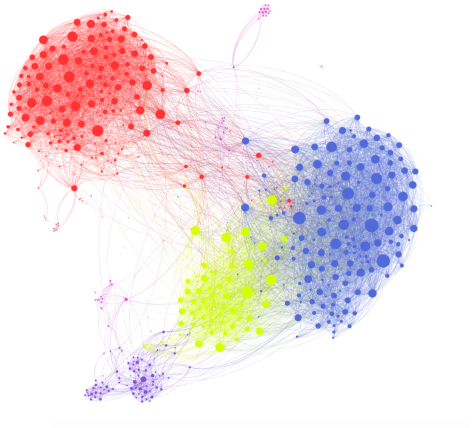
# Mathematical techniques in data science

Lecture 16: Clustering

## Supervised and unsupervised learning

- Supervised learning problems
    - Labelled data $(X, Y)$ with joint density $P(X, Y)$
    - We are mainly interested in the conditional density $P(Y|X)$.
- Unsupervised learning problems
    - Data $X$ is not labelled and has density $P(X)$
    - We want to infer properties of $P(X)$

## Clustering

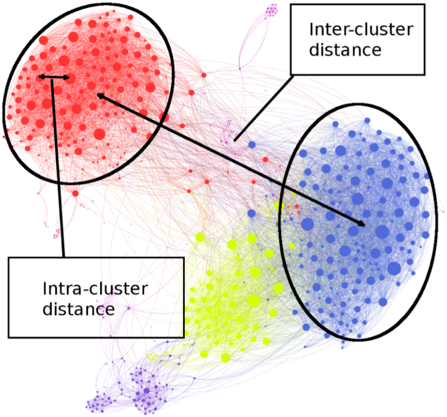


- Unsupervised problem
- Want to label points according to a measure of their similarity

## Clustering

We try to partition observations into “clusters” such that:

- Intra-cluster distance is minimized.
- Inter-cluster distance is maximized.



## K-means clustering

The K-means algorithm is a popular algorithm to cluster a set of points in $\mathbb{R}^p$.

- We are given $n$ observations $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$.
- We are given a number of clusters $K$.
- We want a partition $\hat{S} = \{S_1, \ldots, S_K\}$ of $\{x_1, \ldots, x_n\}$ such that
$$\hat{S} = \operatorname*{argmin}_{S} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$
where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ is the mean of the points in $S_i$ (the "center" of $S_i$).

## K-means clustering

- We want a partition $\hat{S} = \{S_1, \dots, S_K\}$ of $\{x_1, \dots, x_n\}$ such that
$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$
where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ is the mean of the points in $S_i$ (the "center" of $S_i$).
- The above problem is NP hard.
- Efficient approximation algorithms exist (converge to a local minimum though).

## Lloyd's algorithm

Lloyds's algorithm for K-means clustering

- Denote by $C(i)$ the cluster assigned to $x_i$.
- Lloyds's algorithm provides a heuristic method for optimizing the K-means objective function.

Start with a "cluster centers" assignment $m_1^{(0)}, \ldots, m_K^{(0)}$. Set $t := 0$. Repeat:

1. Assign each point $x_j$ to the cluster whose mean is closest to $x_j$:
$$S_i^{(t)} := \{x_j : \|x_j - m_i^{(t)}\|^2 \le \|x_j - m_k^{(t)}\|^2 \ \forall k = 1, \ldots, K\}.$$
2. Compute the average $m_i^{(t+1)}$ of the observations in cluster $i$:
$$m_i^{(t+1)} := \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j.$$

Example, Dense

## Convergence of Lloyd's algorithm

Note that Lloyds's algorithm uses a greedy approach to sequentially minimize:

$$\sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - m_i\|^2.$$

- Both steps of the algorithm decrease the objective.
- Thus, Lloyds's algorithm converges a local minimum of the objective function.

There is no guarantee that Lloyds' algorithm will find the **global** optimum.

