = 2 \sum_{i=1}^{n} y_i^2 \sum_{j=1}^{n} w_{ij} - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} y_i y_j.$$

Note: $\sum_{j=1}^{n} w_{ij} = d_i$ is the degree of vertex v_i, and $D_{ii} = \sum_{j=1}^{n} w_{ij}$.

Let $y = (y_1, \ldots, y_n)^T$. Note that $cost(y) = 2y^T Dy - 2y^T Wy = 2y^T Ly$.

Let’s summarize:

Let's summarize:

- We want to cluster the vertices of a graph using k-means.

Let's summarize:

- We want to cluster the vertices of a graph using k-means.
- We assign a real number y_i to every vertex so that we can compute distances and means.

Let's summarize:

- We want to cluster the vertices of a graph using k-means.
- We assign a real number y_i to every vertex so that we can compute distances and means.
- Choosing $y = (y_1, \ldots, y_n)^T$ to minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(y_i - y_j)^2$ seems like a good idea.

Let's summarize:

- We want to cluster the vertices of a graph using k-means.
- We assign a real number y_i to every vertex so that we can compute distances and means.
- Choosing $y = (y_1, \dots, y_n)^T$ to minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(y_i - y_j)^2$ seems like a good idea.
- We define $L = D - W$ and show that minimizing $y^T L y$ is equivalent.

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).
2. 0 is an eigenvalue of L.

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).
2. 0 is an eigenvalue of L.
3. If the graph is connected, $(1, 1, \ldots, 1)^T$ is an eigenvector with eigenvalue 0.

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).
2. 0 is an eigenvalue of L.
3. If the graph is connected, $(1, 1, \ldots, 1)^T$ is an eigenvector with eigenvalue 0.
4. The multiplicity of the eigenvalue 0 equals the number of connected components.

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).
2. 0 is an eigenvalue of L.
3. If the graph is connected, $(1, 1, \ldots, 1)^T$ is an eigenvector with eigenvalue 0.
4. The multiplicity of the eigenvalue 0 equals the number of connected components.

Remember we want to minimize $y^T Ly$.

- By property 1, the minimum is at least 0. $y = (0, 0 \ldots, 0)^T$ is a trivial solution, so we impose the constraint $y^T y = 1$.

Some properties of the Laplacian

1. L is symmetric and positive semidefinite (all eigenvalues are real and ≥ 0).
2. 0 is an eigenvalue of L.
3. If the graph is connected, $(1, 1, \ldots, 1)^T$ is an eigenvector with eigenvalue 0.
4. The multiplicity of the eigenvalue 0 equals the number of connected components.

Remember we want to minimize $y^T L y$.

- By property 1, the minimum is at least 0. $y = (0, 0 \ldots, 0)^T$ is a trivial solution, so we impose the constraint $y^T y = 1$.
- If the graph is connected, the vector $\mathbf{1} = (1, 1, \ldots, 1)^T$ is a solution with $\mathbf{1}^T L \mathbf{1} = 0$. We impose the constraint $y^T \mathbf{1} = 0$.

Objective

$$\begin{aligned} \min_{y} \quad & y^T L y \\ \text{subject to} \quad & y^T y = 1 \\ & y^T \mathbf{1} = 0 \end{aligned}$$

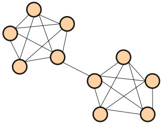
Since the vector $\mathbf{1} = (1, 1, \ldots, 1)^T$ is an eigenvector corresponding to the smallest eigenvalue, the above is solved (in a connected graph) by the **eigenvector corresponding to the second smallest eigenvalue.**

Spectral clustering protoalgorithm for 2-way connected graph partitioning.

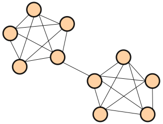
Input: Graph $G = (V, E)$ with adjacency matrix W.

1. Compute the Laplacian $L = D - W$.
2. Compute y, the eigenvector of L corresponding to the second smallest eigenvalue.
3. Run k-means treating the entries of y as one-dimensional data points.

Let’s try to cluster our graph.

