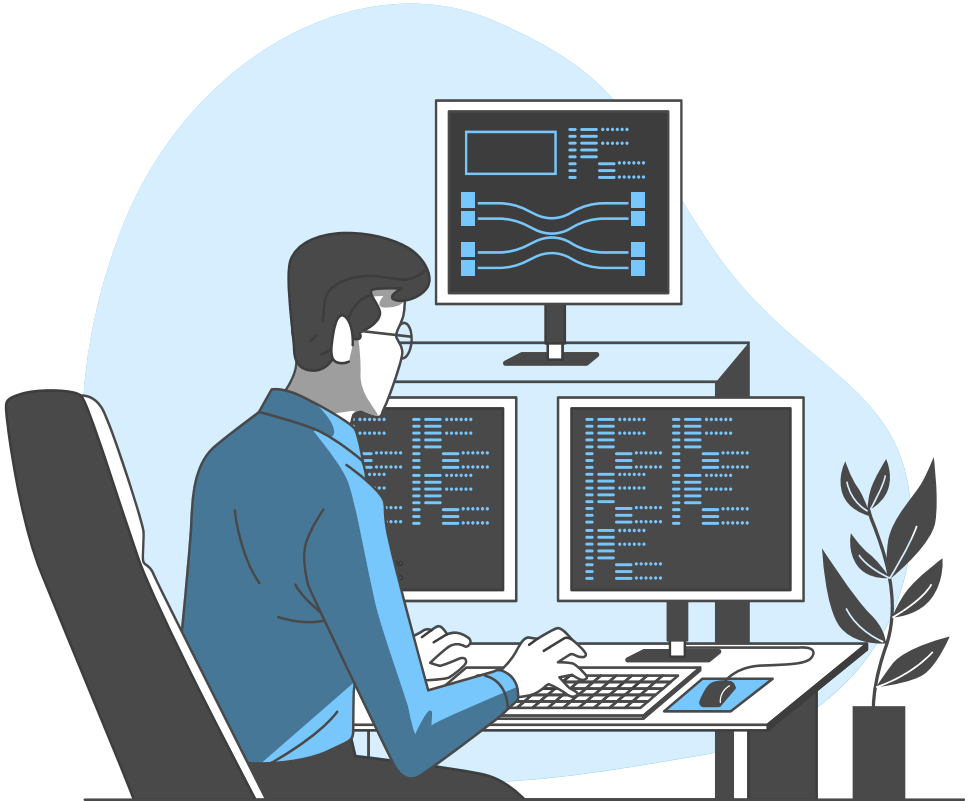


Spectral Clustering

Group 7

Instructors

PhD. Lương Ngọc Hoàng

Members of group

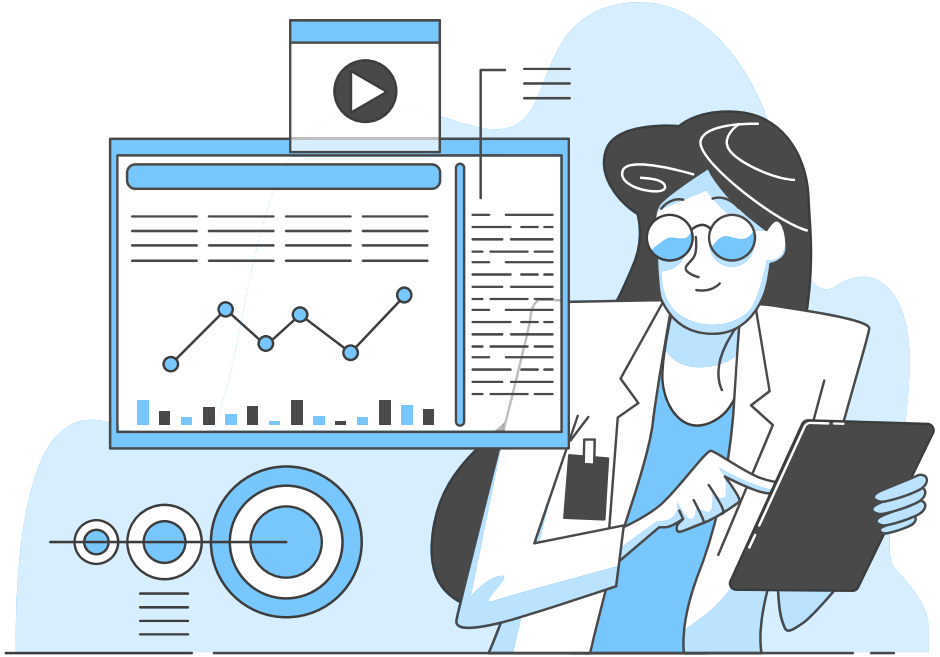
Trần Hoàng Bảo Ly
21521109

Đoàn Nguyễn Trần Hoàn
21520239

Lê Thanh Minh
21520063

Nguyễn Quốc Trường
21521604

Table of content

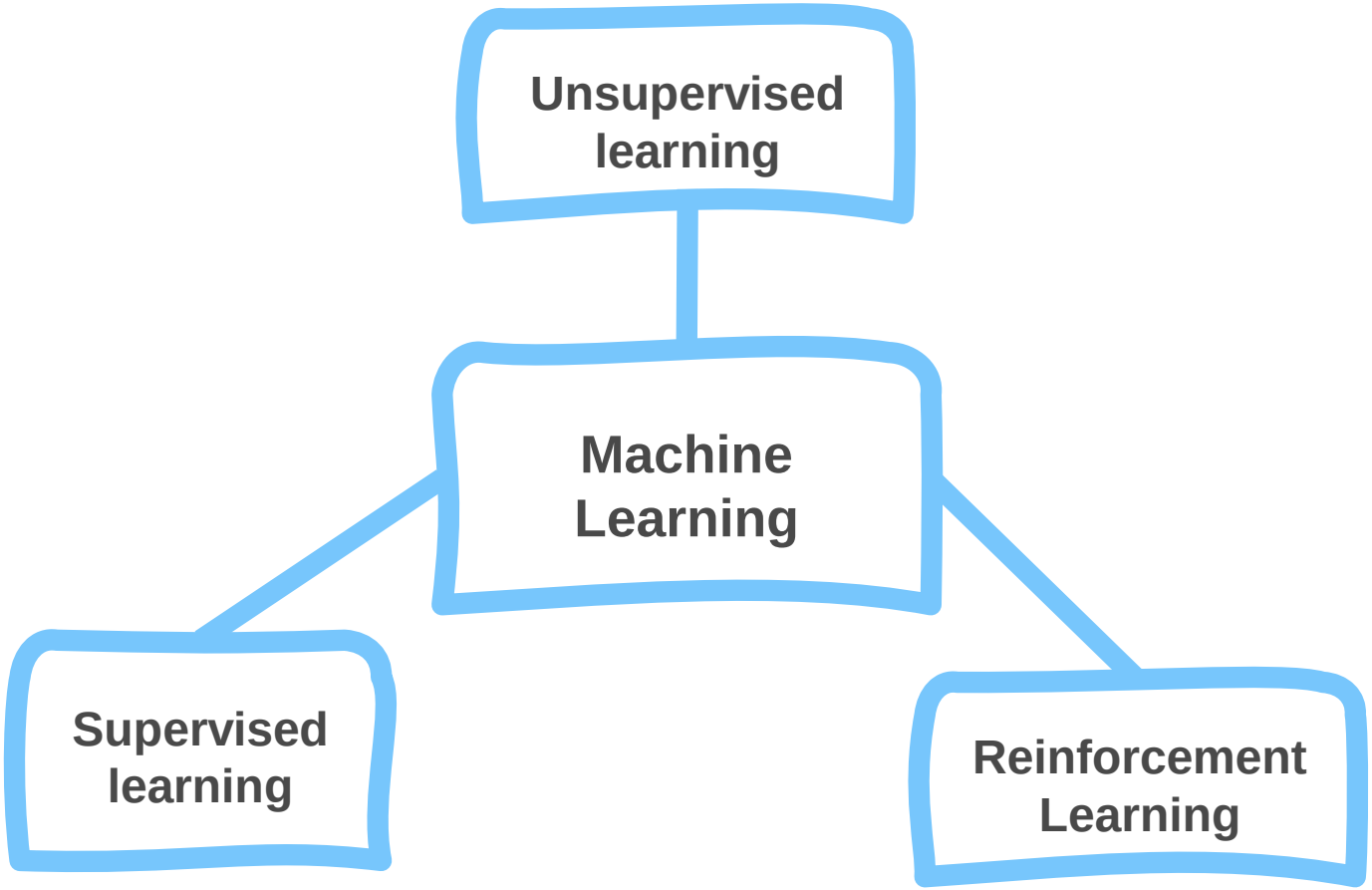


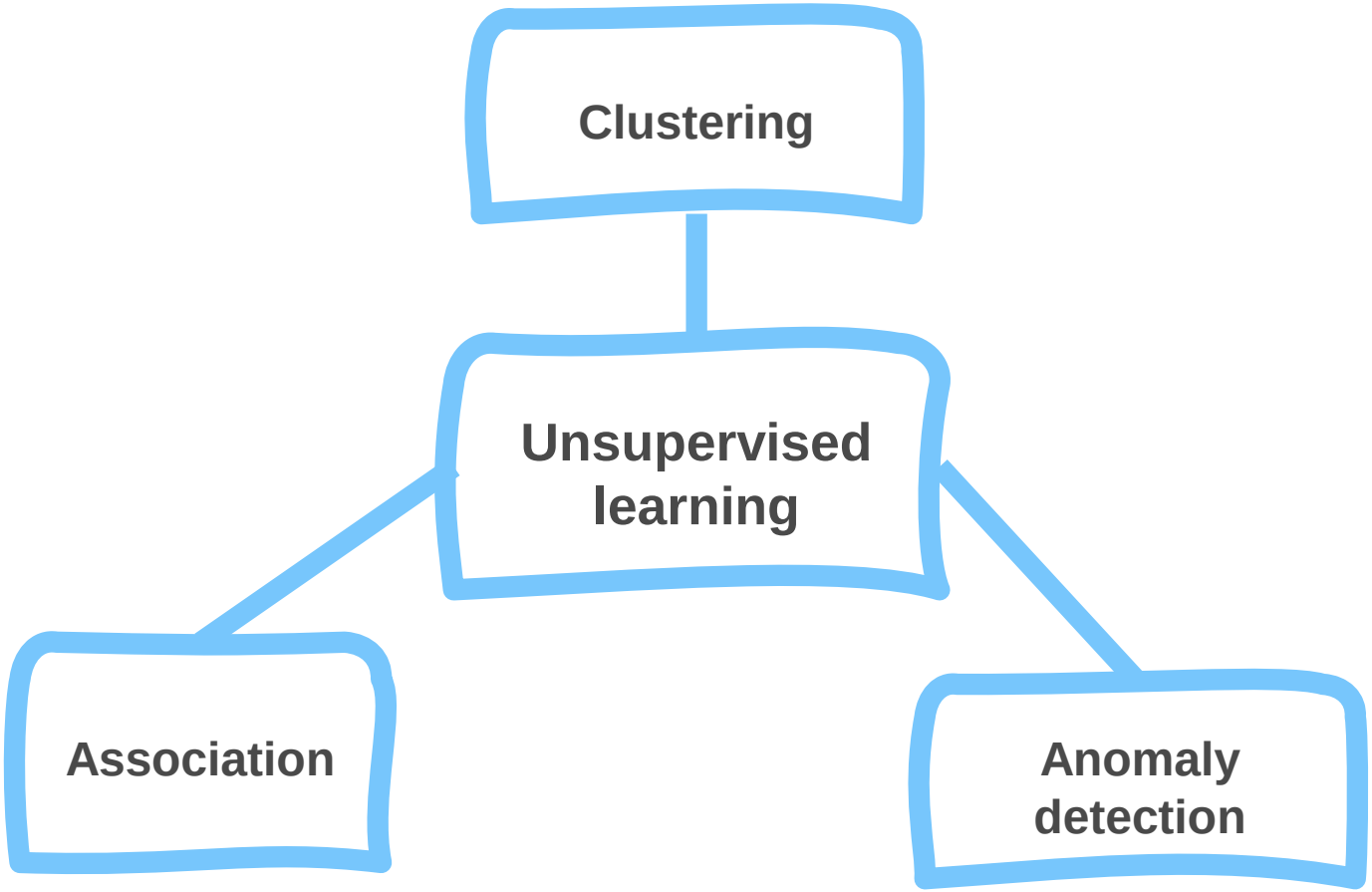
01

Abstract

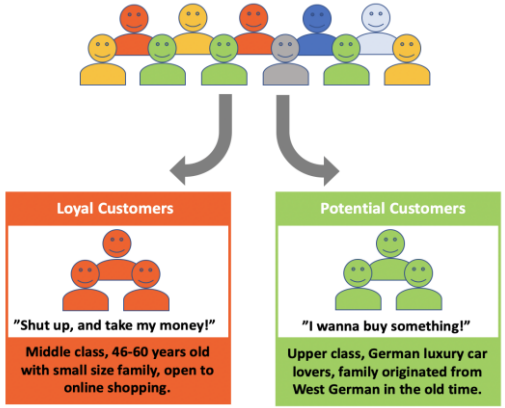
1.1

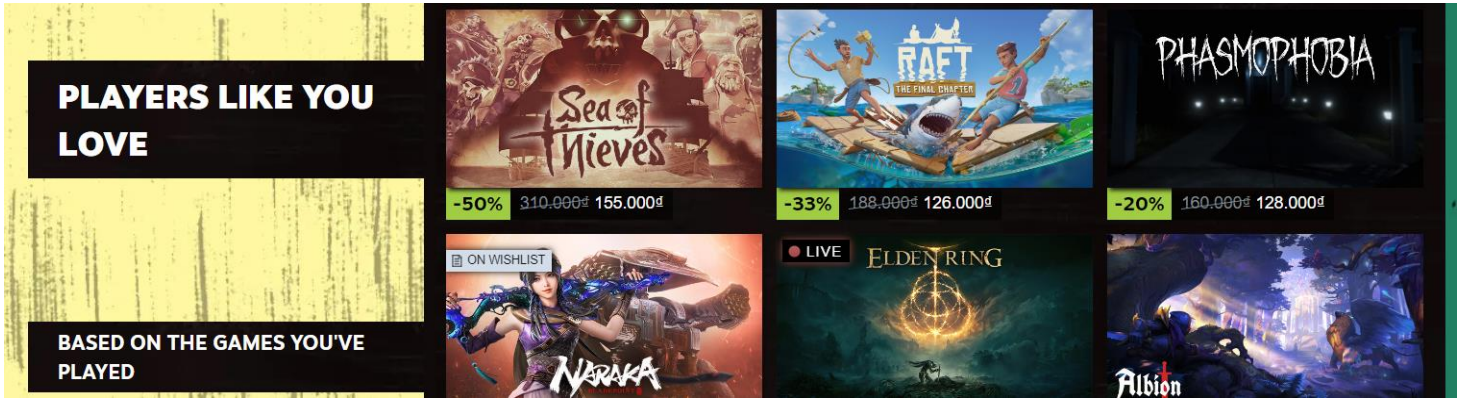
Reason for writing


Unsupervised learning
Machine Learning
Supervised learning
