Clustering

Eugeny Malyutin

Motivation

Suppose we have large amount of unlabelled texts, based on which we are able to:

- Create a useful content-recommendation service
  («It looks like you love to read about battle-rap. Do you want more? »)

- make conclusions on topics represented in textual data -- or other data structure features
  (trending topics, emerging trends, blogosphere analysis)

- find duplicate content
  ("the same piece of news, can be skipped", "plagiarism!")

And so on, and so on… **Another ideas?**

Clustering

Problems?

There are:

- Objects (documents) space **X**
- Set of objects (documents) **X^{l}**

Our task:

- split document collection into groups (clusters) so that the documents in one group were similar to each other, whereas documents from different groups should be dissimilar

Clustering

There are:

- Objects (documents) space **X**
- Set of objects (documents) $\mathbf{X}^l$

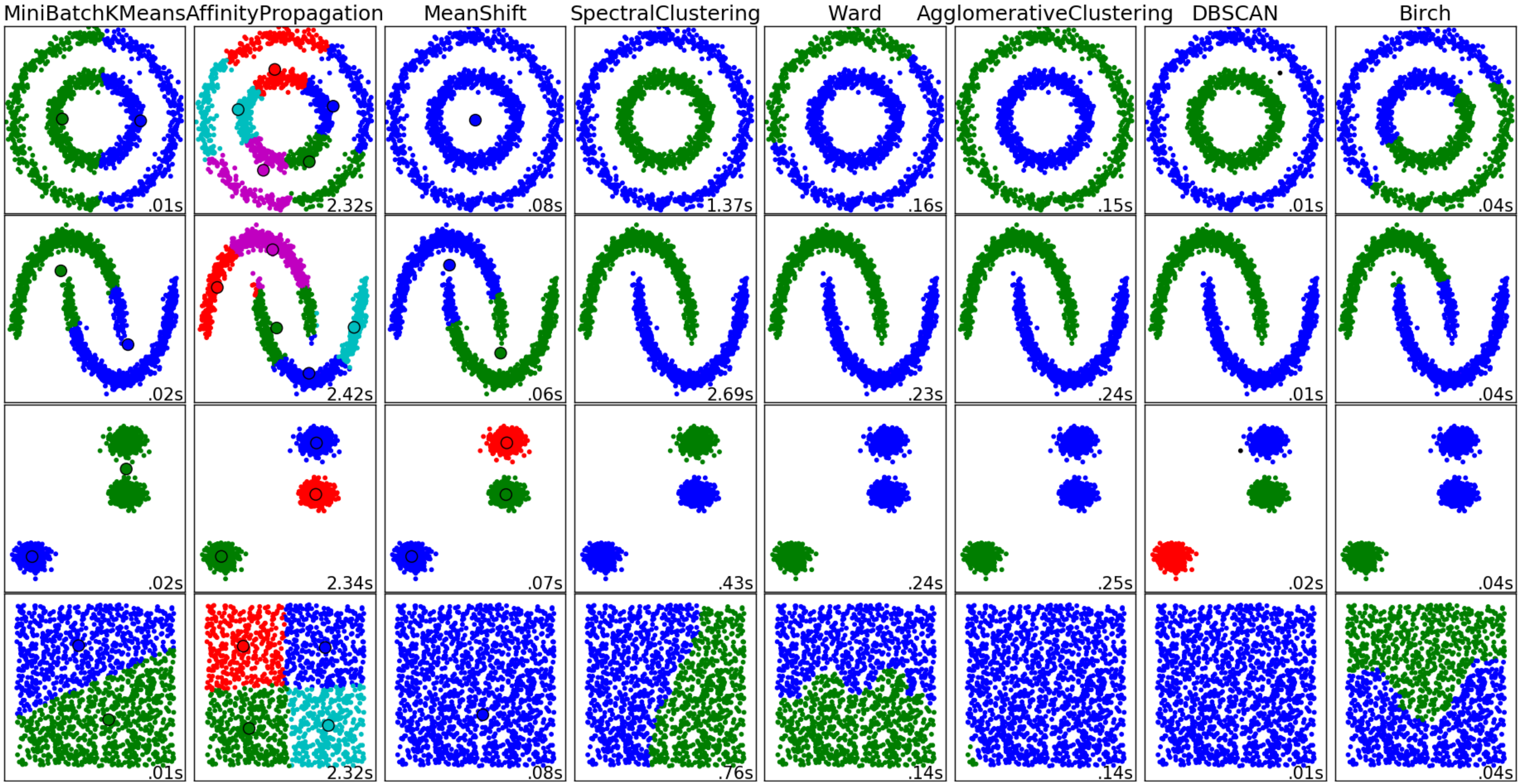
Our task:

- split document collection into groups (clusters) so that the documents in one group were similar to each other, whereas documents from different groups should be dissimilar

Problems?

