Clustering

Groups together “similar” instances in the data sample

**Basic clustering problem:**
- distribute data into $k$ different groups such that data points similar to each other are in the same group
- Similarity between data points is defined in terms of some distance metric (can be chosen)

Clustering is useful for:
- **Similarity/dissimilarity analysis**
  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label
Clustering example

- We see data points and want to partition them into groups
- What data points belong together?
Clustering example

- We see data points and want to partition them into the groups
- Requires **a dissimilarity or a similarity measure** to tell us what points are close (similar) to each other and are in the same group

**Euclidean distance**

---

Clustering example

- A set of patient cases
- We want to partition them into groups based on similarities

<table>
<thead>
<tr>
<th>Patient #</th>
<th>Age</th>
<th>Sex</th>
<th>Heart Rate</th>
<th>Blood pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient 1</td>
<td>55</td>
<td>M</td>
<td>85</td>
<td>125/80</td>
</tr>
<tr>
<td>Patient 2</td>
<td>62</td>
<td>M</td>
<td>87</td>
<td>130/85</td>
</tr>
<tr>
<td>Patient 3</td>
<td>67</td>
<td>F</td>
<td>80</td>
<td>126/86</td>
</tr>
<tr>
<td>Patient 4</td>
<td>65</td>
<td>F</td>
<td>90</td>
<td>130/90</td>
</tr>
<tr>
<td>Patient 5</td>
<td>70</td>
<td>M</td>
<td>84</td>
<td>135/85</td>
</tr>
</tbody>
</table>
Clustering example

- A set of patient cases
- We want to partition them into the groups based on similarities

<table>
<thead>
<tr>
<th>Patient #</th>
<th>Age</th>
<th>Sex</th>
<th>Heart Rate</th>
<th>Blood pressure …</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient  1</td>
<td>55</td>
<td>M</td>
<td>85</td>
<td>125/80</td>
</tr>
<tr>
<td>Patient  2</td>
<td>62</td>
<td>M</td>
<td>87</td>
<td>130/85</td>
</tr>
<tr>
<td>Patient  3</td>
<td>67</td>
<td>F</td>
<td>80</td>
<td>126/86</td>
</tr>
<tr>
<td>Patient  4</td>
<td>65</td>
<td>F</td>
<td>90</td>
<td>130/90</td>
</tr>
<tr>
<td>Patient  5</td>
<td>70</td>
<td>M</td>
<td>84</td>
<td>135/85</td>
</tr>
</tbody>
</table>

How to design the dissimilarity/similarity measure to quantify similarities?

Similarity and dissimilarity measures

- **Dissimilarity measure**
  - Numerical measure of how different two data objects are
  - Often expressed in terms of a distance metric
  - Euclidean:
    \[
    d(a, b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2}
    \]

- **Similarity measure**
  - Numerical measure of how alike two data objects are
  - Examples:
    - **Gaussian kernel:**
      \[
      K(a, b) = \frac{1}{(2\pi h^2)^{d/2}} \exp \left( -\frac{\|a - b\|^2}{2h^2} \right)
      \]
    - **Cosine similarity:**
      \[
      K(a, b) = a^T b
      \]

Distance metrics

Dissimilarity is often measured with the help of a distance metrics.

Properties of distance metrics:
Assume 2 data entries $a, b$

- **Positiveness:** $d(a, b) \geq 0$
- **Symmetry:** $d(a, b) = d(b, a)$
- **Identity:** $d(a, a) = 0$
- **Triangle inequality:** $d(a, c) \leq d(a, b) + d(b, c)$

Distance metrics

Assume pure real-valued data-points:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>34.5</td>
<td>78.5</td>
<td>89.2</td>
<td>19.2</td>
</tr>
<tr>
<td>23.5</td>
<td>41.4</td>
<td>66.3</td>
<td>78.8</td>
<td>8.9</td>
</tr>
<tr>
<td>33.6</td>
<td>36.7</td>
<td>78.3</td>
<td>90.3</td>
<td>21.4</td>
</tr>
<tr>
<td>17.2</td>
<td>30.1</td>
<td>71.6</td>
<td>88.5</td>
<td>12.5</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What distance metric to use?
Distance metrics

Assume pure real-valued data-points:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>34.5</td>
<td>78.5</td>
<td>89.2</td>
<td>19.2</td>
</tr>
<tr>
<td>23.5</td>
<td>41.4</td>
<td>66.3</td>
<td>78.8</td>
<td>8.9</td>
</tr>
<tr>
<td>33.6</td>
<td>36.7</td>
<td>78.3</td>
<td>90.3</td>
<td>21.4</td>
</tr>
<tr>
<td>17.2</td>
<td>30.1</td>
<td>71.6</td>
<td>88.5</td>
<td>12.5</td>
</tr>
</tbody>
</table>
...

