Lecture 16: Clustering

Machine Learning, Summer Term 2019

Michael Tangermann   Frank Hutter   Marius Lindauer

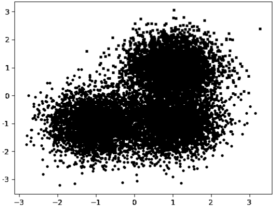
University of Freiburg

Lecture Overview

What is Cluster Analysis?

Also called: clustering, segmentation analysis, taxonomy analysis, automatic classification, numerical taxonomy, botryology, typological analysis, community detection, ...

(Plot modified from scikit-learn clustering tutorial)



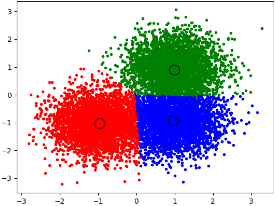
Unsupervised task:

Group objects such that objects within a group are more similiar/related to each other *(in some sense)* than to objects of another group.

What is Cluster Analysis?

Also called: clustering, segmentation analysis, taxonomy analysis, automatic classification, numerical taxonomy, botryology, typological analysis, community detection, ...

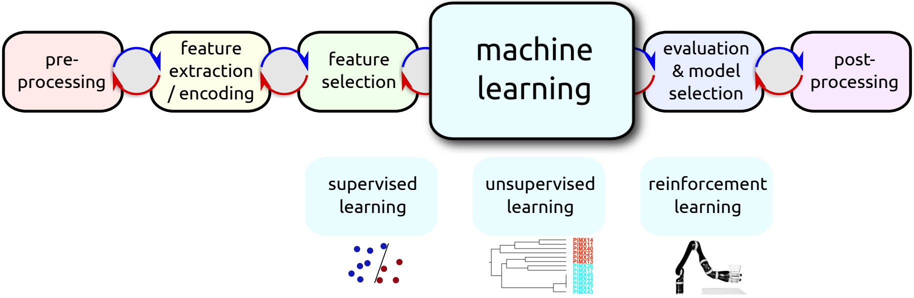
(Plot modified from scikit-learn clustering tutorial)



Unsupervised task:

Group objects such that objects within a group are more similiar/related to each other *(in some sense)* than to objects of another group.

ML Design Cycle



Cluster analysis is an unsupervised learning task

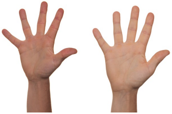
- Ground truth about clusters is not provided $\rightarrow$ evaluation is tricky!

Given:

- N high dimensional data points $\mathbf{x}_i \in \mathbb{R}^D$ with $i = 1 \ldots N$.
- Data is collected in matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$

y missing

Applications



Example Applications (I)

- Medical imaging (fMRI, CT, PET): differentiate between different types of tissues, find tissue boundaries
- Biology: determine communities of organisms in space and time, compute data-driven phylogenetic trees
- Genetics: group DNA sequences into gene families
- Biochemistry / chemistry / pharmacology: group compounds according to their reaction mechanism
- Market research: detect clusters of customers with similar behavior, find market segments
- Social networks: recognize communities
- Search engines: Post-processing of search results into groups of hits that refer to vastly different topics

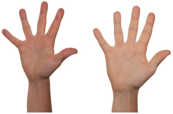
Example Applications (II)

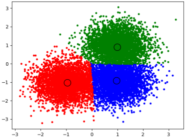
- Image segmentation: border detection, track objects
- Anomaly detection: identify outliers in data streams, network attacks, misbehaving software, sensor failures (robotics, production lines), predictive maintenance
- Finance: find stock clusters of similar behaviour
- Text analysis: clustering of documents into topics
- ...

Lecture Overview

How could Clusters be Determined?

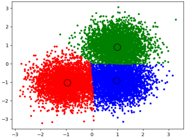
Which metrics might be used to define clusters?





How could Clusters be Determined?

Which metrics might be used to define clusters?



Zoo of clustering methods available, that exploit e.g.:

- distance/similarity function (between cluster members, between members of different clusters)
- connectivity structure using distances $\rightarrow$ single/avg/max linkage clustering, graph-based $\rightarrow$ clique
- centroid + neighborhood
- densities
- expected distributions

Lecture Overview

K-Means Clustering (Steinhaus, 1957)

Find set $C = C_1, ..., C_k$ of k clusters represented by cluster centroids μ_k such, that the clusters have equal variance.