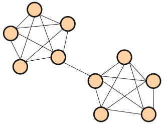


Let’s try to cluster our graph.



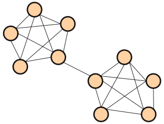
$$W = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix},$$

Let's try to cluster our graph.



$$W = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}, L = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 5 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 5 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix}$$

Let’s try to cluster our graph.

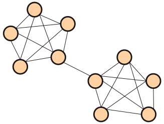


$$W = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}, L = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 5 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 5 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix}$$

Eigenvalues: $(0, \sim 0.3,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5, \sim 6.7)$.

Second smallest eigenvector: $(0.33, 0.33, 0.33, 0.33, 0.23, -0.23, -0.33, -0.33, -0.33, -0.33)^T$.

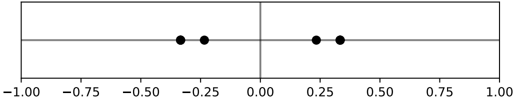
Let's try to cluster our graph.



$$W = \begin{pmatrix} 0&1&1&1&1&0&0&0&0&0\\ 1&0&1&1&1&0&0&0&0&0\\ 1&1&0&1&1&0&0&0&0&0\\ 1&1&1&0&1&0&0&0&0&0\\ 1&1&1&1&0&1&0&0&0&0\\ 0&0&0&0&1&0&1&1&1&1\\ 0&0&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&1&1&0&1&1\\ 0&0&0&0&0&1&1&1&0&1\\ 0&0&0&0&0&1&1&1&1&0 \end{pmatrix}, L = \begin{pmatrix} 4&-1&-1&-1&-1&0&0&0&0&0\\ -1&4&-1&-1&-1&0&0&0&0&0\\ -1&-1&4&-1&-1&0&0&0&0&0\\ -1&-1&-1&4&-1&0&0&0&0&0\\ -1&-1&-1&-1&5&-1&0&0&0&0\\ 0&0&0&0&-1&5&-1&-1&-1&-1\\ 0&0&0&0&0&-1&4&-1&-1&-1\\ 0&0&0&0&0&-1&-1&4&-1&-1\\ 0&0&0&0&0&-1&-1&-1&4&-1\\ 0&0&0&0&0&-1&-1&-1&-1&4 \end{pmatrix}$$

Eigenvalues: $(0, \sim 0.3,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5,\ 5, \sim 6.7)$.

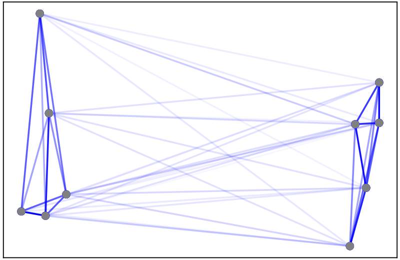
Second smallest eigenvector: $(0.33, 0.33, 0.33, 0.33, 0.23, -0.23, -0.33, -0.33, -0.33, -0.33)^T$.



$\longleftarrow$ We can cluster this with k-means.

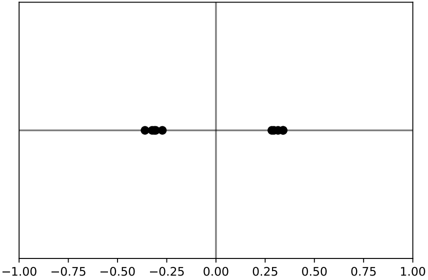
Another example

Fully connected weighted graph.



Eigenvector:

$(0.32, 0.34, 0.28, 0.34, 0.29, -0.32, -0.27, -0.31, -0.36, -0.31)^T$



Optimizing graph cuts

Clustering with cuts

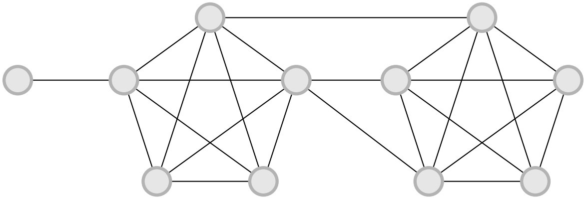
Given a graph $G = (V, E)$, and a vertex subset $S \subseteq V$, with adjacency matrix A,

$$cut(S, \overline{S}) = E(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} A_{ij}.$$