What distance metric to use?

**Euclidian:** works for an arbitrary k-dimensional space

\[ d(a, b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2} \]

---

Distance metrics

Assume pure real-valued data-points:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>34.5</td>
<td>78.5</td>
<td>89.2</td>
<td>19.2</td>
</tr>
<tr>
<td>23.5</td>
<td>41.4</td>
<td>66.3</td>
<td>78.8</td>
<td>8.9</td>
</tr>
<tr>
<td>33.6</td>
<td>36.7</td>
<td>78.3</td>
<td>90.3</td>
<td>21.4</td>
</tr>
<tr>
<td>17.2</td>
<td>30.1</td>
<td>71.6</td>
<td>88.5</td>
<td>12.5</td>
</tr>
</tbody>
</table>

What distance metric to use?

**Squared Euclidian:** works for an arbitrary k-dimensional space

\[ d^2(a, b) = \sum_{i=1}^{k} (a_i - b_i)^2 \]

---
Distance metrics

Assume pure real-valued data-points:

\[
\begin{array}{cccccc}
12 & 34.5 & 78.5 & 89.2 & 19.2 \\
23.5 & 41.4 & 66.3 & 78.8 & 8.9 \\
33.6 & 36.7 & 78.3 & 90.3 & 21.4 \\
17.2 & 30.1 & 71.6 & 88.5 & 12.5 \\
\end{array}
\]

**Manhattan distance:**

works for an arbitrary k-dimensional space

\[
d(a, b) = \sum_{i=1}^{k} |a_i - b_i|
\]

Etc. ..

Distance measures

**Generalized distance metric:**

\[
d^2 (a, b) = (a - b)^T \Gamma^{-1} (a - b)
\]

\(\Gamma\) semi-definite positive matrix

\(\Gamma^{-1}\) is a matrix that weights attributes proportionally to their importance. Different weights lead to a different distance metric.

If \(\Gamma = I\) we get **squared Euclidean**

\[
\Gamma = \Sigma \quad \text{(covariance matrix)} \quad \text{– we get the Mahalanobis distance} \text{ that takes into account correlations among attributes}
\]
## Distance measures

Assume categorical data where integers represent the different categories:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What distance metric to use?

---

**Hamming distance**: The number of values that need to be changed to make them the same
Distance measures.

Assume pure binary values data:

0 1 1 0 1
1 0 1 0 1
0 1 1 0 1
1 1 1 1 1

... 

One metric is the **Hamming distance**: The number of bits that need to be changed to make the entries the same

How about squared Euclidean?

\[
 d^2 (a, b) = \sum_{i=1}^{k} (a_i - b_i)^2
\]

---

Distance measures.

Assume pure binary values data:

0 1 1 0 1
1 0 1 0 1
0 1 1 0 1
1 1 1 1 1

...

One metric is the **Hamming distance**: The number of bits that need to be changed to make the entries the same

How about the squared Euclidean?

\[
 d^2 (a, b) = \sum_{i=1}^{k} (a_i - b_i)^2
\]

**The same as Hamming distance.**
Distance measures

Combination of real-valued and categorical attributes

<table>
<thead>
<tr>
<th>Patient #</th>
<th>Age</th>
<th>Sex</th>
<th>Heart Rate</th>
<th>Blood pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient 1</td>
<td>55</td>
<td>M</td>
<td>85</td>
<td>125/80</td>
</tr>
<tr>
<td>Patient 2</td>
<td>62</td>
<td>M</td>
<td>87</td>
<td>130/85</td>
</tr>
<tr>
<td>Patient 3</td>
<td>67</td>
<td>F</td>
<td>80</td>
<td>126/86</td>
</tr>
<tr>
<td>Patient 4</td>
<td>65</td>
<td>F</td>
<td>90</td>
<td>130/90</td>
</tr>
<tr>
<td>Patient 5</td>
<td>70</td>
<td>M</td>
<td>84</td>
<td>135/85</td>
</tr>
</tbody>
</table>

What distance metric to use?

One solution: A weighted sum approach: e.g. a mix of Euclidian and Hamming distances for subsets of attributes

More complex solutions:
• using tensors and decompositions

Distance measures.