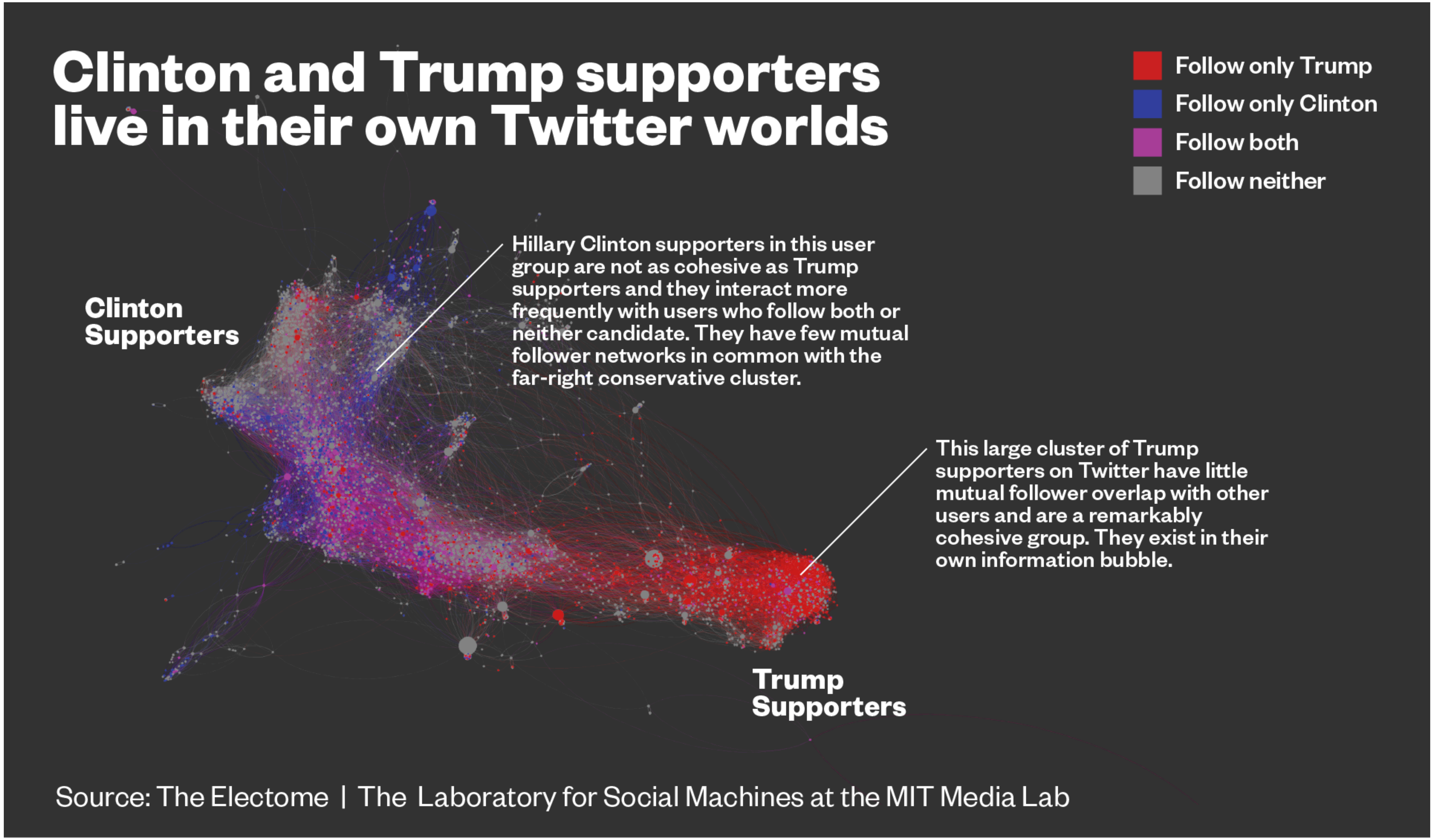
- This task implicitly introduce some *similarity* measure on documents space
- We know nothing about number of clusters K (usually this is the main parameter for the algorithm)
- It's hard to introduce and estimate clustering quality
- There are some problems with huge number of dimensions (curse of dimensionality)

Examples



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

Examples



https://news2-images.vice.com//uploads/2016/12/TwitterData1-01.png

Clustering quality evaluation

Ideas:

- Annotate and check **by hand**
- Apply to already labeled dataset
- extrinsic evaluation: estimate the **‘usefulness’ increase** for some application
- intrinsic evaluation: estimate some clustering ‘**quality index**’

Can you broke any of this methods?

Clustering quality evaluation

Ideas:

- Annotate and check **by hand**
  - and spend all your life labeling wikipedia
- Apply to already labeled dataset
  - so why should we cluster and not train classification model?
- extrinsic evaluation: estimate the **‘usefulness’ increase** for some application
  - but this way we don’t look at clusters quality
- intrinsic evaluation: estimate some clustering ‘**quality index**’
  - we look at one index when we optimize, then we look at a ‘better’ one… why not use the better one for optimization?

Clustering quality evaluation

…with labeled corpora;

Option 1:

Annotate each pair of objects in the test set:
with 1 if they are in the same cluster
or 0 if they are in different ones;
Then we do the same with our predictions

Thus we can evaluate quality the same way as we can do with classification:

1) we can compute **Accuracy** (how many pairs are correctly/incorrectly put into the same cluster)

2) or we can compute **Precision, Recall, F-measure**

Clustering quality evaluation

…with labeled corpora;

Option 2:

'How pure is each cluster': max share of some true cluster in each of the predicted ones:

$$\frac{1}{N}\sum_{m \in M} \max_{d \in D} |d \cap m|$$

D — 'true' clusters
M — predicted clusters

Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. Introduction to Information Retrieval. Cambridge University Press.

Clustering quality evaluation

Pairs:

- $n(n-1)/2$ pairs **is a lot**, the size of dataset (n) can’t be large due to that

Purity:

- large number of clusters delivers large purity value!
- if every element is a cluster, purity = 1.0

A lot of clustering evaluation indices were invented, each of them is ugly in its own way :)

For a good start one may take a look at the Wikipedia article

Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. Introduction to Information Retrieval. Cambridge University Press.

Sorts of ~~sh~~clustering

We can compare clustering algorithms in terms of:

- computational complexity
- do they build flat or hierarchical clustering?
- can the shape of clustering be arbitrary?
    - if not is it symmetrical?
    - can clusters be of dierent size?
    - clusters vary in density of contained objects?
- robustness to outliers
- Hard or soft cluster assignment?

We can compare clustering algorithms in terms of:

- Distance matrix and/or measure function?
- Can we scale it?
- Which intuition these algorithms are based on?
    - graph-based? distribution?
    - density? hierarchy?
- Can we interpret our result?