Clustering with cuts

Given a graph $G = (V, E)$, and a vertex subset $S \subseteq V$, with adjacency matrix A,

$$cut(S, \overline{S}) = E(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} A_{ij}.$$



Clustering with cuts

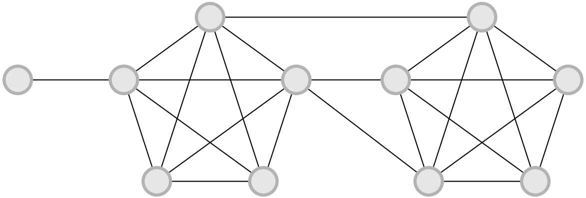
Given a graph $G = (V, E)$, and a vertex subset $S \subseteq V$, with adjacency matrix A,

$$\mathrm{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|} \right) \mathrm{cut}(S, \overline{S})$$

Clustering with cuts

Given a graph $G = (V, E)$, and a vertex subset $S \subseteq V$, with adjacency matrix A,

$$\text{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|} \right) \text{cut}(S, \overline{S})$$



$$\mathrm{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right) \mathrm{cut}(S, \overline{S}).$$

$$\text{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right) \text{cut}(S, \overline{S}).$$

Define a vector x as

$$x_i = \begin{cases} \sqrt{|\overline{S}|/|S|} & \text{if } v_i \in S, \\ -\sqrt{|S|/|\overline{S}|} & \text{if } v_i \in \overline{S}. \end{cases}$$

Define a vector x as

$$\mathrm{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right) \mathrm{cut}(S, \overline{S}).$$

$$x_i = \begin{cases} \sqrt{|\overline{S}|/|S|} & \text{if } v_i \in S, \\ -\sqrt{|S|/|\overline{S}|} & \text{if } v_i \in \overline{S}. \end{cases}$$

We have

- $x^T L x = |V| \cdot \mathrm{RatioCut}(S)$,
- $x^T 1 = 0$,
- $\|x\|_2 = \sqrt{|V|}$.

$$\text{RatioCut}(S, \overline{S}) = \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|}\right) \text{cut}(S, \overline{S}).$$

We have

- $x^T L x = |V| \cdot \text{RatioCut}(S)$,
- $x^T \mathbf{1} = 0$,
- $\|x\|_2 = \sqrt{|V|}$.

Define a vector x as

$$x_i = \begin{cases} \sqrt{|\overline{S}|/|S|} & \text{if } v_i \in S, \\ -\sqrt{|S|/|\overline{S}|} & \text{if } v_i \in \overline{S}. \end{cases}$$

Objective:

$$\begin{aligned} \min_x \quad & x^T L x \\ \text{subject to} \quad & x^T \mathbf{1} = 0, \\ & \|x\|_2 = \sqrt{|V|}, \\ & x_i \text{ as defined above.} \end{aligned}$$

k clusters:

$$\mathrm{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(S_i, \overline{S}_i)}{|S_i|}.$$

k clusters:

$$\text{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S}_i)}{|S_i|}.$$

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{|S_i|}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

Clustering with cuts

Optimizing RatioCut

k clusters:

$$\mathrm{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(S_i, \overline{S}_i)}{|S_i|}.$$

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{|S_i|}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

We have

- $x_i^T L x_i = \frac{\mathrm{cut}(S_i, \overline{S}_i)}{|S_i|}$,
- $\sum_i x_i^T L x_i = \sum_i \frac{\mathrm{cut}(S_i, \overline{S}_i)}{|S_i|}$,
- $x_i^T x_j = 0, i \neq j$,
- $\|x_i\|_2 = 1$.

k clusters:

$$\text{RatioCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S}_i)}{|S_i|}.$$

We have

- $x_i^T L x_i = \frac{\text{cut}(S_i, \overline{S}_i)}{|S_i|}$,
- $\sum_i x_i^T L x_i = \sum_i \frac{\text{cut}(S_i, \overline{S}_i)}{|S_i|}$,
- $x_i^T x_j = 0, i \neq j$,
- $\|x_i\|_2 = 1$.