Combination of real-valued and categorical attributes

<table>
<thead>
<tr>
<th>Patient #</th>
<th>Age</th>
<th>Sex</th>
<th>Heart Rate</th>
<th>Blood pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient 1</td>
<td>55</td>
<td>M</td>
<td>85</td>
<td>125/80</td>
</tr>
<tr>
<td>Patient 2</td>
<td>62</td>
<td>M</td>
<td>87</td>
<td>130/85</td>
</tr>
<tr>
<td>Patient 3</td>
<td>67</td>
<td>F</td>
<td>80</td>
<td>126/86</td>
</tr>
<tr>
<td>Patient 4</td>
<td>65</td>
<td>F</td>
<td>90</td>
<td>130/90</td>
</tr>
<tr>
<td>Patient 5</td>
<td>70</td>
<td>M</td>
<td>84</td>
<td>135/85</td>
</tr>
</tbody>
</table>

What distance metric to use?

One solution: A weighted sum approach: e.g. a mix of Euclidian and Hamming distances for subsets of attributes

More complex solutions:
• using tensors and decompositions
Distance metrics and similarity

- **Dissimilarity/distance measure**
  - Numerical measure of how different two data objects are
  - Expressed in terms of distance metrics

- **Similarity measure**
  - Numerical measure of how alike two data objects are
  - Example: Gaussian kernel:
    \[
    K(a,b) = \frac{1}{(2\pi h^2)^{d/2}} \exp\left[-\frac{||a-b||^2}{2h^2}\right]
    \]
  - Cosine similarity:
    \[
    K(a,b) = a^T b
    \]
  - Do not have to satisfy the properties like the ones for the distance metric

Clustering

**Clustering is useful for:**

- **Similarity/Dissimilarity analysis**
  Analyze what data points in the sample are close to each other

- **Dimensionality reduction**
  High dimensional data replaced with a group (cluster) label

- **Data reduction:** Replaces many data-points with a point representing the group mean

**Challenges:**

- How to measure similarity (problem/data specific)?
- How to choose the number of groups?
  - Many clustering algorithms require us to provide the number of groups ahead of time
Clustering algorithms

• **K-means algorithm**
  – *suitable* only when data points have continuous values; groups are defined in terms of cluster centers (also called *means*). Refinement of the method to categorical values: K-medoids

• **Probabilistic methods (with EM) = soft clustering**
  – *Latent variable models*: class (cluster) is represented by a latent (hidden) variable value
  – Every point goes to the class with the highest posterior
  – *Examples*: mixture of Gaussians, Naïve Bayes with a hidden class

• **Hierarchical methods**
  – Agglomerative
  – Divisive
K-means clustering algorithm

- an iterative clustering algorithm
- works in the d-dimensional $R$ space representing $x$

**K-Means clustering algorithm:**

**Initialize** randomly $k$ values of means (centers)

**Repeat**
- Partition the data according to the current set of means (using the similarity measure)
- Move the means to the center of the data in the current partition

**Until** no change in the means

---

**K-means: example**

- **Initialize** the cluster centers

---
K-means: example

• Calculate the distances of each point to all centers

K-means: example

• For each example pick the best (closest) center
K-means: example

- Recalculate the new mean from all data examples assigned to the same cluster center

K-means: example

- Shift the cluster center to the new mean
K-means: example

• Shift the cluster centers to the new calculated means

K-means: example

• And repeat the iteration …
• Till no change in the centers
K-means clustering algorithm

K-Means algorithm:

Initialize randomly $k$ values of means (centers)
Repeat
  – Partition the data according to the current set of means (using the similarity measure)
  – Move the means to the center of the data in the current partition
Until no change in the means

Properties:
• Minimizes the sum of squared center-point distances for all clusters
  $$\min_s \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - u_i||^2 \quad u_i = \text{center of cluster } S_i$$

K-means clustering algorithm

• Properties:
  – converges to centers minimizing the sum of squared center-point distances (still local optima)
  – The result is sensitive to the initial means’ values
• Advantages:
  – Simplicity
  – Generality – can work for more than one distance measure
• Drawbacks:
  – Can perform poorly with overlapping regions
  – Lack of robustness to outliers
  – Good for attributes (features) with continuous values
    • Allows us to compute cluster means
    • k-medoid algorithm used for discrete data
Clustering algorithms

- **K-means algorithm**
  - suitable only when data points have continuous values; groups are defined in terms of cluster centers (also called means). Refinement of the method to categorical values: K-medoids

- **Probabilistic methods (with EM) = soft clustering**
  - **Latent variable models**: class (cluster) is represented by a latent (hidden) variable value
  - Every point goes to the class with the highest posterior
  - **Examples**: mixture of Gaussians, Naïve Bayes with a hidden class

- **Hierarchical methods**
  - Agglomerative
  - Divisive

Probabilistic (EM-based) algorithms

- **Latent variable models**
  Examples: Naïve Bayes with hidden class
  - Mixture of Gaussians