Local mean

## Lloyd's algorithm: initiation step

- There is no guarantee that Lloyds' algorithm will find the global optimum
- As a result, we use different starting points
- Common initiation schemes:
    - The Forgy method: Pick K observations at random and use these as the initial means
    - Random partition: Randomly assign a cluster to each observation and compute the mean of each cluster
    - kmeans++ (default in sklearn)

## kmeans++ initiation

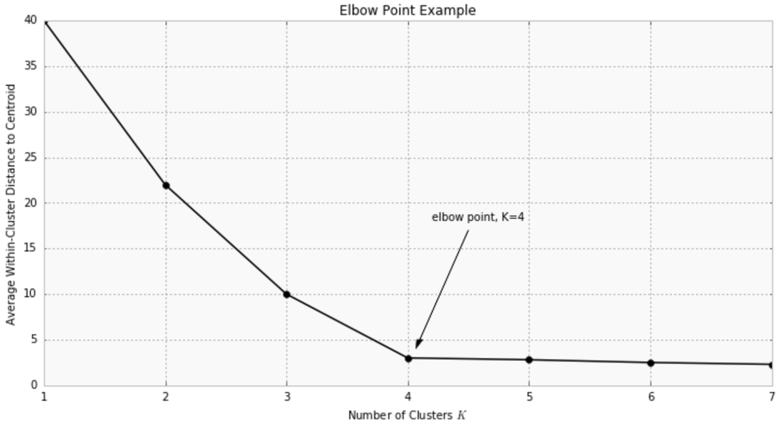
Intuition: spreading out the k initial cluster centers is a good thing

- Choose one center uniformly at random from among the data points.
- For each data point $x$, compute $D(x)$, the distance between $x$ and the nearest center that has already been chosen.
- Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$
- Repeat Steps 2 and 3 until $k$ centers have been chose

## Choosing $k$

- Elbow method
- Cross-validation
- Average silhouette method
- Gap statistic method

# Elbow method



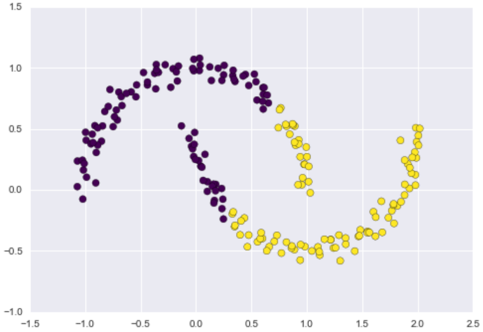
## Silhouette method

- a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation)
- ranges from $[-1, 1]$
- The Silhouette coefficient is defined for each sample and is composed of two scores:
  - $a$: The mean distance between a sample and all other points in the same class.
  - $b$: The mean distance between a sample and all other points in the next nearest cluster
- The Silhouette coefficient (sklearn.metrics.silhouette_score) for a single sample is then given as:

$$s = \frac{b - a}{\max(a, b)}$$

## Issues with k-means

- k-means is limited to linear cluster boundaries



- Solution: adding non-linearities to the model
  - kernel k-means
  - spectral clustering

## Kernel k-means

## Kernel k-means = kernel trick + k-means

- Ideas:
  - maps the data to a high-dimensional space (called feature space) by a non-linear function $\phi$ to separate the clusters linearly
  - Using this high-dimensional representation to run k-means
  - Project the data back to the original space to identify the clusters
- Note: the kernel trick works best if we don't have to construct $\phi(x)$ explicitly, but can compute

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

- For k-means, we need to compute

$$\|\phi(x_i) - m_j\|^2$$

## Kernels

| | |
|---|---|
| Polynomial Kernel | $\kappa(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b}+c)^d$ |
| Gaussian Kernel | $\kappa(\mathbf{a},\mathbf{b}) = \exp(-\|\|\mathbf{a}-\mathbf{b}\|\|^2/2\sigma^2)$ |
| Sigmoid Kernel | $\kappa(\mathbf{a},\mathbf{b}) = \tanh(c(\mathbf{a}\cdot\mathbf{b})+\theta)$ |

## Kernel k-means = kernel trick + k-means

Note that

$$\begin{aligned}\|\phi(x_i) - m_j\|^2 &= \langle \phi(x_i) - m_j, \phi(x_i) - m_j \rangle \\ &= \langle \phi(x_i), \phi(x_i) \rangle - 2\langle \phi(x_i), m_j \rangle + \langle m_j, m_j \rangle\end{aligned}$$