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{|S_i|}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

Objective:

$$\begin{aligned} \min_X \quad & Tr(X^T L X) \\ \text{subject to} \quad & X^T X = I, \\ & X_{ij} = x_{ij} \text{ as defined above.} \end{aligned}$$

Clustering with cuts

Optimizing NCut

$$\mathrm{NCut}(S, \overline{S}) = \left(\frac{1}{\mathrm{vol}(S)} + \frac{1}{\mathrm{vol}(\overline{S})} \right) \mathrm{cut}(S, \overline{S}).$$

$\mathrm{vol}(S) = \sum_{v \in S} d(v).$

Clustering with cuts

Optimizing NCut

Define a vector x as

$$\mathrm{NCut}(S, \overline{S}) = \left(\frac{1}{\mathrm{vol}(S)} + \frac{1}{\mathrm{vol}(\overline{S})}\right) \mathrm{cut}(S, \overline{S}).$$

$$\mathrm{vol}(S) = \sum_{v \in S} d(v).$$

$$x_i = \begin{cases} \sqrt{\mathrm{vol}(\overline{S})/\mathrm{vol}(S)} & \text{if } v_i \in S, \\ -\sqrt{\mathrm{vol}(S)/\mathrm{vol}(\overline{S})} & \text{if } v_i \in \overline{S}. \end{cases}$$

Clustering with cuts

Optimizing NCut

$$\mathrm{NCut}(S, \overline{S}) = \left(\frac{1}{\mathrm{vol}(S)} + \frac{1}{\mathrm{vol}(\overline{S})}\right) \mathrm{cut}(S, \overline{S}).$$

$$\mathrm{vol}(S) = \textstyle\sum_{v \in S} d(v).$$

Define a vector x as

$$x_i = \begin{cases} \sqrt{\mathrm{vol}(\overline{S})/\mathrm{vol}(S)} & \text{if } v_i \in S, \\ -\sqrt{\mathrm{vol}(S)/\mathrm{vol}(\overline{S})} & \text{if } v_i \in \overline{S}. \end{cases}$$

We have

- $x^T L x = \mathrm{vol}(V) \cdot \mathrm{NCut}(S)$,
- $(Dx)^T 1 = 0$,
- $x^T D x = \mathrm{vol}(V)$.

$$\mathrm{NCut}(S,\overline{S}) = \left(\frac{1}{\mathrm{vol}(S)} + \frac{1}{\mathrm{vol}(\overline{S})}\right)\mathrm{cut}(S,\overline{S}).$$

$\mathrm{vol}(S) = \sum_{v \in S} d(v)$.

We have

- $x^T L x = \mathrm{vol}(V) \cdot \mathrm{NCut}(S)$,
- $(Dx)^T 1 = 0$,
- $x^T D x = \mathrm{vol}(V)$.

Define a vector x as

$$x_i = \begin{cases} \sqrt{\mathrm{vol}(\overline{S})/\mathrm{vol}(S)} & \text{if } v_i \in S, \\ -\sqrt{\mathrm{vol}(S)/\mathrm{vol}(\overline{S})} & \text{if } v_i \in \overline{S}. \end{cases}$$

Objective:

$$\begin{aligned} \min_x \quad & x^T D^{-1/2} L D^{-1/2} x \\ \text{subject to} \quad & (D^{1/2}x)^T 1 = 0, \\ & x^T D x = \mathrm{vol}(V), \\ & x_i \text{ as defined above.} \end{aligned}$$

Note: $D^{-1/2} L D^{-1/2}$ is known as the normalized Laplacian.

