MATH 567: Mathematical Techniques in Data Science
Clustering I

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

May 1, 2017

Supervised and unsupervised learning

Supervised learning problems:

- Data (X, Y) is "labelled" (input/output) with joint density $P(X, Y)$.
- We are mainly interested by the conditional density $P(Y|X)$.
- Example: regression problems, classification problems, etc..

Supervised and unsupervised learning

Supervised learning problems:

- Data (X, Y) is "labelled" (input/output) with joint density $P(X, Y)$.
- We are mainly interested by the conditional density $P(Y|X)$.
- Example: regression problems, classification problems, etc..

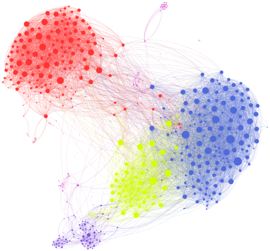
Unsupervised learning problems:

- Data X is **not** labelled and has density $P(X)$.
- We want to infer properties of $P(X)$ without the help of a "supervisor" or "teacher".
- Examples: Density estimation, PCA, ICA, sparse autoencoder, clustering, etc..

Clustering

Wikipedia - Chire.

→

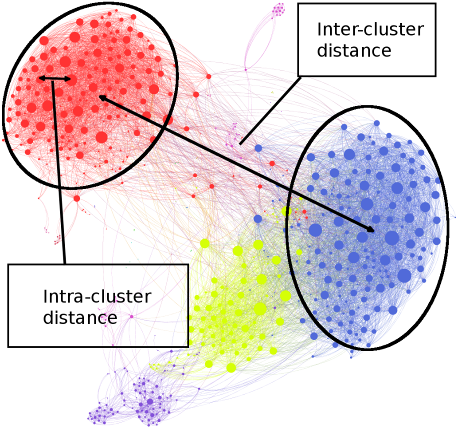


- Unsupervised problem.
- Work only with features/independent variables.
- Want to label points according to a measure of their similarity.

What is a cluster?

We try to partition observations into "clusters" such that:

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



For graphs, we want vertices in the same cluster to be highly connected, and vertices in different clusters to be mostly disconnected.

The K-means algorithm

- Goes back to Hugo Steinhaus (of the Banach–Steinhaus theorem) in 1957.

Source: Wikipedia.

Steinhaus authored over 170 works. Unlike his student, Stefan Banach, who tended to specialize narrowly in the field of functional analysis, Steinhaus made contributions to a wide range of mathematical sub-disciplines, including geometry, probability theory, functional analysis, theory of trigonometric and Fourier series as well as mathematical logic. He also wrote in the area of applied mathematics and enthusiastically collaborated with engineers, geologists, economists, physicians, biologists and, in Kac's words, "even lawyers".

The K-means algorithm (cont.)

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

The K-means algorithm (cont.)

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

- We are given n observations $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$.

The K-means algorithm (cont.)

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.

The K-means algorithm (cont.)

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

$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

  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 K-means algorithm (cont.)

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

$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

  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.

The K-means algorithm (cont.)

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

$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

  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).

Lloyds's algorithm for K-means clustering

Lloyds's algorithm

Lloyds's algorithm for K-means clustering

- Denote by $C(i)$ the cluster assigned to x_i.

Lloyds'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.

Lloyds'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:

Lloyds'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 $m_1^{(0)}, \dots, m_K^{(0)}$. Set $t := 0$. Repeat:

1. Assign each point x_j to the cluster whose mean is closest to x_j:

$$S_i^{(t)} := \{x_j : \|x_j - m_i^{(t)}\|^2 \leq \|x_j - m_k^{(t)}\|^2 \ \forall k = 1, \dots, K\}.$$

Lloyds'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 $m_1^{(0)}, \dots, m_K^{(0)}$. Set $t := 0$. Repeat:

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

Lloyds'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:
$$m_i^{(t+1)} := \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j.$$
3. $t \leftarrow t + 1$.

Until convergence.

Convergence of Lloyds's algorithm

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

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

Convergence of Lloyds'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.

Convergence of Lloyds'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.

Convergence of Lloyds'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.

Convergence of Lloyds'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.

As a result, we use different **starting points** (i.e., different choices for the initial means $m_i^{(0)}$).

