Computational Tools for Data Science 02807, E 2018

Paul Fischer

Institut for Matematik og Computer Science
Danmarks Tekniske Universitet

Efterår 2018

Clustering

Today's schedule

- What is clustering
- Hierarchical clustering
- The k-means algorithm
- The DBSCAN algorithm (not in the book)
- Evaluating clusterings

What is Clustering

Clustering is the task of grouping objects from a large set in such a way that objects in the same group are more “similar” to each other than to those in other groups. The groups are called *clusters*.

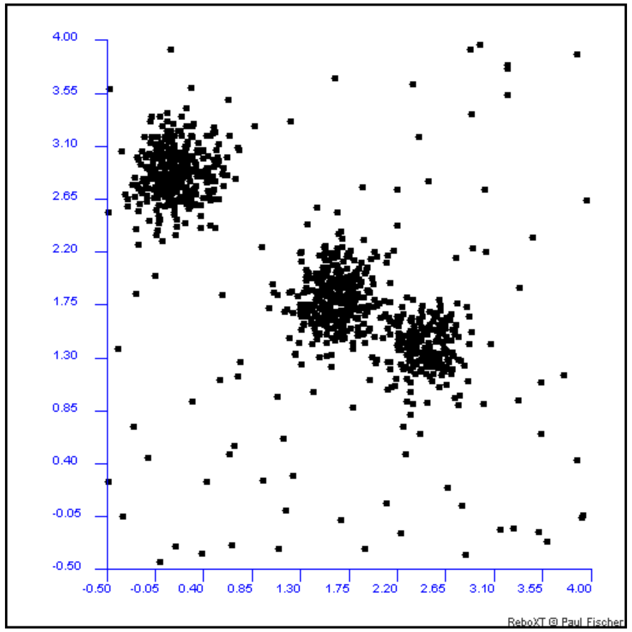
The measure of similarity has to be specified according to the problem under consideration.

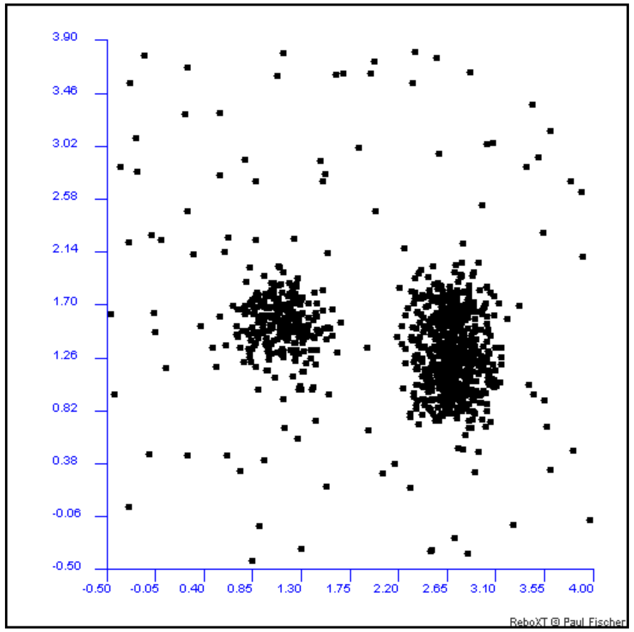
Example

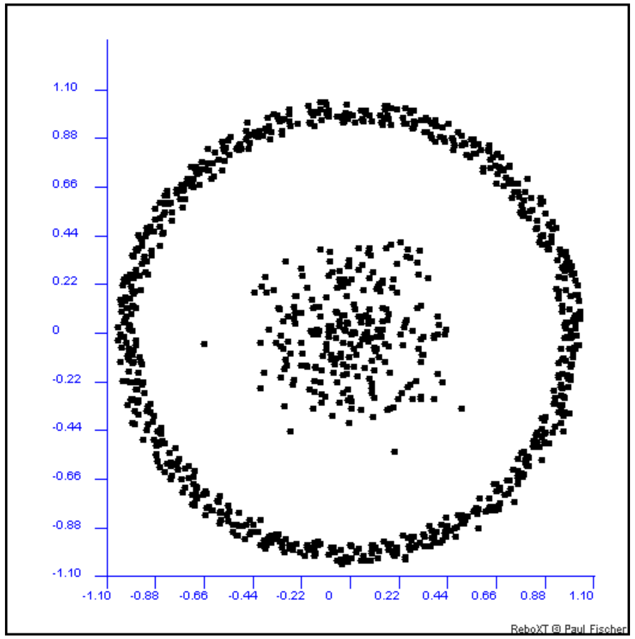
- People with similar interests in social media.
- People with similar taste for movies in a streaming provider.
- Detecting similarities in medical tests.
- Detection of groups in statistical data.