Given a cluster $C_j$, its center (on feature space) is

$$m_j = \frac{1}{|C_j|} \sum_{b \in C_j} \phi(b)$$

Thus

$$\langle \phi(x_i), m_j \rangle = \frac{1}{|C_j|} \sum_{b \in C_j} \langle \phi(x_i), \phi(b) \rangle = \frac{1}{|C_j|} \sum_{b \in C_j} K(x_i, b)$$

## Kernel k-means = kernel trick + k-means

Note that

$$\begin{aligned}\|\phi(x_i) - \mu_j\|^2 &= \langle \phi(x_i) - u_j, \phi(x_i) - u_j \rangle \\ &= \langle \phi(x_i), \phi(x_i) \rangle - 2\langle \phi(x_i), u_j \rangle + \langle u_j, u_j \rangle\end{aligned}$$

Given a cluster $C_j$, its center (on feature space) is

$$m_j = \frac{1}{|C_j|} \sum_{b \in C_j} \phi(b)$$

Thus

$$\langle m_j, m_j \rangle = \frac{1}{|C_j|^2} \sum_{b,c \in C_j} K(b, c)$$

## Kernel k-means

**Input:** $K$: kernel $\qquad$ $k$: number of clusters

**Output:** $C_1, ...., C_k$: partitioning of the points

1. Initialize the $k$ clusters: $C_1^{(0)}, ..., C_k^{(0)}$.
2. Set $t = 0$.
3. For each point $\mathbf{a}$, find its new cluster index as

$$j^*(\mathbf{a}) = \text{argmin}_j \|\phi(\mathbf{a}) - \mathbf{m}_j\|^2, \text{ using (2)}.$$

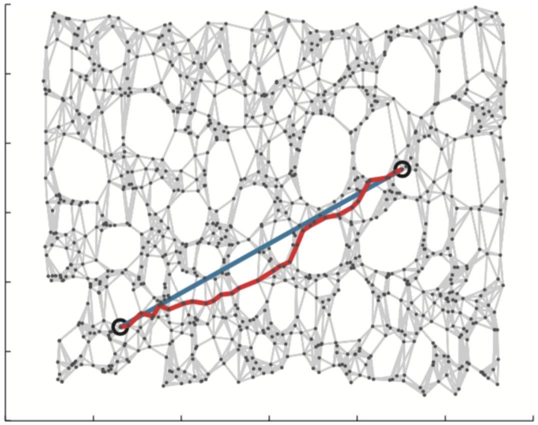
4. Compute the updated clusters as

$$C_j^{t+1} = \{\mathbf{a} \ : \ j^*(\mathbf{a}) = j\}.$$

5. If not converged, set $t = t + 1$ and go to Step 3; Otherwise, stop.

## Spectral clustering

## Recall: Graph-based manifold embedding



## Spectral embedding (Laplace eigenmap)

Step 1: Construct the neighbor graph

- For each point, determine either
  - $K$ nearest neighbors
  - all points in a fixed radius
- each point is connected to its neighbours
- edge length equal to ~~Euclidean distance between the points~~

$$W_{ij} = e^{-\frac{|x_i - x_j|^2}{4t}}$$

## Spectral embedding (Laplace eigenmap)

Step 2: Embedding by Laplace operator's eigenvectors

- Define $L = D - W$
- We want to minimize

$$\min_{\langle Df, f\rangle = 1} \langle Lf, f\rangle$$

- Solve for eigenvectors $\{f_1, f_2, \ldots, f_m\}$
- Map

$$x \to (\langle f_1, x\rangle, \langle f_2, x\rangle, \ldots, \langle f_m, x\rangle)$$

## Spectral clustering: overview

1. Construct a *similarity matrix* measuring the similarity of pairs of objects.
2. Use the similarity matrix to construct a (weighted or unweighted) graph.
3. Compute eigenvectors of the *graph Laplacian* (builds an embedding of the graph into $\mathbb{R}^p$).
4. Cluster the graph