Clustering algorithms:

- Representative-based clustering
- Probabilistic clustering
- Hierarchical clustering
- Density-based clistering

Note 1: We already know how to represent text as a vector; all methods we will discuss are of course applicable in other domains and for other data types

Note 2: NB! The algorithms we are going to discuss have numerous modifications and implementations can differ greatly. Take care when carrying out experiments and training models for production environment!

K-means:

1. **Applicability:** vectors
   Goal: minimize the sum of squares of distances from centroid of each cluster to each element of the cluster

1. Set the number of clusters .

2. Choose documents at random -- clusters centroids.

3. Include the remaining documents into the closest cluster.

4. Compute new cluster centroids as a mean vector in the cluster.

5. Repeat steps 3-4, until

    1. Centroids stop to change?

    2. The partition of the dataset stops to change?

    3. Reach num. iteration threshold

number of clusters

number of cases

centroid for cluster *j*

case *i*

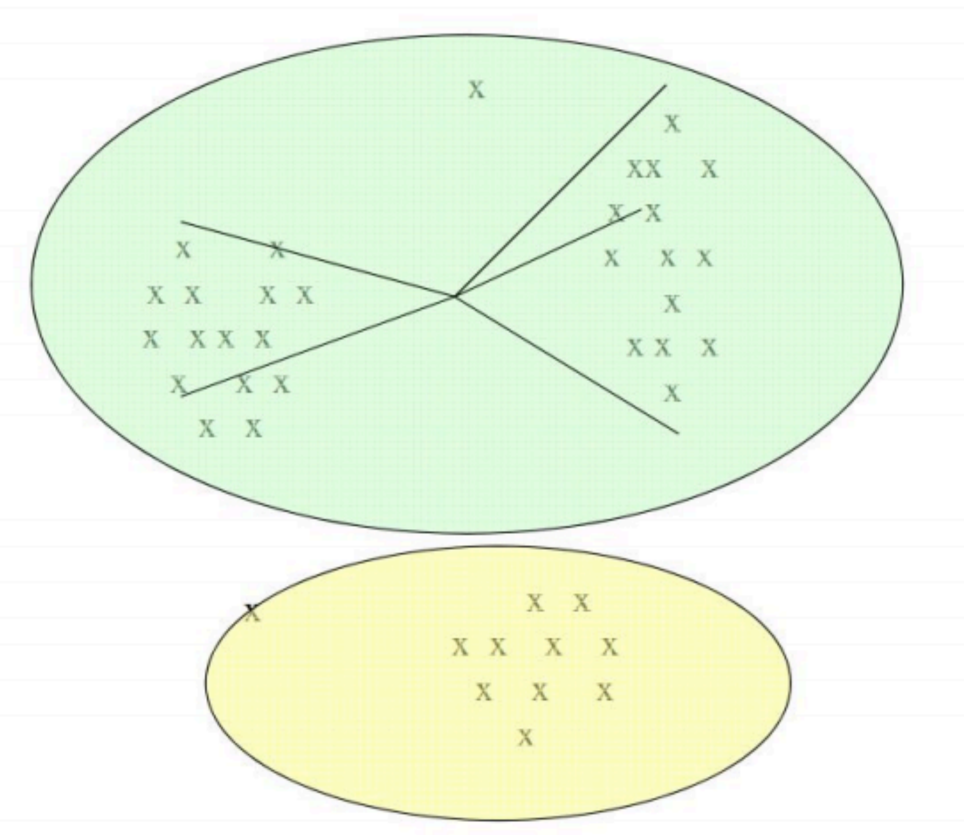
objective function $\leftarrow J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$

K-means:

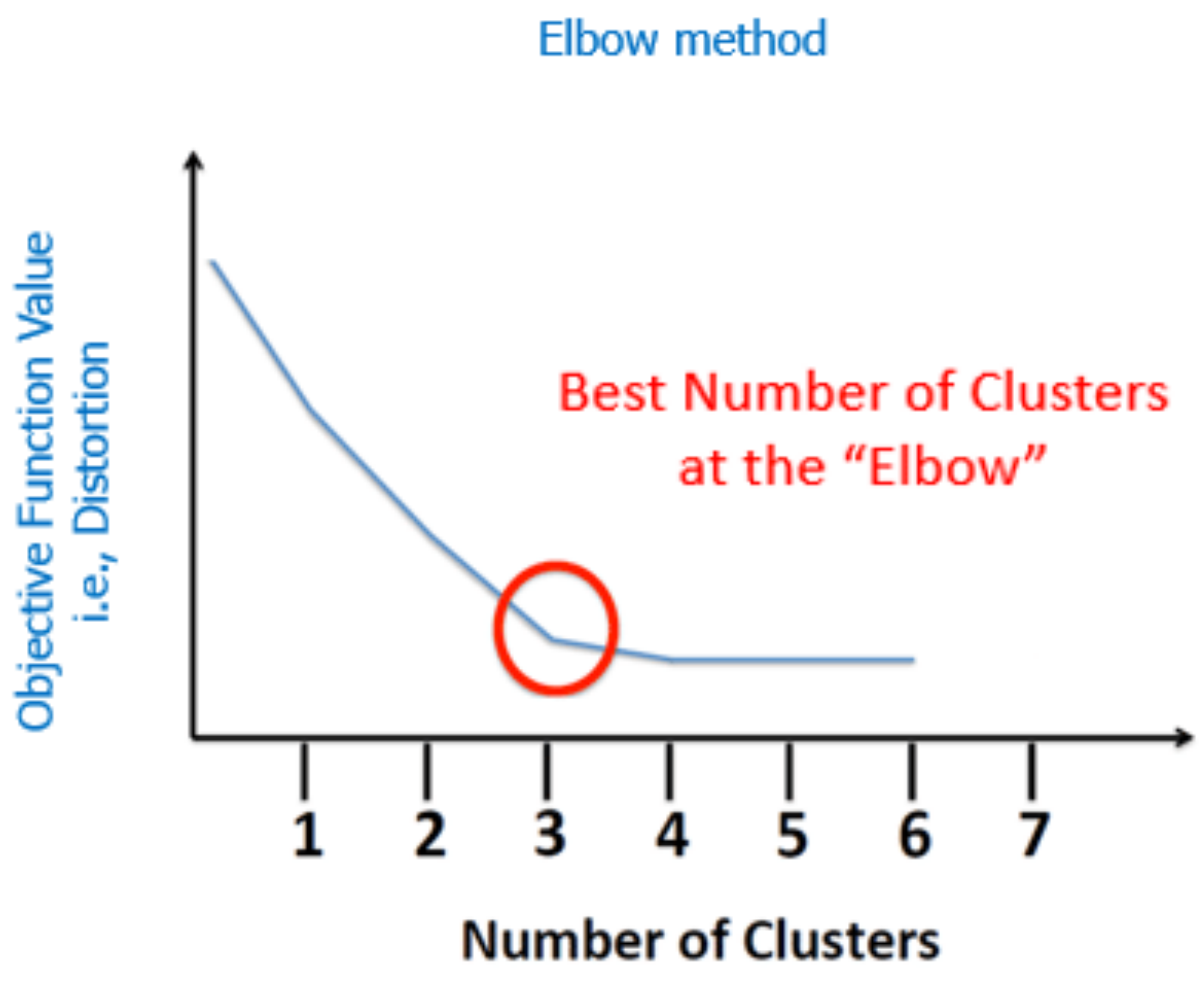
1. Gradually increase **K**

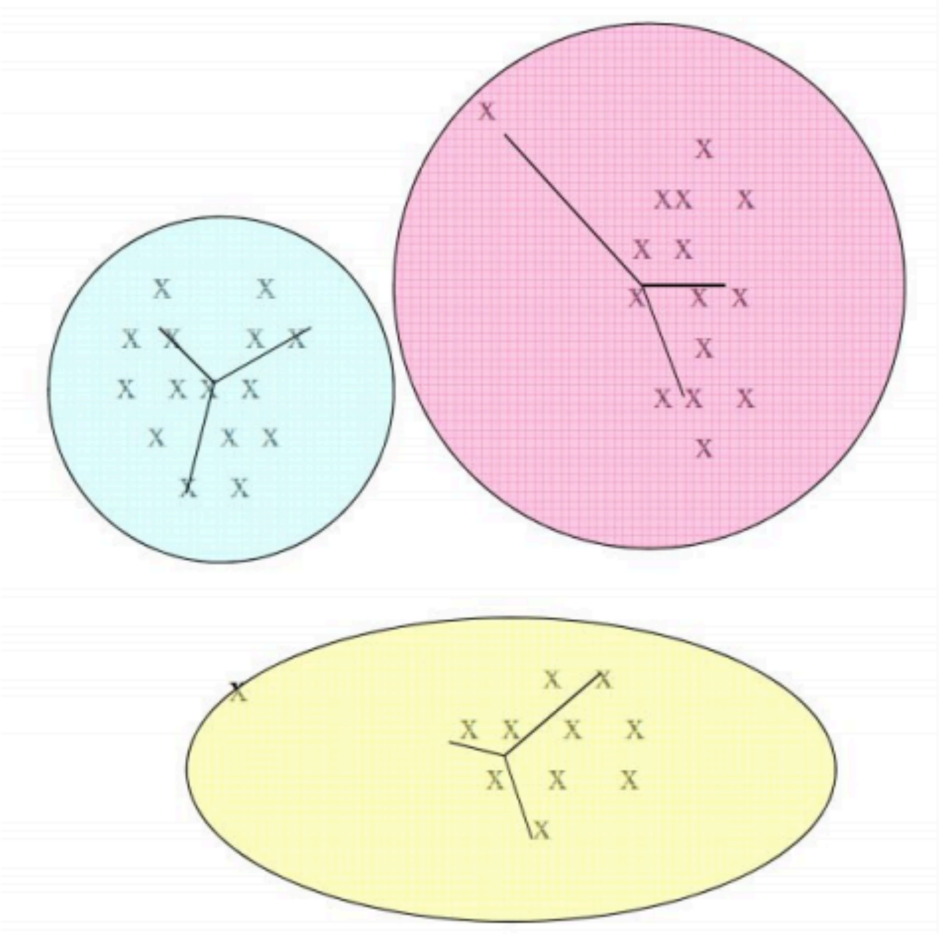
2. Look at the average distance to centroid

3. At some value of K it will stop to drop fast; this is the recommended K value



NOT BRO





BRO

K-means:

Restrictions:

- Sometimes it’s hard to find «nearest neighbours»
- can’t apply to the domain where there is no such thing as an ‘average object’
- is prone to ball-like clusters detection
- **always** finds K clusters
- sometimes heavily depends on initial centroids candidates choice
- However, there are quite a few modifications useful in real life

Probabilistic: EM

Definitions:

- Our space X: $X = R^n$, our objects $x_i = (f_1(x_i), f_2(x_i) \ldots f_k(x_i)) \in X$
- The are **center points** for each cluster: $\mu_y = (\mu_{y1} \, . \, . \, \mu_{yn})$
- Let $\omega_y \mid \sum_{y \in Y} \omega_y = 1$, **prior probability** for each cluster
- **Assume** that Xl i.i.d from mixture of distributions: $p(x) = \sum_{y \in Y} \omega_y p(x)$
- Assume that all distributions are Gaussian;

$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$$

Probabilistic: EM

Definitions:

1. There are start points for $\omega_y, \mu_y, \Sigma_y, \forall y \in Y$
2. E(expectation)-step for y, i=1..l $$g_{iy} \leftarrow P(y \mid x_i) = \frac{\omega_i p_y(x_i)}{\sum_{z \in Z} \omega_z p_z(x)}$$
3. M(maximisation)-step

$$\omega_y \leftarrow \frac{1}{l} \sum_{i=1}^{l} g_{iy} \qquad \mu_{yi} = \frac{1}{l * \omega_y} \sum_{i=1}^{l} g_{iy} f_j(x_i) \qquad \sigma_{yj} \leftarrow \frac{1}{l * w_y} \sum_{i=1}^{l} g_{iy} (f_j(x_i) - \mu_{yj})^2$$

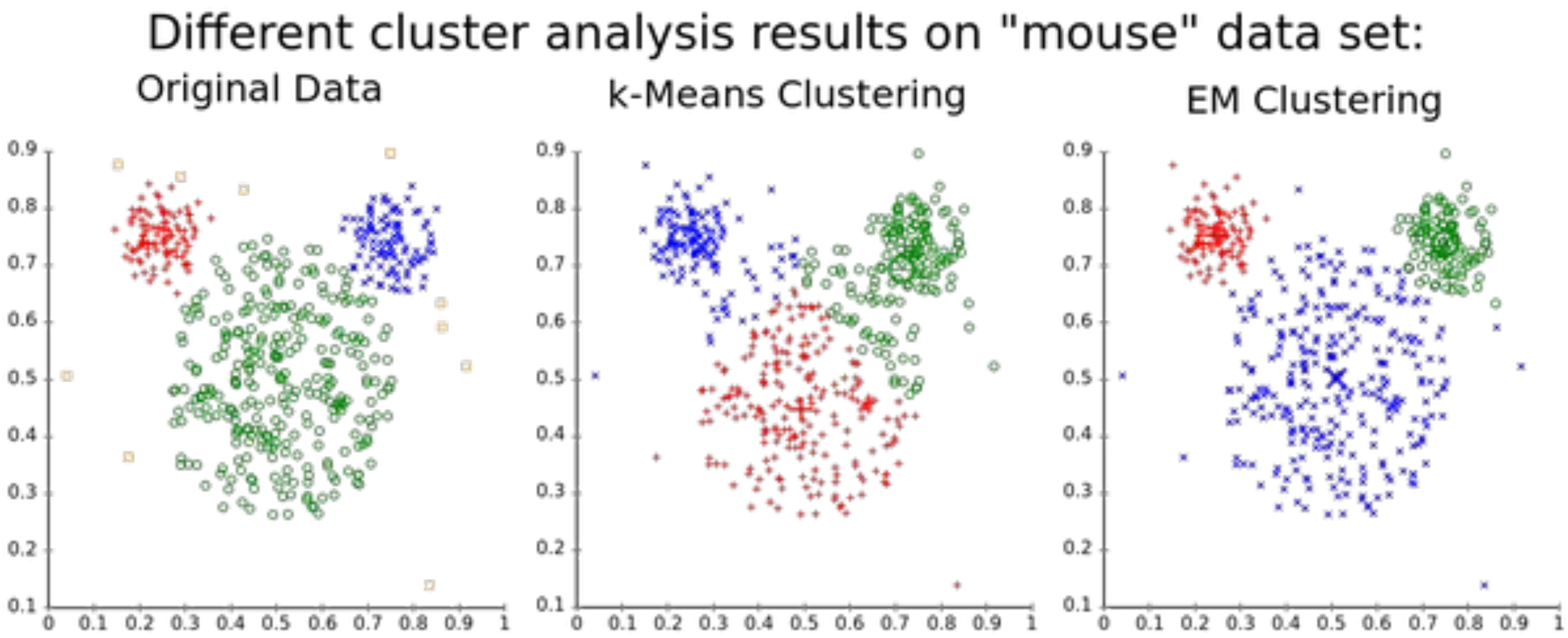
4. $y_i = argmax_{y \in Y} g_{iy}$
5. Repeat until y_i wouldn't fix;