Clustering with cuts

Optimizing NCut

k clusters:

$$\text{NCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S}_i)}{\text{vol}(S_i)}.$$

Clustering with cuts

Optimizing NCut

k clusters:

$$\text{NCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S}_i)}{\text{vol}(S_i)}.$$

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{\text{vol}(S_i)}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

k clusters:

$$\mathrm{NCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(S_i, \overline{S}_i)}{\mathrm{vol}(S_i)}.$$

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{\mathrm{vol}(S_i)}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

We have

- $x_i^T L x_i = \frac{\mathrm{cut}(S_i, \overline{S}_i)}{\mathrm{vol}(S_i)}$,
- $\sum_i x_i^T L x_i = \sum_i \frac{\mathrm{cut}(S_i, \overline{S}_i)}{\mathrm{vol}(S_i)}$,
- $x_i^T x_j = 0, i \neq j$,
- $x_i^T D x_i = 1$.

k clusters:

$$\text{NCut}(S_1, \ldots, S_k) = \sum_{i=1}^{k} \frac{\text{cut}(S_i, \overline{S}_i)}{\text{vol}(S_i)}.$$

We have

- $x_i^T L x_i = \frac{\text{cut}(S_i, \overline{S}_i)}{\text{vol}(S_i)}$,
- $\sum_i x_i^T L x_i = \sum_i \frac{\text{cut}(S_i, \overline{S}_i)}{\text{vol}(S_i)}$,
- $x_i^T x_j = 0, i \neq j$,
- $x_i^T D x_i = 1$.

Define k vectors $(x_1, \ldots, x_k)$ as

$$x_{ij} = \begin{cases} \frac{1}{\sqrt{\text{vol}(S_i)}} & \text{if } v_j \in S_i, \\ 0 & \text{if } v_j \in \overline{S}_i. \end{cases}$$

Objective:

$$\begin{aligned} \min_X \quad & Tr(X^T D^{-1/2} L D^{-1/2} X) \\ \text{subject to} \quad & X^T X = I, \\ & X_{ij} = x_{ij} \text{ as defined above.} \end{aligned}$$

Unnormalized spectral clustering .

Input: Graph $G = (V, E)$ with adjacency matrix W, number of clusters k.

1. Compute the Laplacian $L = D - W$.
2. Compute the eigenvectors $v_1, \ldots, v_k$ of L corresponding to the k smallest eigenvalues.
3. Consider a matrix X whose columns are $v_1, \ldots, v_k$.
4. Run k-means on X.

Normalized variants:

Spectral clustering using normalized Laplacian.

Input: Graph $G = (V, E)$ with adjacency matrix W, number of clusters k.

1. Compute the **normalized** Laplacian $L_{sym} = D^{-1/2}(D - W)D^{-1/2}$.
2. Compute the eigenvectors $v_1, \ldots, v_k$ of L_{sym} corresponding to the k smallest eigenvalues.
3. Consider a matrix X whose columns are $v_1, \ldots, v_k$. Normalize the rows of X to unit norm.
4. Run k-means on X.

Note: $L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$.

Further reading:

- Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007)
- Ng, Jordan, and Weiss. "On spectral clustering: Analysis and an algorithm." NIPS 2002.

Normalized variants:

Spectral clustering using random walk Laplacian.

Input: Graph $G = (V, E)$ with adjacency matrix W, number of clusters k.

1. Compute the **random walk** Laplacian $L_{rw} = D^{-1}(D - W)$.
2. Compute the **right** eigenvectors $v_1, \ldots, v_k$ of L_{rw} corresponding to the k smallest eigenvalues.
3. Consider a matrix X whose columns are $v_1, \ldots, v_k$. Normalize the rows of X to unit norm.
4. Run k-means on X.

Note: $L_{rw} = D^{-1}L = I - D^{-1}W$. This is related to the random walk matrix.

Further reading:

- Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007)
- Dhillon, Inderjit S., Yuqiang Guan, and Brian Kulis. "Kernel k-means: spectral clustering and normalized cuts." KDD 2004.

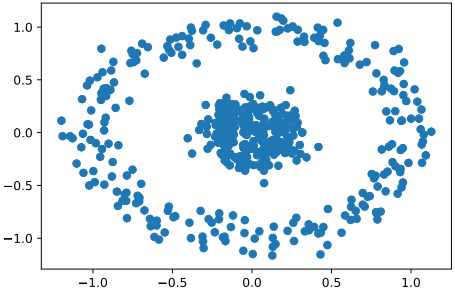
Recommended read: "A tutorial on spectral clustering." by Ulrike Von Luxburg.

