Introduction to Algorithmic Data Analysis

Esther Galbrun
Autumn 2022
Part II

Clustering Basics
Problem
A simple example

Consider a bunch of dry beans
A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups.
A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups.
A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups.
Let us take a closer look at the beans...

Measurements, in number of pixels, can be extracted automatically from digital images of the beans

**data points:** Dry beans

**attributes:** physical properties
  - Major Axis Length
  - Minor Axis Length
  - Eccentricity
  - Roundness
  - Extent
  - Shape Factor 2

See [https://doi.org/10.1016/j.compag.2020.105507](https://doi.org/10.1016/j.compag.2020.105507) for details
A simple example

Given measurements of physical properties of the beans...
A simple example

...we would like to divide the beans into a few coherent groups
A simple example

...we would like to divide the beans into a few coherent groups
A simple example

This is a clustering task
The problem, informally

To put it simply, the goal of **clustering** is to divide a collection of objects, or data points, into a small number of groups, such that the objects *within a group* are *similar* to each other whereas objects *from different groups* are *dissimilar*.
Some obvious questions arise

How many groups?

How to measure similarity?
The problem, informally

Some obvious questions arise

**How many groups?**

typically chosen by the user, i.e. input parameter, but determining the most appropriate number of groups can also be seen as part of the problem

**How to measure similarity?**
calculating distances is a crucial ingredient in many clustering methods
The problem, informally

Some obvious questions arise

**How many groups?**

typically chosen by the user, i.e. input parameter, but determining the most appropriate number of groups can also be seen as part of the problem

**How to measure similarity?**
calculating distances is a crucial ingredient in many clustering methods

Next, we will look in turn at different clustering methods
Then, we will look at ways to compare and evaluate clusterings
A small example data set

For illustrative purposes, we will take as an example a data set that consists of a handful of the beans. We focus on a pair of measurement variables at a time, so that we can easily visualize our data.
A small example data set

For illustrative purposes, we will take as an example a data set that consists of a handful of the beans. We normalize the data, rescaling each variable so that its values fall within the unit interval.
Some notations

The data set, denoted as $\mathcal{D}$, contains $n$ data points and $m$ attributes, i.e. it is a $n \times m$ matrix.

A data point is a $m$-dimensional vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_m \rangle$.

We denote $\mathbf{x}^{(j)}$ the $j^{th}$ data point of $\mathcal{D}$, i.e. the $j^{th}$ row.

Data points are sometimes called instances or examples.

We consider subsets of the data set.

For instance, we denote the subset consisting of the first, third and fourth data points as $S = \{x_1, x_3, x_4\}$.

For simplicity, when there is no ambiguity about the underlying data set, we might specify a subset by listing the indices of the data points it contains.

The size of a subset $S$, denoted $|S|$, is the number of data points it contains.
A clustering is a collection of subsets of data points, called clusters. We write $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ to denote a clustering consisting of $k$ clusters. The size of a clustering $\mathcal{C}$, denoted $|\mathcal{C}|$, is the number of clusters it contains.

Typically, the clusters form a partition of the data set. That is, seeing $\mathcal{D}$ as a set of data points, i.e. ignoring the order, the clusters are such that:

(i) they cover the entire data set, i.e. $\bigcup_{C \in \mathcal{C}} C = \mathcal{D}$, and
(ii) they are pairwise disjoint, i.e. $C_i \cap C_j = \emptyset$ for any pair of distinct clusters $C_i$ and $C_j$ from $\mathcal{C}$.
A clustering is a collection of subsets of data points, called clusters.

We write $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ to denote a clustering consisting of $k$ clusters.

The size of a clustering $\mathcal{C}$, denoted $|\mathcal{C}|$, is the number of clusters it contains.

Typically, the clusters form a partition of the data set.

A clustering, i.e. an assignment of the data points to $k$ clusters, can be represented as a $n$-dimensional vector:

$$y = \langle y_1, y_2, \ldots, y_n \rangle \in [1..k]^n,$$

where $y_j$ is the index of the cluster to which data point $x^{(j)}$ is assigned. That is, $y_j = s$ if and only if $x^{(j)} \in C_s$. 

Methods
A representative (a.k.a. center) is associated to each cluster

Representatives can be

- synthetic vectors from the domain, or
- existing points from the data set

Given a distance function $d$, the goal is to find a chosen number $k$ of representatives so that all points are as close as possible from a representative

Find $R = \{r^{(1)}, r^{(2)}, \ldots r^{(k)}\}$ to minimize $\sum_{x \in D} \min_{r \in R} d(x, r)$

Each data point is assigned to the cluster associated to its closest representative
Representative-based algorithms

The representatives and the assignment of data points are unknown a priori, but depend on each other in a circular way

- if the representatives are fixed, it is easy to assign each data point to the closest one
- if the assignment of data points is fixed, it is easy to determine a representative for each group

Such problems are typically solved using an iterative algorithm that alternates between refining the representatives or the assignment, keeping the other one fixed
When the chosen distance function is the Euclidean distance ($\ell_2$ norm), i.e.

$$d(x, x') = \sqrt{\sum_{i=1}^{i=m} (x_i - x_i')^2},$$

it can be shown that the optimal representative is the mean of the data points assigned to it.
When the chosen distance function is the Euclidean distance ($\ell_2$ norm), i.e.

$$d(x, x') = \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2},$$

it can be shown that the optimal representative is the mean of the data points assigned to it.

Considering the data points assigned to cluster $C_j$, the associated representative is $r^{(j)} = \langle r_1^{(j)}, r_2^{(j)}, \ldots, r_m^{(j)} \rangle$ where

$$r_i^{(j)} = \frac{\sum_{x \in C_j} x_i}{|C_j|}.$$
$k$-means algorithm

centers $\leftarrow k$ points sampled from the domain

repeat
  Assign points to closest center
  centers $\leftarrow$ mean of assigned points
until convergence

for $k = 4$
Step# 1
The $k$-means algorithm

centers ← $k$ points sampled from the domain

repeat

Assign points to closest center

centers ← mean of assigned points

until convergence

for $k = 4$

Step# 2
**k-means algorithm**

```plaintext
centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence
```

![Diagram showing a scatter plot with points labeled for k = 4, Step# 3.](image)
**k-means algorithm**

centers ← \( k \) points sampled from the domain

repeat

Assign points to closest center

centers ← mean of assigned points

until convergence

---

**Diagram:**

For \( k = 4 \) in Step #4
The $k$-means algorithm is defined as follows:

1. Initialize $k$ centers randomly sampled from the domain.
2. Repeat the following steps until convergence:
   - Assign each point to the closest center.
   - Update the centers to the mean of the points assigned to each center.

For $k = 4$, the algorithm is illustrated in the graph with points colored by their assigned cluster and markers indicating the updated centers.
`k`-means algorithm

\[ \text{centers} \leftarrow k \text{ points sampled from the domain} \]

repeat

Assign points to closest center
\[ \text{centers} \leftarrow \text{mean of assigned points} \]

until convergence

for \( k = 4 \)

Step# 6
\textit{k}-means algorithm

\begin{verbatim}
centers $\leftarrow$ \textit{k} points sampled from the domain

\textbf{repeat}

\hspace{1em} Assign points to closest center

\hspace{1em} centers $\leftarrow$ mean of assigned points

\textbf{until} convergence
\end{verbatim}
**k-means algorithm**

\[ \text{centers} \leftarrow k \text{ points sampled from the domain} \]

**repeat**

- Assign points to closest center
- \[ \text{centers} \leftarrow \text{mean of assigned points} \]

**until** convergence

**Graph:** For \( k = 4 \)
- Step #8
The $k$-means algorithm is defined as follows:

1. **Initialization**: Choose $k$ points from the domain as initial centers.
2. **Assignment**: Assign each point to the closest center.
3. **Update**: Update the centers to the mean of the points assigned to each center.
4. **Convergence**: Repeat steps 2 and 3 until convergence.

