Lecture 8: Density-based and high-dimensional clustering

Felix Held, Mathematical Sciences

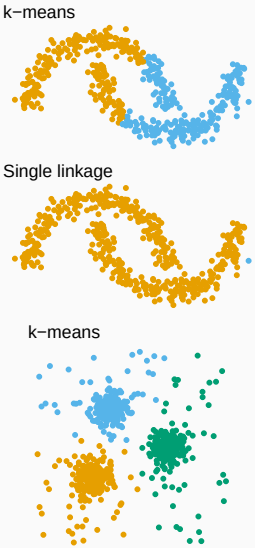
MSA220/MVE440 Statistical Learning for Big Data

23rd April 2020

Density-based clustering

Yet another approach to clustering

- Most methods discussed so far have problems with odd, non-convex shapes
- What about **noise**? Some observations might not fit into any cluster
- Clusters are dense regions in feature space
  - What is dense?
  - How to find groups and separate the noise?
- **Naive approach:** Find points surrounded by many other points and connect them to a cluster. Points that do not end up in a cluster are noise.



Notation in density-based clustering

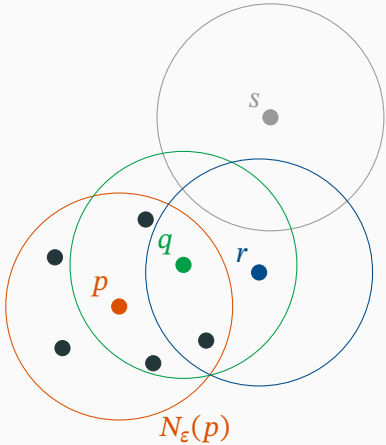
Let $\varepsilon > 0$ and $n_{\min} \in \mathbb{N}$ be two **tuning parameters**. Assume each observation is a **point p in a database/dataset D** and there is a **distance measure** $d(p, q)$.

- **ε-neighbourhood of p:** $N_\varepsilon(p) = \{q \in D | d(p, q) \leq \varepsilon\}$
- **Core point:** A $p \in D$ s.th. $|N_\varepsilon(p)| \geq n_{\min}$
- p is **directly density-reachable** from a core-point q if $p \in N_\varepsilon(q)$
- p is **density-reachable** from a core-point q if there is a chain $q = p_1, p_2, \dots, p_m = p$ s.th. p_{i+1} is directly density-reachable from p_i
- p and q are **density-connected** if there is a core-point o s.th. p and q are density-reachable from o

Concepts in density-based clustering

Let $n_{\min} = 5$ then

- p and q are core points, r is not
- q is directly density-reachable from p and *vice versa*
- r is density-reachable from p
- p and r are density-connected
- s is neither density-connected nor density-reachable



Density-based clusters

A **cluster** C is a set of points in D s.th.

1. If $p \in C$ and q is density-reachable from p then $q \in C$ (**maximality**)
2. For all $p, q \in C$: p and q are density-connected (**connectivity**)

This leads to **three types of points**

1. **Core points:** Part of a cluster and at least $n_{\min}$ points in neighbourhood
2. **Border points:** Part of a cluster but not core points
3. **Noise:** Not part of any cluster

Note: Border points can have **non-unique cluster assignments**

DBSCAN algorithm

Computational procedure:

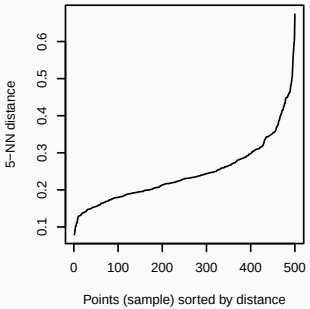
1. Go through each point p in the dataset D
2. If it has already been processed take the next one
3. Else determine its ε-neighbourhood. If less than $n_{\min}$ points in neighbourhood, label as noise. Otherwise, start a new cluster.
4. Find all points that are density-reachable from p and add them to the cluster, unless they were previously assigned to a different cluster.

Dependence on $n_{\min}$

- Controls how easy it is to connect components in a cluster
  - Too small and most points are core points, creating many small clusters
  - Too large and few points are core points, leading to many noise-labelled observations
- A cluster has by definition at least $n_{\min}$ points
- Choice of $n_{\min}$ is very dataset dependent
- Tricky in high-dimensional data (**curse of dimensionality**, everything is far apart)

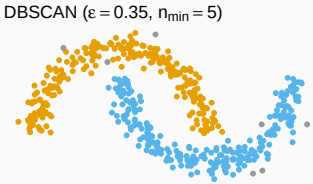
Dependence on ε

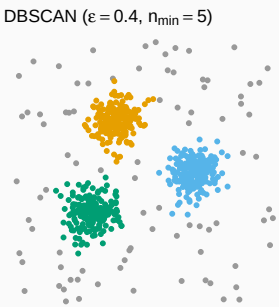
- Controls how much of the data will be clustered
  - Too small and small gaps in clusters cannot be bridged, leading to isolated islands in the data
  - Too large and everything is connected
- Choice of ε is also dataset dependent but there is a **decision tool**.
  - For each point in the dataset, determine distance to its k nearest neighbours (typically $k = n_{\min}$) and pick the largest distance
  - Sort the distances from smallest to largest and plot
  - The optimal ε will be roughly at the elbow