Examples

How many clusters do you see?







General assumptions

We assume that the data to be considered is numerical. Each data point is a d-dimensional vector

$$\boldsymbol{x} = (x_0, \dots, x_{d-1})$$

The input to clustering is a multi-set $S = \langle \boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1} \rangle$ of n data points.

For a multi-set $S = \langle \boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1} \rangle$ the *centroid* (center of gravity) $\text{cent}(S)$ is defined by

$$\text{cent}(S) = \frac{1}{n} \sum_{i=0}^{n-1} \boldsymbol{x}_i$$

where the sum is componentwise. That is for $\boldsymbol{x} = (x_0, x_1, \cdots, x_{d-1})$ and $\boldsymbol{y} = (y_0, y_1, \cdots, y_{d-1})$:

$$\boldsymbol{x} + \boldsymbol{y} = (x_0 + y_0\, , \, x_1 + y_1\, , \, \dots \, , \, x_{d-1} + y_{d-1})$$

A distance measure $\{dist(\cdot,\cdot)\}$ is defined on $\mathbb{R}$, where $dist(x,y) \geq 0$, $dist(x,y) = \text{dist}(y,x)$, and $dist(x,y) \leq \text{dist}(x,z) + \text{dist}(z,y)$, i.e., dist is a metric.

Outline of hierarchical clustering

The algorithm joins cluster, which are close to each other. Let $\boldsymbol{c}_i$ be the centroid of cluster C_i.

- *Initialisation* Each data point is a cluster by itself, i.e., $C_i = \{\boldsymbol{x_i}\}$ and $\boldsymbol{c}_i = \boldsymbol{x}_i$.
- *Merging* Find clusters C_i and C_j where $\text{dist}(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is minimal (breaking ties, e.g., randomly). Merge C_i and C_j into a new cluster C_k, where the indexing is done by new numbers or re-using existing ($k = i$). Remove C_i an C_j. Note, that merging is multi-set union, denoted $\uplus$.
- Stop the process when some criterium is satisfied, e.g., a certain number of clusters is reached.

Pseudo code

```
for $i = 0, \dots, n-1$ do
    $C_i \leftarrow \{\boldsymbol{x}_i\}$;
    $\boldsymbol{c}_i \leftarrow \boldsymbol{x}_i$;
end
$goon \leftarrow true$;
while $goon$ do
    find $i \neq j$ with $\text{dist}(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is minimal;
    $C_k = C_i \uplus C_j$;
    $\boldsymbol{c}_k = \text{cent}(C_k)$;
    Remove $C_i$ and $C_j$ as clusters and $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ as centers ;
    Update $goon$
end
```

Note that in general $\boldsymbol{c}_k \neq (\boldsymbol{c}_i + \boldsymbol{c}_j)/2$, but the summands habe to weighted by the size of the clusters. $\boldsymbol{c}_k \neq \frac{(|C_i|\boldsymbol{c}_i + |C_j|\boldsymbol{c}_j)}{|C_i| + |C_j|}$