Reinforcement Learning


Clustering
Unsupervised learning
Association
Anomaly detection

Application of Clustering





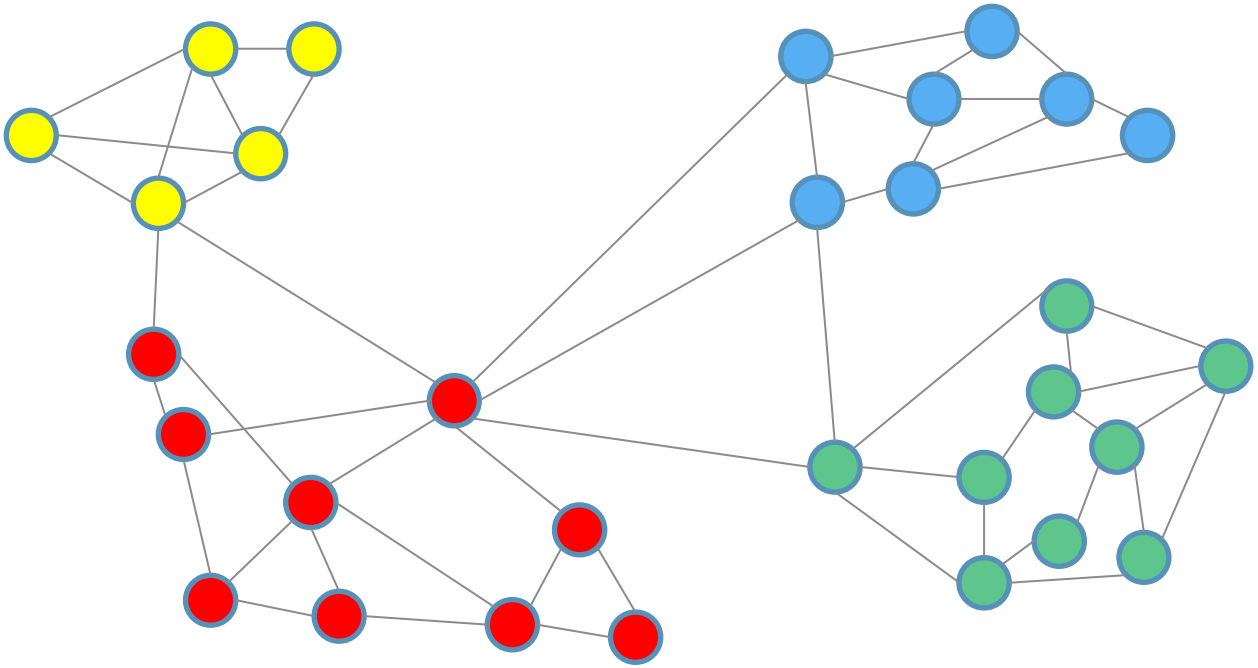
1.2

Introduction

What is clustering problem

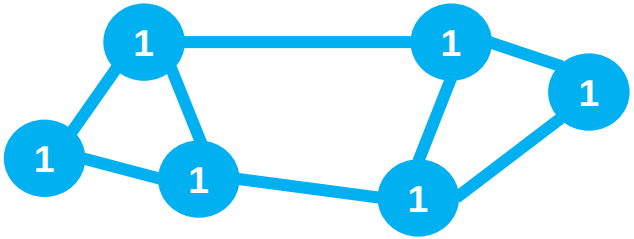
- Clustering is a type of unsupervised learning method.
- It’s related to graph theory and graph clusterting.
- Clustering is the task dividing datapoint into different groups, such that data points inner same group similar each other and different with data points outer that group. Based on a certain criterion.

