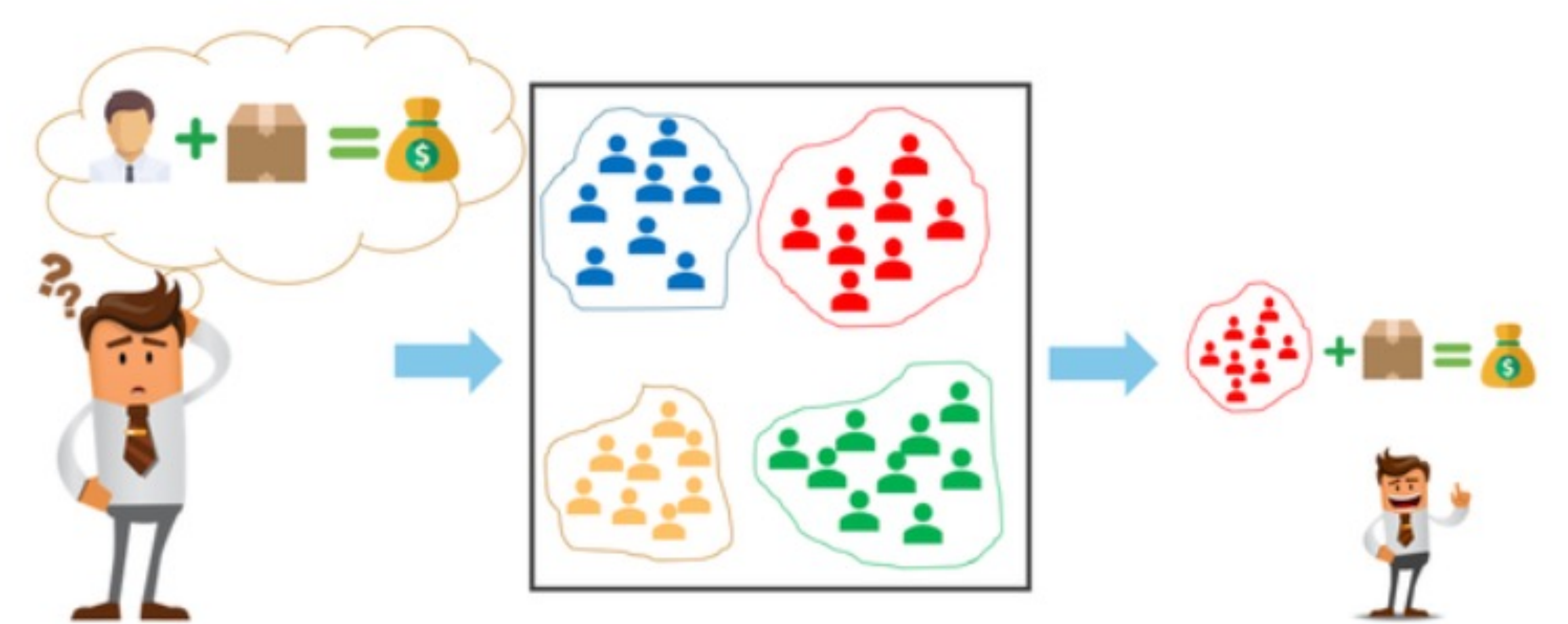
Clustering

ECE57000: Artificial Intelligence

David I. Inouye

2021

Clustering application: Market segmentation to group customers



https://medium.com/analytics-vidhya/customer-segmentation-for-differentiated-targeting-in-marketing-using-clustering-analysis-3ed0b883c18b

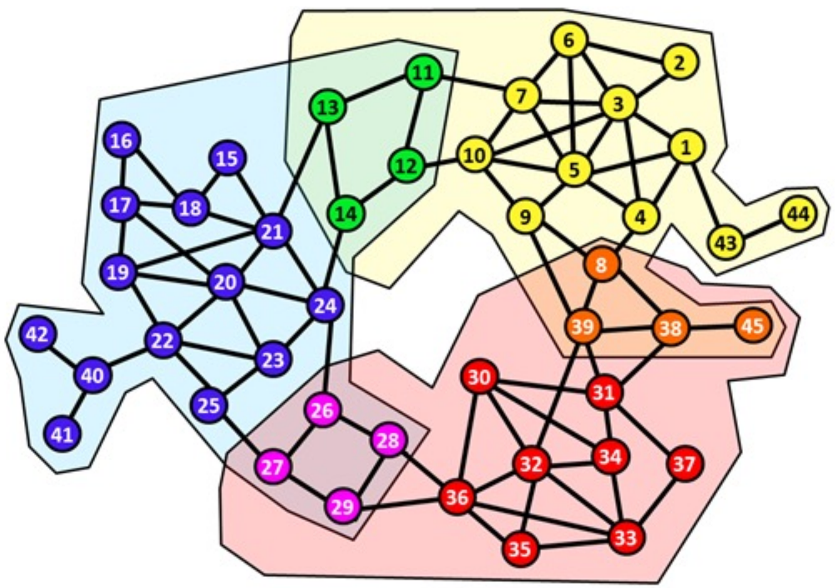
Clustering applications: Discretization of colors for compression