Probabilistic: EM

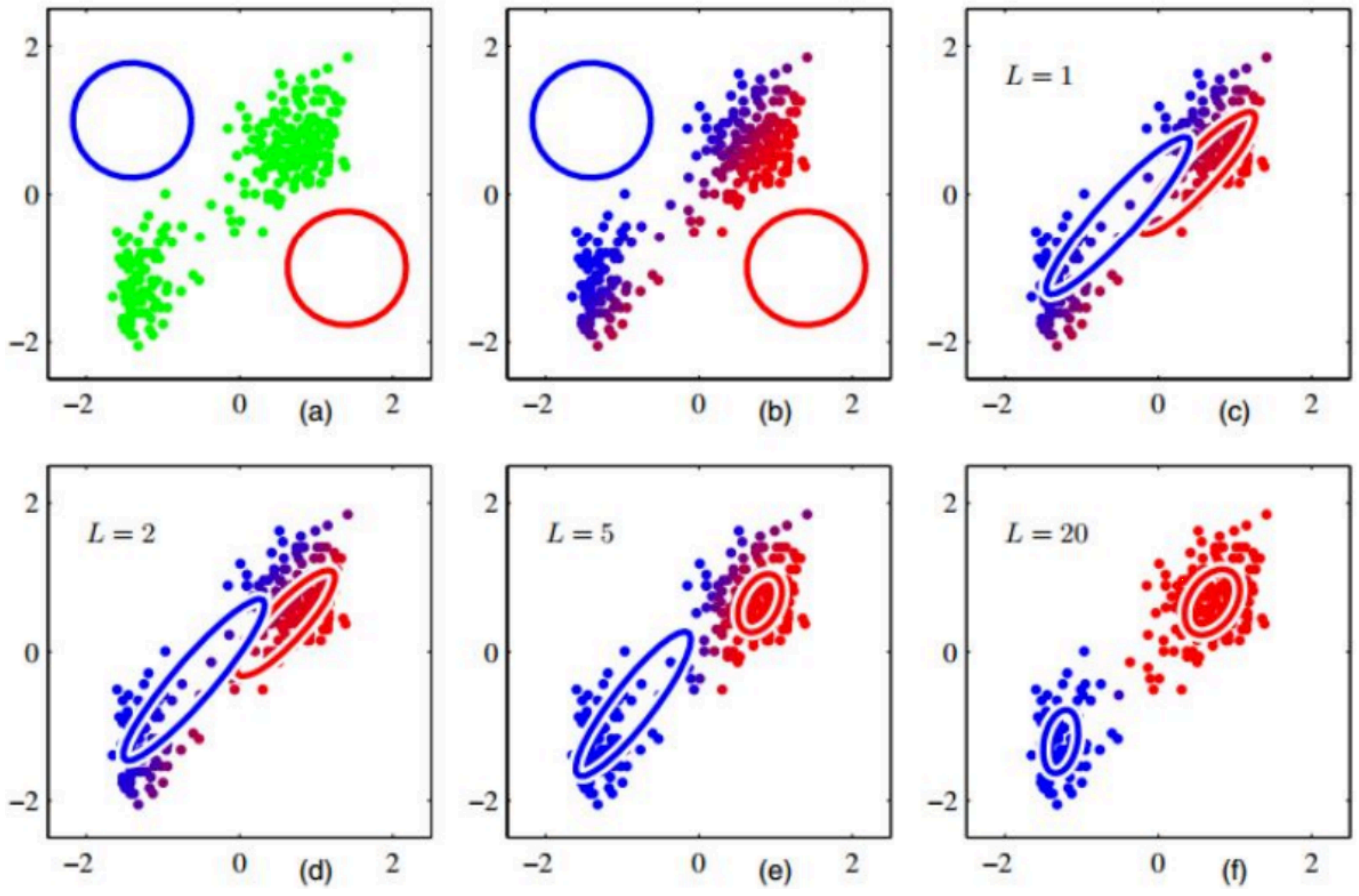
This way we'll have

1. 'fuzzy clustering' (soft clustering): clusters probabilities for each document,

2. possibility to restrict/give some hint on the possible shapes (distribution family) of the cluster



Probabilistic: EM



Hierarchical clustering

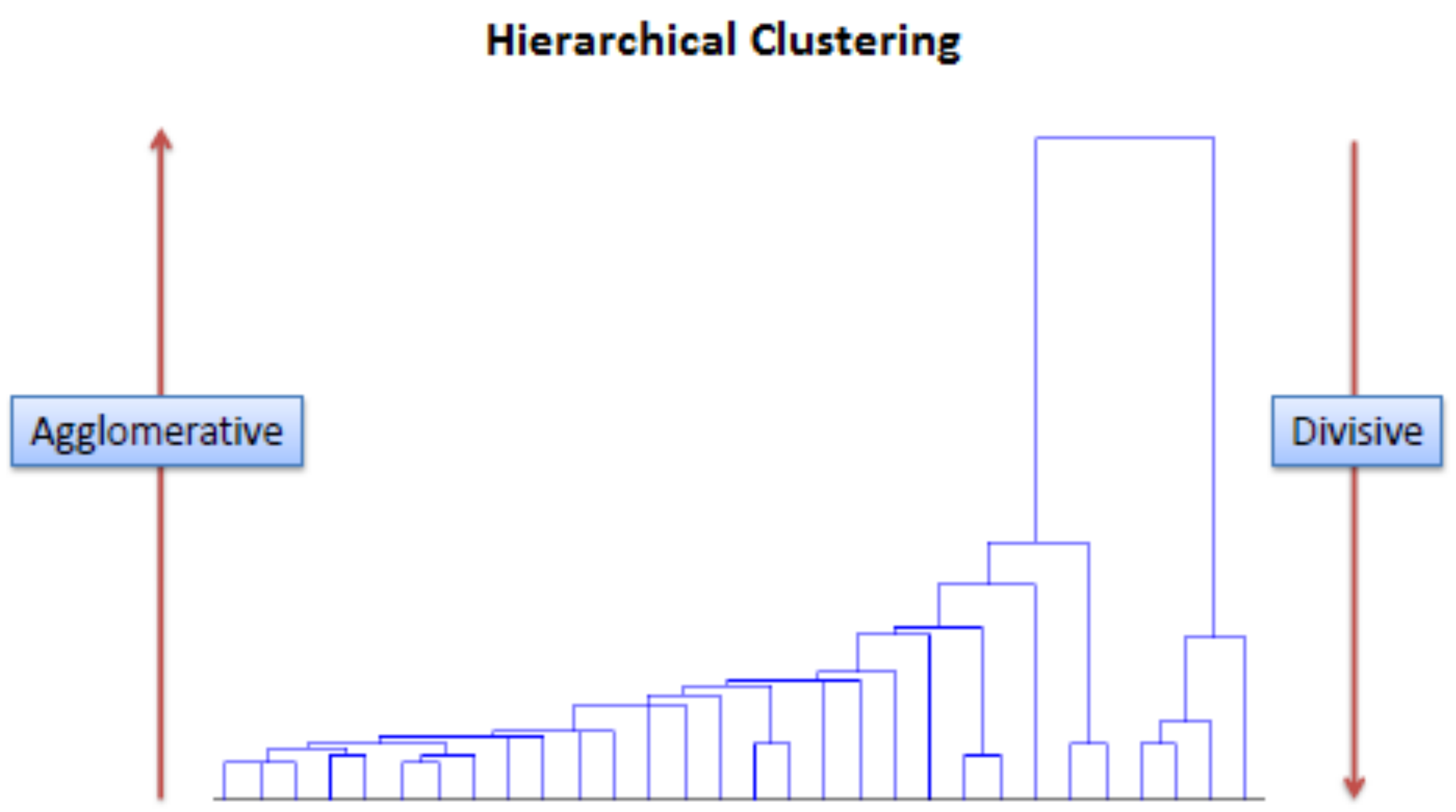
Two methods types

1. agglomerative

2. divisive

Each hierarchical method builds a **dendrogram** for further pruning = clusters selection.