- **Example**

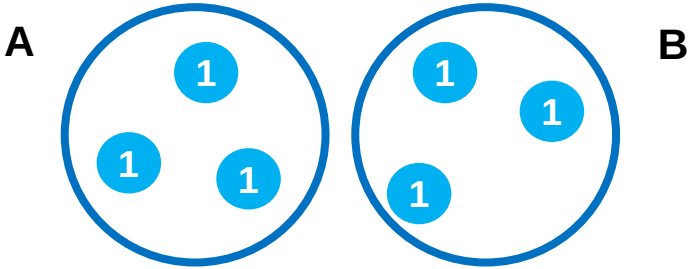


What make a good cluster

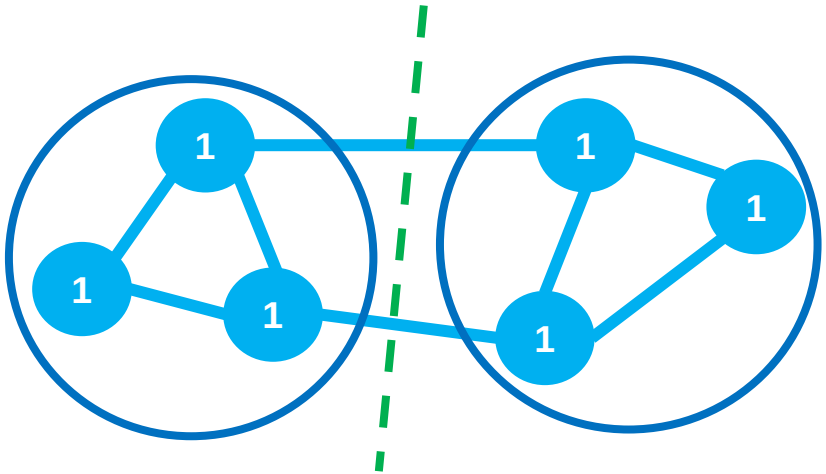
- **We have:** a undirected graph G(V,E):



- **Task:** Divide G into two clusters A, B. Such that A = V/B and B = V/A



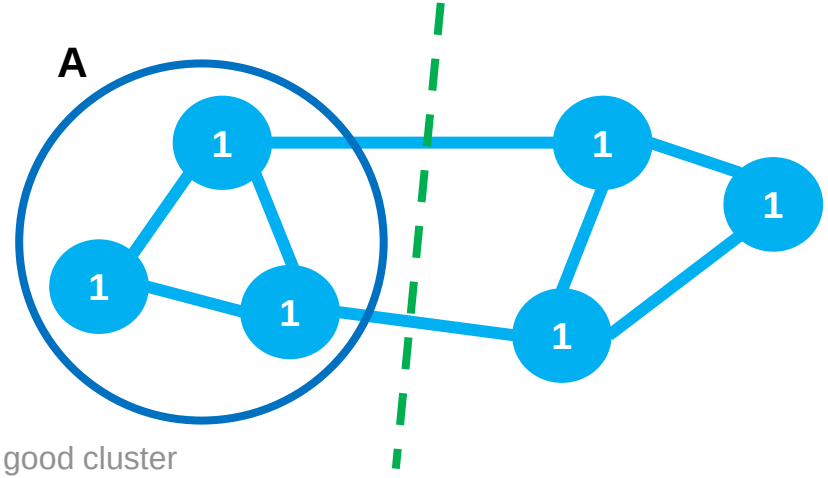
- **Desire**
  - Maximize the number of edges within cluster.
  - Minimize the number of edges between two clusters.



Approach 1:

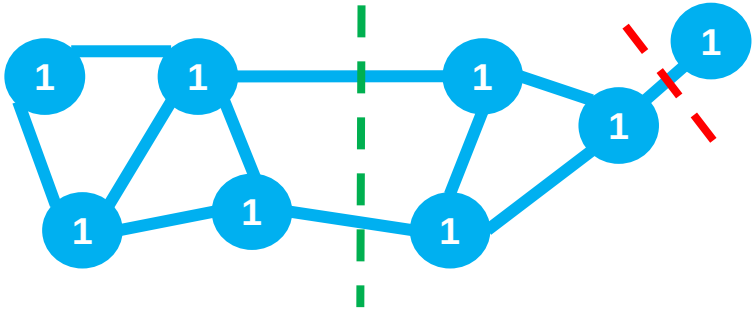
- Suppose that quality of the cluster A is evaluated by function *cut(A)* of the "edge cut" of the cluster.
- "Edge cut" is a set of edges that has only one node in the cluster.

$$cut(A) = \sum_{i \in A, j \notin A} W_{ij}$$



$$cut(A) = 2$$

- **So:** The quality of the cluster is total sum of weight of edge with one node outside of the cluster
- **Degenerate case:**



- **Problem:** Cut() function just consider edges that connect two clusters and doesn't consider edges that connect node within cluster.

Second approach :

- **Conductance**
  - Vol(A) is the total degree of all nodes in cluster A:

$$Vol(A) = \sum_{i \in A} d_i : degree\ of\ node\ i$$

  - Conductance score is the ratio of *Cut(A) / Vol (A)*
  - Conduction score can be expressed in below

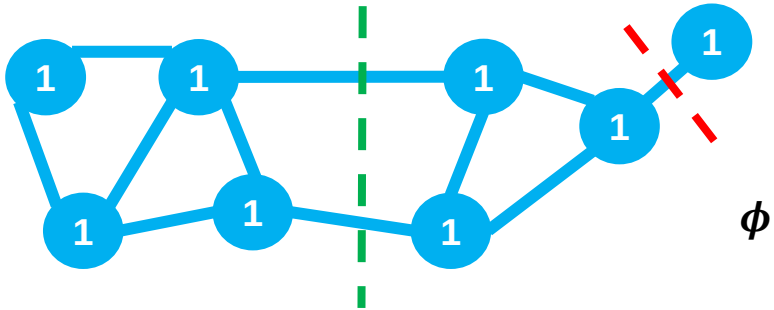
$$\phi(A) = \frac{|\{(i,j) \in E; i \in A, j \notin A\}|}{min\big(vol(A), 2m - vol(A)\big)}$$

  - m : total degree of all nodes
  - $|A| \leq \frac{|N|}{2}$

⇒ Conductance score : the less the better

○ Example

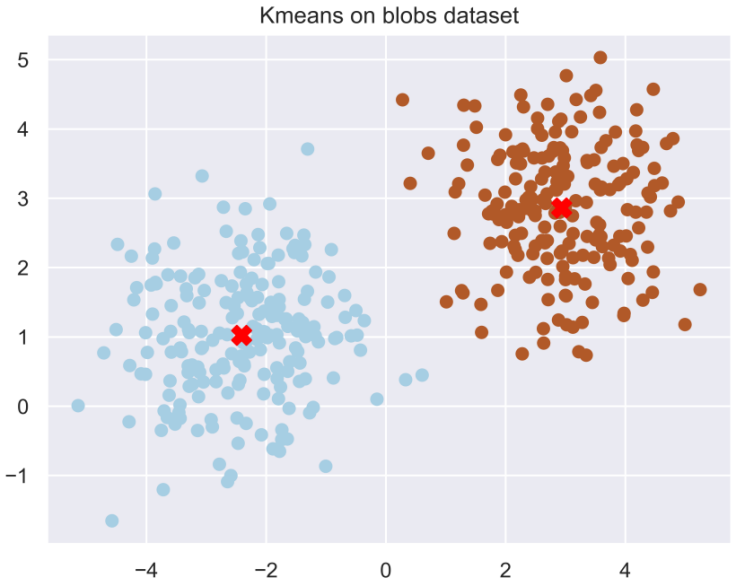
$$\phi = \frac{2}{10} = 0.2$$

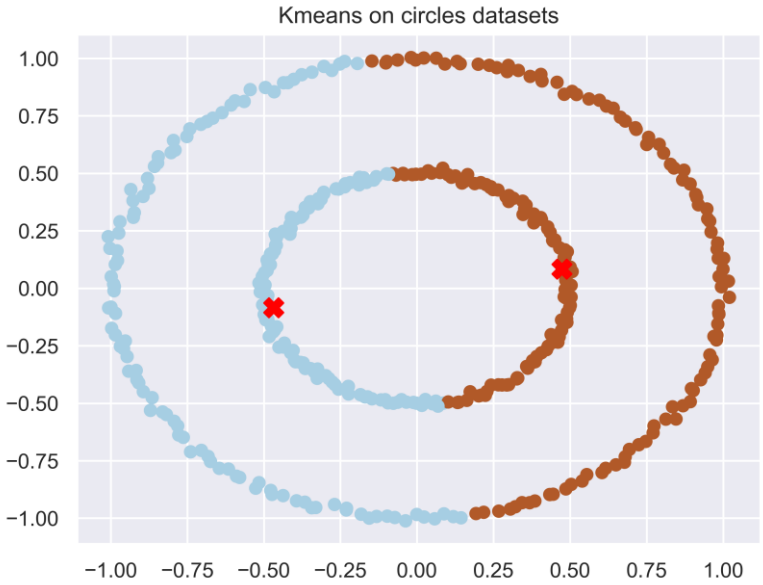


$$\phi = \frac{1}{1} = 1$$

⇒ Green line cut is better than red line cut (conductance score less than)

How many clustering algorithm ?





Kmeans algorithm

Need another clustering algorithm to solve problem ?

Spectral Clustering

02

Theoretical basis

2.1

Similarity graphs

Similarity graphs

- Similarity graph: determining the degree of similarity between these two graphs.