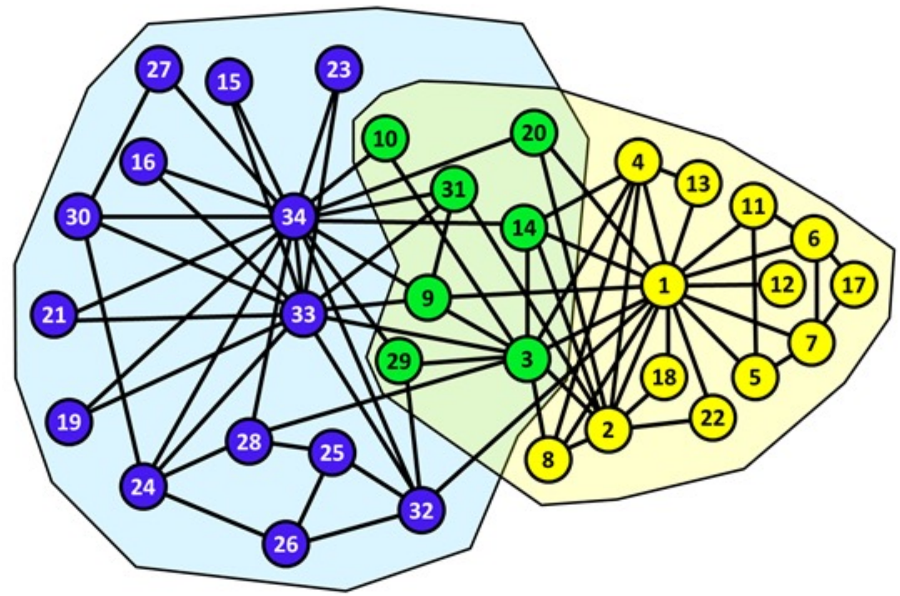
https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html

Clustering applications: Unsupervised image segmentation

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274-2282, Nov. 2012, doi: 10.1109/TPAMI.2012.120.

Another clustering application: Clustering people in social networks





Zachary's Karate Club Network

https://en.wikipedia.org/wiki/Zachary%27s_karate_club

https://bigdata.oden.utexas.edu/project/graph-clustering/

Outline

- Clustering applications
- K-means algorithm (Python demo)
- K-means relation to PCA
- Graph clustering
- Spectral clustering

The K-means algorithm is the most common clustering algorithm

- (See demo of k-means algorithm and derivation)

Recap: Principal Component Analysis (PCA) can be formalized as minimizing the linear reconstruction error of the data using only $k \leq d$ dimensions

- PCA can be formalized as

$$\min_{Z,W} \|X_c - ZW^T\|_F^2$$

  - where
    $X_c = X - \mathbf{1}_n \mu_x^T \in \mathbb{R}^{n \times d}$ (centered input data)
    $Z \in \mathbb{R}^{n \times k}$ (latent representation or "scores")
    $W^T \in \mathbb{R}^{k \times d}$ (principal components)
    $w_s^T w_t = 0, w_s^T w_s = \|w_s\|_2 = 1, \forall s, t$
    (orthogonal constraint)

- Solution
  - $W^T = V_{1:k}^T$ where $X_c = USV^T$ is the **SVD** of X_c
  - $Z = X_c W$

K-means relation to PCA: One-hot vectors vs continuous vectors

- k-means clustering can be seen as reducing the dimensionality to k latent categories
  - Each category can be represented by a one-hot vector of length k
    e.g., if $k = 3$, $z_i \in \{[1,0,0], [0,1,0], [0,0,1]\}, \forall i$
  - Every instance can only "belong" to one category
- In dimensionality reduction techniques, the latent vectors can have non-zeros for all k latent dimensions
  - e.g., if $k = 3$, $z_i \in \mathbb{R}^3, \forall i$

K-means objective can be reformulated as seeking the best approximation to X with low rank constraint ($k < d$)

- Original k-means objective

$$\min_{\substack{\mathcal{C}_1,\dots,\mathcal{C}_k \\ \mu_1,\dots,\mu_k}} \sum_{j=1}^{k} \sum_{x \in \mathcal{C}_j} \left\| x - \mu_j \right\|_2^2$$

- Equivalent to the following objective

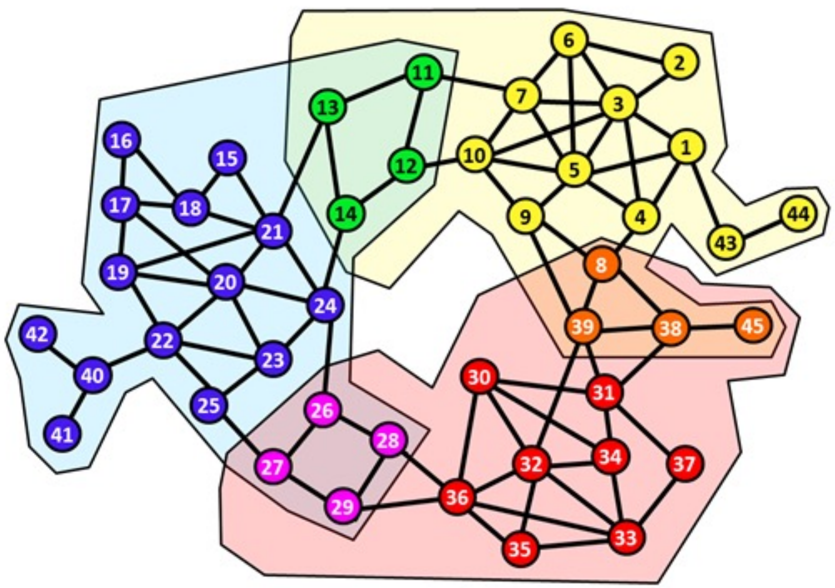
$$\min_{Z,M} \|X - ZM\|_F^2$$
$$\text{where } Z \in \{0,1\}^{n \times k}, \sum_j z_{ij} = 1, \forall i$$
$$\text{and } M \in \mathbb{R}^{k \times d}$$