For $k = 4$:

- **Step #9**: The algorithm has converged to the current set of centers.
The $k$-means algorithm

- **Initialization:** $k$ points sampled from the domain
- **Main Loop:**
  - **Assignment:** Assign points to closest center
  - **Update Centers:** $\text{centers} \leftarrow \text{mean of assigned points}$
- **Termination:** Until convergence

For $k = 4$

Step # 10
$k$-means algorithm

1. **centers** ← $k$ points sampled from the domain

2. **repeat**
   - Assign points to closest center
   - **centers** ← mean of assigned points

3. **until** convergence

---

**Graph:**

- **Step #11:** for $k = 4$

---

UEF//School of Computing

JADE: Clustering Basics

13/53
**$k$-means algorithm**

centers $\leftarrow k$ points sampled from the domain

repeat

   Assign points to closest center

   centers $\leftarrow$ mean of assigned points

until convergence

for $k = 4$
Step# 12
$k$-means algorithm

centers ← $k$ points sampled from the domain

repeat
    Assign points to closest center
    centers ← mean of assigned points

until convergence
$k$-means algorithm

centers ← $k$ points sampled from the domain
repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence

for $k = 4$
Step# 14
\textit{k}-means algorithm

centers $\leftarrow k$ points sampled from the domain

\textbf{repeat}

Assign points to closest center
centers $\leftarrow$ mean of assigned points

\textbf{until} convergence

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{chart.png}
\caption{For $k = 4$ Step# 15}
\end{figure}
**k-means algorithm**

\[ \text{centers} \leftarrow k \text{ points sampled from the domain} \]

repeat

Assign points to closest center

center \( \leftarrow \text{mean of assigned points} \)

until convergence

---

![Diagram](attachment:image.png)

for \( k = 4 \)

Step# 16
The text in the image is a description of the \textit{k}-means algorithm. Here is the code snippet converted to natural text:

```plaintext
\texttt{k}\textendash\textit{means} algorithm

centers \leftarrow k \text{ points sampled from the domain}

\texttt{repeat}

\texttt{Assign points to closest center}

\texttt{centers} \leftarrow \text{mean of assigned points}

\texttt{until convergence}
```

The diagram visualizes the \textit{k}-means algorithm for \( k = 4 \). The plot shows points in two dimensions assigned to different clusters represented by different colors.
When the chosen distance function is the Manhattan distance (£_1 norm), i.e.

\[ d(x, x') = \sum_{i=1}^{i=m} |x_i - x'_i| , \]

it can be shown that the optimal representative is the median of the data points assigned to it.
When the chosen distance function is the Manhattan distance ($\ell_1$ norm), i.e.

$$d(x, x') = \sum_{i=1}^{i=m} |x_i - x'_i| ,$$

it can be shown that the optimal representative is the median of the data points assigned to it.

Given a set of values $A$, we denote $a^{<p>}$ the $p^{th}$ smallest value in $A$, that is, $a^{<1>} \leq a^{<2>} \leq \ldots \leq a^{<|A|>}$.

Then the median of $A$ is

$$\text{median}(A) = \begin{cases} a^{<(|A|+1)/2>} & \text{if } |A| \text{ is odd} \\ (a^{<|A|/2>} + a^{<|A|/2+1>})/2 & \text{if } |A| \text{ is even} \end{cases}$$
When the chosen distance function is the Manhattan distance ($\ell_1$ norm), i.e.

$$d(x, x') = \sum_{i=1}^{i=m} |x_i - x'_i|,$$

it can be shown that the optimal representative is the median of the data points assigned to it.

Considering the data points assigned to cluster $C_j$, the associated representative is $r^{(j)} = \langle r_1^{(j)}, r_2^{(j)}, \ldots, r_m^{(j)} \rangle$ where

$$r_i^{(j)} = \text{median}(\{x_i, x \in C_j\})$$
$k$-medians algorithm

centers ← $k$ points sampled from the domain

repeat
    Assign points to closest center
    centers ← median of assigned points

until convergence

for $k = 4$
Step# 1
$k$-medians algorithm

centers ← $k$ points sampled from the domain

repeat
  Assign points to closest center
  centers ← median of assigned points
until convergence

for $k = 4$
Step# 2
$k$-medians algorithm

\begin{align*}
\text{centers} & \leftarrow k \text{ points sampled from the domain} \\
\text{repeat} & \\
\text{Assign points to closest center} & \\
\text{centers} & \leftarrow \text{median of assigned points} \\
\text{until} & \text{convergence}
\end{align*}

for $k = 4$  
Step# 3
$k$-medians algorithm

centers ← $k$ points sampled from the domain
repeat
   Assign points to closest center
   centers ← median of assigned points
until convergence

for $k = 4$
Step# 4
$k$-medians algorithm

centers $\leftarrow$ $k$ points sampled from the domain

repeat
    Assign points to closest center

    centers $\leftarrow$ median of assigned points

until convergence

for $k = 4$
Step# 5
The $k$-medians algorithm:

1. Initialize $k$ points sampled from the domain.
2. **Repeat**
   - Assign points to the closest center.
   - Update centers to the median of the assigned points.
3. **Until** convergence.

**Graph:**
- For $k = 4$.
- Step #6.
The $k$-medians algorithm can be described as follows:

1. Initialize $k$ centers as $k$ points sampled from the domain.
2. Repeat the following steps until convergence:
   - Assign each point to the closest center.
   - Update the centers to the median of the points assigned to each center.

The algorithm cycles between the assignment of points to centers and the update of the centers until the centers no longer change significantly or a predetermined number of iterations is reached.
$k$-medians algorithm

centers ← $k$ points sampled from the domain

repeat
    Assign points to closest center
    centers ← median of assigned points

until convergence

for $k = 4$

Step# 8
$k$-medians algorithm

centers ← $k$ points sampled from the domain
repeat
  Assign points to closest center
  centers ← median of assigned points
until convergence

for $k = 4$
### Representative-based algorithms

<table>
<thead>
<tr>
<th>Distance</th>
<th>Euclidean distance ( \ell_2 ) norm</th>
<th>Manhattan distance ( \ell_1 ) norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d(x, x') )</td>
<td>( \sqrt{\sum_{i=1}^{m} (x_i - x'_i)^2} )</td>
<td>( \sum_{i=1}^{m}</td>
</tr>
<tr>
<td>Representative</td>
<td>mean</td>
<td>median</td>
</tr>
<tr>
<td>( r_i^{(j)} )</td>
<td>( \sum_{x \in C_j} x_i /</td>
<td>C_j</td>
</tr>
<tr>
<td>Algorithm</td>
<td>( k )-means</td>
<td>( k )-medians</td>
</tr>
</tbody>
</table>
A representative might not be assigned any data point, because it is not closest to any one

for $k = 4$

Step# 1
Initialization

A representative might not be assigned any data point, because it is not closest to any one
A representative might not be assigned any data point, because it is not closest to any one.

