Clustering

Statistical Natural Language Processing 1

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2025/2026

version: 91bfade+ @2026-01-26

Unsupervised learning

- In unsupervised learning, we do not have labels in our training data
- Our aim is to find useful patterns/structure in the data
  - for exploratory study of the data
  - for augmenting / complementing supervised methods
- Close relationships with 'data mining', 'data science / analytics', 'knowledge discovery'
- Most unsupervised methods can be cast as graphical models with hidden variables
- Evaluation is difficult: no 'true' labels/values

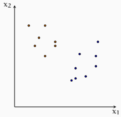
Today's lecture (and later)

- Today: *clustering*, finding related groups of instances
  - k-means
  - hierarchical clustering
  - evaluation
- Later: *clustering*, finding related groups of instances
  - *Density estimation*: finding a probability distribution that explains the data
  - *Dimensionality reduction*: find an accurate/useful lower dimensional representation of the data
  - Unsupervised learning in ANNs (RBMs, autoencoders)

Clustering: why do we do it?

- The aim is to find groups of instances/items that are similar to each other
- Applications include
  - Clustering languages, dialects for determining their relations
  - Clustering (literary) texts, for e.g., authorship attribution
  - Clustering words for e.g., better parsing
  - Clustering documents, e.g., news into topics
  - ...

Clustering in two dimensional space



- Unlike classification, we do not have labels
- We want to find 'natural' groups in the data
- Intuitively, similar or closer data points are grouped together

Similarity and distance

- The notion of distance (similarity) is important in clustering. A distance measure D,
  - is symmetric: $D(a, b) = D(b, a)$
  - non-negative: $D(a, b) \geqslant 0$ for all a, b, and it $D(a, b) = 0$ iff $a = b$
  - obeys triangle inequality: $D(a, b) + D(b, c) \geqslant D(a, c)$
- The choice of distance is application specific
- We will often face with defining distance measures between linguistic units (letters, words, sentences, documents, ...)

Distance measures in Euclidean space

- Euclidean distance:

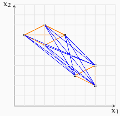
$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{\sum_{j=1}^{k} (a_j - b_j)^2}$$

- Manhattan distance:

$$\|\mathbf{a} - \mathbf{b}\|_1 = \sum_{j=1}^{k} |a_j - b_j|$$

How to do clustering

Most clustering algorithms try to minimize the scatter within each cluster. Which is equivalent to maximizing the scatter between clusters.



$$\sum_{k=1}^{K} \sum_{a \in C_k} \sum_{b \in C_k} d(a, b)$$

$$\sum_{k=1}^{K} \sum_{a \in C_k} \sum_{b \notin C_k} d(a, b)$$

K-means algorithm

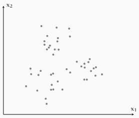
K-means is a popular method for clustering.

1. Randomly choose *centroids*, $m_1, \ldots, m_K$, representing K clusters
2. Repeat until convergence
   - Assign each data point to the cluster of the nearest centroid
   - Re-calculate the centroid locations based on the assignments

Effectively, we are finding a *local minimum* of the sum of squared Euclidean distance within each cluster

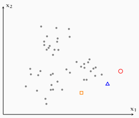
$$\frac{1}{2} \sum_{k=1}^{K} \sum_{a \in C_k} \sum_{b \in C_k} \|a - b\|^2$$

K-means clustering: visualization



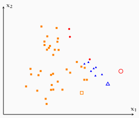
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



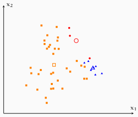
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



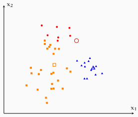
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



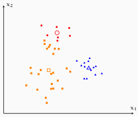
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



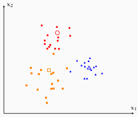
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



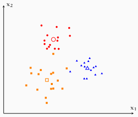
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



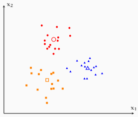
- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

K-means clustering: visualization



- The data
- Set cluster centroids randomly
- Assign data points to the closest centroid
- Recalculate the centroids

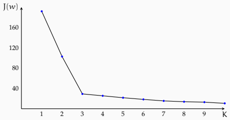
K-means: some issues

- K-means requires the data to be in an Euclidean space
- K-means is sensitive to outliers
- The results are sensitive to initialization
  - There are some smarter ways to select initial points
  - One can do multiple initializations, and pick the best (with lowest within-group squares)
- It works well with approximately equal-size round-shaped clusters
- We need to specify number of clusters in advance

How many clusters?