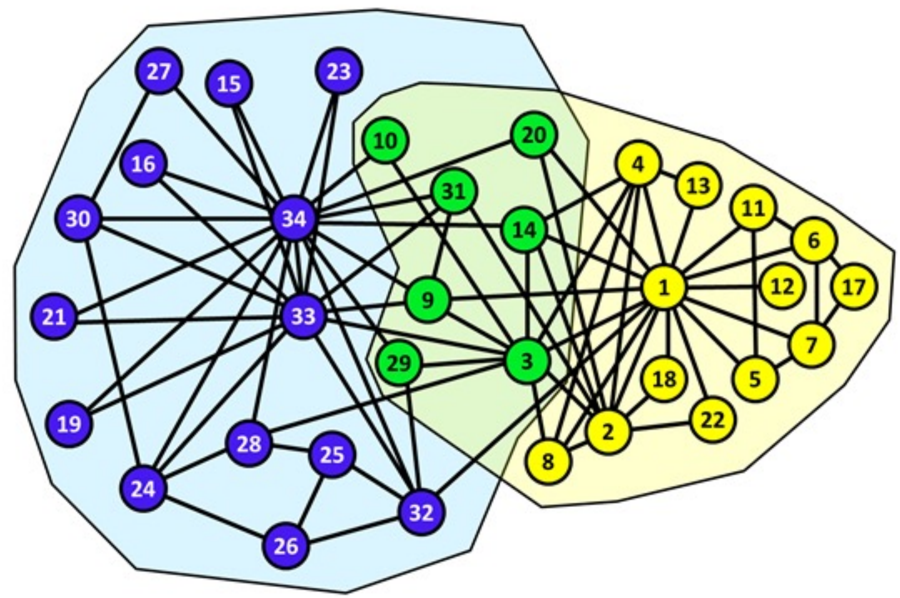
- Notice the similarity and differences with PCA objective and constraints

Derivation of equivalence between two objectives for k-means

- $y_i \in \{1, \dots, k\}$ is the cluster label for each instance
- z_i is the corresponding one hot vector to y_i
- $M = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_k \end{bmatrix}$ is the matrix of mean vectors
- $\sum_{j=1}^{k} \sum_{x \in \mathcal{C}_j} \left\| x - \mu_j \right\|_2^2$
- $= \sum_{i=1}^{n} \left\| x_i - \mu_{y_i} \right\|_2^2$
- $= \sum_{i=1}^{n} \left\| x_i^T - z_i^T M \right\|_2^2$ (row vector form)
- $= \sum_{i=1}^{n} \sum_{s=1}^{d} \left(x_{is} - z_i^T m_s \right)^2$ (m_s is a column of M)
- $= \left(\sqrt{\sum_{i=1}^{n} \sum_{s=1}^{d} \left(x_{is} - z_i^T m_s \right)^2} \right)^2$
- $= \|X - ZM\|_F^2$

How can we cluster the nodes of a network (a.k.a. a graph) instead of a set of points?





Zachary's Karate Club Network

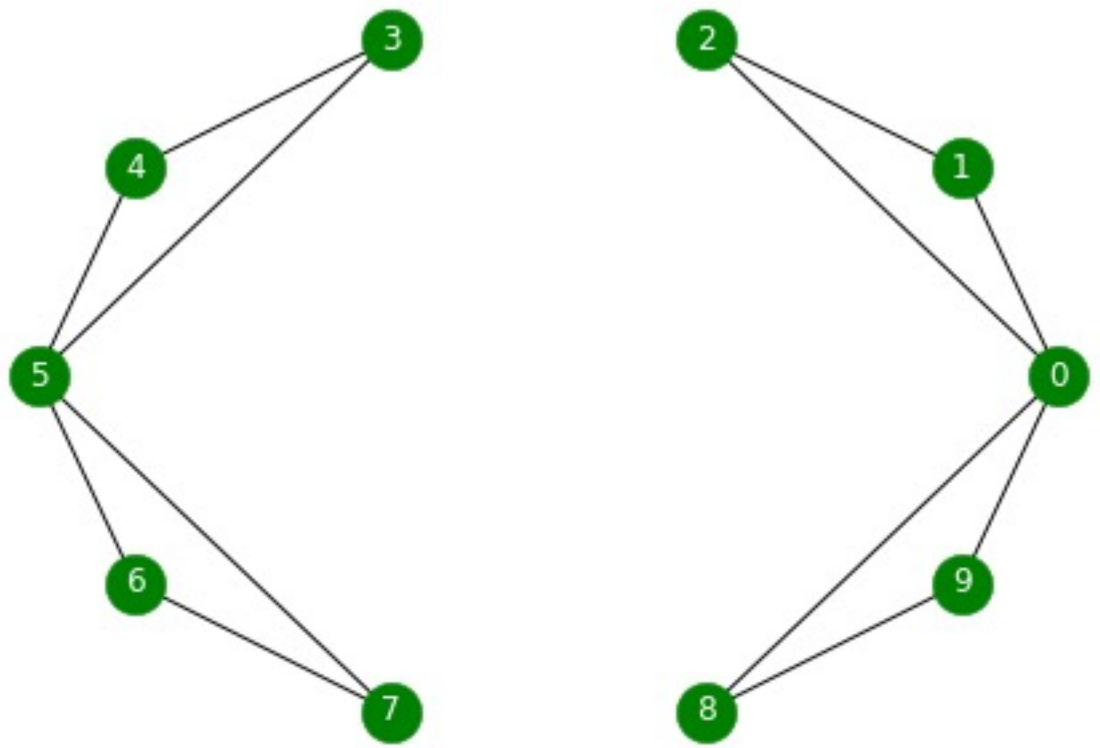
https://en.wikipedia.org/wiki/Zachary%27s_karate_club