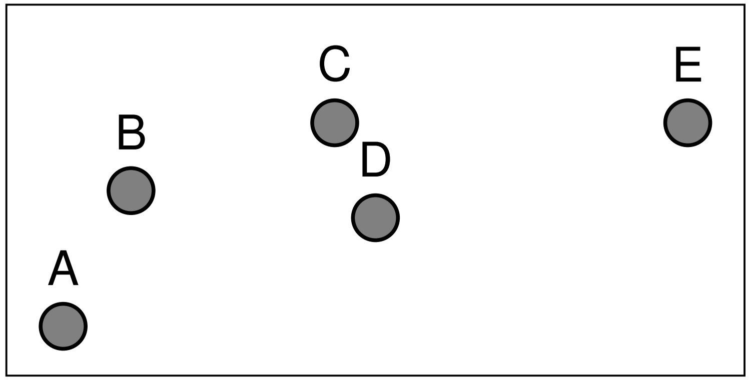
Stop criteria

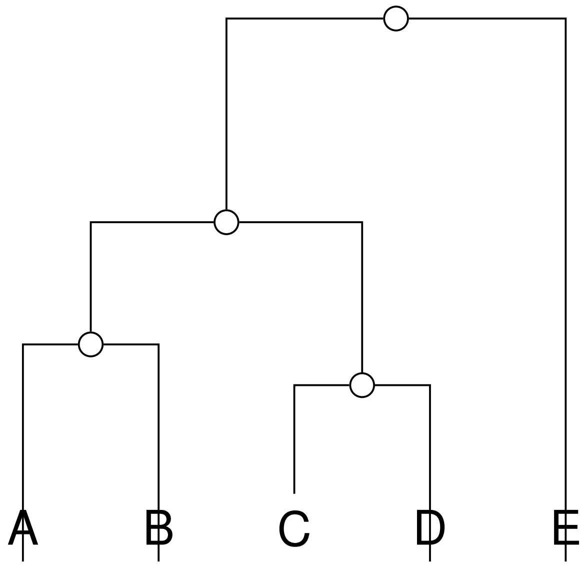
- A number of clusters has been specified beforehand. When only this number is left, the algoritm terminates.
- The *density* of the cluster resulting from a merger is bad. The density is the average distance between points in a cluster. This can also be used to reject mergers in course of the algorithm.
- See more in the book.

Without further features, which “guide” the algorithm, hierarchical clustering might perform bad on larger data sets.

Phylogenetic Trees

Hierarchical clustering is useful to generate *phylogenetic trees* (on small data sets).





The k-means algorithm

The k-means algorithm requires the user to provide the number k of clusters and delivers a partition of S into k clusters, $C_0, \ldots, C_{k-1}$.

Idea:

0 Randomly select k points $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{k-1}$ from S. These are the centers of the clusters.

1 For each $\boldsymbol{x}_i \in S$, assign $\boldsymbol{x}_i$ to that cluster the center of which is closest.

2 Re-compute the centers $\boldsymbol{c}_j$ to be the centroids of C_j.

- Iterate steps 1 and 2 until no (only very small) changes occur.

The k-means algorithm

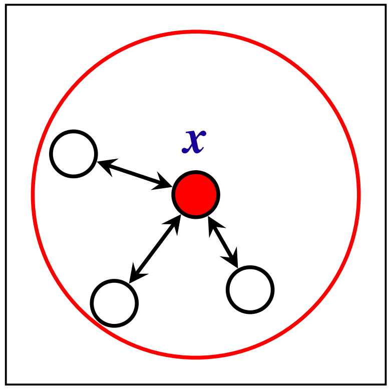
```
Input: A multi-set $S = \langle \boldsymbol{x}_0, \dots, \boldsymbol{x}_{n-1} \rangle$ and a positive integer $k$
Randomly select $k$ distinct points $\boldsymbol{c}_i$ from $S$;
while goon do
    for $j = 0, \dots, k-1$ do
        $C_j \leftarrow \emptyset$;
    end
    for $i = 0, \dots, n-1$ do
        $\ell = \arg\min\{\text{dist}(\boldsymbol{x}_i, \boldsymbol{c}_j) \mid j = 0, \dots, k-1\}$;
        $C_\ell = C_\ell \uplus \{\boldsymbol{x}_i\}$;
    end
    for $j = 0, \dots, k-1$ do
        $\boldsymbol{c}_j \leftarrow \text{cent}(C_j)$;
    end
    Update $goon$;
end
```

DBSCAN, Idea

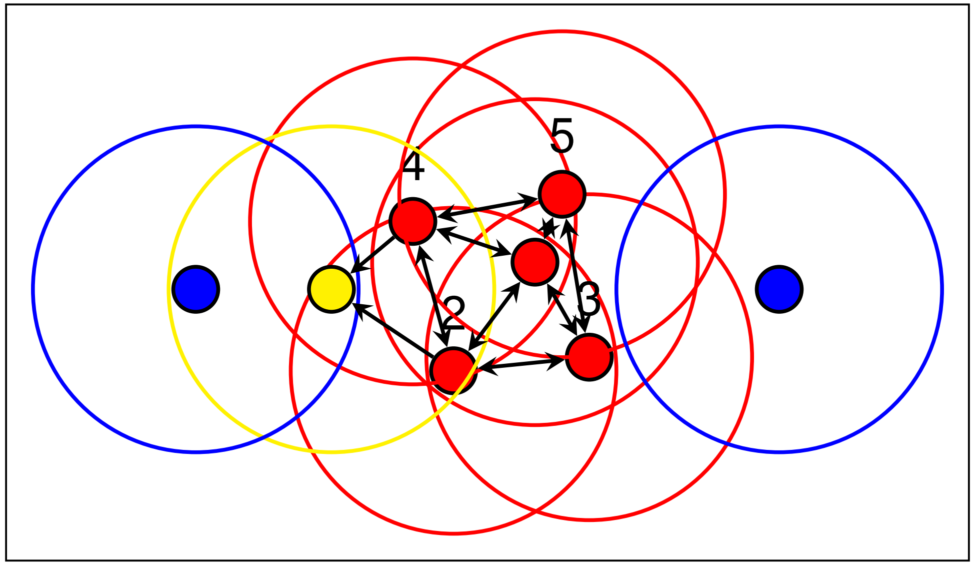
Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN)