- The number of clusters is defined for some problems, e.g., classifying news into a fixed set of topics/interests
- For others, there is no clear way to select the best number of clusters
- The error (within cluster scatter) decreases with increasing number of clusters, using a test set or cross validation is not useful either
- A common approach is clustering for multiple K values, and picking where there is an 'elbow' in the graph of the error function

How many clusters?



This plot is sometimes called a *scree plot*.

K-medoids

- K-medoids algorithm is an alternation of K-means
- Instead of calculating centroids, we try to find most typical data point (medoids) at each iteration
- K-medoids can work with distances, does not need feature vectors to be in an Euclidean space
- It is less sensitive to outliers
- It is computationally more expensive than K-means

Hierarchical clustering

- Instead of a flat division to clusters as in K-means, hierarchical clustering builds a hierarchy based on similarity of the data points
- There are two main 'modes of operation':

Bottom-up or *agglomerative* clustering
- starts with individual data points,
- merges the clusters until all data is in a single cluster

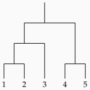
Top-down or *divisive* clustering
- starts with a single cluster,
- and splits until all leaves are single data points

Hierarchical clustering

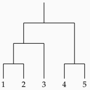
- Hierarchical clustering operates on distances (or similarities)
- The result is a binary tree called *dendrogram*
- Dendrograms are easy to interpret (especially if data is hierarchical)
- The algorithm does not commit to the number of clusters K from the start, the dendrogram can be 'cut' at any height for determining the clusters

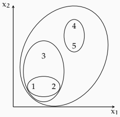
Agglomerative clustering

1. Compute the similarity/distance matrix
2. Assign each data point to its own cluster
3. Repeat until no clusters left to merge
   - Pick two clusters that are most similar to each other
   - Merge them into a single cluster



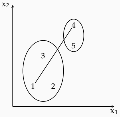
Agglomerative clustering demonstration





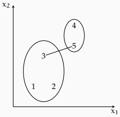
How to calculate between cluster distances

- **Complete** maximal inter-cluster distance
- Single minimal inter-cluster distance
- Average mean inter-cluster distance
- Centroid distance between the centroids



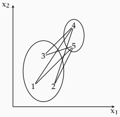
How to calculate between cluster distances

- Complete maximal inter-cluster distance
- **Single** minimal inter-cluster distance
- Average mean inter-cluster distance
- Centroid distance between the centroids



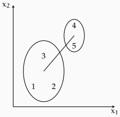
How to calculate between cluster distances

- Complete maximal inter-cluster distance
- Single minimal inter-cluster distance
- **Average** mean inter-cluster distance
- Centroid distance between the centroids



How to calculate between cluster distances

- Complete maximal inter-cluster distance
- Single minimal inter-cluster distance
- Average mean inter-cluster distance
- **Centroid** distance between the centroids



Note: we only need distances, (feature) vectors are not necessary

Clustering evaluation

Evaluating clustering results is often non-trivial

- Internal evaluation is based a metric that aims to indicate 'good clustering': e.g., *Dunn index*, *gap statistic*, *silhouette*
- External metrics can be useful if we have labeled *test* data: e.g., *V-measure*, *B*^{3}*ed F-score*
- The results can be tested on the target application: e.g., word-clusters evaluated based on their effect on parsing accuracy
- Human judgments, manual evaluation – 'looks good to me'

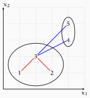
Clustering evaluation

internal metric example: silhouette

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

where

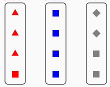
- $a(i)$ average distance between object i and and objects in the same cluster
- $b(i)$ average distance between object i and and objects in the *closest* cluster



Clustering evaluation

external metrics: general intution

- We want clusters that contain members of a single gold-standard class (homogeneity)
- We want all members of a class to be in a single cluster (completeness)



Note the similarity with precision and recall.

Clustering: some closing notes

- Clustering evaluation is not straightforward
- Some clustering methods are unstable, slight changes in the data or parameter choices may change the results drastically
- Approaches against instability include some validation methods, or producing 'probabilistic' dendrograms by running clustering with different options
- Reading suggestion: James et al. (2023, section 12.4)