- Sign : G (V, E, W)
- Which : $\mathrm{V} = \{X_1, X_2, X_3, \dots X_n\}$ *is a set of vertices*

  E *is a set of* $\{X_i, X_j\}$ $(i, j \in E)$ *which* $W(X_i, X_j) > 0$

  W *is* the weight of similarity graph

Create similarity graph

- A first step in Spectral Clustering
- We can use this graph to create similarity graph:

1

Epsilon-Neighbourhood

2

K-Nearest Neighbours

3

Fully connected graph

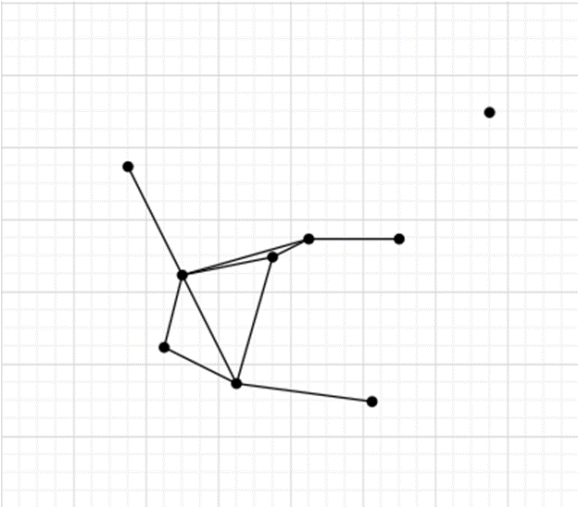
Epsilon-Neighbourhood graph

- To build the Epsilon-Neighbourhood graph, we connect all vertices which its distance less than ε (we defined ε depend on what we want).
- Similarity weight :

$$W_{ij} = \begin{cases} 1, \left|x_i - x_j\right| \leq \varepsilon \\ 0, otherwise \end{cases}$$

Epsilon-Neighbourhood graph

- Example of Epsilon-neighbourhood graph with ε = 2 :



Epsilon-Neighbourhood graph

- This is an undirectional graph
- When the distance between adjacent vertices is approximately the same level (maximum ε), weighting the edges cannot incorporate more information from the data into the graph. -> Graphs are generally considered to be unweighted graphs

- It really hard to choose a suitable ε in Epsilon-neighbourhood graph.

- Epsilon-neighbourhood graph cannot connect data points from different data levels

K-Nearest Neighbours Graph

- Aim : connect vertex x_i with vertex x_j if x_j in k-Nearest neighbour of x_i.
- However, this will make a graph become a directional graph (may be x_j in k-nearest neighbour of x_i but x_i not in k-nearest neighbour of x_j.
- There is 2 method to make this graph become undirectional graph again

...

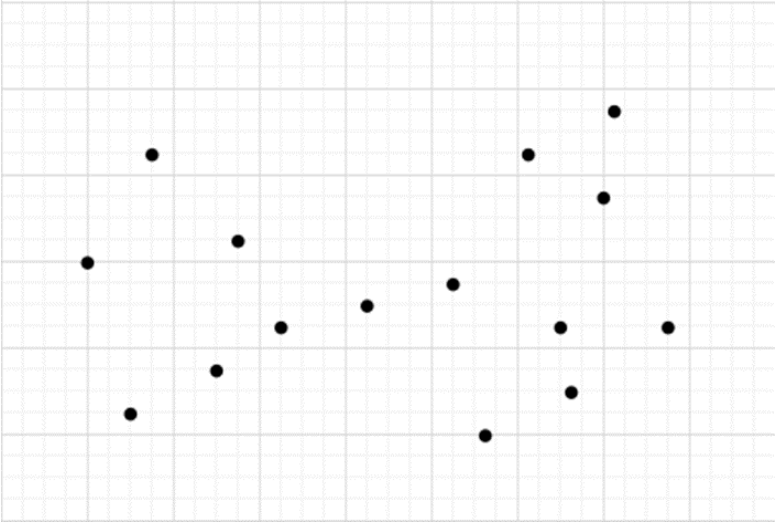
K-Nearest Neighbours Graph

- Method 1: Ignore the direction of edge. We connect x_i and x_j by undirectional edge if x_i in k-nearest neighbour of x_j or vice versa
- Method 2: We connect x_i and x_j only if both of them in k-nearest neighbour of each other

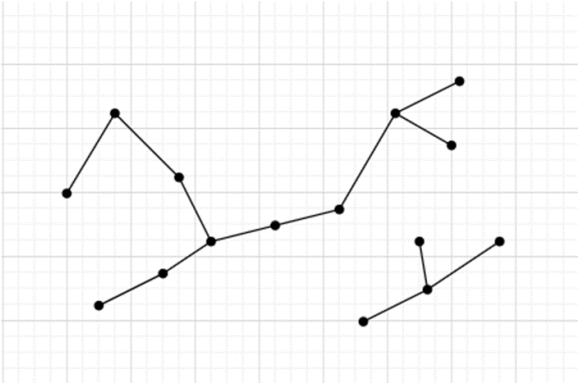
...

K-Nearest Neighbours Graph

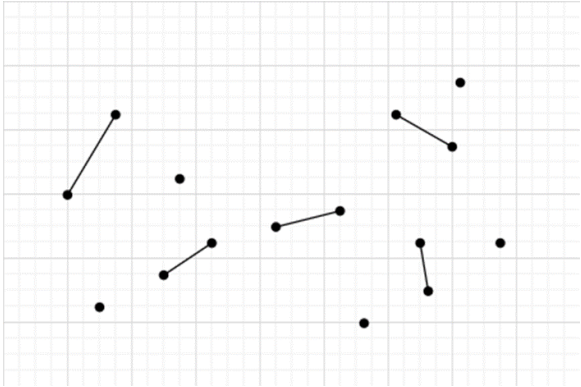
Example : Given dataset



K-Nearest Neighbours Graph



kNN graph was created by using method 1



kNN graph was created by using method 2

K-Nearest Neighbours Graph

- The kNN graph obtained in first method can connect data points on different data levels.
- The kNN graph obtained in second method tends to join data points in regions of the same density, but not in regions of different density.

Therefore, it can be considered as a kind of graph between epsilon-neighbourhood graph and kNN graph.

...

Fully connected graph

- To create this graph, every node in graph will be connected with another unidirectional edge having weight
- The relativity of 2 node X_i, X_j can be evaluated based on Gauss's distribution:

$$W_{ij} = e^{\frac{|x_i - x_j|^2}{2\sigma^2}}$$

- σ : standard deviation.

- W_{ij} use to adjust the size of cluster. The higher the value W_{ij} is, the more the connectivity of x_i and x_j.

...

Fully connected graph

- The distance of two node is calculated by Euclide's distance:

$$d_{ij} = \left| x_i - x_j \right|$$

⇒ The higher the connectivity of nodes, the less the length of distance

…

2.2

Projecting the data onto a lower Dimensional Space

Our Solutions

01 Adjacency Matrix

From graph, create adjacency matrix (A)

02 Degree Matrix

From adjacency matrix, create degree matrix (D)

03 Laplacian Matrix

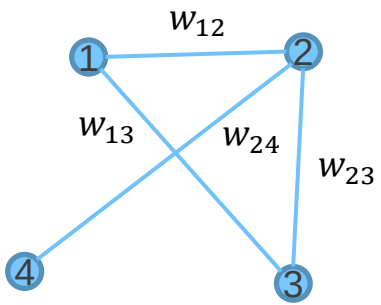
L = D-A

04 Eigenvector and Eigenvalue

Calculate eigenvectors and eigenvalue of Laplacian Matrix

Adjacency matrix

$$A = \begin{cases} w_{ij} & : Weight\ of\ edge\ between\ 2\ nodes\ (v_i, v_j) \\ 0 & : if\ there\ is\ \text{no edge between 2 nodes}\ (\mathrm{v_i}, \mathrm{v_j}) \end{cases}$$



	1	2	3	4
1	0	w_{12}	w_{13}	0
2	w_{12}	0	w_{23}	w_{24}
3	w_{13}	w_{23}	0	0
4	0	w_{24}	0	0

Important properties :

- Is a symmetric matrix
  Has n eigenvector (n: size of matrix)
- Orthogonal eigenvectors

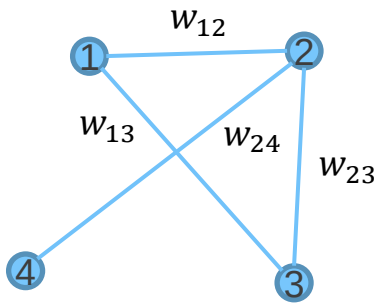
Degree Matrix

$$\boldsymbol{d_i} = \sum_{\{j|(i,j)\in E\}} \boldsymbol{w_{ij}}$$

$$d_1 = w_{12} + w_{13}$$
$$d_2 = w_{12} + w_{23} + w_{13}$$
$$d_3 = w_{13} + w_{23}$$
$$d_4 = w_{24} + w_{13}$$

	1	2	3	4
1	d_1	0	0	0
2	0	d_2	0	0
3	0	0	d_3	0
4	0	0	0	d_4

$Which\ is, d_i$ is an degree of node i



Laplacian Matrix

$$L = D - A = \begin{cases} d_i \; if \; i = j \\ -w_{i,j} \; if \, (i,j) \; is \; an \; edge \\ 0 \; if \; there \; is \; no \; edge \; between \; (i,j) \end{cases}$$

	1	2	3	4
1	d_1	$-w_{12}$	$-w_{13}$	0
2	$-w_{12}$	d_2	$-w_{23}$	$-w_{24}$
3	$-w_{13}$	$-w_{23}$	d_3	0
4	0	$-w_{24}$	0	d_4

Laplacian matrix

Properties:

- Always have the eigenvector (1,1,…,1) with eigenvalue 0
- The number of eigenvalue equals to 0 is the number of connected components of the graph(**Kirchhoff theory**)