DBSCAN example

- DBSCAN is able to cluster points in the situations advertised and correctly identifies noise points
- Very sensitive to the choice of tuning parameters





High-dimensional clustering

One version of the curse of dimensionality

Samples tend to be further away from the origin

Let $\mathbf{x} \in [-1,1]^p$ be a uniformly distributed random variable. For $0 \leq t \leq 1$ consider

$$q = \mathbb{P}(-t \leq x^{(1)} \leq t, \dots, -t \leq x^{(p)} \leq t) = \left(\frac{2t}{2}\right)^p$$

$$\Rightarrow \quad t = q^{1/p}$$

In a large enough sample about q percent of observations will be in $[-t,t]^p$.

In high dimensions, most data points are far away from the origin.

How should t be chosen so that about q percent of observations lie in $[-t,t]^p$?

p	$q = 1\%$	$q = 10\%$
2	$t \approx 0.01$	$t \approx 0.32$
3	$t \approx 0.22$	$t \approx 0.46$
10	$t \approx 0.63$	$t \approx 0.79$
100	$t \approx 0.95$	$t \approx 0.98$

Another version of the curse of dimensionality

Pairwise distances grow with dimension

If $\mathbf{x}, \mathbf{y} \in [0,1]^p$ uniformly distributed, then their pairwise distance $||\mathbf{x} - \mathbf{y}||_2$ grow with p.

The last column suggests that the mean pairwise distance grows as $O(\sqrt{p})$.

The standard deviations stay constant suggesting that observations have increasingly similar pairwise distances in high dimensions.

Mean and standard deviation of the pairwise distances of $n = 500$ simulated observations.

p	Mean	SD	Mean / $\sqrt{p}$
2	0.52	0.25	0.37
3	0.66	0.25	0.38
10	1.28	0.25	0.40
100	4.07	0.24	0.41
500	9.13	0.25	0.41
1000	12.91	0.24	0.41

High-dimensional clustering

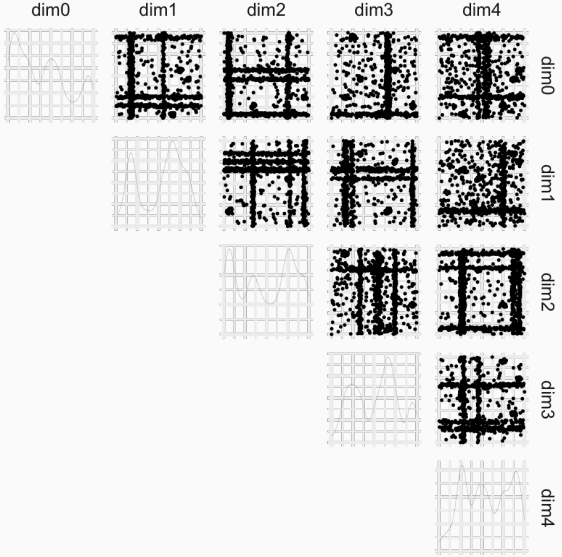
All clustering relies on pairwise distances between observations. For increasing feature space dimension, these become increasingly meaningless.

What can be done about this dilemma?

1. **Feature selection:** Deciding on a subset of the original features
   - There is no response in clustering, making it harder to judge feature quality
   - Selecting features with large variance across all samples can be used. However, features very relevant to only a subset of samples might get filtered out.
2. **Feature transformation:** Combining existing features while reducing dimension (e.g. PCA)
   - Clusters might become hard to interpret in the original context
   - The feature transformation might destroy/obscure relationships in the original data that it cannot capture
   - Since features are transformed, it is not guaranteed that uninformative features are actually filtered out

Subspace clustering

Data in many subspaces



Subspace clustering

Instead of selecting single features, sometimes it would be better if we could select whole subspaces of features.

How can these be found?

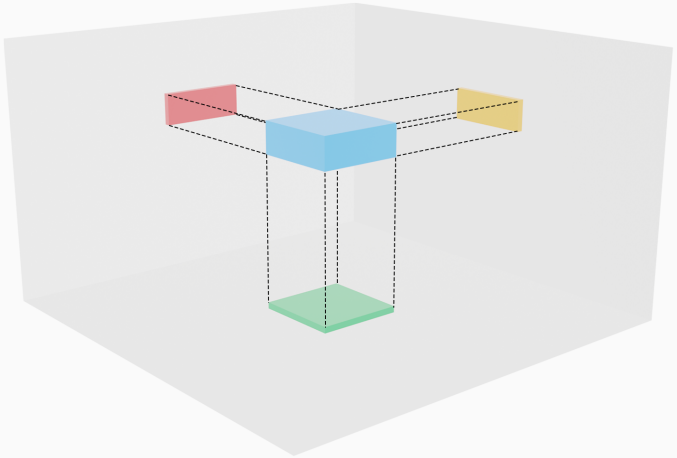
- It is infeasible to look at all possible subspaces
- As in combinatorial clustering or stepwise selection methods in regression there are two approaches
  1. **Top-down:** Start with all dimensions and search for relevant dimensions
  2. **Bottom-up:** Start with a grid in each dimension and combine them step-wise