In such cases, the representative might be dropped altogether, resulting in a smaller number of clusters, or re-initialised for $k = 4$.
A representative might not be assigned any data point, because it is not closest to any one.

In such cases, the representative might be dropped altogether, resulting in a smaller number of clusters, or re-initialised.
A representative might not be assigned any data point, because it is not closest to any one
The initialization of the representatives is a crucial step. Instead of initializing the representatives as random vectors from the domain of the variables, one might sample existing points from the data set.
$k$-means algorithm

centers $\leftarrow k$ points sampled from the data
repeat
  Assign points to closest center
  centers $\leftarrow$ mean of assigned points
until convergence

for $k = 4$
Step# 1
$k$-means algorithm

centers $\leftarrow k$ points sampled from the data

repeat
  Assign points to closest center
  centers $\leftarrow$ mean of assigned points
until convergence

for $k = 4$
Step# 2
**k-means algorithm**

centers ← \( k \) points sampled from the data

repeat

Assign points to closest center

centers ← mean of assigned points

until convergence

---

**UEF//School of Computing**

**JADe:Clustering Basics**

19/53
$k$-means algorithm

centers ← $k$ points sampled from the data
repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence

for $k = 4$

Step# 4
$k$-means algorithm

\[
\text{centers} \leftarrow k \text{ points sampled from the data}
\]

\[
\text{repeat}
\]

Assign points to closest center

\[
\text{centers} \leftarrow \text{mean of assigned points}
\]

\[
\text{until convergence}
\]
**$k$-means algorithm**

centers ← $k$ points sampled from the data

repeat

Assign points to closest center

centers ← mean of assigned points

until convergence

---

for $k = 4$

Step# 6
$k$-means algorithm

centers ← $k$ points sampled from the data

repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence

for $k = 4$
The initialization of the representatives is a crucial step. It is desirable that the initial representatives are spread through different regions of the data. When sampling, penalize data points that are close to previously selected representatives.
The initialization of the representatives is a crucial step. It is desirable that the initial representatives are spread through different regions of the data. When sampling, penalize data points that are close to previously selected representatives.

→ $k$-means++ algorithm
$r^{(1)} \leftarrow$ sample from data points uniformly at random
for $j = 2..k$ do
    $r^{(j)} \leftarrow$ sample from remaining data points with probability $P(x) \propto \min_{i=1..j} d(x, r_i)$
...

for $k = 4$
Step# 1
**k-means++ algorithm**

\[ r^{(1)} \leftarrow \text{sample from data points uniformly at random} \]

for \( j = 2 \ldots k \) do

\[ r^{(j)} \leftarrow \text{sample from remaining data points with probability } P(x) \propto \min_{i=1..j} d(x, r_i) \]

...

---

**for \( k = 4 \)**

**Step# 2**
**k-means++ algorithm**

\[ r^{(1)} \leftarrow \text{sample from data points uniformly at random} \]

for \( j = 2 \ldots k \) do

\[ r^{(j)} \leftarrow \text{sample from remaining data points with probability } P(x) \propto \min_{i=1..j} d(x, r_i) \]

\[ \ldots \]

---

For \( k = 4 \)

Step# 3
$k$-means++ algorithm

$r^{(1)} \leftarrow$ sample from data points uniformly at random
for $j = 2..k$ do
  $r^{(j)} \leftarrow$ sample from remaining data points with probability $P(x) \propto \min_{i=1..j} d(x, r_i)$
...

for $k = 4$
Step# 4
... repeat

Assign points to closest center
centers ← mean of assigned points
until convergence

for \( k = 4 \)
Step# 5
\textit{k}-means++ algorithm

\begin{verbatim}
... repeat
    Assign points to closest center
    centers ← mean of assigned points
until convergence
\end{verbatim}

\textit{for } k = 4 \\
\textit{Step# 6}
\( k \)-means++ algorithm

... 
repeat
\begin{align*}
&\text{Assign points to closest center} \\
&\text{centers} \leftarrow \text{mean of assigned points}
\end{align*}
until convergence

for \( k = 4 \) 
Step# 7
\textit{k-means++ algorithm}

\begin{align*}
&\text{repeat} \\
&\quad \text{Assign points to closest center} \\
&\quad \text{centers} \leftarrow \text{mean of assigned points} \\
&\text{until convergence}
\end{align*}

\[\text{for } k = 4, \text{ Step# 8}\]
$k$-means++ algorithm

... repeat

Assign points to closest center

centers ← mean of assigned points

until convergence

for $k = 4$

Step# 9
\textit{k-means++ algorithm}

... repeat
  Assign points to closest center
  \texttt{centers} ← mean of assigned points
until convergence

for \( k = 4 \)
Step# 10
$k$-means++ algorithm

\[ ... \]

\[ \text{repeat} \]
\[ \text{Assign points to closest center} \]
\[ \text{centers} \leftarrow \text{mean of assigned points} \]

\[ \text{until convergence} \]

for $k = 4$
Step# 11
\textit{k-means++} algorithm

... 
repeat 
\quad Assign points to closest center 
\quad centers ← mean of assigned points 
until convergence

for $k = 4$
The initialization of the representatives is a crucial step.

More careful initialization, as with $k$-means++ might be more expensive, but the algorithm then typically and more reliably converges in fewer iterations, and to better solutions.
Underlying assumption: the data was generated from a mixture of $k$ probability distributions

A probabilistic model (a.k.a. *mixture component*) is associated to each cluster

Each data point is generated by the mixture model as follows

(i) a mixture component is selected
(ii) the data point is generated from this component
Probabilistic model-based algorithms

Each data point is generated by the mixture model as follows
(i) a mixture component is selected
(ii) the data point is generated from this component

Each mixture component $M^{(i)}$ has a prior probability $\alpha_j$, a set of parameters $\theta_j$ and a probability density function $f_{\theta_j}$

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the probability of data point $x$ is

$$p(x \mid \mathcal{M}) = \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(x)$$

and the probability of the data is

$$p(D \mid \mathcal{M}) = \prod_{x \in D} p(x \mid \mathcal{M})$$
Probabilistic model-based algorithms

Each mixture component $M^{(i)}$ has a prior probability $\alpha_j$, a set of parameters $\theta_j$ and a probability density function $f_{\theta_j}$.

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the probability of the data is

$$p(\mathcal{D} | \mathcal{M}) = \prod_{x \in \mathcal{D}} p(x | \mathcal{M})$$

It is generally more convenient to work with the log likelihood

$$\mathcal{L}(\mathcal{D} | \mathcal{M}) = \log \left( p(\mathcal{D} | \mathcal{M}) \right) = \sum_{x \in \mathcal{D}} \log \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(x)$$
Probabilistic model-based algorithms

Each mixture component $M^{(i)}$ has a prior probability $\alpha_j$, a set of parameters $\theta_j$ and a probability density function $f_{\theta_j}$.

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the log likelihood of the data is

$$
\mathcal{L}(\mathcal{D} \mid \mathcal{M}) = \sum_{x \in \mathcal{D}} \log \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(x)
$$

The goal is to find the model parameters that maximize the fit of the model to the data, as measured by the log likelihood

$$
\arg \max_{\mathcal{M}} \mathcal{L}(\mathcal{D} \mid \mathcal{M})
$$
The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way.