- One defines the concept of *(density) reachable* for the data points.
- The algorithm uses two parameters: $\varepsilon > 0$, the *neighborhood radius*, and $m \in \mathbb{N}^+$, the minimum required neighbourhood size.
- The algorithms classifies points as **core** (centrally in a cluster), **rim** (at the edge of a cluster) and **noise** not belonging to any cluster.
- The number of clusters is not fixed beforehand, it is implicitly controlled by ε and m.
- A point $\boldsymbol{x}$ is **core** if there area at least m points (incl. $\boldsymbol{x}$) within distance ε, i.e., $|\{\boldsymbol{z} \mid \mathrm{dist}(\boldsymbol{x}, \boldsymbol{z}) \leq \varepsilon\}| \geq m$.
- A point $\boldsymbol{z}$ is *directly reachable* from $\boldsymbol{x}$ if $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{z}) \leq \varepsilon$ and $\boldsymbol{x}$ is **core**.
- A point $\boldsymbol{z}$ is *reachable* from $\boldsymbol{x}$ if there are points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$, such that $\boldsymbol{x} = \boldsymbol{x}_1$, $\boldsymbol{z} = \boldsymbol{x}_k$, $\boldsymbol{x}_{i+1}$ is reachable from $\boldsymbol{x}_i$, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{k-1}$ are **core**. If $\boldsymbol{z}$ is not **core**, it is **rim**.

DBSCAN, Idea



Point x is **core** for $m = 4$.



For $m = 4$: **core** points in red, **rim** points in yellow, **noise** points in blue.

DBSCAN Pseudo Code

Algorithm 1: DBSCAN(S, ε, m)

```
Mark all $x_i \in S$ as **unvisited**;
for $i = 0, \ldots, n-1$ do
    if $x_i$ is *unvisited* then
        $N \leftarrow$ neigh$(x_i, \varepsilon)$;
        if $|N| < m$ then
            Mark $x_i$ as **noise**;
        else
            $C \leftarrow \emptyset$ ;
            Mark $x_i$ as **core**;
            expand$(x_i, N, C, \varepsilon, m)$;
        end
    end
end
```

Algorithm 2: neigh(x, ε)

```
return all points $z$ with dist$(x, z) \leq \varepsilon$
```

Algorithm 3: expand$(x, N, C, \varepsilon, m)$

```
$C \leftarrow C \uplus \{x\}$;
for $z \in N$ do
    if $z$ is not *visited* then
        Mark $z$ as **visited**;
        $N' \leftarrow$ neigh$(z, \varepsilon)$;
        if $|N'| \geq m$ then
            $N \leftarrow N \uplus N'$;
        end
    end
    if $z$ is not in any cluster then
        $C \leftarrow C \uplus \{z\}$;
        if $|N'| \geq m$ then
            Mark $z$ as **core**;
        else
            Mark $z$ as **rim**;
        end
    end
end
```

Evaluating the result

One way is the *DaviesBouldin index*

$$DB = \frac{1}{k}\sum_{i=0}^{k-1} \max\left\{\left(\frac{\sigma_i + \sigma_j}{\text{dist}(\boldsymbol{c}_i, \boldsymbol{c}_j)}\right) \mid j \neq i\right\}$$

where $\boldsymbol{c}_i = \text{cent}(C_i)$ and $\sigma_i = \frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i} \text{dist}(\boldsymbol{x}, \boldsymbol{c}_i)$ the average distance of points in cluster C_i form its center.

This index is low if the distances (σ_i) in the clusters are low and the distances between the clusters ($\text{dist}(\boldsymbol{c}_i, \boldsymbol{c}_j)$) are large.

Final remarks

Some algorithms depend on user supplied parameters. For DBSCAN you can find some guideline on `https://en.wikipedia.org/wiki/DBSCAN`.

Most clustering algorithms require finding "close by" points (nearest neighbours). Computing the distance form one point to every other one is very time consuming $O(n)$. Computing the distances for all points beforehand and storing them requires $O(n^2)$ space, which is infeasible for medium n.

There are sophisticated data structures (e.g., Voronoi diagrams) for the nearest neighbor problem. However the suffer the "curse of dimensionality".

How does one represent clusters? 1) sets of points, 2) use an integer array C where $C[i]$ is the number of the cluster in which $\boldsymbol{x}_i$ is located, 3) use some other data structure.