As a rule of thumb, use the **random walk** variant.

Spectral clustering in practice

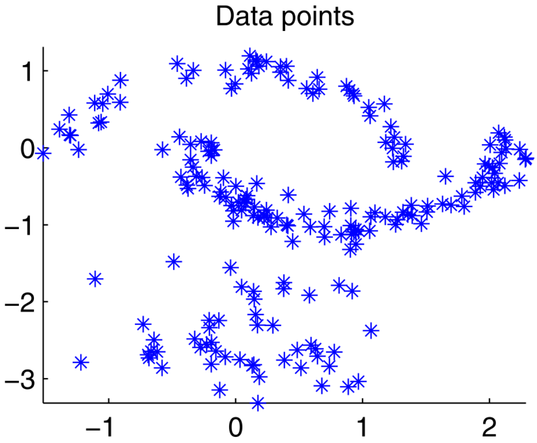
We have seen how to use spectral clustering on graphs.

But can we use it on any type of data? E.g. points in $\mathbb{R}^d$.



Example by Von Luxburg[1].

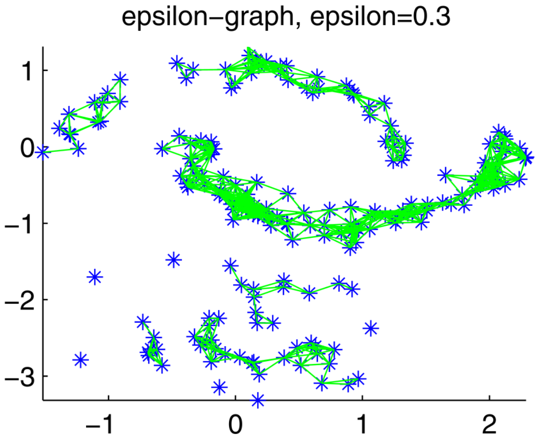
We can create a graph based on these data.



[1] Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

Example by Von Luxburg[1].

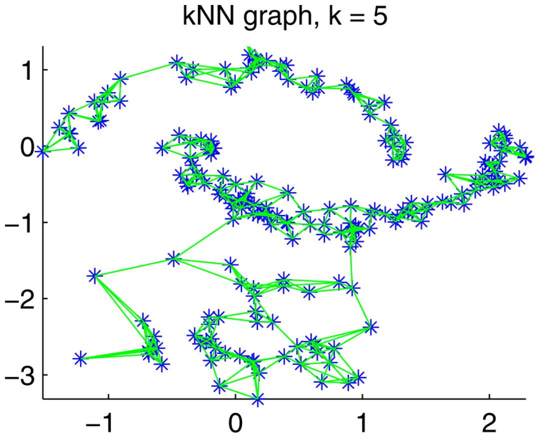
epsilon-graph: there is an edge between x and y if and only if $\|x - y\|_2 \leq \epsilon$.



[1] Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

Example by Von Luxburg[1].

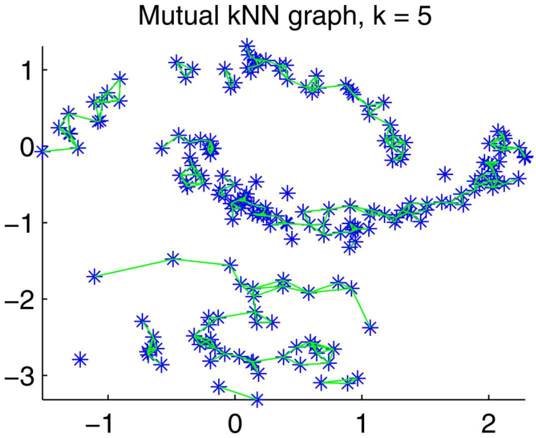
k-nearest-neighbours:
there is an edge between x and y
if and only if
x is one of the k nearest neighbours of y
or
y is one of the k nearest neighbours of x.