- if the parameters of the model are fixed, it is easy to compute the probabilities of each data point to be generated by each mixture component.
- if the probabilities of each data point to be generated by each mixture component are fixed, it is relatively easy to determine the parameters of the model.
Probabilistic model-based algorithms

The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way.

Such problems are typically solved using an iterative algorithm that alternates between two steps:

**Expectation** estimate the posterior probability that point $\mathbf{x}$ was generated by each mixture component $M^{(i)}$.

**Maximization** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment.
Probabilistic model-based algorithms

The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way.

Such problems are typically solved using an iterative algorithm that alternates between two steps:

- **Expectation** estimate the posterior probability that point $x$ was generated by each mixture component $M^{(i)}$.

- **Maximization** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment.

→ **Expectation-Maximization** (or EM) algorithm
In particular, Gaussian distributions might be used as mixture components.

A Gaussian distribution (a.k.a. normal distribution) has two parameters, the mean $\mu$ and the variance $\sigma^2$, i.e. $\theta = (\mu, \sigma^2)$, and is typically denoted $\mathcal{N}(\mu, \sigma^2)$. 
In particular, Gaussian distributions might be used as mixture components.

A Gaussian distribution (a.k.a. normal distribution) has two parameters, the mean $\mu$ and the variance $\sigma^2$, i.e. $\theta = (\mu, \sigma^2)$, and is typically denoted $\mathcal{N}(\mu, \sigma^2)$.

The probability of data point $x$ under mixture component $M$ with $\mu = \langle \mu_1, \ldots, \mu_m \rangle$ and $\sigma^2 = \langle \sigma_1^2, \ldots, \sigma_m^2 \rangle$ is

$$f_\theta(x) = \prod_{i=1}^{i=m} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}}$$
Let $\Theta$ denote all parameters of the model, including the mean and variance of each mixture component, as well as their prior probabilities.

Assuming that the parameter values in $\Theta$ are fixed, the posterior probability that data point $x$ was generated by mixture component $M^{(j)}$ is

$$p(M^{(j)} \mid x, \Theta) = \frac{\alpha_j \cdot f_{\theta_j}(x)}{\sum_{s=1}^{S=k} \alpha_s \cdot f_{\theta_s}(x)}$$

This can be interpreted as a *soft assignment* of the data point to the mixture component, and used to weigh the contribution of the data point to the mixture component.
Given a soft assignment of data points to mixture component $M^{(j)}$ as a vector of probabilities over the data points $w^{(j)} \in [0, 1]^n$, such that $w_q^{(j)} = p(M^{(j)} \mid x^{(q)}, \Theta)$, the parameters of the mixture component are estimated as

$$
\mu_i^{(j)} = \frac{\sum_{q=1}^{n} w_q^{(j)} \cdot x_i^{(q)}}{\sum_{q=1}^{n} w_q} 
\text{ and } \sigma_i^{(j)} = \sqrt{\frac{\sum_{q=1}^{n} w_q^{(j)} \cdot (x_i^{(q)} - \mu_i)^2}{\sum_{q=1}^{n} w_q}}
$$

And the prior probability of mixture component $M^{(j)}$ is estimated as

$$
\alpha_j = \frac{\sum_{q=1}^{n} w_q^{(j)}}{n}
$$
EM clustering with Gaussian mixture model

**Initialize mixture components:** sample $k$ normal distribution parameter pairs and set prior probabilities

- $\theta_j \leftarrow$ initialize $(\mu, \sigma)$, $j \in [1..k]$
- $\alpha_j \leftarrow 1/k$, $j \in [1..k]$

repeat
- compute points assignment
- compute parameters $\Theta$
until convergence
EM clustering with Gaussian mixture model

**E(xpectation) step:** estimate the posterior probability that point $x$ was generated by each mixture component $M(j)$

- $E(x) = \sum_{j=1}^{k} \alpha_j \phi_j(x)$

**Parameters Initialization:**
- $\theta_j \leftarrow$ initialize $(\mu, \sigma), \; j \in [1..k]$
- $\alpha_j \leftarrow 1/k, \; j \in [1..k]$

**Iteration:**
- Repeat for $k = 3$
- Compute points assignment
- Compute parameters $\Theta$
- Until convergence
**EM clustering with Gaussian mixture model**

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment.

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\quad \text{compute points assignment} \\
\quad \text{compute parameters } \Theta \\
\text{until convergence}
\]
EM clustering with Gaussian mixture model

E(xpectation) step: estimate the posterior probability that point $\mathbf{x}$ was generated by each mixture component $M^{(j)}$

\[ \theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \]
\[ \alpha_j \leftarrow 1/k, \quad j \in [1..k] \]
repeat
compute points assignment
compute parameters $\Theta$
until convergence
**EM clustering with Gaussian mixture model**

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment.

- $\theta_j \leftarrow$ initialize $(\mu, \sigma), \; j \in [1..k]$
- $\alpha_j \leftarrow 1/k, \; j \in [1..k]$
- repeat
  - compute points assignment
  - compute parameters $\Theta$
- until convergence

For $k = 3$

Step# 5
EM clustering with Gaussian mixture model

E(xpectation) step: estimate the posterior probability that point $x$ was generated by each mixture component $M(j)$

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\quad \text{compute points assignment} \\
\quad \text{compute parameters } \Theta \\
\text{until convergence}
\]
**EM clustering with Gaussian mixture model**

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\text{compute points assignment} \\
\text{compute parameters } \Theta \\
\text{until convergence}
\]
**EM clustering with Gaussian mixture model**

**E(xpectation) step:** estimate the posterior probability that point $x$ was generated by each mixture component $M^{(j)}$.

\[ \theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \]

\[ \alpha_j \leftarrow 1/k, \quad j \in [1..k] \]

**repeat**

compute points assignment

compute parameters $\Theta$

**until** convergence

For $k = 3$

Step# 8
EM clustering with Gaussian mixture model

M(aximization) step: estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment

$\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k]$

$\alpha_j \leftarrow 1/k, \quad j \in [1..k]$

repeat

compute points assignment

compute parameters $\Theta$

until convergence

for $k = 3$

Step# 9
EM clustering with Gaussian mixture model

**E(xpectation) step:** estimate the posterior probability that point \( x \) was generated by each mixture component \( M^{(j)} \)

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\quad \text{compute points assignment} \\
\quad \text{compute parameters } \Theta \\
\text{until convergence}
\]
EM clustering with Gaussian mixture model

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment.