Convergence of Lloyds'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.

As a result, we use different **starting points** (i.e., different choices for the initial means $m_i^{(0)}$).

Common initialization methods:

1. **The Forgy method:** Pick K observations at random from $\{x_1, \ldots, x_n\}$ and use these as the initial means.

Convergence of Lloyds'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.

As a result, we use different **starting points** (i.e., different choices for the initial means $m_i^{(0)}$).

Common initialization methods:

1. **The Forgy method:** Pick K observations at random from $\{x_1, \dots, x_n\}$ and use these as the initial means.
2. **Random partition:** Randomly assign a cluster to each observation and compute the mean of each cluster.

Illustration of the K-means algorithm

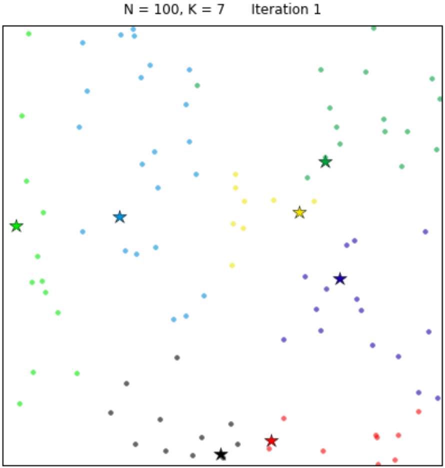
- 100 random points in $\mathbb{R}^2$.

Illustration of the K-means algorithm

- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).

Illustration of the K-means algorithm

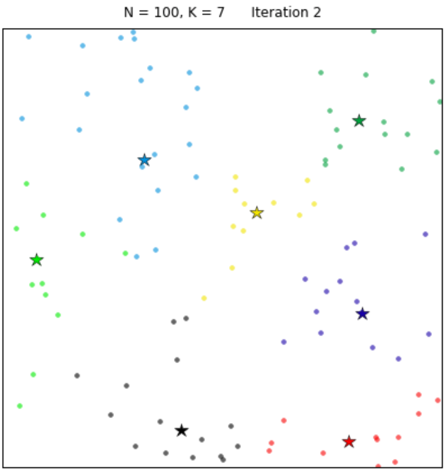
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

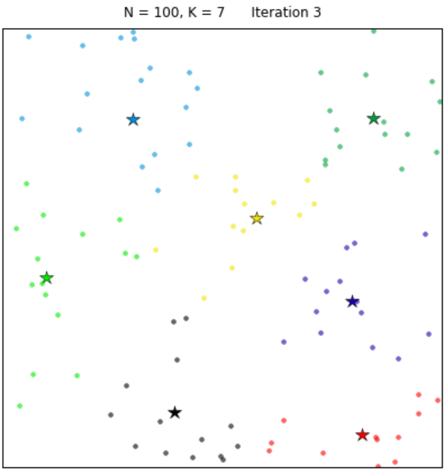
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

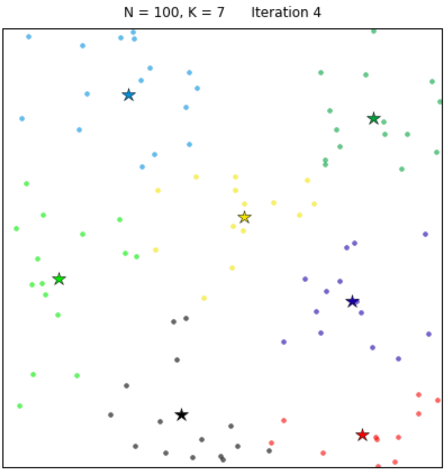
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

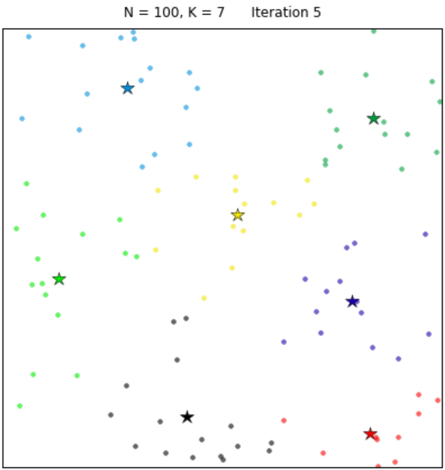
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