## Neighbor graph

Step 1: Construct the neighbor graph

- For each point, determine either
  - $K$ nearest neighbors
  - all points in a fixed radius
- each point is connected to its neighbours
- edge length equal to ~~Euclidean distance between the points~~

$$W_{ij} = e^{-\frac{|x_i - x_j|^2}{4t}}$$

## Remarks

$$W_{ij} = e^{-\frac{|x_i - x_j|^2}{4t}}$$

$\rightarrow$ the RBF (Gaussian) kernel

- If $x_i \approx x_j$, then $\langle \phi(x_i), \phi(x_j) \rangle \approx 0$
- If $x_i$ is far from $x_j$, then $\phi(x_i) \perp x_j$
- This means that when the RBF kernel is used, we are virtually mapping the dataset to an infinite dimensional space, where the points are clustered around some vector that are perpendicular to each other

## Notations

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \dots, v_n\}$ and edge set $E \subset V \times V$.
- Each edge carries a *weight* $w_{ij} \geq 0$.
- The adjacency matrix of $G$ is $W = W_G = (w_{ij})_{i,j=1}^n$. We will assume $W$ is symmetric (undirected graphs).
- The *degree* of $v_i$ is
$$d_i := \sum_{j=1}^n w_{ij}.$$
- The *degree matrix* of $G$ is $D := \operatorname{diag}(d_1, \dots, d_n)$.
- We denote the complement of $A \subset V$ by $\overline{A}$.

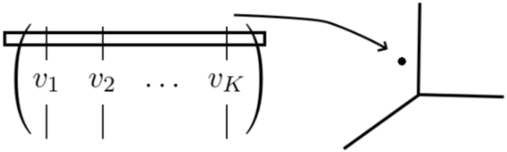
## Graph embedding

Step 2:

- Compute eigenvectors of the (normalized or unnormalized) graph Laplacian

$$L = D - W, \quad L_{sym} = D^{-1/2} L D^{-1/2}$$

- Construct a matrix containing the smallest $K$ eigenvectors of $L$ or $L_{sym}$ as columns
- Normalize the rows to have norm 1
- Each row identifies a vertex of the graph to a point in $\mathbb{R}^K$



How does spectral clustering work?

## Ideal case

- Suppose that $k = 3$, and the three cluster of size $n_1$, $n_2$, $n_3$ are $S_1, S_2, S_3$
- Assume that the data $S = \{x_1, x_2, \ldots, x_n\}$ are arranged in such a way that the first $n_1$ points are in $S_1$, the next $n_2$ in $S_2$, and so on
- Assume further that the cluster are infinitely far apart

## Ideal case

$$A = D - W = \begin{bmatrix} A^{11} & 0 & 0 \\ 0 & A^{22} & 0 \\ 0 & 0 & A^{33} \end{bmatrix}$$

and

$$L_{sym} = \begin{bmatrix} L^{11} & 0 & 0 \\ 0 & L^{22} & 0 \\ 0 & 0 & L^{33} \end{bmatrix}$$

where

$$L^{ii} = (D^{ii})^{-1/2}(A^{ii})(D^{ii})^{-1/2}$$

Note: $L^{ii}$ is non-negative definite and have eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$

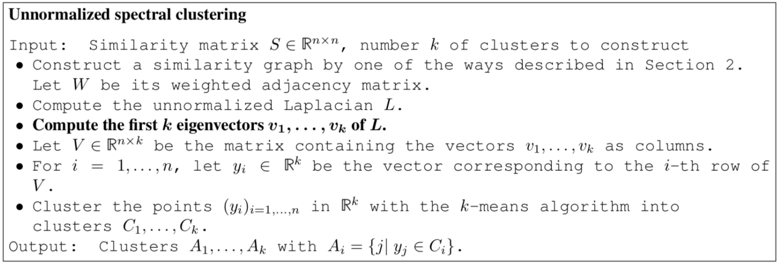
## Ideal case

- The 3 smallest eigenvectors have the forms $(x_1, 0, 0)$, or $(0, x_2, 0)$, or $(0, 0, x_3)$
- Stack them by columns

$$x = \begin{bmatrix} x_1 & 0 & 0 \\ 0 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} \in \mathbb{R}^{n \times 3}$$

- If we normalize the row, then all points in the first cluster are mapped to $(1, 0, 0)$