https://bigdata.oden.utexas.edu/project/graph-clustering/

Graph clustering puts the nodes of a graph into clusters

- What is a graph?
- How do we represent a graph?
  - Adjacency matrix
  - Graph Laplacian
- How do we use graph Laplacian to cluster?

A **graph**/**network** is composed of nodes and weighted edges between the nodes

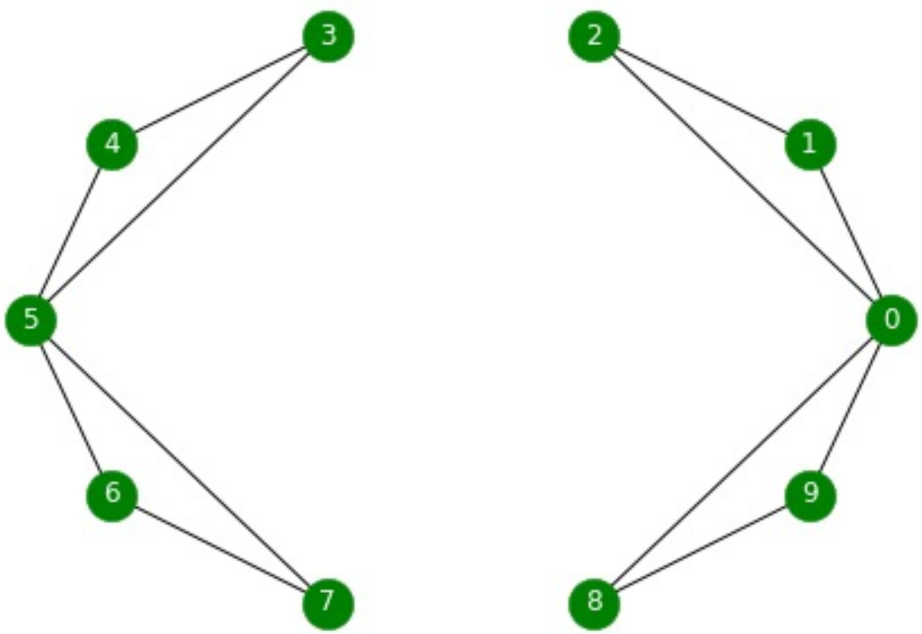


10 node graph with 2 disconnected components.

https://towardsdatascience.com/spectral-clustering-aba2640c0d5b

A graph can be represented as an <u>adjacency matrix</u>

- Nodes are represented by rows/columns
- Edges are encoded as 1s




```
[[0 1 1 0 0 0 0 0 1 1]
 [1 0 1 0 0 0 0 0 0 0]
 [1 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0 0 0]
 [0 0 0 1 0 1 0 0 0 0]
 [0 0 0 1 1 0 1 1 0 0]
 [0 0 0 0 0 1 0 1 0 0]
 [0 0 0 0 0 1 1 0 0 0]
 [1 0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 1 0]]
```

https://towardsdatascience.com/spectral-clustering-aba2640c0d5b

The <u>graph Laplacian</u> is formed by subtracting the adjacency from the degree matrix

- The degree matrix is a diagonal matrix whose elements are the sum of the rows:

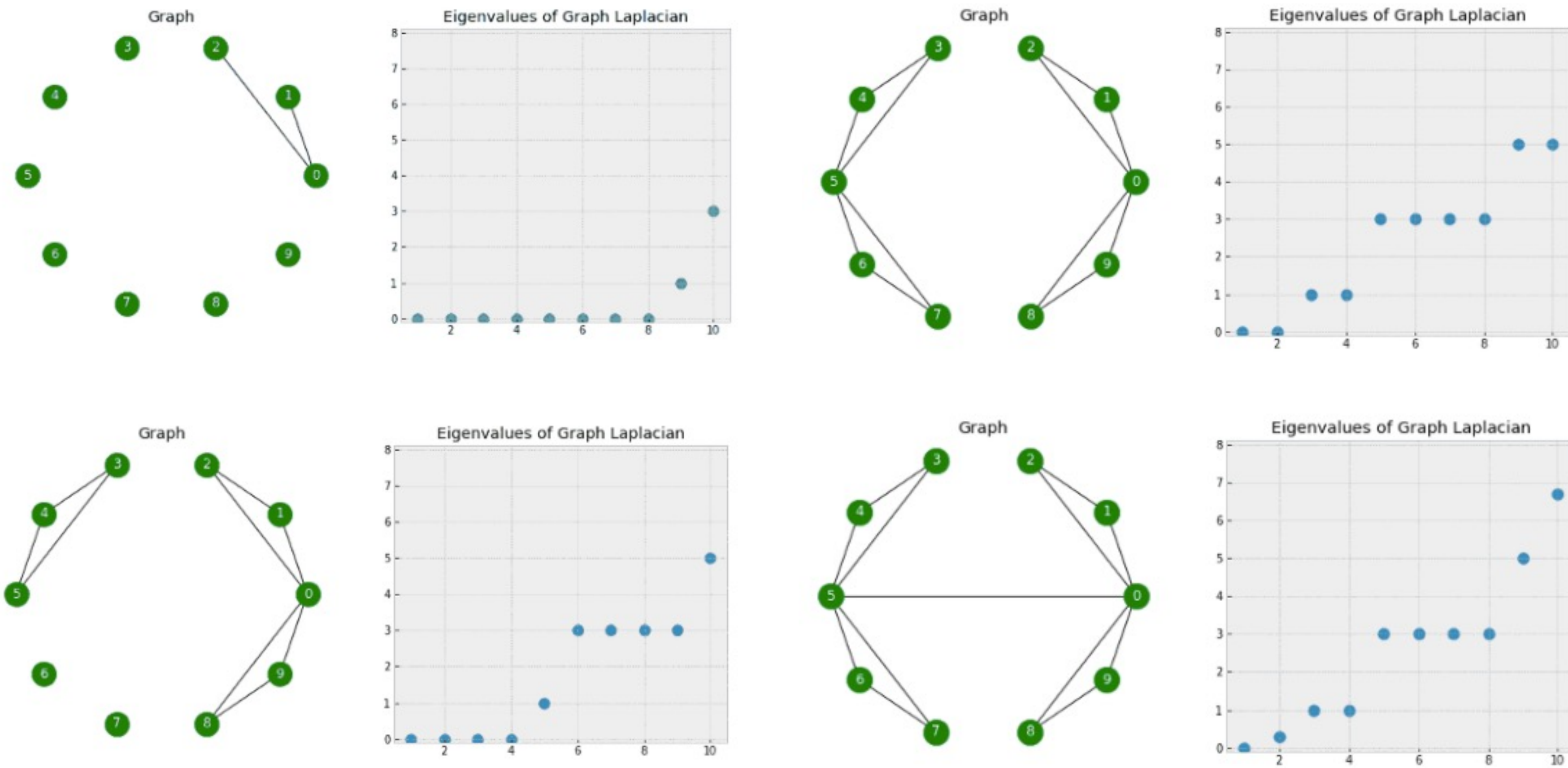
$$D = \mathrm{diag}(A1)$$

- Graph Laplacian is defined as:

$$L = D - A$$

$$\begin{bmatrix} 4 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The number of 0 eigenvalues of the Laplacian is the number of connected components



https://towardsdatascience.com/spectral-clustering-aba2640c0d5b

The Fiedler vector (2^{nd} to last eigenvector) can be used to create 2 clusters

- Intuitively, we could 0 the 2^{nd} to last eigenvalue to get 2 components instead

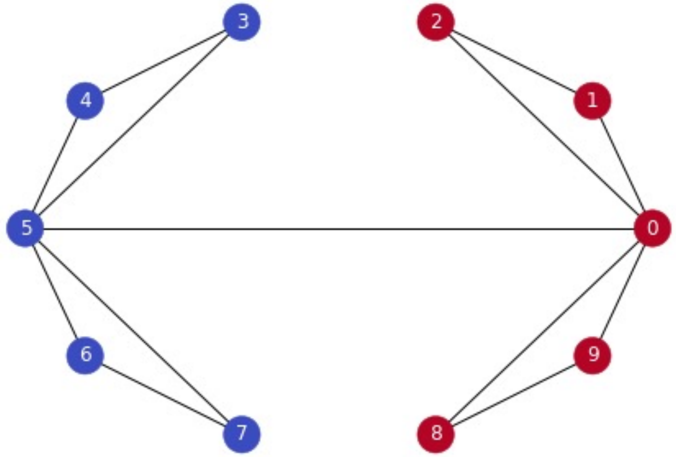
- Nodes are clustered based on whether their values in the Fiedler vector