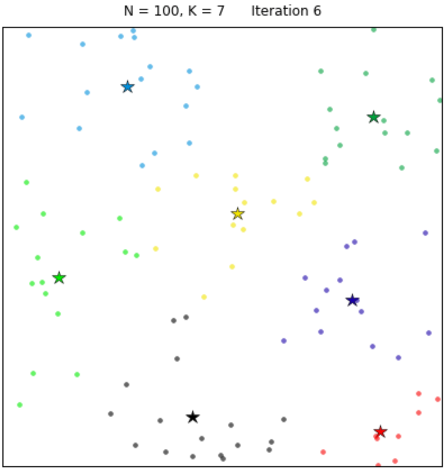
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

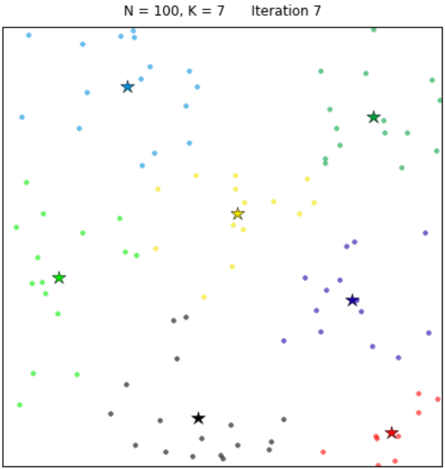
- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Illustration of the K-means algorithm

- 100 random points in $\mathbb{R}^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).



Source: https://datasciencelab.wordpress.com

Example: clustering the zip data

Is there a nice cluster structure in the zip dataset?

Experiment:

- Find 10 clusters using K-means.
- Compute the percentage p_{ij} of samples labelled i having "true" label j.

Example: clustering the zip data

Is there a nice cluster structure in the zip dataset?

Experiment:

- Find 10 clusters using K-means.
- Compute the percentage p_{ij} of samples labelled i having "true" label j.

$$p_{ij} = \begin{pmatrix}
0.00 & 0.00 & 2.45 & 0.38 & 0.94 & 0.57 & 0.00 & \mathbf{83.96} & 0.19 & 11.51 \\
14.78 & 0.00 & 0.77 & 0.26 & 0.77 & 14.40 & \mathbf{68.64} & 0.00 & 0.39 & 0.00 \\
1.08 & 0.46 & 7.57 & 11.13 & 0.77 & 10.66 & 0.31 & 0.62 & \mathbf{66.46} & 0.93 \\
\mathbf{90.37} & 0.00 & 2.28 & 0.18 & 0.18 & 1.23 & 5.08 & 0.00 & 0.70 & 0.00 \\
\mathbf{88.96} & 0.00 & 0.51 & 0.34 & 0.00 & 2.72 & 7.13 & 0.00 & 0.34 & 0.00 \\
1.08 & 0.00 & \mathbf{86.15} & 1.85 & 2.15 & 1.38 & 5.54 & 0.31 & 1.54 & 0.00 \\
1.41 & 0.00 & 5.66 & 1.13 & \mathbf{62.23} & 5.66 & 1.41 & 3.25 & 1.41 & 17.82 \\
1.63 & 0.00 & 3.69 & \mathbf{59.22} & 0.00 & 32.00 & 0.00 & 0.00 & 3.25 & 0.22 \\
0.00 & \mathbf{93.03} & 0.37 & 0.09 & 3.90 & 0.00 & 0.84 & 0.28 & 1.02 & 0.46 \\
0.00 & 0.12 & 1.10 & 1.46 & 16.93 & 0.61 & 0.24 & 20.46 & 4.99 & \mathbf{54.08}
\end{pmatrix}$$

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Overview of spectral clustering:

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Overview of spectral clustering:

1. Construct a *similarity matrix* measuring the similarity of pairs of objects.

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Overview of spectral clustering:

1. Construct a *similarity matrix* measuring the similarity of pairs of objects.
2. Use the similarity matrix to construct a (weighted or unweighted) graph.

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Overview of spectral clustering:

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 $\mathbb{R}^p$).

Spectral clustering: overview

We saw how K-means can be used to cluster points in $\mathbb{R}^p$.