Examples:

- CLIQUE: Bottom-up algorithm; grid-based and density-based
- ProClus: Top-down algorithm; variant of k-medoids

Apriori principle

A dense region in q dimensions should lead to dense regions in every $(q-1)$-dimensional projection ('Clusters cast shadows')

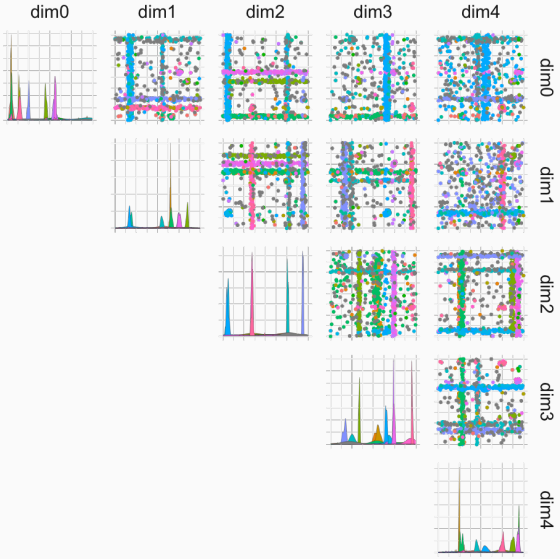


The CLIQUE algorithm

Input parameters: A positive integer m and $1 > \tau > 0$

1. **For 1D:**
   - 1.1 Partition each dimension $i = 1, \ldots, p$ into m intervals and compute the proportions of contained one-dimensional projections of the data (*one-dimensional histograms*)
   - 1.2 Keep those intervals containing proportions $> \tau$
   - 1.3 Among the remaining intervals, merge neighbouring ones
2. **Moving from dimension $q-1$ to q:**
   - 2.1 Create volumes in q dimensions by combining those found in $q-1$ dimensions.
   - 2.2 Recompute proportions per constructed volume
   - 2.3 Keep those volumes containing proportions $> \tau$
   - 2.4 Among the remaining volumes, merge neighbouring ones
3. **Post-processing:** Filter out remaining volumes with low contained proportion and try to enlarge found clusters as much as possible

Subspace clustering: CLIQUE

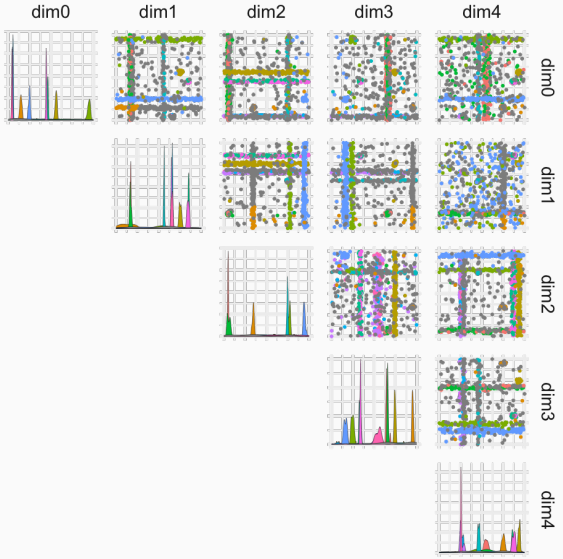


The ProClus algorithm

Input parameters: Number K of clusters to be found and the average number of dimensions $d > 0$ in which clusters reside

1. **Initialisation:**
   - Find $M > K$ medoids in a greedy fashion
   - Use a random sample of size K
2. **Iterations:** Until no change within some threshold
   - 2.1 For each medoid: Locally find best dimensions where data is dense
     - Dimensions in which average distance to the medoid $<$ overall avg. distance
     - d influences how many dimensions are picked
   - 2.2 Assign data points to medoids measuring distance only in the selected dimensions of the medoid
   - 2.3 Evaluate clustering quality and remove medoids with small numbers of points. Replace them with others from the initial set of medoids
3. **Post-Processing:** Determine the best dimensions once more for each medoid and assign points to clusters

Subspace clustering: ProClus



Notes on subspace clustering

- **Pro:** Can deal with complex structures and high-dimensions
- **Pro:** The variable selection/subspace discovery that is performed per cluster can lead to mechanistic insight into a problem
- **Con:** Hard to tune since it is difficult to get an understanding for the data (e.g. grid-size, average number of subspace dimensions, …)
- Some adaptive algorithms exist to e.g. estimate optimal grid-size from data

Take-home message

- Density-based clustering allows to capture complex shapes and the identification of noise during clustering
- The curse of dimensionality makes pairwise distances in high-dimensions relatively meaningless
- Subspace clustering attempts to find clusters that are only active in some dimensions of the feature space