$$y = 1(f > 0)$$

- In theory, this is known as the **minimal cut**

Fiedler vector

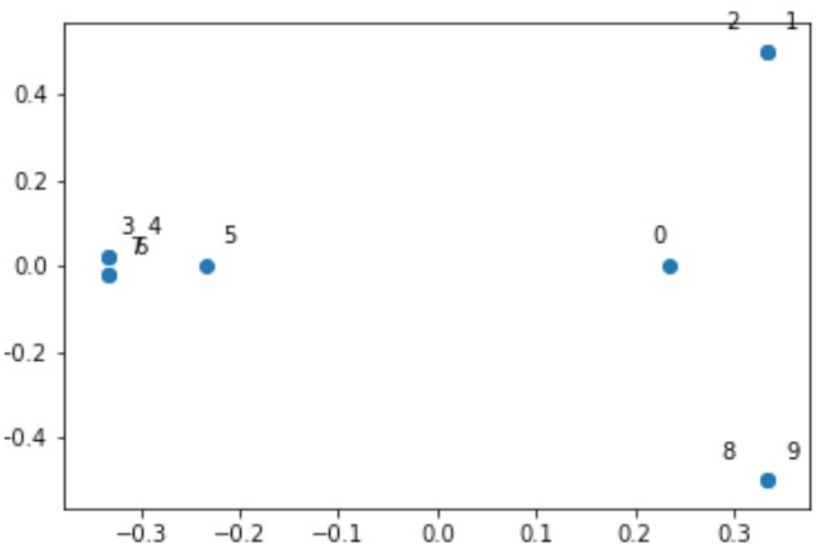
```
0   0.234
1   0.333
2   0.333
3  -0.333
4  -0.333
5  -0.234
6  -0.333
7  -0.333
8   0.333
9   0.333
```

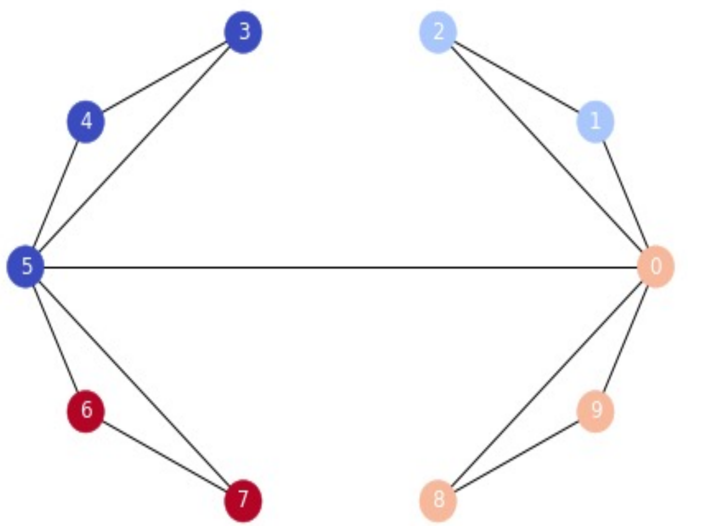


Spectral clustering generalizes to $k > 2$ clusters by taking the lowest eigenvectors as a new node representation and then doing K-means

- We take the m-lowest eigenvectors to represent the data
- Then just run K-means

2 lowest eigenvector representations of nodes

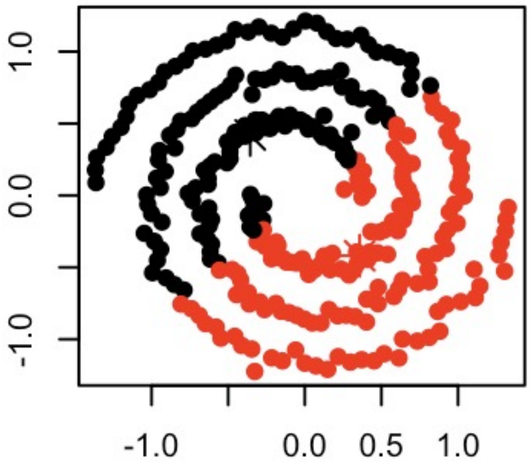


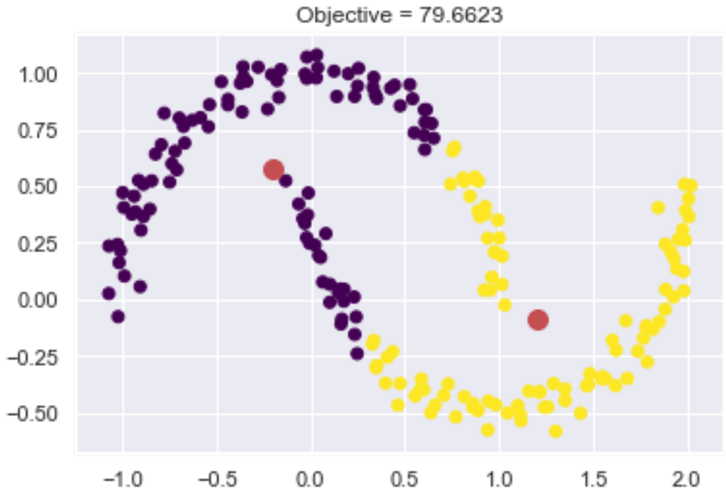


https://towardsdatascience.com/spectral-clustering-aba2640c0d5b

Standard K-means clustering is limited to circular clusters with linear boundaries between clusters

- K-means is based on the clustering assumption of "compactness"
  - Points in a cluster are close to one another
  - Squared error objective within cluster
- This assumption may not be appropriate

