GL Applied Data Science Program

Unsupervised Learning - Clustering

August 27, 2021

Overview

Overview of this week / module:

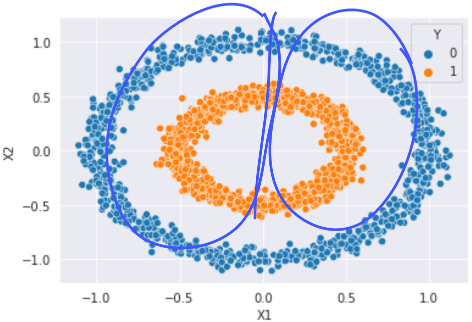
- Data collection and visualization for exploratory data analysis
- Network analysis
- Unsupervised learning - clustering

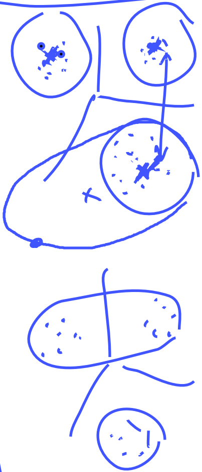
Overview of this lecture:

- Clustering methods
- Community detection in networks

Case study: clustering

- Find groups, so that elements within cluster are very similar and elements between clusters are very different
- **Examples:**
  - Find customer groups to adjust advertisement
  - Find subtypes of diseases to fine-tune treatment
- Our eye is very good at identifying cluster





K-means

$R = \frac{1}{n} X^T X$ correlation matrix

$R = V \Lambda V^T$

$$\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{bmatrix}$$

$$V = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_p \\ | & & | \end{bmatrix}$$

$$\boxed{\lambda_1 + \dots + \lambda_p = p}$$

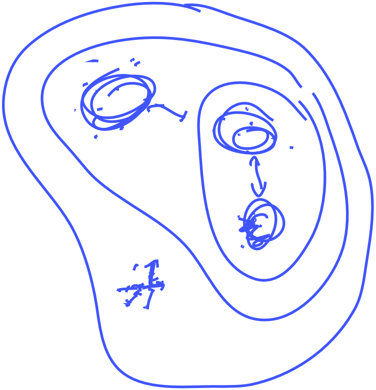
$$\boxed{\lambda_i \geq 1}$$

1.1
1.1
1.1
1.1

$\boxed{0.6}$

Gaussian mixture models

Clustering



hierarchical
clustering

Clustering

N samples, k clusters: k^N possible assignments

- E.g., $N = 100$, $k = 3$: $3^{100} = 5 * 10^{47}$!!

  $\Rightarrow$ impossible to search through all assignments

We will discuss:

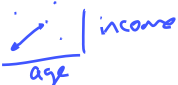
- k-means clustering
- Gaussian mixture models
- Hierarchical clustering
- DBSCAN

Examples of dissimilarity measures between samples

Examples of dissimilarity measures between samples

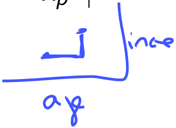
- Euclidean distance (i.e., ℓ_2 - norm)

$$d(x^{(i)}, x^{(j)}) = \sqrt{(x_1^{(i)} - x_1^{(j)})^2 + (x_2^{(i)} - x_2^{(j)})^2 + \cdots + (x_p^{(i)} - x_p^{(j)})^2}$$



- Manhattan distance (i.e., ℓ_1 - norm)

$$d(x^{(i)}, x^{(j)}) = |x_1^{(i)} - x_1^{(j)}| + |x_2^{(i)} - x_2^{(j)}| + \cdots + |x_p^{(i)} - x_p^{(j)}|$$



- Maximum distance (i.e., ℓ_∞ - norm)

$$d(x^{(i)}, x^{(j)}) = \max_{k=1,\dots,p} |x_k^{(i)} - x_k^{(j)}|$$

- or more flexible dissimilarity satisfying

$$d(x^{(i)}, x^{(j)}) \geq 0, \;\; d(x^{(i)}, x^{(i)}) = 0, \;\; d(x^{(i)}, x^{(j)}) = d(x^{(j)}, x^{(i)})$$