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\text{compute points assignment} \\
\text{compute parameters } \Theta \\
\text{until convergence}
\]
**EM clustering with Gaussian mixture model**

**E(xpectation) step:** estimate the posterior probability that point \( x \) was generated by each mixture component \( M^{(j)} \)

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\quad \text{compute points assignment} \\
\quad \text{compute parameters } \Theta \\
\text{until convergence}
\]
EM clustering with Gaussian mixture model

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment

\[\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k]\]
\[\alpha_j \leftarrow 1/k, \quad j \in [1..k]\]

repeat

compute points assignment

compute parameters $\Theta$

until convergence
EM clustering with Gaussian mixture model

**E(xpectation) step:** estimate the posterior probability that point $x$ was generated by each mixture component $M^{(j)}$

\[
\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \\
\alpha_j \leftarrow 1/k, \quad j \in [1..k] \\
\text{repeat} \\
\quad \text{compute points assignment} \\
\quad \text{compute parameters } \Theta \\
\text{until convergence}
\]
EM clustering with Gaussian mixture model

**M(aximization) step:** estimate model parameters $\Theta$ to maximize the log-likelihood fit under the current assignment

\[ \theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k] \]
\[ \alpha_j \leftarrow 1/k, \quad j \in [1..k] \]

repeat

compute points assignment

compute parameters $\Theta$

until convergence
EM clustering with Gaussian mixture model

Convergence: the parameters of the model stabilize

for $k = 3$

Step# 16

$$\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k]$$

$$\alpha_j \leftarrow 1/k, \quad j \in [1..k]$$

repeat

compute points assignment

compute parameters $\Theta$

until convergence
EM clustering with Gaussian mixture model

**Soft assignment:** compute the final posterior probabilities for each data point \(x\)

\[
p(M^{(i)} \mid x, \Theta), M^{(i)} \in \mathcal{M}
\]

\[\theta_j \leftarrow \text{initialize } (\mu, \sigma), \quad j \in [1..k]
\]
\[\alpha_j \leftarrow 1/k, \quad j \in [1..k]
\]
repeat
   compute points assignment
   compute parameters \(\Theta\)
until convergence

\[\text{for } k = 3\]
EM clustering with Gaussian mixture model

**Hard assignment:** assign each data point to the model under which it has the highest probability

$$\arg \max_{M^{(j)} \in \mathcal{M}} p(M^{(j)} | x, \Theta)$$

θᵢ ← initialize (µ, σ),   \( j \in [1..k] \)
αᵢ ← 1/k,   \( j \in [1..k] \)
repeat
  compute points assignment
  compute parameters Θ
until convergence

for \( k = 3 \)
The **bisecting** $k$-means algorithm, as the name suggests, works by *bisecting* clusters, i.e. splitting them into two, recursively by applying the $k$-means algorithm with $k = 2$.
Bisecting $k$-means

Start with a single cluster containing all data points
Bisecting $k$-means

Recursively split non-singleton clusters into two

Step# 1
Bisecting $k$-means

Recursively split non-singleton clusters into two

Step# 2
Bisecting \( k \)-means

Recursively split non-singleton clusters into two

Step# 3
Bisecting $k$-means

Recursively split non-singleton clusters into two

Step# 4
Bisecting \( k \)-means

Recursively split non-singleton clusters into two
Bisecting $k$-means

Recursively split non-singleton clusters into two
Bisecting $k$-means

Recursively split non-singleton clusters into two

Step# 7
Bisecting $k$-means

Until each data point is in its own cluster
Bisecting $k$-means

The clusters obtained through the successive iterations of the algorithm form a hierarchy, with higher-level clusters containing lower-level clusters.

→ Bisecting $k$-means is a top-down divisive hierarchical clustering algorithm.
Bisecting \( k \)-means

The clusters obtained through the successive iterations of the algorithm form a hierarchy, with higher-level clusters containing lower-level clusters.

The tree diagram depicting this hierarchical structure is called a dendrogram.
Bisecting $k$-means

Rather than further splitting all current non-singleton clusters, we can split one cluster at a time.

In particular, the *least cohesive* current cluster can be selected to be split next.

At each step, the number of clusters increases by one. The process can be repeated until (i) the desired number of clusters is obtained, or (ii) a desired cohesiveness threshold is reached for all clusters.
Bisecting $k$-means

The cohesiveness of a cluster can be evaluated using an aggregate of the distances between pairs of points in the cluster, such as the maximum of pairwise distances.

Let $C_m$ denote the current collection of non-singleton clusters. The next cluster to split can be selected as

$$\arg \max_{C \in C_m} \max_{(x, x')} \in C^2 d(x, x')$$

This way, the cluster containing the pair of nodes furthest apart will be selected to be split in the next step.
Bisecting $k$-means

Start with a single cluster containing all data points

for $k = 7$
Bisecting $k$-means

Split the initial cluster into two

The horizontal position of a node indicates the cohesiveness of the corresponding cluster. The cohesiveness of the old and of the new clusters is indicated below the dendrogram, and the corresponding pairwise distances are depicted with arrows in the scatter plot. The number in a circle above a node indicates its position in the priority queue.

for $k = 7$
Step# 1
Bisecting $k$-means

Take the first cluster from the queue, split it into two, and add the new clusters to the queue.
Bisecting $k$-means

Take the first cluster from the queue, split it into two, and add the new clusters to the queue for $k = 7$

Step# 3
Bisecting $k$-means

Take the first cluster from the queue, split it into two, and add the new clusters to the queue

for $k = 7$

Step# 4
Bisecting \( k \)-means

Take the first cluster from the queue, split it into two, and add the new clusters to the queue for \( k = 7 \)

Step# 5
Bisecting $k$-means

Take the first cluster from the queue, split it into two, and add the new clusters to the queue for $k = 7$

Step# 6
Bisecting $k$-means

Until the desired number of clusters is obtained

for $k = 7$
Hierarchical agglomerative algorithms

Contrary to *divisive* clustering methods that proceed in a *top-down* manner, hierarchical *agglomerative* clustering methods proceed from the *bottom up*.

They start with each data point in its own cluster, and iteratively merge the pair of clusters that are the closest, until a single cluster containing all data points is obtained.
Hierarchical agglomerative algorithms

Contrary to *divisive* clustering methods that proceed in a *top-down* manner, hierarchical *agglomerative* clustering methods proceed from the *bottom up*

They start with each data point in its own cluster, and iteratively merge the pair of clusters that are the closest, until a single cluster containing all data points is obtained.

Which clusters are considered to be the closest and selected to be merged in the next step depends on the chosen inter-cluster distance, called the *linkage function*.

Different linkage functions correspond to algorithm variants of agglomerative clustering.
Inter-cluster distances

Considering two clusters \( C_u \) and \( C_v \), the inter-cluster distance between them, \( d(C_u, C_v) \), is often defined as a function of the pairwise point distances, that is, of the distances between all pairs of points \( x \) and \( x' \) respectively from \( C_u \) and \( C_v \).

Furthermore, the distance between cluster \( C_{uv} \), resulting from the merger of \( C_u \) and \( C_v \), and any other cluster \( C_s \), \( d(C_{uv}, C_s) \), can often accordingly be computed as a function of the distances between \( C_u \) and \( C_s \) and between \( C_v \) and \( C_s \).