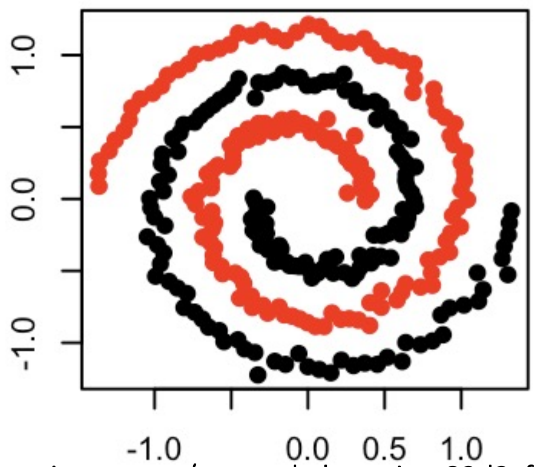


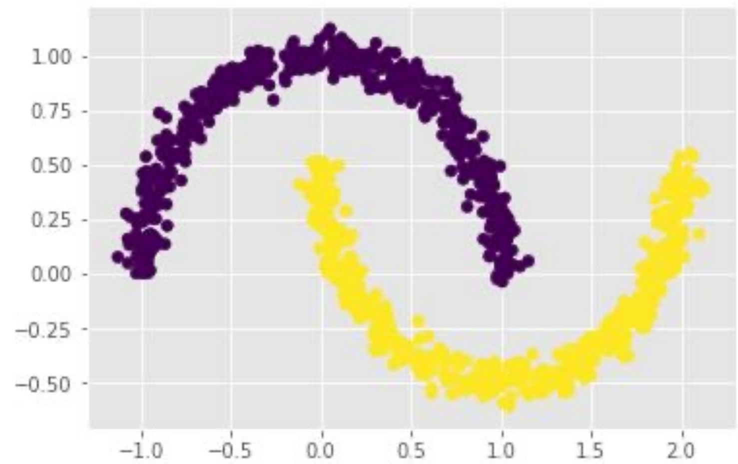


https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7

Spectral clustering applied to vector data can be used to learn clusters based on "connectivity"

- Points that are "connected" to each other are clustered together
- This allows non-circular clustering





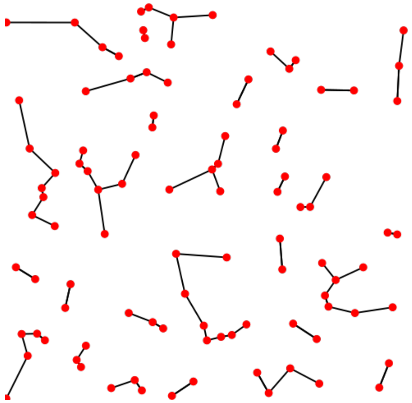
https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7

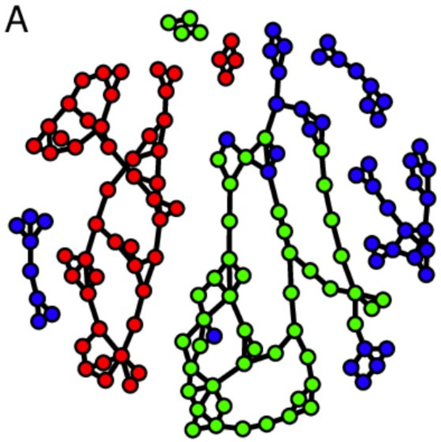
Spectral clustering: First create similarity graph based on data

- K-nearest neighbor graph
  - Add edge for all k-nearest neighbors

- General similarity graph
  - Compute all pairwise similarity between points such as:

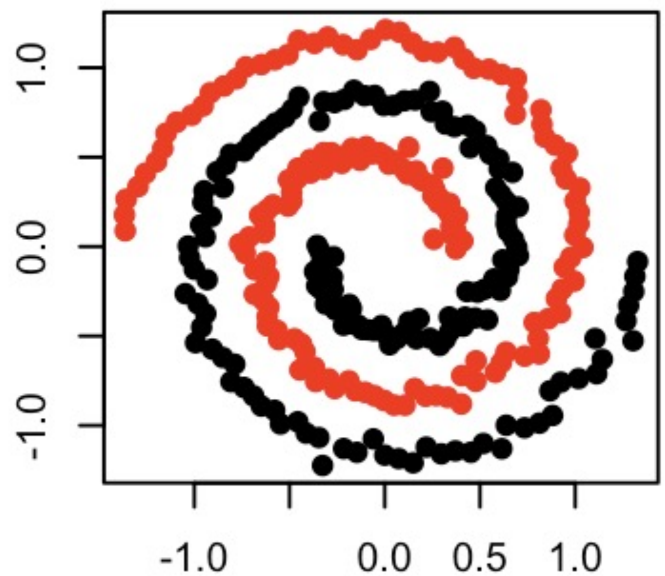
$$s(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$