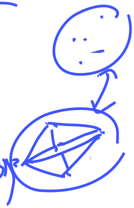
- K (fixed!) Clusters are obtained by minimizing some loss function
- Natural loss function given by within-groups sum of squares (WGSS):

minimize

$$W(C) = \sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k} d(x^{(i)}, x^{(j)})^2$$

maximize

$$\sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k'} d(x^{(i)}, x^{(j)})$$



- K (fixed!) Clusters are obtained by minimizing some loss function
- Natural loss function given by within-groups sum of squares (WGSS):

$$\boxed{W(C) = \sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k} d(x^{(i)}, x^{(j)})^2}$$

- $W(C)$ characterizes the extent to which observations assigned to the same cluster tend to be close to one another or observations between different clusters are further apart from each other

$$d(x^{(i)}, x^{(j)})^2 = \|x^{(i)} - x^{(j)}\|_2^2$$

$$= \|x^{(i)} - \mu_c - (x^{(j)} - \mu_c)\|_2^2$$

K-means clustering

- K (fixed!) Clusters are obtained by minimizing some loss function
- Natural loss function given by within-groups sum of squares (WGSS):

$$W(C) = \sum_{k=1}^{K} \sum_{C(x^{(i)})=k} \sum_{C(x^{(j)})=k} d(x^{(i)}, x^{(j)})^2$$

- $W(C)$ characterizes the extent to which observations assigned to the same cluster tend to be close to one another or observations between different clusters are further apart from each other
- K-means clustering: $d(x^{(i)}, x^{(j)})^2 = \|x^{(i)} - x^{(j)}\|_2^2$
- Then WGSS becomes: $W(C) = \sum_{k=1}^{K} 2N_k \sum_{C(x^{(i)})=k} \|x^{(i)} - \mu_k\|_2^2$, where N_k is the total number of points in cluster k

K-means clustering

- Exact solution of minimization problem is computationally infeasible
  - Use greedy algorithm
  - Use random restarts to avoid local optima
- Leads to spherical shaped clusters of similar radii

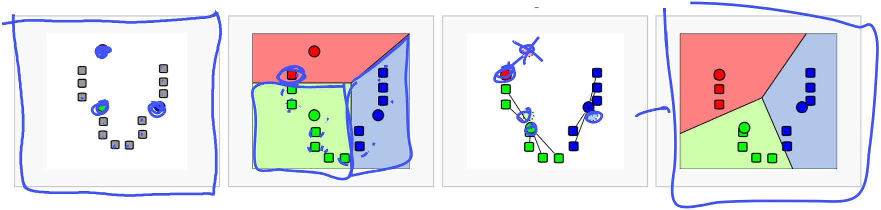
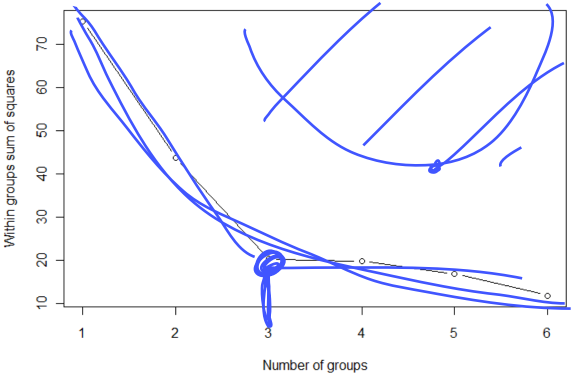


Image source: Wikipedia

Choosing the number of clusters

- Run K-means clustering for several number of groups K
- Plot WGSS versus the number of groups
- Choose number of groups after the last big drop of the curve

Example:



Partitioning around medoids (PAM)

- K-Means: Cluster centers μ_k can be arbitrary points in space

  $\Rightarrow$ very sensitive to outliers!