Spectral clustering:

- Very popular clustering method.
- Often outperforms other methods such as K-means.
- Can be used for various "types" of data (not only points in $\mathbb{R}^p$).
- Easy to implement. Only uses basic linear algebra.

Overview of spectral clustering:

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 using the eigenvectors of the graph Laplacian using the K-means algorithm.

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E \subset V \times V$.

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E \subset V \times V$.
- Each edge carries a *weight* $w_{ij} \geq 0$.

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ 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).

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ 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

$$d_i := \sum_{j=1}^{n} w_{ij}.$$

Notation

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 $D := \operatorname{diag}(d_1, \dots, d_n)$.

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ 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 $D := \mathrm{diag}(d_1, \ldots, d_n)$.
- We denote the complement of $A \subset V$ by $\overline{A}$.

Notation

We will use the following notation/conventions:

- $G = (V, E)$ a graph with vertex set $V = \{v_1, \ldots, v_n\}$ 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 $D := \mathrm{diag}(d_1, \ldots, d_n)$.
- We denote the complement of $A \subset V$ by $\overline{A}$.
- If $A \subset V$, then we let $\mathbb{1}_A = (f_1, \ldots, f_n)^T \in \mathbb{R}^n$, where $f_i = 1$ if $v_i \in A$ and 0 otherwise.

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \ldots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \ldots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

- We denote by $s_{ij} := s(x_i, x_j)$ the *measure of similarity* between x_i and x_j.

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \ldots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

- We denote by $s_{ij} := s(x_i, x_j)$ the *measure of similarity* between x_i and x_j.
- Equivalently, we may assume we have a measure of *distance* between data points (e.g. $(\mathcal{X}, d)$ is a metric space).

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \ldots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

- We denote by $s_{ij} := s(x_i, x_j)$ the *measure of similarity* between x_i and x_j.
- Equivalently, we may assume we have a measure of *distance* between data points (e.g. $(\mathcal{X}, d)$ is a metric space).
- Let $d_{ij} := d(x_i, x_j)$, the distance between x_i and x_j.

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \dots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

- We denote by $s_{ij} := s(x_i, x_j)$ the *measure of similarity* between x_i and x_j.
- Equivalently, we may assume we have a measure of *distance* between data points (e.g. $(\mathcal{X}, d)$ is a metric space).
- Let $d_{ij} := d(x_i, x_j)$, the distance between x_i and x_j.
- From d_{ij} (or s_{ij}), we naturally build a *similarity graph*.

Similarity graphs

- We assume we are given a measure of similarity s between data points $x_1, \ldots, x_n \in \mathcal{X}$:

$$s : \mathcal{X} \times \mathcal{X} \to [0, \infty).$$

- We denote by $s_{ij} := s(x_i, x_j)$ the *measure of similarity* between x_i and x_j.
- Equivalently, we may assume we have a measure of *distance* between data points (e.g. $(\mathcal{X}, d)$ is a metric space).
- Let $d_{ij} := d(x_i, x_j)$, the distance between x_i and x_j.
- From d_{ij} (or s_{ij}), we naturally build a *similarity graph*.
- We will discuss 3 popular ways of building a similarity graph.

Similarity graphs (cont.)

Vertex set = $\{v_1, \ldots, v_n\}$ where n is the number of data points.

Similarity graphs (cont.)

Vertex set = $\{v_1, \ldots, v_n\}$ where n is the number of data points.

1. **The ϵ-neighborhood graph:** Connect all points whose pairwise distances are smaller than some $\epsilon > 0$. We usually don't weight the edges. The graph is thus a simple graph (unweighted, undirected graph containing no loops or multiple edges).

Similarity graphs (cont.)

Vertex set = $\{v_1, \ldots, v_n\}$ where n is the number of data points.

1. **The ϵ-neighborhood graph:** Connect all points whose pairwise distances are smaller than some $\epsilon > 0$. We usually don't weight the edges. The graph is thus a simple graph (unweighted, undirected graph containing no loops or multiple edges).
2. **The k-nearest neighbor graph:** The goal is to connect v_i to v_j if x_j is among the k nearest neighbords of x_i. However, this leads to a directed graph. We therefore define:

Similarity graphs (cont.)