Property

a) All eigenvalues are not negative: $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ \ldots \leq \lambda_n$

b) $x^T L x = \sum_{ij} L_{ij} x_i x_j \geq 0$, with any x.

c) L can be written as $L = N^T . N$

Proof:

$$(c) \Rightarrow (b): x^T L x = x^T N^T N x = (Nx)^T (Nx) \geq 0$$

$$(b) \Rightarrow (a): \lambda x^T x = x^T \lambda x = x^T L x \geq 0 \Rightarrow \lambda \geq 0$$

$$(a) \Rightarrow (c): using\ \boldsymbol{Incidence\ matrix}$$

Optimize problem

$$\boldsymbol{\lambda_2 = min\left(\frac{x^T L x}{x^T x}\right)}$$

Which is:

$$x^T L x$$
$$= \sum_{i,j=1}^{n} L_{ij} x_i x_j = \sum_{i,j=1}^{n} (D_{ij}$$
$$- A_{ij}) x_i x_j = \sum_i d_i x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j = \sum_{(i,j)\in E} (x_i - x_j)^2$$

Node i has degree d_i. So, value x_i^2 needs to be summed up d_i times.
But each edge (i,j) has two endpoints so we need $x_i^2 + x_j^2$

Optimize problem

Beside :

x is a unit vector: $\sum_i x_i^2 = 1$

x orthogonal with first eigenvector (1,1,…,1) : $\sum_i x_i = 0$

$$If\ 0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_n,\ then:$$

$$\lambda_2 = argmin_x \frac{\sum_{(i,j)\in E}\left(x_i - x_j\right)^2}{\sum_i x_i^2}$$

2.3

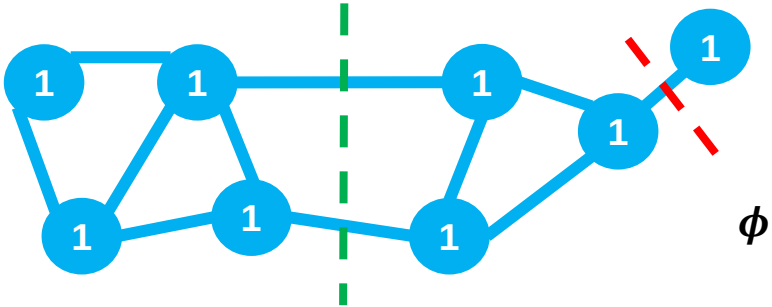
Clustering data

Summarize

Criterion $$\phi(A) = \frac{|\{(i,j) \in E; i \in A, j \notin A\}|}{min\big(vol(A), 2m - vol(A)\big)}$$

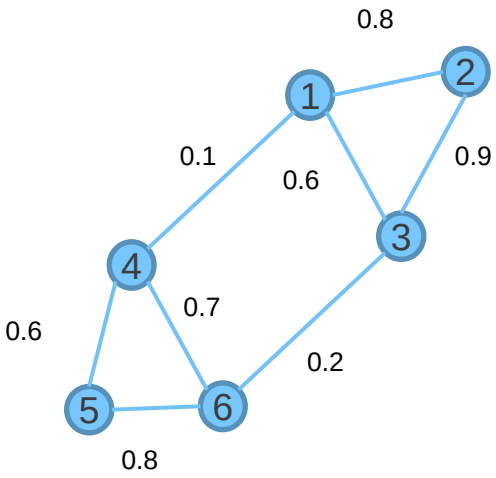
$$\phi = \frac{2}{10} = 0.2$$

How to find optimal cut ? : NP hard



$$\phi = \frac{1}{1} = 1$$

- Transform graph (with weight) into Adjacency matrix



Graph A

	1	2	3	4	5	6
1	0	0.8	0.6	0.1	0	0
2	0.8	0	0.9	0	0	0
3	0.6	0.9	0	0	0	0.2
4	0.1	0	0	0	0.6	0.7
5	0	0	0	0.6	0	0.8
6	0	0	0.2	0.7	0.8	0

Adjacency matrix of A

- Transform Adjacency matrix into Laplace matrix
- Compute Eigenvalues and eigenvector of Laplace matrix

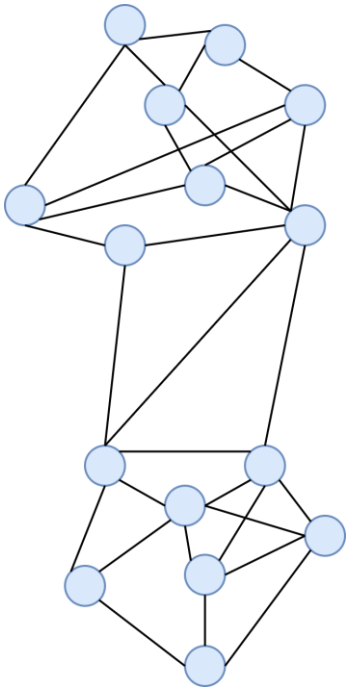
	1	2	3	4	5	6
1	1.5	-0.8	-0.6	-0.1	0	0
2	-0.8	1.7	-0.9	0	0	0
3	0-.6	-0.9	1.7	0	0	-0.2
4	-0.1	0	0	1.4	-0.6	-0.7
5	0	0	0	-0.6	1.4	-0.8
6	0	0	-0.2	-0.7	-0.8	1.7

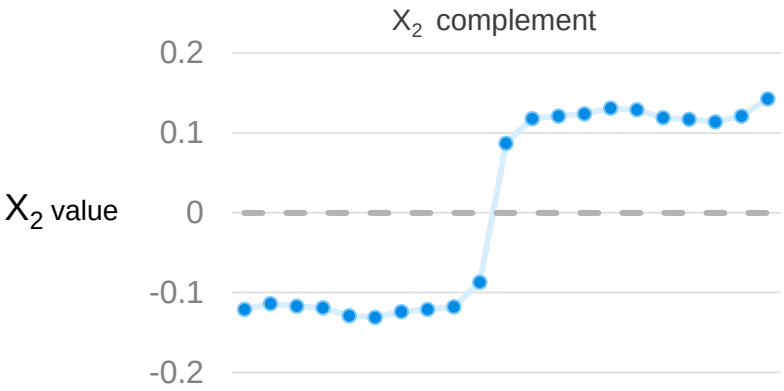
Laplacian matrix of A

$\lambda_1 = 0$	$x_1 = [0.408\ \ 0.408\ \ -0.291\ \ -0.643\ \ -0.149\ \ 0.383]$
$\lambda_2 = 0.189$	$x_2 = [0.408\ \ 0.439\ \ 0.085\ \ 0.110\ \ 0.686\ \ -0.388]$
$\lambda_3 = 1.963$	$x_3 = [0.408\ \ 0.374\ \ 0.233\ \ 0.517\ \ -0.609\ \ -0.026]$
$\lambda_4 = 2.147$	$x_4 = [0.408\ \ -0.403\ \ -0.737\ \ 0.182\ \ -0.090\ \ -0.295]$
$\lambda_5 = 2.673$	$x_5 = [0.408\ \ -0.446\ \ 0.527\ \ -0.444\ \ -0.160\ \ -0.367]$
$\lambda_6 = 2.429$	$x_6 = [0.408\ \ -0.373\ \ 0.182\ \ 0.278\ \ 0.321\ \ 0.693]$

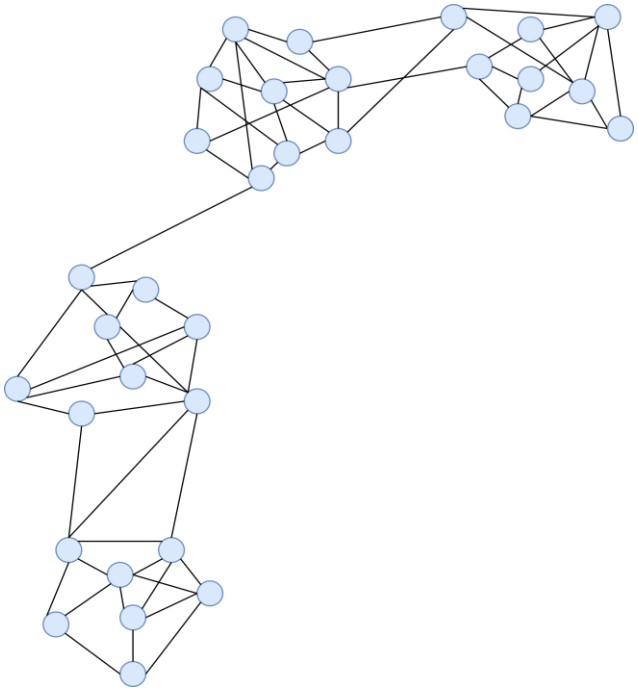
Eigenvalues and eigenvectors of Laplacian matrix