Partitioning around medoids (PAM)

- K-Means: Cluster centers μ_k can be arbitrary points in space

  $\Rightarrow$ very sensitive to outliers!

- Robust alternative: Partitioning around medoids (PAM)
  - Cluster center must be an observation ("medoid")
  - More robust against outliers
  - Also gives a representative object for each cluster (e.g., for easy interpretation)

Gaussian mixture model

- Soft version of k-means clustering based on a statistical model

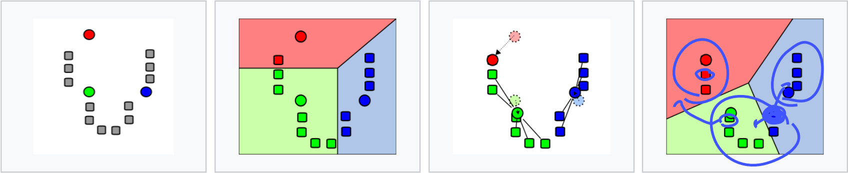


Image source: Wikipedia

Gaussian mixture model

- Assume underlying statistical model:

$$P(x) = \sum_{k=1}^{K} P(\text{cluster } k) P(x \mid \text{cluster } k),$$

where $X \mid \text{cluster } k \sim \mathcal{N}(\mu_k, \Sigma_k)$

- Sample x is assigned to cluster k that maximizes $P(\text{cluster } k \mid x)$
- Estimating $P(\text{cluster } k)$, μ_k and Σ_k by maximum likelihood estimation is difficult (leads to a non-convex optimization problem)
- Parameter estimates are usually found using the Expectation-Maximization (EM) algorithm

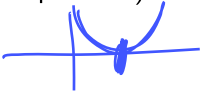
Gaussian mixture model

- Assume underlying statistical model:

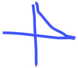
$$P(x) = \sum_{k=1}^{K} P(\text{cluster } k) P(x \mid \text{cluster } k),$$

where $X \mid \text{cluster } k \sim \mathcal{N}(\mu_k, \Sigma_k)$

- Sample x is assigned to cluster k that maximizes $P(\text{cluster } k \mid x)$
- Estimating $P(\text{cluster } k)$, μ_k and Σ_k by maximum likelihood estimation is difficult (leads to a non-convex optimization problem)

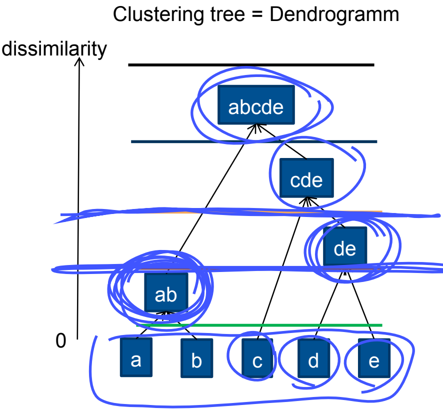


- Parameter estimates are usually found using the Expectation-Maximization (EM) algorithm
- Number of clusters is found for example by maximizing the Bayesian information criterion



$$\boxed{\text{BIC} = \log\text{-likelihood} - \frac{\log(n)}{2} \cdot (\#\text{ of parameters})}$$

Hierarchical clustering

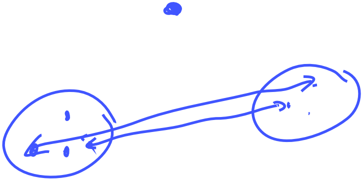


- Agglomerative clustering: Build up clusters from individual observations
- Divisive clustering: Start with whole group of observations and split off clusters

Advantage of hierarchical clustering:

- Solve clustering for all possible numbers of cluster $1, 2, \ldots, n$ at once
- Choose desired number of clusters later

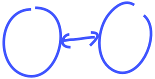
Examples of dissimilarity measures between clusters



Examples of dissimilarity measures between clusters

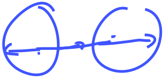
- single linkage (i.e., minimum distance)

$$d(C_r, C_s) = \min_{x^{(i)} \in C_r, x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$