$\rightarrow$ Minimize the *inertia* or *within-cluster sum-of-squares* criterion:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_j||^2$$

Observations:

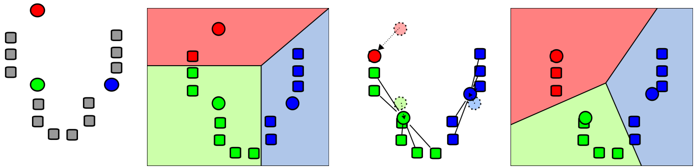
- Cluster centroids μ_j do not need to be points of the training data sets
- Unfortunately NP-hard problem!
- Clustering can be represented by Voronoi tesselation

K-Means Clustering

Practical solution by heuristic approximation
(e.g. Lloyd's algorithm (1957, 1982), similar to expectation-maximization):

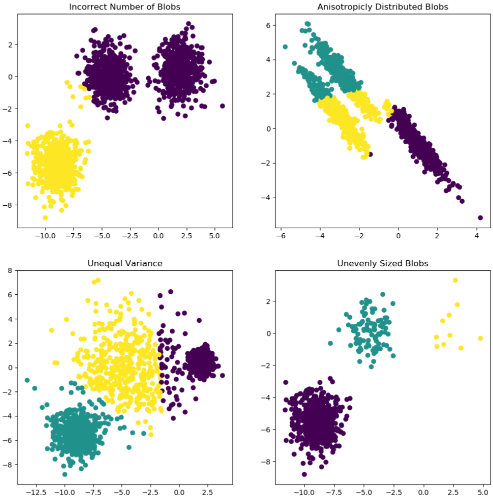
`Initialize` k `data points as cluster centroids.`
`Then iterate these two steps until convergence of the centroids:`

1. `Assign each data point to its nearest centroid`
   `(`$\rightarrow$ `approximations necessary for high dimensions!)`
2. `Create k new centroids by taking the mean value of all of the data points assigned to each novel centroid`
   `(`$\rightarrow$ `approximations necessary for many data points)`



K-Means Clustering

Problematic data sets for k-means:



K-Means Clustering

Pros:

- conceptually simple algorithm
- mini-batches and different kind of initialization strategies are available ($\rightarrow$ k-means++)
- scales to many data points (if approximations are utilized)

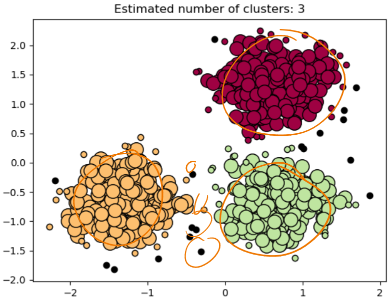
Cons:

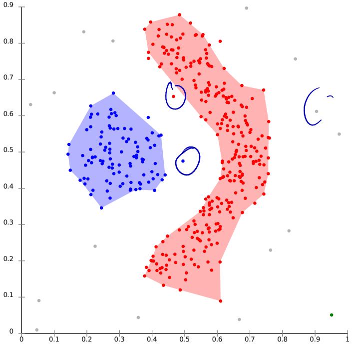
- sensitive to initialization of centroids
- can not model noise or outliers
- concave cluster shapes are problematic
- can not deal with uneven variance between clusters
- number k of clusters needs to be provided

Lecture Overview

DBSCAN

- DBSCAN (Ester et al., 1996) is a density-based, non-parameteric clustering method.
- It received the **test of time award** at KDD conference in 2014.





DBSCAN: A Cluster has High Density

DBSCAN: A Cluster has High Density

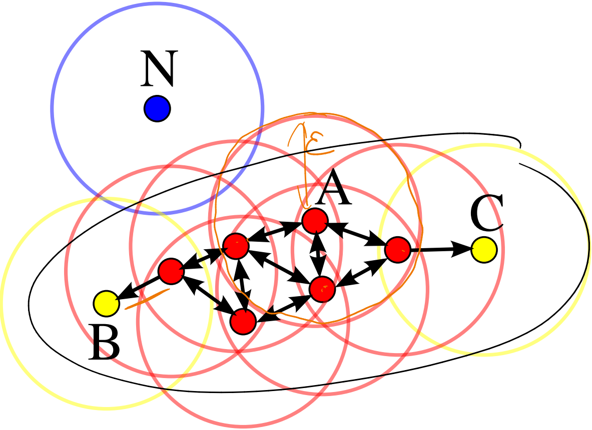
- A **core point** p is a data point in an area of high density. It is defined by having at least `minPts` points (including p) within an `eps`-neighborhood.