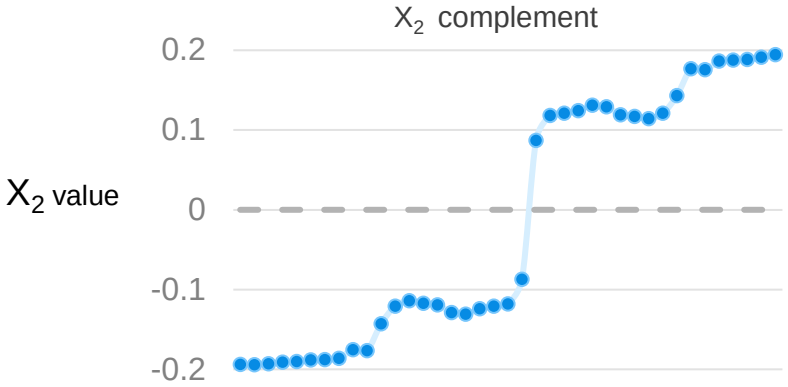
Summarize



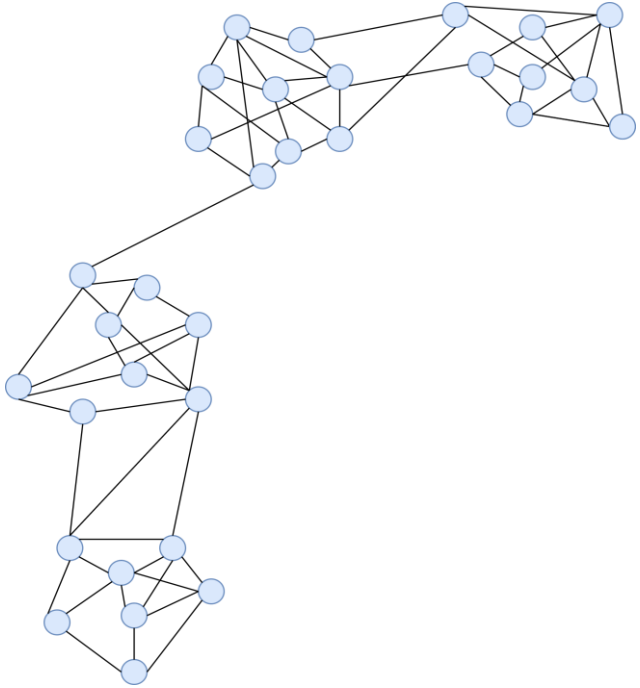


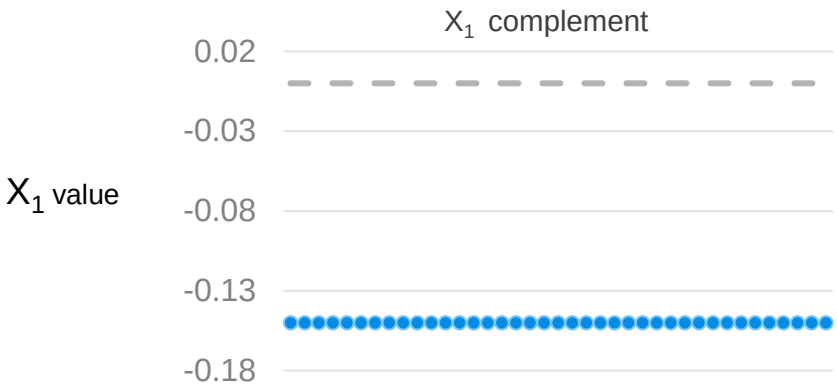
Summarize

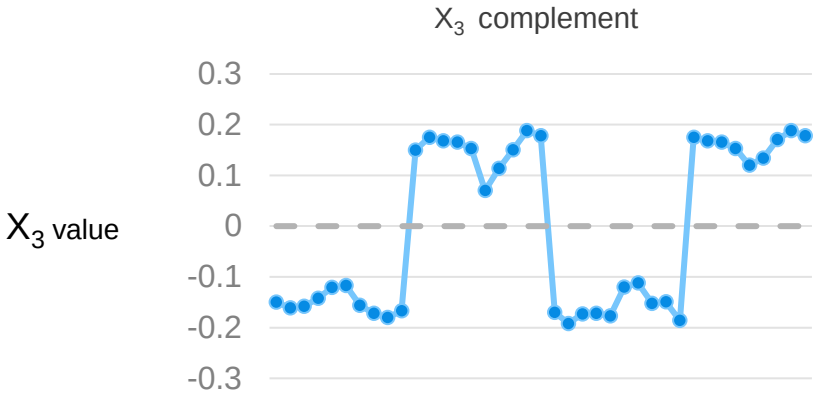




Summarize



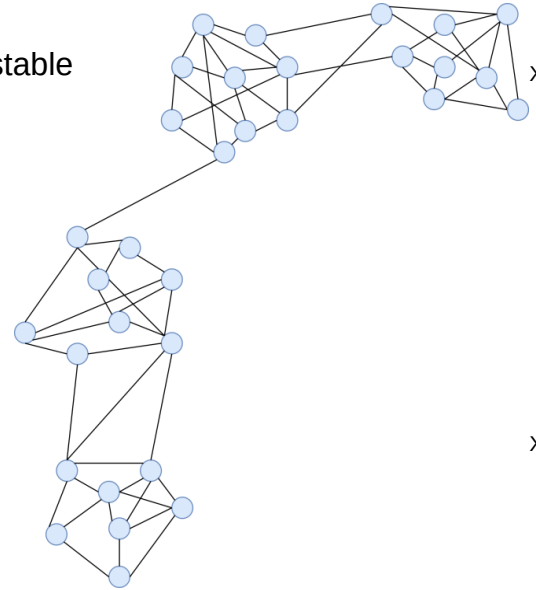

X_1 complement
0.02
-0.03
-0.08
-0.13
-0.18
X_1 value

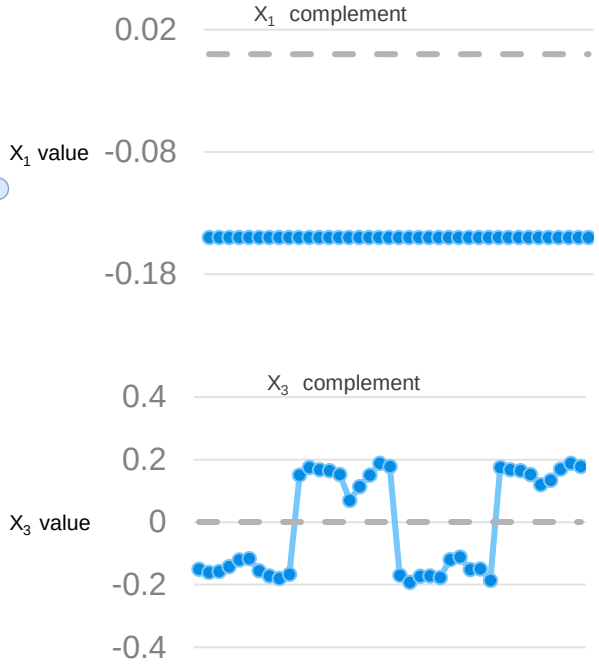

X_3 complement
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
X_3 value

Summarize

How do we partition a graph into k clusters ?
There are 2 basic approaches :

- Recursive bi-partioning : Inefficient and unstable
- Cluster multiple eigenvectors :
  + Build a reduced dimension-space from multiple eigenvectors
  + Commonly used in recent papers
  + A preferable approach...





Optimization

$c^{(i)}$: index of cluster (1,2,... K) which points $x^{(i)}$ currently assigned

μ_K: cluster centroid k ($\mu_K \in R^n$)

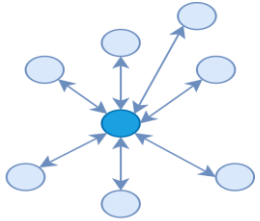
$\mu_{c^{(i)}}$: cluster centroid of cluster to which example $x^{(i)}$ has been assigned

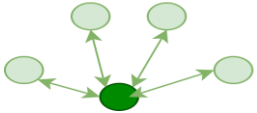
$$\Rightarrow \text{ Optimization : } J(c^1, c^2, \dots c^m \;\; , \;\; \mu_1, \mu_2, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} || x^{(i)} - \mu_{c^{(i)}} ||^2$$

Optimization

$\Rightarrow$ Optimization : $J(c^1, c^2, \dots c^m \;\; , \;\; \mu_1, \mu_2, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} || x^{(i)} - \mu_{c^{(i)}} ||^2$

Repeat {
  For i=1 to m :
    $c^{(i)}$: Index (from 1 to m) of cluster centroid closet to $x^{(i)}$
  For i=1 to n :
    μ_K: Average (means) of points assign to cluster k
  } until μ_K remain unchanged





$\Rightarrow$ Debugging, ensure our algorithm running correctly (advanced :reduce local optima case)

Clustering

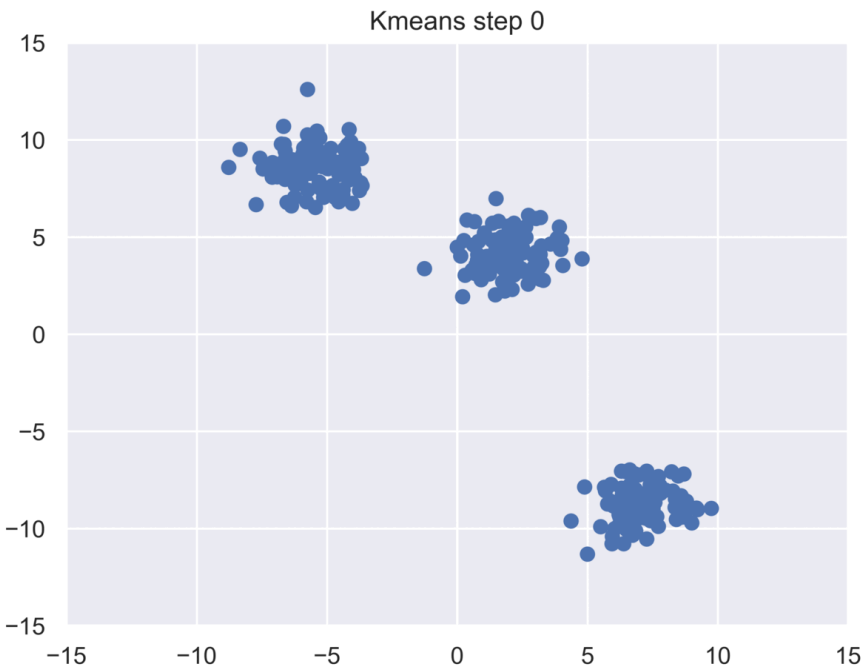
Kmeans algorithm :

...