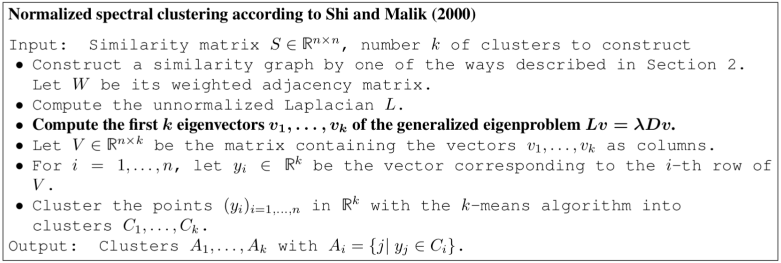
## Variations



**Unnormalized spectral clustering**

```
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct
  • Construct a similarity graph by one of the ways described in Section 2.
    Let $W$ be its weighted adjacency matrix.
  • Compute the unnormalized Laplacian $L$.
  • Compute the first $k$ eigenvectors $v_1, \dots, v_k$ of $L$.
  • Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \dots, v_k$ as columns.
  • For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of
    $V$.
  • Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into
    clusters $C_1, \dots, C_k$.
Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.
```

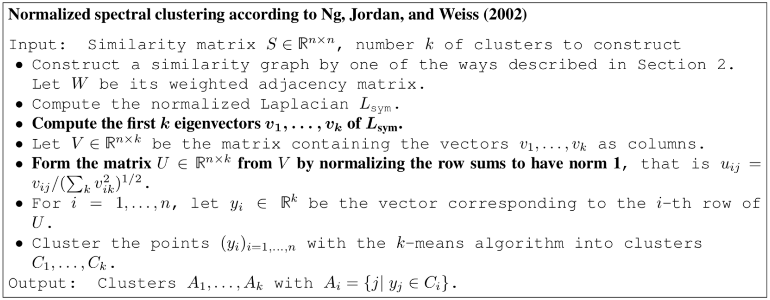
# Variations



**Normalized spectral clustering according to Shi and Malik (2000)**

```
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct
  • Construct a similarity graph by one of the ways described in Section 2.
    Let $W$ be its weighted adjacency matrix.
  • Compute the unnormalized Laplacian $L$.
  • Compute the first $k$ eigenvectors $v_1, \dots, v_k$ of the generalized eigenproblem $Lv = \lambda Dv$.
  • Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \dots, v_k$ as columns.
  • For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of
    $V$.
  • Cluster the points $(y_i)_{i=1,\dots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into
    clusters $C_1, \dots, C_k$.
Output: Clusters $A_1, \dots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.
```

## Variations



**Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)**

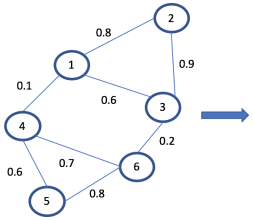
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct

- Construct a similarity graph by one of the ways described in Section 2. Let $W$ be its weighted adjacency matrix.
- Compute the normalized Laplacian $L_{\text{sym}}$.
- **Compute the first $k$ eigenvectors $v_1, \ldots, v_k$ of $L_{\text{sym}}$.**
- Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \ldots, v_k$ as columns.
- **Form the matrix $U \in \mathbb{R}^{n \times k}$ from $V$ by normalizing the row sums to have norm 1,** that is $u_{ij} = v_{ij} / (\sum_k v_{ik}^2)^{1/2}$.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $U$.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ with the $k$-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

## Example

$G =$

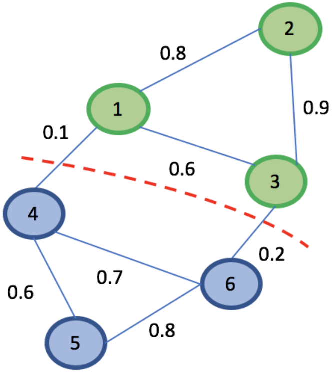


$L =$

| | 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 |

## Example

| | v2 |
|---|---|
| 1 | 0.41 |
| 2 | 0.44 |
| 3 | 0.37 |
| 4 | -0.40 |
| 5 | -0.45 |
| 6 | -0.37 |


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