Dendrogram shows measures of closeness between objects and sets of objects



Divisive approach:

Example: let’s choose some flat clustering method A (e.g. KMeans)

1. Initially — just 1 cluster containing all elements, root of the tree

2. Apply method A to the leaf of the tree (chosen by some rule).

3. Add resulting clusters as leaves (x being their ‘parent’).

4. Repeat 2-3, until the cardinality of each ‘leaf’ is equal to 1.

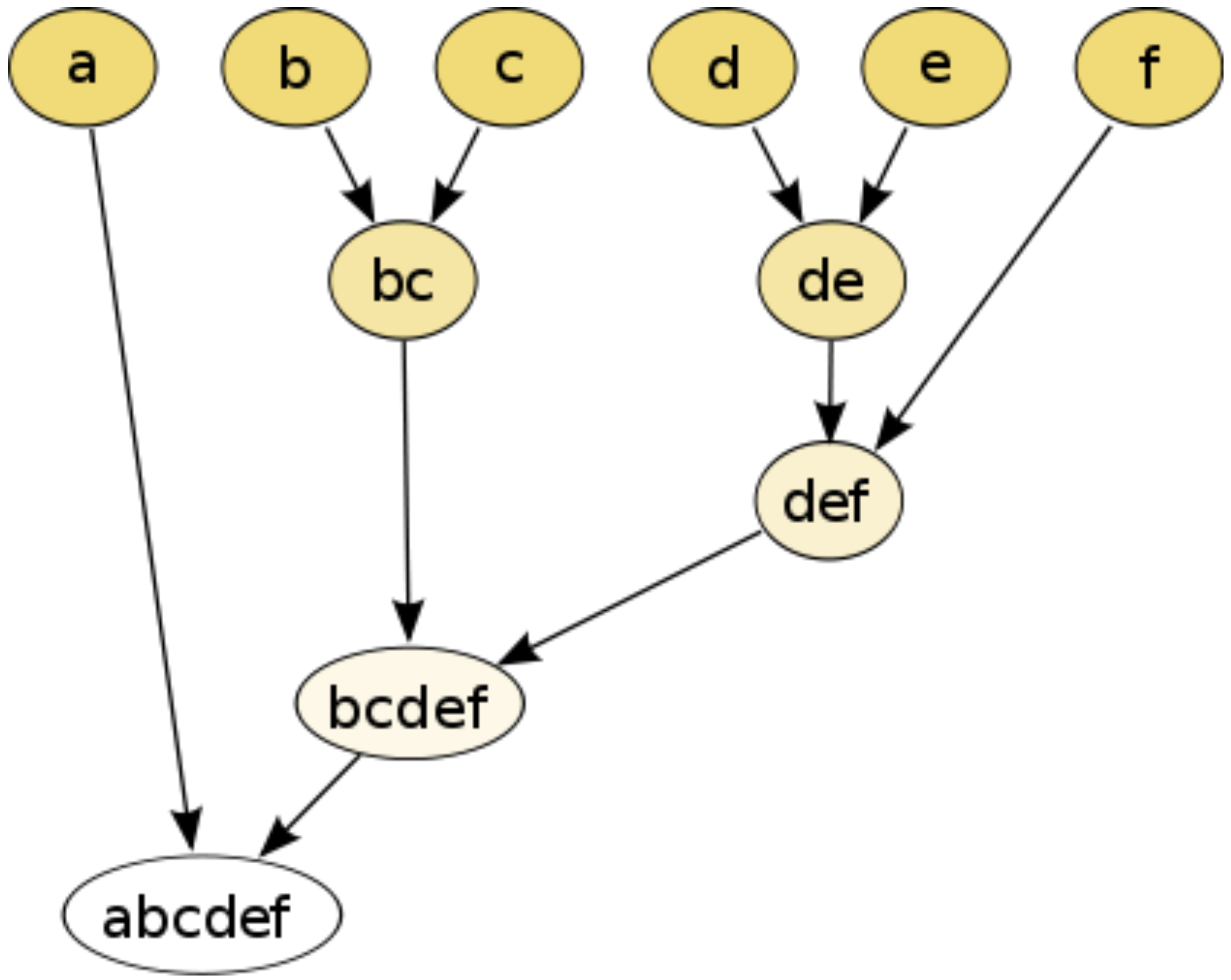
Divisive approach:

A more popular and intuitive approach

1. Initially each element is a cluster of size 1

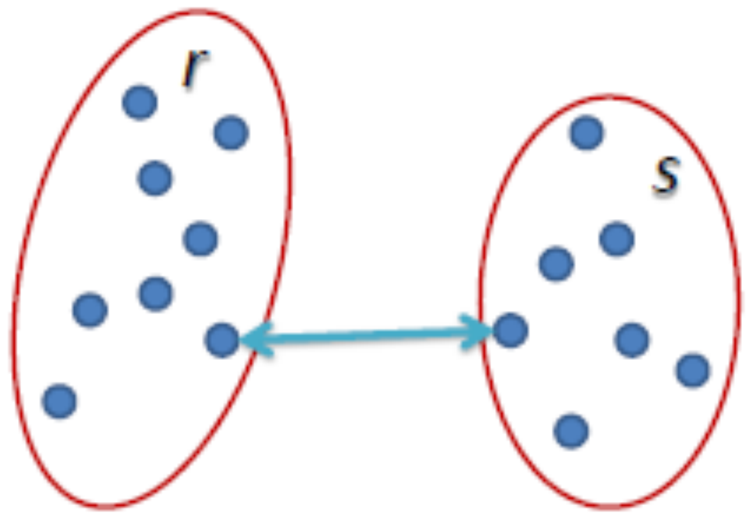
2. Using a certain rule, we choose two closest clusters and merge them into one