[1] Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

Example by Von Luxburg[1].

mutual k-nearest-neighbours:
there is an edge between x and y
if and only if
x is one of the k nearest neighbours of y
and
y is one of the k nearest neighbours of x.



[1] Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007): 395-416.

Alternatively, we can consider a fully-connected weighted graph, by using a similarity function.

We will use the Gaussian (or RBF) kernel, defined as follows:

$$\begin{aligned} \kappa : \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R} \\ x, y &\mapsto \kappa(x, y) = \exp\left(\frac{-\|x - y\|_2^2}{2\sigma^2}\right). \end{aligned}$$

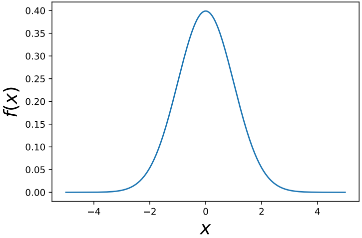
Alternatively, we can consider a fully-connected weighted graph, by using a similarity function.

We will use the Gaussian (or RBF) kernel, defined as follows:

$$\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

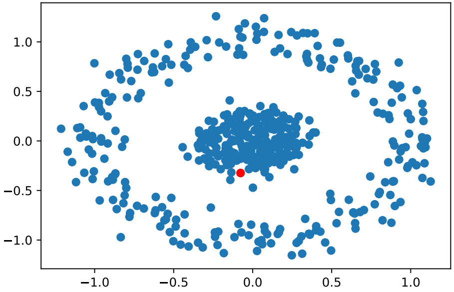
$$x, y \mapsto \kappa(x, y) = \exp\left(\frac{-\|x - y\|_2^2}{2\sigma^2}\right).$$

Recall density for $\mathcal{N}(\mu, \sigma) : f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$



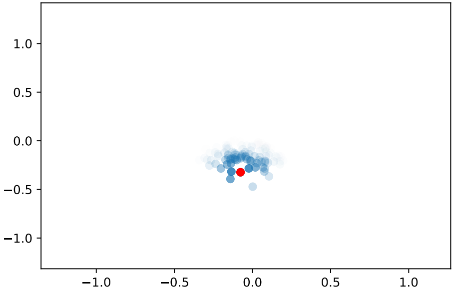
$$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$$

We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.



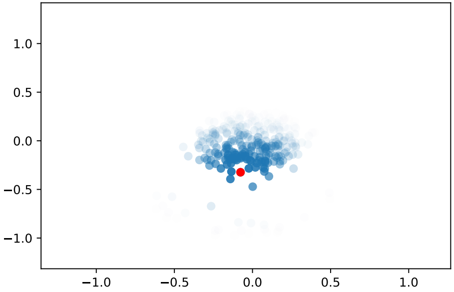
$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$
We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.

- $\sigma = 0.1$



$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$
We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.

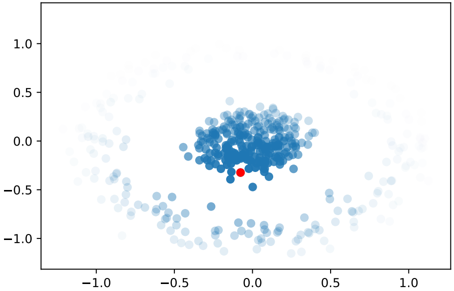
- $\sigma = 0.2$



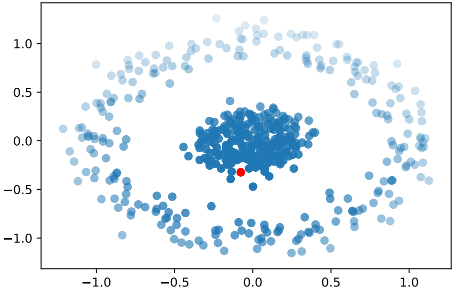
$\kappa(x,y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$

We set x to be the red point and compute the value of $\kappa(x,y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x,y)$.

- $\sigma = 0.4$

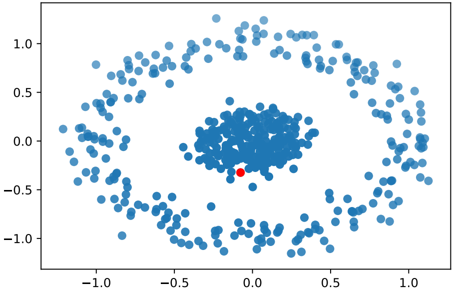


$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$
We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.