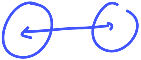
- complete linkage (i.e., maximum distance)

$$d(C_r, C_s) = \max_{x^{(i)} \in C_r, x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$



- average linkage (i.e., average distance)

$$d(C_r, C_s) = \frac{1}{n_r} \frac{1}{n_s} \sum_{x^{(i)} \in C_r} \sum_{x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$



Examples of dissimilarity measures between clusters

- single linkage (i.e., minimum distance)

$$d(C_r, C_s) = \min_{x^{(i)} \in C_r, x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$

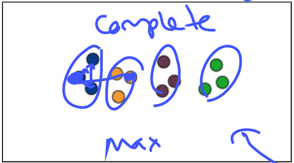
- complete linkage (i.e., maximum distance)

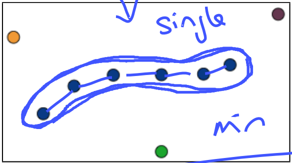
$$d(C_r, C_s) = \max_{x^{(i)} \in C_r, x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$

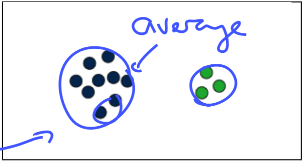
- average linkage (i.e., average distance)

$$d(C_r, C_s) = \frac{1}{n_r} \frac{1}{n_s} \sum_{x^{(i)} \in C_r} \sum_{x^{(j)} \in C_s} d(x^{(i)}, x^{(j)})$$

How do the resulting clusters look like? Which one is which?



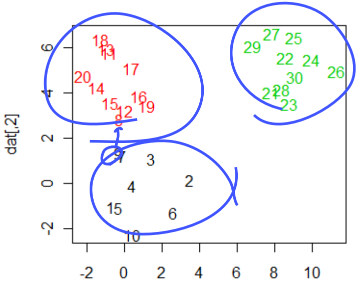


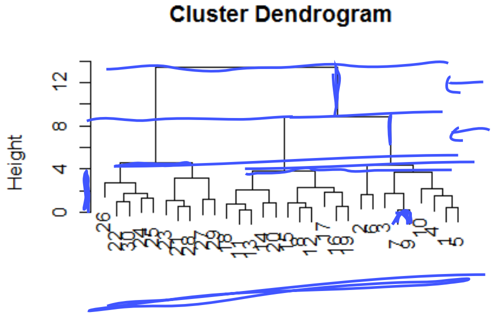


Choosing the number of clusters

- No strict rule
- Find the largest vertical "drop" in the tree

Example:





DBSCAN

- Uses 2 parameters: minPts (minimum number of points) and ϵ (radius of neighborhood)
- Core points have at least minPts within distance ϵ
- Clusters are defined by looking at all points reachable from a core point

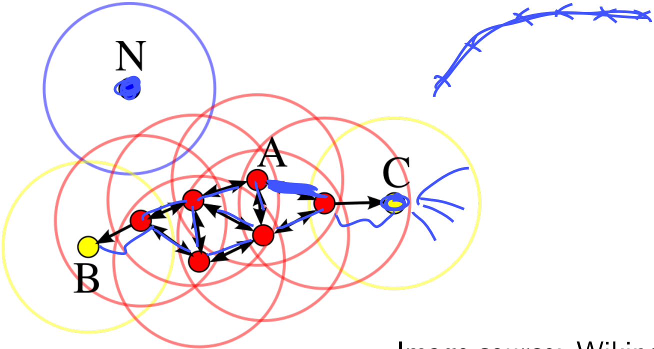


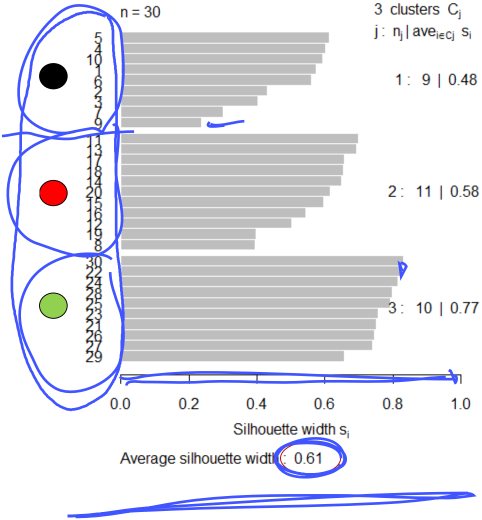
Image source: Wikipedia

Quality of clustering: Silhouette plot

Compute for each sample $x^{(i)}$:

- $a(x^{(i)})$ = average dissimilarity between $x^{(i)}$ and all other points in its cluster
- $b(x^{(i)})$ = average dissimilarity between $x^{(i)}$ and the closest cluster it does not belong to
- $S(x^{(i)}) \in [-1, 1]$ with

$$\boxed{S(x^{(i)}) = \frac{(b(x^{(i)}) - a(x^{(i)}))}{\max(a(x^{(i)}), b(x^{(i)}))}}$$

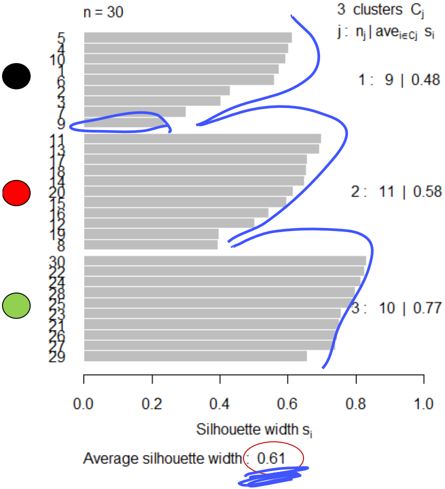


Quality of clustering: Silhouette plot

Compute for each sample $x^{(i)}$:

- $a(x^{(i)})$ = average dissimilarity between $x^{(i)}$ and all other points in its cluster
- $b(x^{(i)})$ = average dissimilarity between $x^{(i)}$ and the closest cluster it does not belong to
- $S(x^{(i)}) \in [-1, 1]$ with

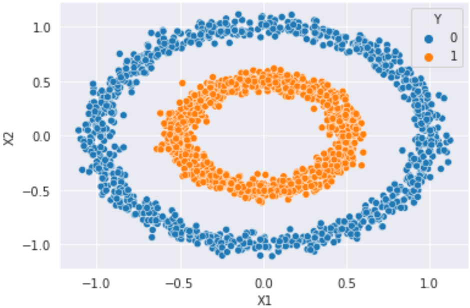
$$S(x^{(i)}) = \frac{(b(x^{(i)}) - a(x^{(i)}))}{\max(a(x^{(i)}), b(x^{(i)}))}$$



Note: $S(x^{(i)})$ large (0.5 is often used as cut-off): well clustered; $S(x^{(i)})$ small: badly clustered; $S(x^{(i)}) < 0$: assigned to wrong cluster

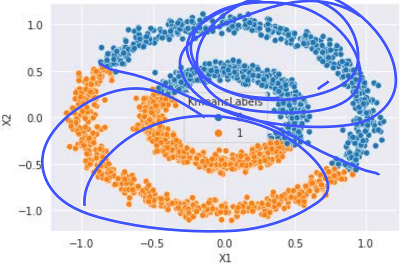
Case study: clustering

Which clustering methods are able to identify the two clusters?