Among the most common linkage functions are

- **single linkage**  minimum of pairwise point distances
- **complete linkage** maximum of pairwise point distances
- **average linkage**  average of pairwise point distances
Inter-cluster distances, linkage functions

<table>
<thead>
<tr>
<th>Linkage function</th>
<th>single</th>
<th>complete</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.k.a. minimum</td>
<td></td>
<td>a.k.a. maximum</td>
<td></td>
</tr>
</tbody>
</table>

**Inter-cluster distance** (as function of pairwise point distances)

\[
d(C_u, C_v) = \min_{(x, x') \in C_u \times C_v} d(x, x') \quad \max_{(x, x') \in C_u \times C_v} d(x, x') \quad \frac{\sum_{(x, x') \in C_u \times C_v} d(x, x')}{|C_u| \cdot |C_v|}
\]

**Distance merging** (for \(C_{uv} = C_u \cup C_v\) and any other cluster \(C_s\))

\[
d(C_{uv}, C_s) = \min_{C \in \{C_u, C_v\}} d(C, C_s) \quad \max_{C \in \{C_u, C_v\}} d(C, C_s) \quad \frac{\sum_{C \in \{C_u, C_v\}} |C| \cdot d(C, C_s)}{|C_u| + |C_v|}
\]
Agglomerative clustering with complete linkage

Start with each data point in its own cluster
Distances must be computed for all pairs of points
The lower triangle of the symmetric distance matrix is depicted, with darker shades of gray indicating larger values.
Select the two clusters that are closest and merge them

The clusters corresponding to the smallest value in the distance matrix are selected (blue and red). They are merged into a new cluster (purple), and the distance matrix is updated. Specifically, the distance vectors for the two clusters are aggregated according to the linkage function, to compute distances for the new cluster (as depicted below the distance matrix). The two vectors are removed and the new one is added to the matrix.
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

Step# 2

complete link.
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link.
Step# 3
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 4
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.  
Step# 5
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 6
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them. Iterate...

complete link.  
Step# 7
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 8
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them. Iterate...

complete link. Step# 9
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 10
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate…

Complete link.
Step# 11
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link.
Step# 12
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

complete link.  Step# 13
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them

Iterate...

complete link.

Step# 14
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 15

complete link.
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link. 
Step# 16
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 17
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 18
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.
Step# 19
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

![Graph showing step #20 with complete link]
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link.  Step# 21
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 22
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 23
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

complete link.
Step# 24
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them

Iterate...

complete link.  
Step# 25
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link.
Step# 26
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

Step# 27

complete link.
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them

Iterate...

complete link.  
Step# 28
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

Step# 29
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them. Iterate...

[Diagram showing clusters being merged with complete linkages and step number 30 highlighted]
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them. Iterate...

UEF/School of Computing  JADE: Clustering Basics
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.

Step# 32
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

complete link. Step# 33
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

Step# 35
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 36
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.
Iterate...

complete link.  
Step# 37
Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them.

Iterate...

complete link.  Step# 38
Agglomerative clustering with complete linkage

Until a single cluster containing all data points is obtained

complete link.
Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances.
Agglomerative clustering with complete linkage

Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances.
Agglomerative clustering with complete linkage

Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances for $k = 7$. 

[Graph showing the hierarchy and the complete link for $k = 7$.]
Agglomerative clustering with complete linkage

Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances.
Agglomerative clustering with average linkage

Start with each data point in its own cluster
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate…
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 2
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.
Iterate...

average link.
Step# 3
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 4
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 5
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

Iterate...

average link.

Step# 6
Select the two clusters that are closest and merge them
Iterate...

Agglomerative clustering with average linkage
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.

Step# 8
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 9
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 10
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.
Iterate...
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 12
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 13
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 14
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.  Step# 15
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.
Iterate...
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 17
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

Iterate...

average link.

Step# 18
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

Iterate...

average link.
Step# 19
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 20
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.
Iterate...

average link.
Step# 21
Select the two clusters that are closest and merge them
Iterate...
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 23
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 24
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 25
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.  Step# 26
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

Iterate...

average link.

Step# 27
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 28
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 29
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 30
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.

Step# 31
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 32
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them. Iterate...

average link. Step# 33
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 34
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

average link.

Step# 35
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 36
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them.

Iterate...

Step# 37
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them
Iterate...

average link.
Step# 38
Agglomerative clustering with average linkage

Until a single cluster containing all data points is obtained
Agglomerative clustering with average linkage

Truncate the hierarchy to obtain the desired number of clusters

average link.
Agglomerative clustering with average linkage

Truncate the hierarchy to obtain the desired number of clusters

for $k = 9$

average link.
Agglomerative clustering with average linkage

Truncate the hierarchy to obtain the desired number of clusters

average link. for $k = 7$
Agglomerative clustering with average linkage

Truncate the hierarchy to obtain the desired number of clusters

average link.
for $k = 4$
Agglomerative clustering with single linkage

Start with each data point in its own cluster
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 1
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.
Step# 2
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 3
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.
Step# 4
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...

single link.
Step# 5
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link. Step# 7
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 8
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...

single link.

Step# 9
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 12
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...

single link.

Step# 13
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.  
Step# 14
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.

Step# 15
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 16
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.
Step# 17
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 18
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.

Step# 19
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.

Step# 20
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

UEF/School of Computing  JADE:Clustering Basics
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.
Step# 22
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 23
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.
Step# 24
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them. Iterate...

single link.
Step# 25
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

Step# 26
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 27
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

Step # 28

single link.
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link. Step# 29
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.  Step# 30
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.
Step# 31
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.
Step# 32
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.

Step# 33
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them
Iterate...

single link.  
Step# 34
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.
Iterate...

single link.
Step# 35
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.
Step# 36
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them. Iterate...

single link.  
Step# 37
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them.

Iterate...

single link.

Step# 38
Agglomerative clustering with single linkage

Until a single cluster containing all data points is obtained

single link.
Agglomerative clustering with single linkage

Truncate the hierarchy to obtain the desired number of clusters
Agglomerative clustering with single linkage

Truncate the hierarchy to obtain the desired number of clusters

single link. for $k = 9$
Agglomerative clustering with single linkage

Truncate the hierarchy to obtain the desired number of clusters

single link.
for $k = 7$
Truncate the hierarchy to obtain the desired number of clusters

single link.

for $k = 4$
Hierarchical agglomerative algorithms

Different linkage functions produce different cluster hierarchies

single link.  complete link.  average link.
Density-based algorithms aim to identify connected dense areas of the data as the clusters.

Data points that lie in sparse areas of the data might not be assigned to any cluster.
Density-based algorithms aim to identify connected dense areas of the data as the clusters.

Data points that lie in sparse areas of the data might not be assigned to any cluster.

DBSCAN is a popular example of a density-based algorithm.
The DBSCAN algorithm

The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into three categories, depending on their neighborhood

For chosen parameters \( \epsilon \) and \( \tau \)

**core points** have at least \( \tau \) points within a radius \( \epsilon \)

**border points** have less than \( \tau \) points within a radius \( \epsilon \), but at least one is a core point

**noise points** have less than \( \tau \) points within a radius \( \epsilon \), and none is a core point
The DBSCAN algorithm

For chosen parameters $\epsilon$ and $\tau$

Let $N(x, \epsilon)$ denote the set of points within a radius $\epsilon$ of point $x$ (including the point itself), that is

$$
N(x, \epsilon) = \{ x' \in D, d(x, x') \leq \epsilon \}
$$

Let $D_c$, $D_b$ and $D_n$ denote the sets of core, border and noise points respectively, then

$$
D_c = \{ x \in D, |N(x, \epsilon)| \geq \tau \}
$$

$$
D_b = \{ x \in D, |N(x, \epsilon)| < \tau \text{ and } N(x, \epsilon) \cap D_c \neq \emptyset \}
$$

$$
D_n = \{ x \in D, |N(x, \epsilon)| < \tau \text{ and } N(x, \epsilon) \cap D_c = \emptyset \} 
$$
The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into core, border and noise points

(ii) construct a graph with core points as the vertices and with an edge between two vertices if the corresponding points are within a radius $\epsilon$ of each other, find connected components from this graph
The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into core, border and noise points
(ii) find connected components from the graph of core points
(iii) assign border points to the component they are most strongly connected to
The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into core, border and noise points
(ii) find connected components from the graph of core points
(iii) assign border points to the most relevant component

→ The connected components are returned as the clusters
Noise points are not assigned to any cluster
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices,

and find the connected components

Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices, and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices, and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices, and find the connected components

Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$.