- $\sigma = 0.8$

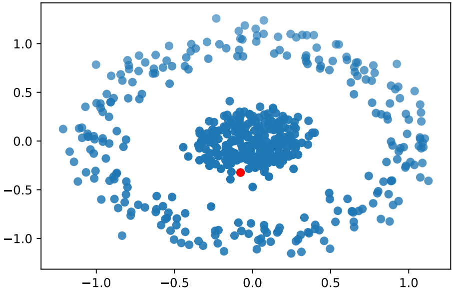
$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$
We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.



- $\sigma = 1.6$

$\kappa(x, y) = \exp\left(\frac{-\|x-y\|_2^2}{2\sigma^2}\right)$

We set x to be the red point and compute the value of $\kappa(x, y)$ for all y, with different values of σ. We plot each point with opacity equal to $\kappa(x, y)$.



- $\sigma = 1.6$

Note: the matrix of a fully connected graph might be too large to store. Consider thresholding or using nearest neighbours.

Practical considerations

Some recommendations:

- Scalability: the similarity (adjacency) matrix is of size $n \times n$. Thus, spectral clustering requires at least $O(n^2)$ computations just for the preliminary phase. The spectral decomposition requires $O(n^3)$ work in general. Always use sparse matrices if possible.

[2] https://en.wikipedia.org/wiki/LOBPCG

[3] Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." KDD 2009.

Practical considerations

Some recommendations:

- Scalability: the similarity (adjacency) matrix is of size $n \times n$. Thus, spectral clustering requires at least $O(n^2)$ computations just for the preliminary phase. The spectral decomposition requires $O(n^3)$ work in general. Always use sparse matrices if possible.
- Use a nearest neighbour graph to avoid storing the $n \times n$ matrix.

---

[2] https://en.wikipedia.org/wiki/LOBPCG

[3] Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." KDD 2009.

Practical considerations

Some recommendations:

- Scalability: the similarity (adjacency) matrix is of size $n \times n$. Thus, spectral clustering requires at least $O(n^2)$ computations just for the preliminary phase. The spectral decomposition requires $O(n^3)$ work in general. Always use sparse matrices if possible.
- Use a nearest neighbour graph to avoid storing the $n \times n$ matrix.
- Compute eigenvectors efficiently[2]

[2] https://en.wikipedia.org/wiki/LOBPCG

[3] Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." KDD 2009.

Practical considerations

Some recommendations:

- Scalability: the similarity (adjacency) matrix is of size $n \times n$. Thus, spectral clustering requires at least $O(n^2)$ computations just for the preliminary phase. The spectral decomposition requires $O(n^3)$ work in general. Always use sparse matrices if possible.
- Use a nearest neighbour graph to avoid storing the $n \times n$ matrix.
- Compute eigenvectors efficiently[2]
- Sample random subgraphs.

[2] https://en.wikipedia.org/wiki/LOBPCG

[3] Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." KDD 2009.

Practical considerations

Some recommendations:

- Scalability: the similarity (adjacency) matrix is of size $n \times n$. Thus, spectral clustering requires at least $O(n^2)$ computations just for the preliminary phase. The spectral decomposition requires $O(n^3)$ work in general. Always use sparse matrices if possible.
- Use a nearest neighbour graph to avoid storing the $n \times n$ matrix.
- Compute eigenvectors efficiently[2]
- Sample random subgraphs.
- Read more[3]

---

[2] https://en.wikipedia.org/wiki/LOBPCG

[3] Yan, Donghui, Ling Huang, and Michael I. Jordan. "Fast approximate spectral clustering." KDD 2009.

Take-aways from this lecture:

- Derivation of spectral clustering from first principles.
- Derivation of spectral clustering from cut objectives:
  - RatioCut
  - NCut
- Spectral clustering algorithms.
- Building a graph from vector data.
- Practical considerations.