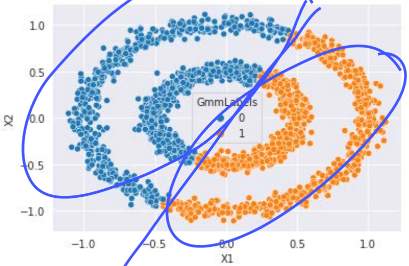


Case study: clustering

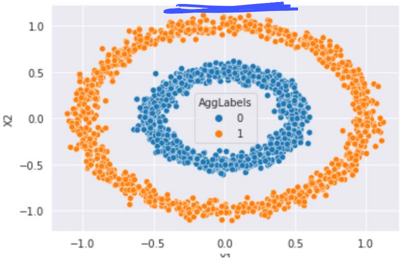
k-means clustering



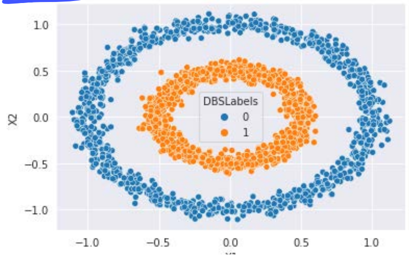
Gaussian mixture model



Hierarchical clustering (single linkage)



DBSCAN



Community detection

Community detection:

- detect subsets of nodes that are more densely connected between each other in the network than outside the community

Clustering

- determine subsets of points that are ‘close’ to each other given a pairwise distance or similarity measure
- can be used also for community detection by defining a vertex similarity measure (e.g., geodesic distance, number of different neighbors, correlation between adjacency matrix columns, etc.)
- can use clustering methods discussed so far based on these similarity measures

Other methods: Divisive algorithm using betweenness

- Intuition: intercommunity edges have a large value of edge betweenness, because many shortest paths connecting vertices of different communities will pass through them

- Algorithm of Girvan and Newman (2002): iteratively remove edges with highest betweenness centrality

- can define betweenness e.g. using geodesic or random walk

Other methods: Modularity maximization

- quality function: function that assigns a number (quality measure) to each partition of a graph
- most popular quality function: modularity

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - P_{ij}) \delta(C_i, C_j),$$

where P_{ij} is expected number of edges between i and j in a null model, for example:

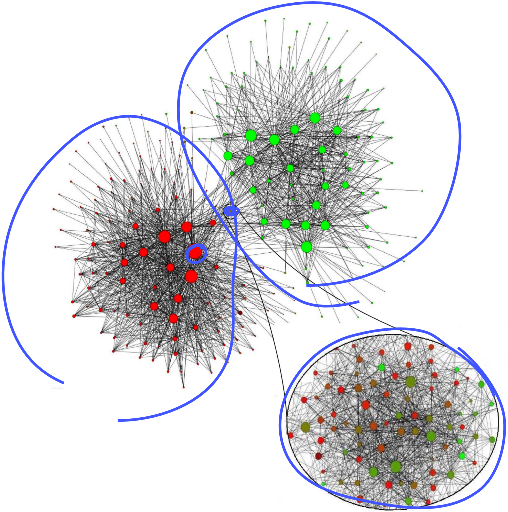
$\frac{m}{\binom{n}{2}}$

- $P_{ij} = \frac{2m}{n(n-1)}$, where m is the total number of edges and n is the total number of nodes in the network

Louvain method (Blondel et al., 2008)

- modularity optimization is NP-complete (Brandes et al., 2006)
- Louvain method: very fast heuristic
  - put each node in its own community
  - put node i into community j that yields biggest increase in modularity
  - replace communities by supernodes, where edge weight between supernodes is sum of edge weights between corresponding nodes
  - iterate process until Q cannot be improved
- provides decomposition of network into communities for different levels of organization
- extremely fast: runs in $\mathcal{O}(m)$

Louvain method (Blondel et al., 2008)



Belgian mobile phone network with 2M customers (red: French-speaking, green: Dutch-speaking).

References

- For clustering
  - Chapter 14 in

    T. Hastie, R. Tibshirani, & J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

- For community detection in networks:
  - V. D. Blondel, et al. *Fast unfolding of communities in large networks*. Journal of Statistical Mechanics: Theory and Experiment 10, 2008.
  - S. Fortunato. *Community detection in graphs*. Physics Reports 486, 2010.
  - Lecture notes on Laplacian and spectral clustering (prominent method not discussed in this module) by T. Roughgarden & G. Valiant:

    `http://web.stanford.edu/class/cs168/l/l11.pdf`