Construct a graph with $D_c$ as vertices, and find the connected components.

Assign each point in $D_b$ to the most relevant component.

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices, and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$ and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $D$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices,

and find the connected components

Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.1$

and $\tau = 6$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $D$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices, and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $D$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices, and find the connected components

Assign each point in $D_b$ to the most relevant component

For $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices, and find the connected components

Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices, and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices,
and find the connected components

Assign each point in $D_b$ to the most relevant component

For $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$

Construct a graph with $D_c$ as vertices, and find the connected components

Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.05$ and $\tau = 3$
The DBSCAN algorithm

Given $\epsilon$ and $\tau$, partition $\mathcal{D}$ into $D_c$, $D_b$ and $D_n$
Construct a graph with $D_c$ as vertices,
and find the connected components
Assign each point in $D_b$ to the most relevant component

for $\epsilon = 0.05$
and $\tau = 3$
The DBSCAN algorithm

Unlike for instance \(k\)-means, DBSCAN is not limited to spherical clusters but can detect clusters of arbitrary shapes.

On the other hand, it is limited to detecting clusters of similar densities.

For \(\epsilon = 0.1\) and \(\tau = 6\),

for \(\epsilon = 0.05\) and \(\tau = 3\)
The DBSCAN algorithm

DBSCAN does not require to provide the number of clusters as input parameter, it is set implicitly based on the connectivity of the graph.

For $\epsilon = 0.1$ and $\tau = 6$

For $\epsilon = 0.05$ and $\tau = 3$
DBSCAN does not require to provide the *number of clusters* as input parameter, it is set implicitly based on the connectivity of the graph.

On the other hand, DBSCAN requires to set parameters $\epsilon$ and $\tau$. While their meaning is relatively intuitive, that is, a smaller radius $\epsilon$ and a greater number of neighbors $\tau$ increase the density needed for an area to be considered a cluster, they might be difficult to adjust for a specific data set.
Evaluation
Clustering evaluation

Given a dataset, we can obtain various clusterings, by applying different methods and using different parameter settings. We need to quantify the quality of clusterings, in order to:

- measure the effectiveness and tune the parameters of a particular algorithm
- compare and select clusterings
Clustering evaluation

Clustering is defined as an *unsupervised* task, and often there is *no ground truth clustering* to compare against.

Hence we often need to rely on *internal* validation criteria.
Internal validation criteria

We often need to rely on *internal* validation criteria, such as

**sum of square distances to centroids** determine a centroid for each cluster (or use the representative, for representative-based methods) and compute the sum of square distances from every point to the associated centroid.

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, let $r^{(u)}$ denote the centroid of cluster $C_u$, then

$$SSDC(\mathcal{C}) = \sum_{C_u \in \mathcal{C}} \sum_{x \in C_u} d(x, r^{(u)})^2$$

Smaller values indicate more cohesive clusters.
Internal validation criteria

We often need to rely on *internal* validation criteria, such as

**Intra-cluster vs. inter-cluster distance ratio** compare the distances between pairs of points in the same vs. in different clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, let

$$D(C_u) = \sum_{(x, x') \in C_u \times C_u} d(x, x')$$

$$D_{\text{intra}} = \sum_{C_u \in \mathcal{C}} D(C_u)$$

$$P_{\text{intra}} = \sum_{C_u \in \mathcal{C}} |C_u| \cdot (|C_u| - 1)$$

$$D(C_u, C_v) = \sum_{(x, x') \in C_u \times C_v} d(x, x')$$

$$D_{\text{inter}} = \sum_{(C_u, C_v) \in \mathcal{C} \times \mathcal{C}} D(C_u, C_v)$$

$$P_{\text{inter}} = \sum_{(C_u, C_v) \in \mathcal{C} \times \mathcal{C}} |C_u| \cdot |C_v|$$

then, $\text{DR}(\mathcal{C}) = \frac{D_{\text{intra}}/P_{\text{intra}}}{D_{\text{inter}}/P_{\text{inter}}}$
We often need to rely on *internal* validation criteria, such as

**Intra-cluster vs. inter-cluster distance ratio** compare the distances between pairs of points in the same vs. in different clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$,

$$DR(\mathcal{C}) = \frac{D_{\text{intra}}/P_{\text{intra}}}{D_{\text{inter}}/P_{\text{inter}}}$$

$D_{\text{intra}}$ and $D_{\text{inter}}$ can be computed for $P_{\text{intra}}$ and $P_{\text{inter}}$ pairs of points sampled at random rather than from all pairs, especially for large datasets

Smaller values indicate more cohesive clusters
Internal validation criteria

We often need to rely on *internal* validation criteria, such as

**Silhouette coefficient** compare for each point the average
distance to other points within the same cluster
and average distance to points in other clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ and data point $x \in C_j$, let

\[
D_s(x) = \sum_{\substack{x' \in C_j \\
x' \neq x}} \frac{d(x, x')}{|C_j| - 1} \quad \text{and} \quad D_o(x) = \min_{\substack{C \in \mathcal{C} \\
C \neq C_j}} \sum_{x' \in C} \frac{d(x, x')}{|C|}
\]

then, the silhouette coefficient for point $x$ is

\[
S(x) = \frac{D_o(x) - D_s(x)}{\max(D_o(x), D_s(x))}
\]
Internal validation criteria

We often need to rely on *internal* validation criteria, such as

**Silhouette coefficient** compare for each point the average
distance to other points within the same cluster
and average distance to points in other clusters

For a clustering $C = \{C_1, C_2, \ldots, C_k\}$, the overall silhouette
coefficient is the average of point-specific coefficients

$$S(C) = \sum_{x \in D} \frac{S(x)}{|D|}$$

The silhouette coefficient takes values in $[-1, 1]$, with large
positive values indicating more clearly separated clusters
whereas negative values indicate more blending
Internal validation criteria

We often need to rely on *internal* validation criteria, such as:

- Sum of square distances to centroids
- Intra-cluster vs. inter-cluster distance ratio
- Silhouette coefficient

These measures are biased towards algorithms that optimize a similar criterion.
We often need to rely on *internal* validation criteria, such as:

**Sum of square distances to centroids**

**Intra-cluster vs. inter-cluster distance ratio**

**Silhouette coefficient**

These measures can be used to select values for the parameters, such as the number of clusters $k$.

One might look at how the value of the validation measure evolves when varying the value of a parameter and look for an inflection point.

Caution is required due to the inherent flaws of the measures.
Comparing clusterings

We might want to compare the specific assignments of data points corresponding to two different clusterings, to evaluate how much they agree.

\( C_A \)

\( k_A = 7 \)

\( C_B \)

\( k_B = 4 \)
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters, respectively.

Note that $k_A$ and $k_B$ might be different, and that no mapping between clusters of $C_A$ and of $C_B$ is assumed a priori.
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters.

Given a data point, let $a$ (respectively $b$) be the index of the cluster to which it is assigned in clustering $C_A$ (respectively $C_B$).