$p \longrightarrow q$

- A point q is **directly reachable** from $\cancel{q}$ p, if p is in the eps neighborhood of a core point q.
- A point q is **reachable** from p if there is a path along the points $p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i. Note that this implies that all points on the path must be core points, with the possible exception of q (if q is on the fringe of a cluster).
- All points not reachable from any other point are considered **outliers** or **noise** points.

If p is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it.

DBSCAN



Example with `minPts`=4. The `eps`-neighborhoods are indicated by circles.

DBSCAN Algorithm

DBSCAN creates clusters by sequentially considering the training data points, starting with an arbitrary point:

1. Find the points in the `eps` neighborhood of every point, and identify the core points with more than `minPts` neighbors.
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an `eps` neighbor, otherwise assign it to noise.

Pseudocode for DBSCAN

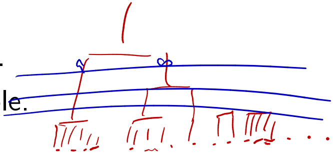
```
DBSCAN(DB, distFunc, eps, minPts) {
   C = 0                                                     /* Cluster counter */
   for each point P in database DB {
      if label(P) ≠ undefined then continue                  /* Previously processed in inner loop */
      Neighbors N = RangeQuery(DB, distFunc, P, eps)         /* Find neighbors */
      if |N| < minPts then {                                 /* Density check */
         label(P) = Noise                                    /* Label as Noise */
         continue
      }
      C = C + 1                                              /* next cluster label */
      label(P) = C                                           /* Label initial point */
      Seed set S = N \ {P}                                   /* Neighbors to expand */
      for each point Q in S {                                /* Process every seed point */
         if label(Q) = Noise then label(Q) = C               /* Change Noise to border point */
         if label(Q) ≠ undefined then continue               /* Previously processed */
         label(Q) = C                                        /* Label neighbor */
         Neighbors N = RangeQuery(DB, distFunc, Q, eps)      /* Find neighbors */
         if |N| ≥ minPts then {                              /* Density check */
            S = S ∪ N                                        /* Add new neighbors to seed set */
         }
      }
   }
}
```

```
RangeQuery(DB, distFunc, Q, eps) {
   Neighbors = empty list
   for each point P in database DB {                         /* Scan all points in the database */
      if distFunc(Q, P) ≤ eps then {                         /* Compute distance and check epsilon */
         Neighbors = Neighbors ∪ {P}                         /* Add to result */
      }
   }
   return Neighbors
}
```

Characteristics of DBSCAN

Pros:

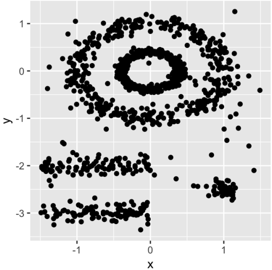
- DBSCAN is fast! With n data points: $O(n^2)$ as worst case (all points belong to single cluster), but $O(n \log(n))$ with good data structures and for typical data.
- DBSCAN is deterministic for a fixed processing sequence.
- Number of clusters is determined automatically.
- Hyperparameter ~~min samples~~ minPts can express prior knowledge about noise.
- Different distance metrics can be utilized.
- Hierarchical variant HDBSCAN is available.



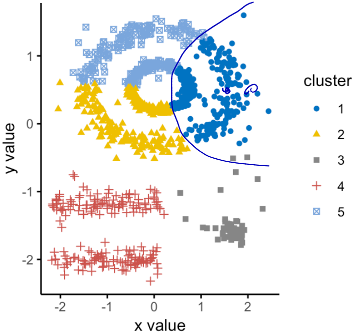
Cons:

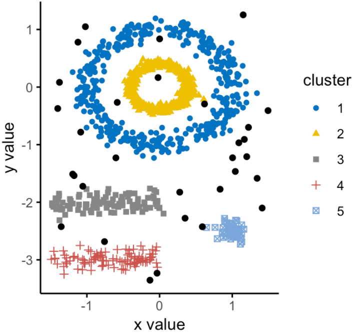
- Varying the sequence of data points processed can lead to different clusterings.
- Hyperparameter eps is critical, no good default!