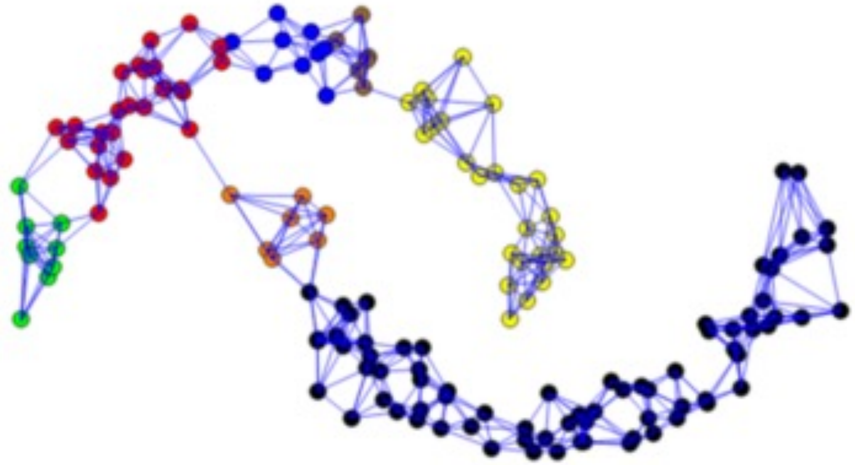


P. Veenstra, C. Cooper and S. Phelps, "Spectral clustering using the kNN-MST similarity graph," *2016 8th Computer Science and Electronic Engineering (CEEC)*, Colchester, 2016, pp. 222-227, doi: 10.1109/CEEC.2016.7835917.

Spectral clustering:
Second, apply spectral clustering to resulting graph

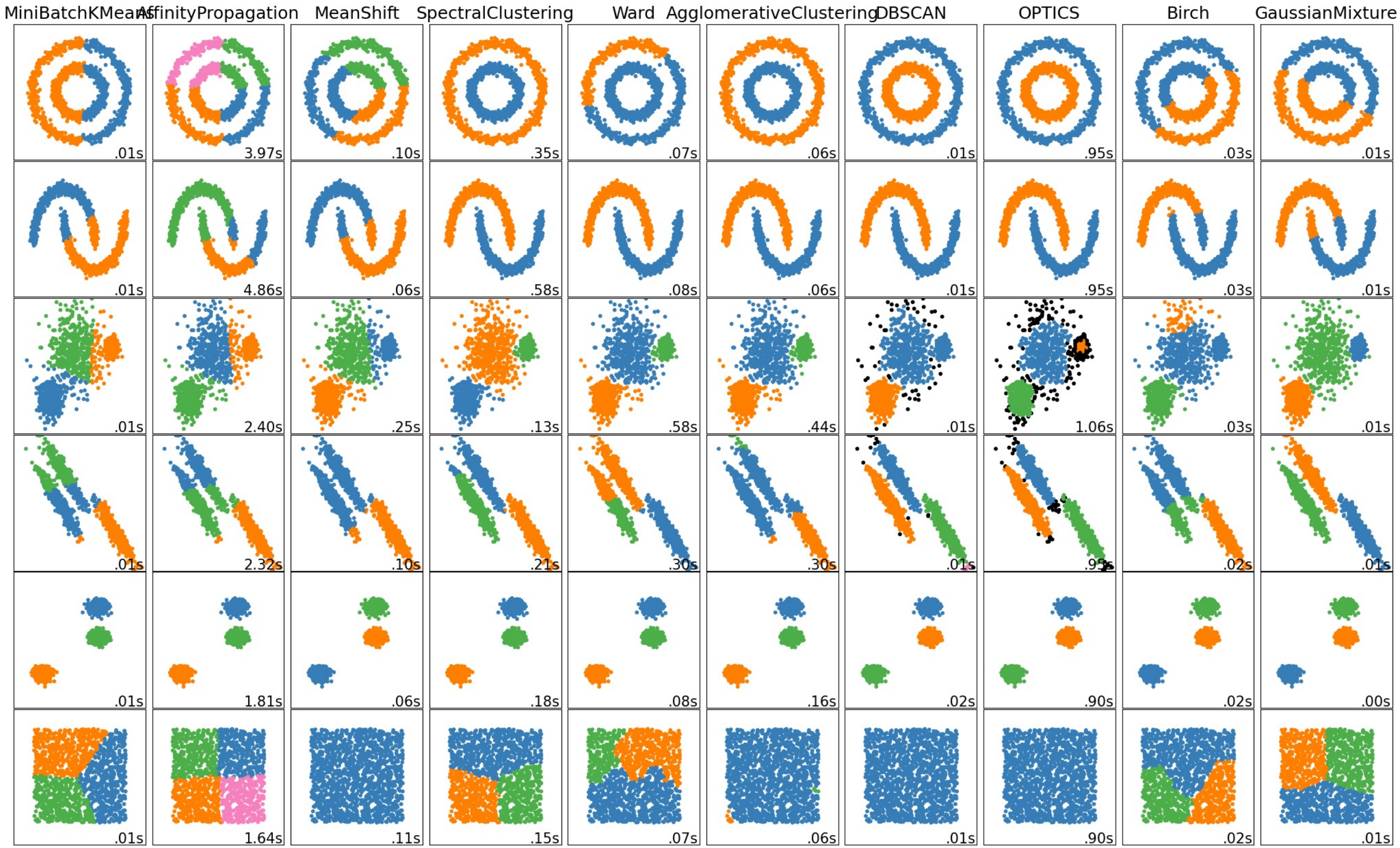


https://towardsdatascience.com/spectral-clustering-82d3cff3d3b7



http://math.ucdenver.edu/~sborgwardt/wiki/index.php/Spectral_clustering

Many other clustering algorithms exist



https://scikit-learn.org/stable/modules/clustering.html

Simple demo of spectral clustering (time permitting)