Vertex set $= \{v_1, \ldots, v_n\}$ where n is the number of data points.

1. **The ϵ-neighborhood graph:** Connect all points whose pairwise distances are smaller than some $\epsilon > 0$. We usually don't weight the edges. The graph is thus a simple graph (unweighted, undirected graph containing no loops or multiple edges).
2. **The k-nearest neighbor graph:** The goal is to connect v_i to v_j if x_j is among the k nearest neighbords of x_i. However, this leads to a directed graph. We therefore define:
   - the k-nearest neighbor graph: v_i is adjacent to v_j iff x_j is among the k nearest neighbords of x_i **OR** x_i is among the k nearest neighbords of x_j.

Similarity graphs (cont.)

Vertex set = $\{v_1, \ldots, v_n\}$ where n is the number of data points.

1. **The ϵ-neighborhood graph:** Connect all points whose pairwise distances are smaller than some $\epsilon > 0$. We usually don't weight the edges. The graph is thus a simple graph (unweighted, undirected graph containing no loops or multiple edges).
2. **The k-nearest neighbor graph:** The goal is to connect v_i to v_j if x_j is among the k nearest neighbords of x_i. However, this leads to a directed graph. We therefore define:
   - the k-nearest neighbor graph: v_i is adjacent to v_j iff x_j is among the k nearest neighbords of x_i **OR** x_i is among the k nearest neighbords of x_j.
   - the mutual k-nearest neighbor graph: v_i is adjacent to v_j iff x_j is among the k nearest neighbords of x_i **AND** x_i is among the k nearest neighbors of x_j.

Similarity graphs (cont.)

Vertex set = $\{v_1, \ldots, v_n\}$ where n is the number of data points.

1. **The ϵ-neighborhood graph:** Connect all points whose pairwise distances are smaller than some $\epsilon > 0$. We usually don't weight the edges. The graph is thus a simple graph (unweighted, undirected graph containing no loops or multiple edges).
2. **The k-nearest neighbor graph:** The goal is to connect v_i to v_j if x_j is among the k nearest neighbords of x_i. However, this leads to a directed graph. We therefore define:
   - the k-nearest neighbor graph: v_i is adjacent to v_j iff x_j is among the k nearest neighbords of x_i **OR** x_i is among the k nearest neighbords of x_j.
   - the mutual k-nearest neighbor graph: v_i is adjacent to v_j iff x_j is among the k nearest neighbords of x_i **AND** x_i is among the k nearest neighbors of x_j.

   We weight the edges by the similarity of their endpoints.

Similarity graphs (cont.)

3. **The fully connected graph:** Connect all points with edge weights s_{ij}.

Similarity graphs (cont.)

3. **The fully connected graph:** Connect all points with edge weights s_{ij}. For example, one could use the *Gaussian similarity function* to represent a local neighborhood relationships:

$$s_{ij} = s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2)) \qquad (\sigma^2 > 0).$$

Note: σ^2 controls the width of the neighborhoods.

3. **The fully connected graph:** Connect all points with edge weights s_{ij}. For example, one could use the *Gaussian similarity function* to represent a local neighborhood relationships:

$$s_{ij} = s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2)) \qquad (\sigma^2 > 0).$$

Note: σ^2 controls the width of the neighborhoods.

All graphs mentioned above are regularly used in spectral clustering.

Graph Laplacians

There are three commonly used definitions of the graph Laplacian:

1. **The unnormalized Laplacian** is

$$L := D - W.$$

Graph Laplacians

There are three commonly used definitions of the graph Laplacian:

1. **The unnormalized Laplacian** is

$$L := D - W.$$

2. **The normalized symmetric Laplacian** is

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}.$$

Graph Laplacians

There are three commonly used definitions of the graph Laplacian:

1. **The unnormalized Laplacian** is

$$L := D - W.$$

2. **The normalized symmetric Laplacian** is

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}.$$

3. **The normalized "random walk" Laplacian** is

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

Graph Laplacians

There are three commonly used definitions of the graph Laplacian:

1. **The unnormalized Laplacian** is

$$L := D - W.$$

2. **The normalized symmetric Laplacian** is

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}.$$

3. **The normalized "random walk" Laplacian** is

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

We will see in the next lecture how these Laplacians can be used to cluster graphs.