Comparison of K-Means and DBSCAN

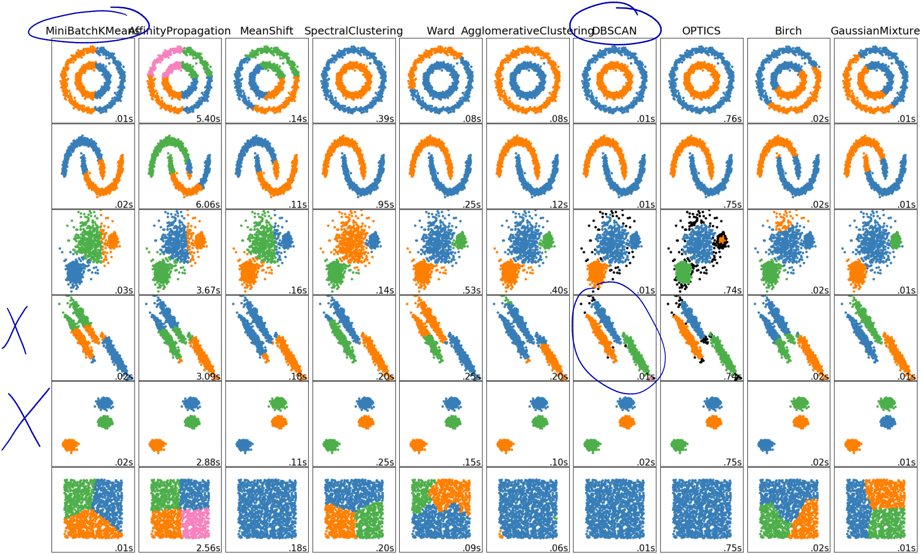


k-means



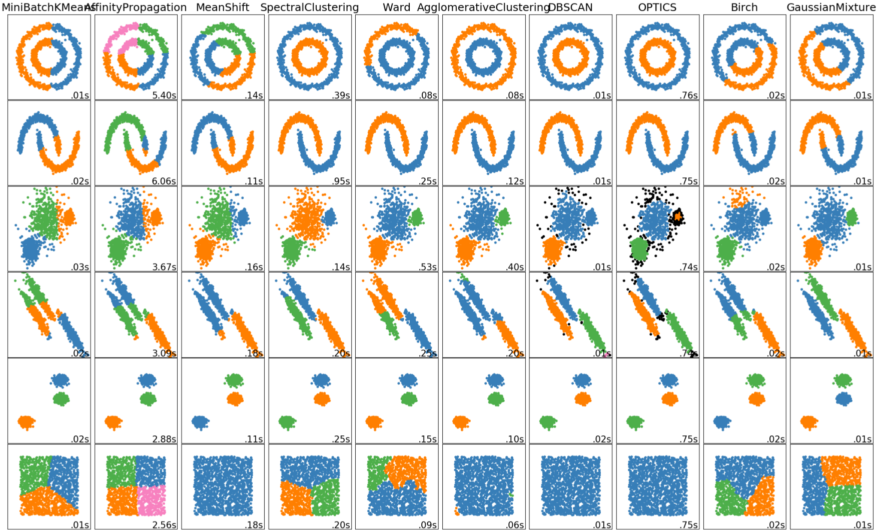


A few Clustering Algorithms on Toy Data



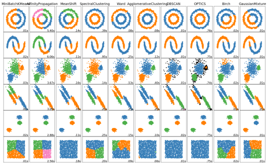
https://scikit-learn.org/stable/modules/clustering.html

A few Clustering Algorithms on Toy Data



https://scikit-learn.org/stable/modules/clustering.html

Criteria for the Choice of Clustering Algorithms



Does the algorithm...

- expects each cluster to follow a specific distribution? (e.g. Gaussian)?
- considers density of data points?
- deal well with noisy data / high-dimensional data / redundant dimensions / irrelevant dimensions?
- deliver hard / soft clustering?
- deliver a strict partitioning (i.e. each object belongs to exactly one cluster)?
- deliver a hierarchical clustering?

Wrap-Up: Summary by Learning Goals

Having heard this lecture and doing the assigment on clustering, you will be able to:

- Explain, which metrics can be used to create a clustering from unlabeled data
- formulate the optimization problem for k-means clustering and implement an iterative heuristic
- Describe pros and cons of k-means and DBSCAN
- Derive a metric for the quality of a given clustering (e.g. via the "**silhouette score**", see assignment)