Randomly initialize K cluster centroid
$\mu_1, \mu_2, \mu_3, \ldots \mu_K \in R$

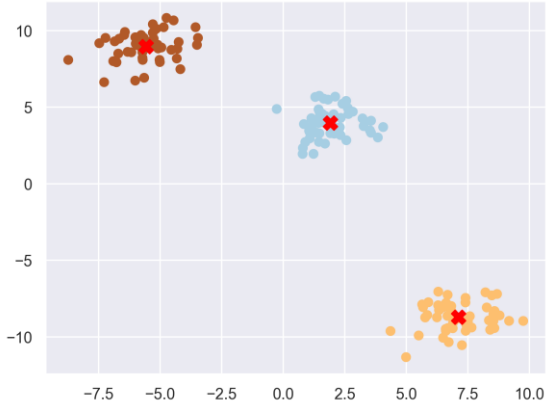
Repeat {
For i=1 to m :
$c^{(i)}$: Index (from 1 to m) of cluster centroid closet to $x^{(i)}$
For i=1 to n :
μ_K: Average (means) of points assign to cluster k
} until μ_K remain unchanged

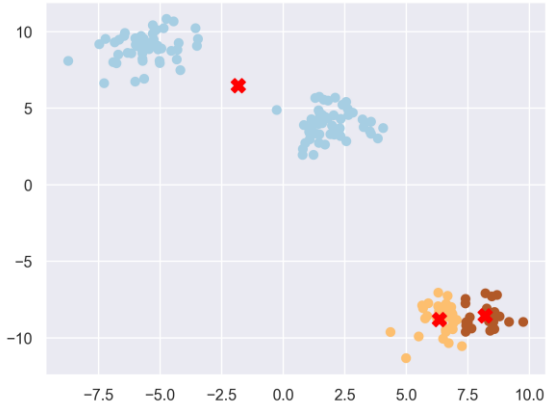
...



Initialize cluster centroid

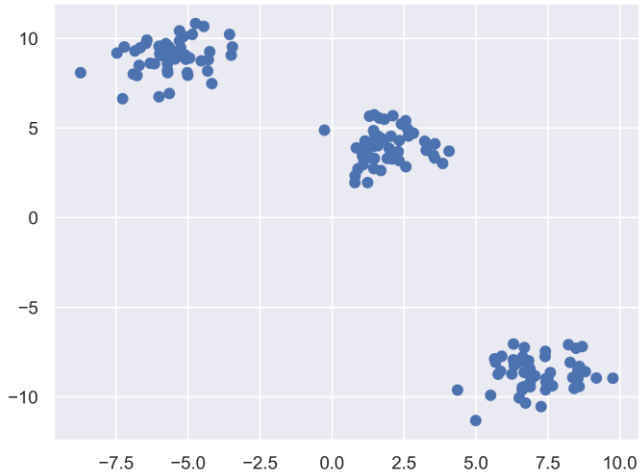
Local optima





$$J(c^1, c^2, \dots c^m \;\; , \;\; \mu_1, \mu_2, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} || \, x^{(i)} - \mu_{c^{(i)}} ||^2$$

Initialize cluster centroid



Loop 50-1000 times :

Generate randomly K-means

Run K-means

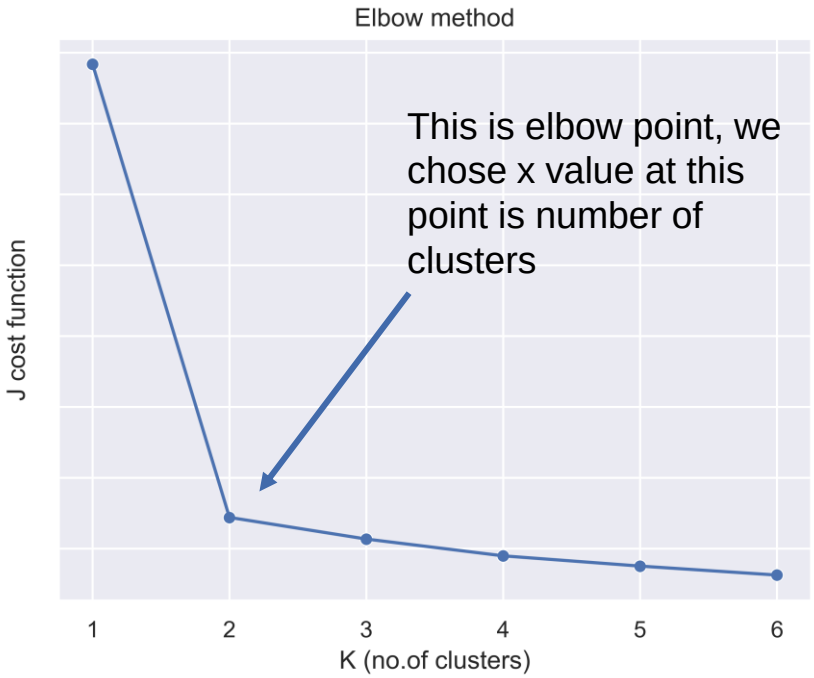
Calculate cost function (distortion)

$J(c^1, c^2, \dots c^m \quad , \quad \mu_1, \mu_2, \dots, \mu_K)$

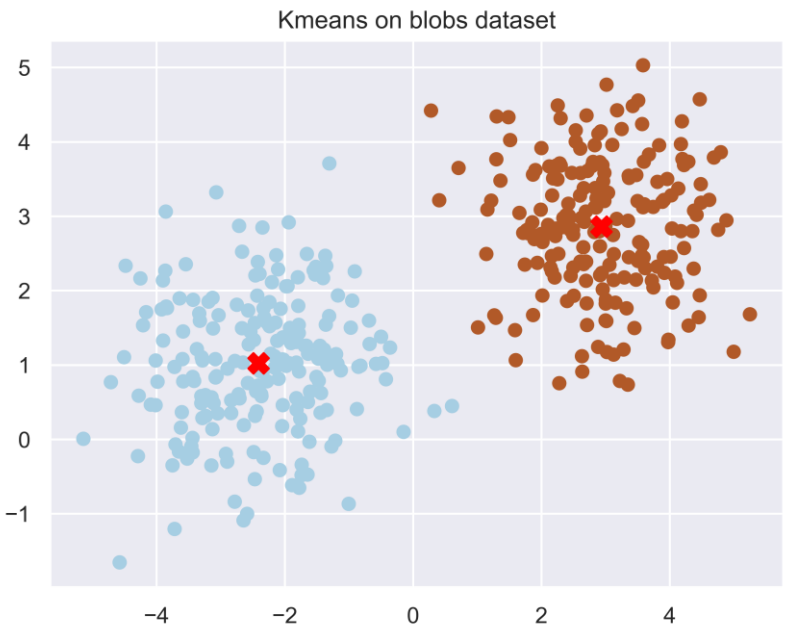
Save the value : $c^1, c^2, \dots c^m, \mu_1, \mu_2, \dots, \mu_K$ and J

=> Choose in saved data the one have smallest J value (Optimal random case)

Choosing number of clusters

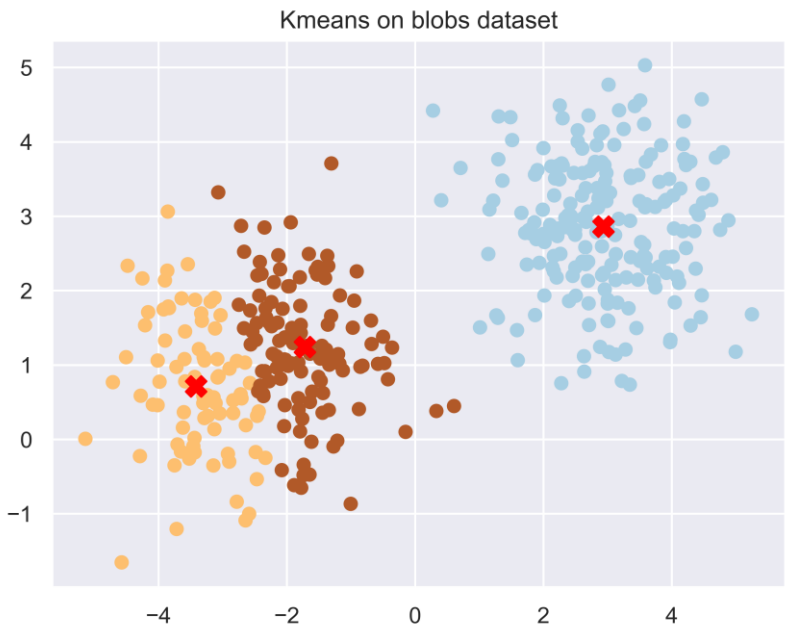


- Example with k = 2 follow elbow method above



We have nearly perfect clusters

- Example with k = 3 follow elbow method above



Cluster yellow and brown almost in the same cluster ⇒ Bad clusters

03

Implementation

Process

- **Step 1:** Create adjacency matrix from data using K-neighbors algorithm. From that, create a Laplacian matrix.
- **Step 2:** Calculate all eigenvalues and eigenvectors of Laplacian matrix, then sort all the eigenvalues (increasing order) to have Laplacian's eigenvalues spectrum.
- **Step 3:** Choosing k (you can choose k) eigenvectors corresponding to first k eigenvalues in spectrum from step 2.
- **Step 4:** Using Kmeans to cluster based on k-vector matrix in step 3. The result of Kmeans algorithm is the solution.