- **Partitioning**: the data point belongs to the class with the highest posterior

- **Advantages**:
  - Good performance on overlapping regions
  - Robustness to outliers
  - Data attributes can have different types of values

- **Drawbacks**:
  - EM is computationally expensive and can take time to converge
  - Density model should be given in advance
Clustering algorithms

- **K-means algorithm**
  - suitable only when data points have continuous values; groups are defined in terms of cluster centers (also called means). Refinement of the method to categorical values: K-medoids
- **Probabilistic methods (with EM) = soft clustering**
  - Latent variable models: class (cluster) is represented by a latent (hidden) variable value
  - Every point goes to the class with the highest posterior
  - Examples: mixture of Gaussians, Naïve Bayes with a hidden class
- **Hierarchical methods**
  - Agglomerative
  - Divisive

Hierarchical clustering

**Can use many different dissimilarity measures**

**Typical dissimilarity measures** $d(a,b)$:

- **Pure real-valued data-points:**
  - Euclidean, Manhattan, Minkowski distances

- **Pure categorical data:**
  - Hamming distance, Number of matching values

**Combination of real-valued and categorical attributes**

- Weighted, or Euclidean
Hierarchical clustering

Two versions of the hierarchical clustering
  – Agglomerative approach
    • Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  – Divisive approach:
    • Splits clusters in top-down fashion, starting from one complete cluster

Hierarchical (agglomerative) clustering

Approach:
  • Compute dissimilarity matrix for all pairs of points
    – uses standard or other distance measures
  • Construct clusters greedily:
    – Agglomerative approach
      • Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  • Stop the greedy construction when some criterion is satisfied
    – E.g. fixed number of clusters
Hierarchical (agglomerative) clustering

Approach:
• Compute dissimilarity matrix for all pairs of points
  – uses standard or other distance measures

N datapoints, O(N^2) pairs, O(N^2) distances
Hierarchical (agglomerative) clustering

Approach:
• **Compute dissimilarity matrix for all pairs of points**
  – uses standard or other distance measures
• **Construct clusters greedily:**
  – **Agglomerative approach**
    • Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
Hierarchical (agglomerative) clustering

Approach:
- Compute dissimilarity matrix for all pairs of points
  - uses standard or other distance measures
- Construct clusters greedily:
  - Agglomerative approach
    - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
Cluster merging

- **Agglomerative approach**
  - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - Merge clusters based on **cluster (or linkage) distances**.
    Defined in terms of point distances. **Examples:**

  **Min distance**
  
  \[
  d_{\text{min}}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)
  \]

  ![Min distance diagram](image)

  **Max distance**
  
  \[
  d_{\text{max}}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)
  \]

  ![Max distance diagram](image)
Cluster merging

- **Agglomerative approach**
  - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - Merge clusters based on **cluster (or linkage) distances**.
    Defined in terms of point distances. **Examples:**

  \[
  \text{Mean distance } d_{\text{mean}}(C_i, C_j) = d\left(\frac{1}{|C_i|} \sum_{i} p_i; \frac{1}{|C_j|} \sum_{j} q_j \right)
  \]

Hierarchical (agglomerative) clustering

**Approach:**
- **Compute dissimilarity matrix for all pairs of points**
  - uses standard or other distance measures
- **Construct clusters greedily:**
  - **Agglomerative approach**
    - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - **Stop the greedy construction** when some criterion is satisfied
    - E.g. fixed number of clusters
Hierarchical (divisive) clustering

Approach:
- **Compute dissimilarity matrix for all pairs of points**
  - uses standard distance or other dissimilarity measures
- **Construct clusters greedily:**
  - **Agglomerative approach**
    - Merge pair of clusters in a bottom-up fashion, starting from singleton clusters
  - **Divisive approach:**
    - Splits clusters in top-down fashion, starting from one complete cluster
- **Stop the greedy construction** when some criterion is satisfied
  - E.g. fixed number of clusters

Hierarchical clustering example

![Hierarchical clustering example](image)
Hierarchical clustering example

- Dendogram

Hierarchical clustering

- **Advantage:**
  - Smaller computational cost; avoids scanning all possible clusterings

- **Disadvantage:**
  - Greedy choice fixes the order in which clusters are merged; cannot be repaired

- **Partial solution:**
  - combine hierarchical clustering with iterative algorithms like k-means algorithm
Other clustering methods

- **Spectral clustering**
  - Uses similarity matrix and its spectral decomposition (eigenvalues and eigenvectors)

- **Multidimensional scaling**
  - Techniques often used in data visualization for exploring similarities or dissimilarities in data.