A data point might be assigned to

- the first cluster of $C_A$ and the first cluster of $C_B$, i.e. $(a = 1, b = 1)$
- the first cluster of $C_A$ and the second cluster of $C_B$, i.e. $(a = 1, b = 2)$
  ...
- the first cluster of $C_A$ and the last cluster of $C_B$, i.e. $(a = 1, b = k_B)$
  ...
- the last cluster of $C_A$ and the last cluster of $C_B$, i.e. $(a = k_A, b = k_B)$

There are $k_A \cdot k_B$ distinct possible outcomes, i.e. pairs $(a, b)$. 
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters.

Given a data point, let $a$ (respectively $b$) be the index of the cluster to which it is assigned in clustering $C_A$ (respectively $C_B$).

There are $k_A \cdot k_B$ distinct possible outcomes, i.e. pairs $(a, b)$.

Let $\#(a = i, b = j)$ denote the number of data points that belong to the $i^{th}$ cluster of $C_A$ and the $j^{th}$ cluster of $C_B$.

Similarly, let $\#(a = i)$ and $\#(b = j)$ respectively denote the total number of data points in the $i^{th}$ cluster of $C_A$ and in the $j^{th}$ cluster of $C_B$.

We assume the clusterings are partitions of the data set, that is:

$$n = \sum_{i \in [1..k_A]} \#(a = i) = \sum_{j \in [1..k_B]} \#(b = j)$$
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters.

Let $(a = i, b = j)$ denote the number of data points that belong to the $i^{th}$ cluster of $C_A$ and the $j^{th}$ cluster of $C_B$.

The assignment outcome of the pair of clusterings for the data set can be summarized in a $k_A \times k_B$ contingency matrix.

<table>
<thead>
<tr>
<th>Clustering $C_A$</th>
<th>$b = 1$</th>
<th>$b = j$</th>
<th>$b = k_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 1$</td>
<td>$(a = 1, b = 1)$</td>
<td>$(a = 1, b = j)$</td>
<td>$(a = 1, b = k_B)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a = i$</td>
<td>$(a = i, b = 1)$</td>
<td>$(a = i, b = j)$</td>
<td>$(a = i, b = k_B)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a = k_A$</td>
<td>$(a = k_A, b = 1)$</td>
<td>$(a = k_A, b = j)$</td>
<td>$(a = k_A, b = k_B)$</td>
</tr>
</tbody>
</table>
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters

<table>
<thead>
<tr>
<th>Clustering $C_A$</th>
<th>Clustering $C_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$b = 1$</td>
</tr>
<tr>
<td>$a = 1$</td>
<td>$(a = 1, b = 1)$</td>
</tr>
<tr>
<td>$a = i$</td>
<td>$(a = i, b = 1)$</td>
</tr>
<tr>
<td>$a = k_A$</td>
<td>$(a = k_A, b = 1)$</td>
</tr>
</tbody>
</table>

For equal $k_A = k_B$, perfect agreement means that the clusterings are identical up to relabeling of the clusters, i.e. that the rows and columns of the contingency matrix can be reordered so that non-zero values appear only on the diagonal.
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters

<table>
<thead>
<tr>
<th>Clustering $C_A$</th>
<th>$b = 1$</th>
<th>...</th>
<th>$b = j$</th>
<th>...</th>
<th>$b = k_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 1$</td>
<td>$(a = 1, b = 1)$</td>
<td>$(a = 1, b = j)$</td>
<td>$(a = 1, b = k_B)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>$(a = i, b = 1)$</td>
<td>$(a = i, b = j)$</td>
<td>$(a = i, b = k_B)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a = k_A$</td>
<td>$(a = k_A, b = 1)$</td>
<td>$(a = k_A, b = j)$</td>
<td>$(a = k_A, b = k_B)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More in general, large values concentrated in few cells of the contingency matrix indicate high agreement, whereas a more uniform distribution of values across the matrix indicate poor agreement.
Comparing clusterings

Consider two clusterings $C_A$ and $C_B$ with $k_A$ and $k_B$ clusters

<table>
<thead>
<tr>
<th>Clustering $C_B$</th>
<th>$b = 1$</th>
<th>$\ldots$</th>
<th>$b = j$</th>
<th>$\ldots$</th>
<th>$b = k_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering $C_A$</td>
<td>$a = 1$</td>
<td>#(a = 1, b = 1)</td>
<td>#(a = 1, b = j)</td>
<td>#(a = 1, b = k_B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a = i$</td>
<td>#(a = i, b = 1)</td>
<td>#(a = i, b = j)</td>
<td>#(a = i, b = k_B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a = k_A$</td>
<td>#(a = k_A, b = 1)</td>
<td>#(a = k_A, b = j)</td>
<td>#(a = k_A, b = k_B)</td>
<td></td>
</tr>
</tbody>
</table>

Hence, the agreement between the clusterings can be intuitively assessed by just looking at the distribution of values across the contingency matrix.

Measures can be computed from this matrix in order to quantify the degree of agreement.
Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*

\[
\text{Purity}(C_A, C_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}
\]

Values close to 1 are desirable, indicating that the clusters of $C_A$ are very pure with respect to those of $C_B$, i.e. a given cluster of $C_A$ mainly contains points from the same cluster of $C_B$. *Cluster purity* only takes into account the majority assignment.
Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*, the *Gini index*

\[
Purity(C_A, C_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}\\
Gini(C_A, C_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot (1 - \sum_{j \in [1..k_B]} \left( \frac{\#(a = i, b = j)}{\#(a = i)} \right)^2)
\]

Values close to 0 are desirable
Larger values indicate that points from the same cluster of \(C_A\) are scattered across several clusters of \(C_B\)
Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*, the *Gini index* and the *entropy*

\[
Purity(C_A, C_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}
\]

\[
Gini(C_A, C_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot (1 - \sum_{j \in [1..k_B]} \left( \frac{\#(a = i, b = j)}{\#(a = i)} \right)^2)
\]

\[
Entropy(C_A, C_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \sum_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{\#(a = i)} \log_2 \left( \frac{\#(a = i, b = j)}{\#(a = i)} \right)
\]

Values close to 0 are desirable

The *entropy* captures similar properties as the *Gini index*
Comparing clusterings

\[ \text{Purity}(C_A, C_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n} \]

\[ \text{Gini}(C_A, C_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot (1 - \sum_{j \in [1..k_B]} \left( \frac{\#(a = i, b = j)}{\#(a = i)} \right)^2) \]

\[ \text{Entropy}(C_A, C_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \sum_{j \in [1..k_B]} \frac{-\#(a = i, b = j)}{\#(a = i)} \log_2 \left( \frac{\#(a = i, b = j)}{\#(a = i)} \right) \]

! Note that these measures are not symmetric

Computing \( \text{Purity}(C_A, C_B) \) corresponds to taking \( C_B \) as reference, evaluating the purity of the clusters of \( C_A \) with respect to \( C_B \)
Computing \( \text{Purity}(C_B, C_A) \), i.e. taking \( C_A \) as reference instead, will not yield the same value in general
One might compute the measure in both directions and take the average
Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings, to evaluate one clustering against another clustering, include *cluster purity*, the *Gini index* and the *entropy*

These measures constitute *external* validation criteria, since they rely on an external reference to evaluate a clustering
Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings, to evaluate one clustering against another clustering, include *cluster purity*, the *Gini index* and the *entropy*.

These measures constitute *external* validation criteria, since they rely on an external reference to evaluate a clustering.

If some ground truth is available, for instance in the case of synthetically generated data, it can be used as reference.

In our example, for instance, we might consider that grouping beans by species provides a useful reference for evaluating the obtained clusters.