MySpectralClustering

```
class MySpectralClustering:
    def __init__(self, n_cluster = 3, random_state = None, n_neighbor =
10):
        self.k = n_cluster
        self.seed = random_state
        self.nneighbor = n_neighbor
        super.__init__
```

code (link in slide 74)

MySpectralClustering

```
def __generateGraphLaplace(self,X):
        #find connect use kneighbors algorithm
        connectivity = kneighbors_graph(X = X,
n_neighbors=self.nneighbor)
        #build adjacenty Matrix
        adjacentyMatrix = (1/2)*(connectivity+ connectivity.T)
        #build laplace matrix
        graphLaplaceS =
sparse.csgraph.laplacian(csgraph=adjacentyMatrix, normed=False)
        graphLaplace = graphLaplaceS.toarray()
        self.laplaceMatrix = graphLaplace
```

code (link in slide 74)

MySpectralClustering

```
def __getEigenVectorMatrix(self):
        #ucalcute eigenvalues and eigenvectors crossponding to
        eigenValues, eigenVectors = linalg.eig(self.laplaceMatrix)
        eigenValues = np.real(eigenValues)
        eigenVectors = np.real(eigenVectors)

        #sort eigenvalues to get spectral
        eigenValuesSortIndices = np.argsort(eigenValues)
        # get k eigenValues first of spectral
        Indices = eigenValuesSortIndices[:self.k]
        return eigenVectors[:, Indices.squeeze()]
```

code (link in slide 74)

MySpectralClustering

```
def fit(self, X):
        self.__generateGraphLaplace(X)
        EigenVectorsMatrix = self.__getEigenVectorMatrix()
        #to avoid  error when data just has 1 fetures
        if (self.k == 1):
            EigenVectorsMatrix = EigenVectorsMatrix.reshape(1,-1)
        #use kmeaans to cluster on k eigenvector first of spetral
eigenvector
        kMeans = KMeans(n_clusters=self.k,
random_state=self.seed).fit(EigenVectorsMatrix)
        self.labels_ = kMeans.labels_
        self.inertia_ = kMeans.inertia_
        return self
```

code (link in slide 74)

Development

- **Notes:** With a symmetric matrix M we have $\boldsymbol{\lambda_2} = \boldsymbol{min_x} \frac{\boldsymbol{x^T M x}}{\boldsymbol{x^T x}}$
- **Recall that by assumption:** $\mathrm{x}^{\mathrm{T}} L x = \sum_{i,j \in E} \left(x_i - x_j\right)^2$
- **Expectation:** find x to minimize λ_2
- **Theory about $\boldsymbol{x}$:**
  - x is an unit vector : $\sum_i x_i^2 = 1$.
  - x is orthogonal to 1st eigenvector λ_1 (1,1,..,1), thus:
  - $\sum_i x_i . 1 = \sum_i x_i = 0$

- In Laplacian Matrix. λ_2 is the second smallest eigenvalue, Or the second eigenvalue in spectral also know as Fiedler eigenvalue after Miroslav Fiedler (Czech Mathematician).
- This property can describe: If this value greater than 0, this is a connected graph. If this value equal to 0. This graph will include 2 connected graph.
- Consequence: The number of eigenvalue in Laplacian Matrix equal to 0 is the number of connected graph.
- In reality, It is hardly to see a fully separated graph. So we usually take eigenvalue that ≈ 0.
- Every Fiedler's eigenvalue has an eigenvector called Fiedler's eigenvector.

=> The number of eigenvalues approximately 0, which is also the number of clusters (In relatively way)

From MySpectralClustering

```
class MySpectralClustering:
    def __init__(self, n_cluster = 3, random_state = None, n_neighbor =
10):
        self.k = n_cluster
        self.seed = random_state
        self.nneighbor = n_neighbor
        super.__init__
```

code (link in slide 74)

To AutoKSpectralClustering

```
class AutoSpectralClustering:
    def __init__(self, threshold = 1e-2, random_state = None, n_neighbor
= 10):
        self.seed = random_state
        self.nneighbor = n_neighbor
        self.threshold = threshold
        super.__init__
```

code (link in slide 74)

From MySpectralClustering

```
def __getEigenVectorMatrix(self):
        #ucalcute eigenvalues and eigenvectors crossponding to
        eigenValues, eigenVectors = linalg.eig(self.laplaceMatrix)
        eigenValues = np.real(eigenValues)
        eigenVectors = np.real(eigenVectors)

        #sort eigenvalues to get spectral
        eigenValuesSortIndices = np.argsort(eigenValues)
        # get k eigenValues first of spectral
        Indices = eigenValuesSortIndices[:self.k]
        return eigenVectors[:, Indices.squeeze()]
```

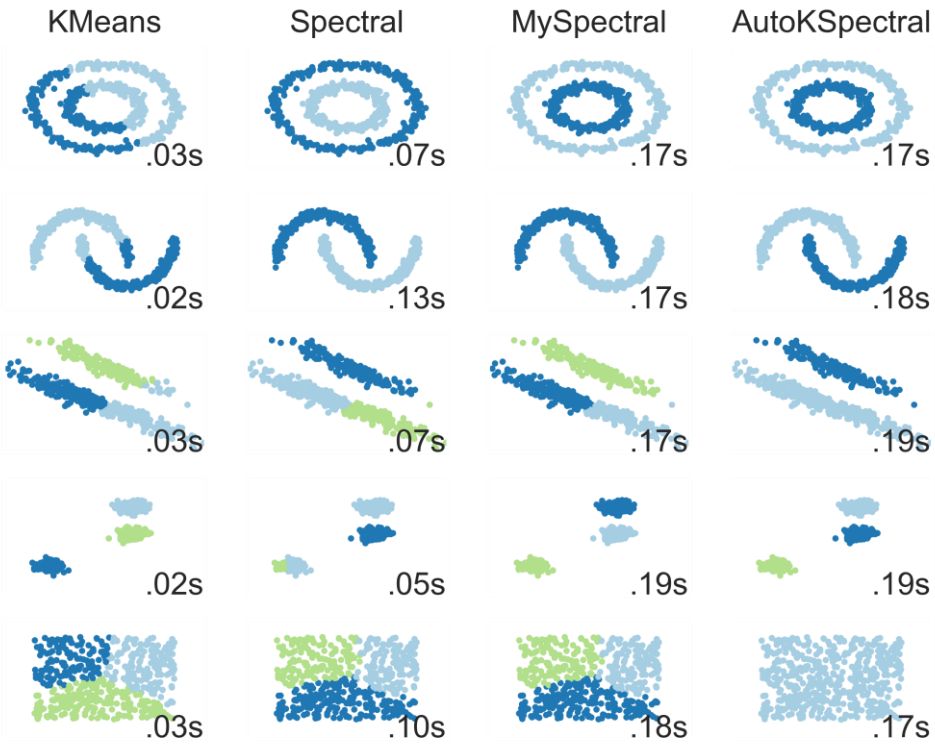
code (link in slide 74)

To AutoKSpectralClustering

```
def __getEigenVectorMatrix(self):
        #ucalcute eigenvalues and eigenvectors crossponding to
        eigenValues, eigenVectors = linalg.eig(self.laplaceMatrix)
        eigenValues = np.real(eigenValues)
        eigenVectors = np.real(eigenVectors)
        #get eigenvalue Fiedler
        zeroEigenValueIndex = np.argwhere(abs(eigenValues) <
self.threshold)
        self.k, _ = zeroEigenValueIndex.shape
        #get eigenvector Fiedler crossponding to eigenvalue Fiedler
        return eigenVectors[:, zeroEigenValueIndex.squeeze()]
```

code (link in slide 74)

Time and algorithm comparasion



Note : Kmeans and Spectral from scikit-learn library, MySpectral and AutoKSpectral was implemented in demo code.

Evaluation

- **Pros**
  - Spectral clustering does not tendency to identify shape of cluster like kmeans
  - High precision clustering
  - Especially, with non-linear problem or unique cluster shape, Spectral clustering is an optimal choice to choose cluster
- **Cons**
  - High complexity $O(n^3)$ compare to $O(n^2)$ of kmeans

Solution

- **Using mini batch.**
  - K-loop.
  - Choosing n (A small part of data point) to divide cluster.

Solution

- **Optimize algorithm:**
  - The complexity of algorithm (big-O notation) spectral clustering come from the calculation of eigenvalues and eigenvectors $(O(n^3))$.
  - To reduce the complexity of this algorithm, we need to reduce the calculation process of finding eigenvalues and eigenvectors by rounding value.[5]

Related link:

- All the demonstrate code and figures were generated by code are saved on Github

References

1) Leskovec, J. (n.d.). *CS224W: Machine Learning with Graphs*. Retrieved December 27, 2022.
2) *sklearn.cluster.SpectralClustering*. (2019). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html
3) Fleshman, W. (2019, February 21). *Spectral Clustering Towards Data Science*. Medium; Towards Data Science. https://towardsdatascience.com/spectral-clustering-aba2640c0d5b
4) Langone, R., & Suykens, J. A. K. (2017). Fast kernel spectral clustering. *Neurocomputing*, *268*, 27–33. https://doi.org/10.1016/j.neucom.2016.12.085
5) *The Nyström Method for Finding Eigenpairs of a Kernel Function Boostedml*. (2020, August 29). Boostedml. https://boostedml.com/2020/08/the-nystrom-method-for-finding-eigenpairs-of-a-kernel-function.html

Thanks for listening our presentation!!