3. **Problems?**



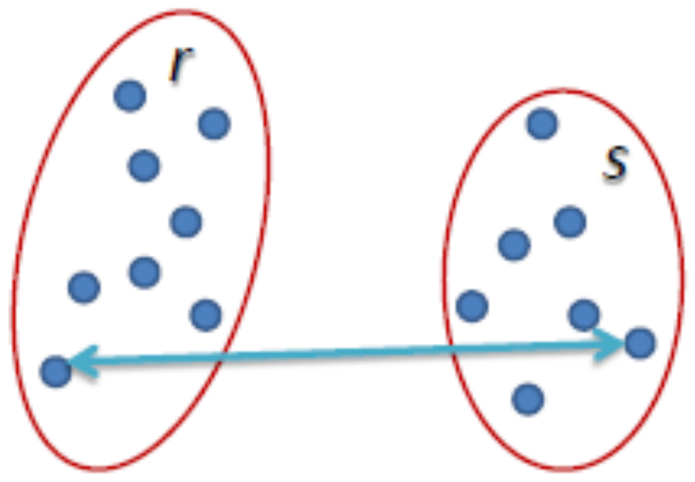
How to compute distance:

- **Single linkage:** the distance between two clusters is defined as the *shortest* distance between two points in each cluster.



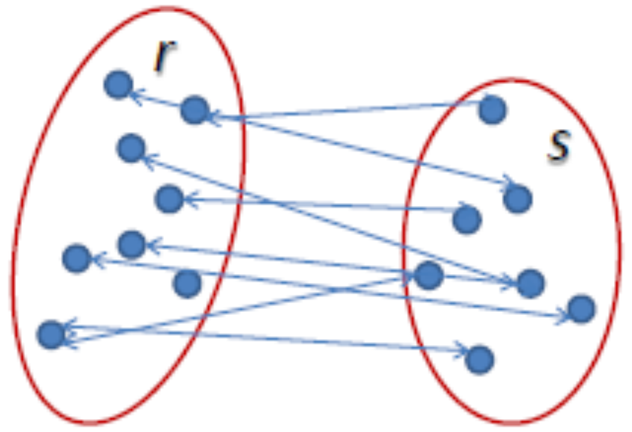
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

- **Complete Linkage:** the distance between two clusters is defined as the *longest* distance between two points in each cluster



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

- **Average Linkage:** the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

- **Ward linkage:** difference between sum(sqr(distances)) inside the possible clusters union and sum(sqr(distances)) inside each of the two clusters separately

Tips and tricks:

- How to choose num clusters?
  //https://stats.stackexchange.com/questions/3685/where-to-cut-a-dendrogram

- Use Ward, practically it better

- Read literature, it’s interesting

Density-based: DBSCAN

Density-Based Spatial Clustering of Applications with Noise the most cited clustering algorithm

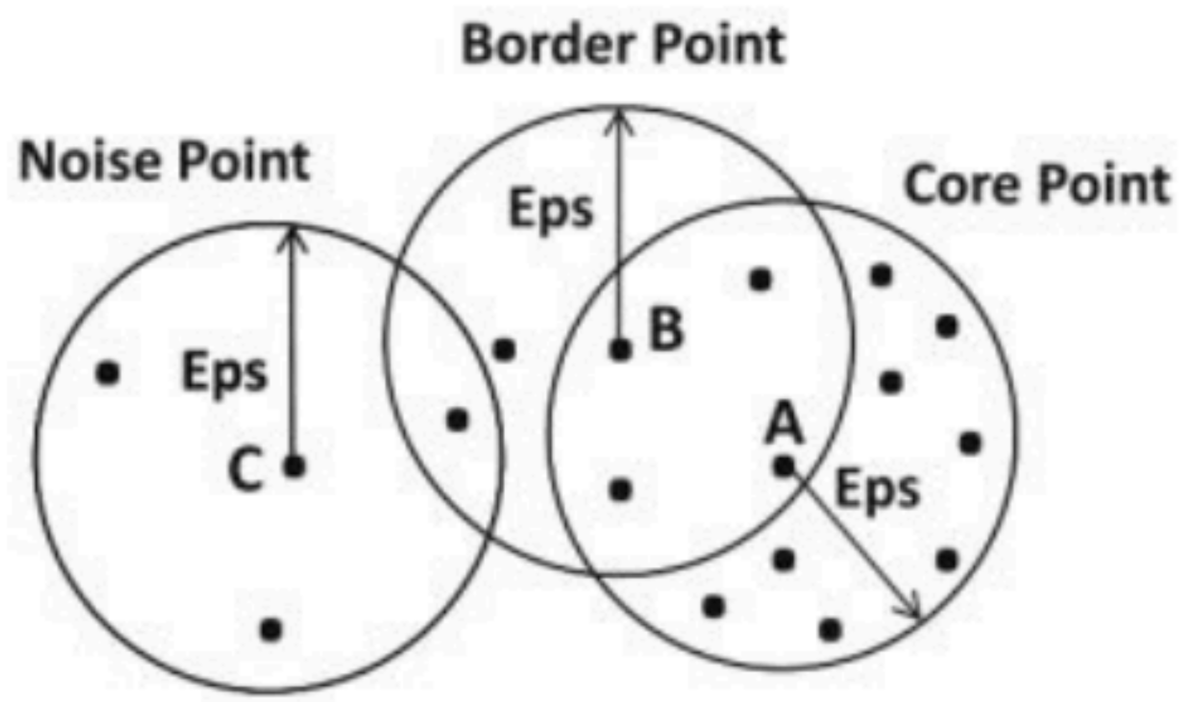
Set ε (distance) and k (an integer)

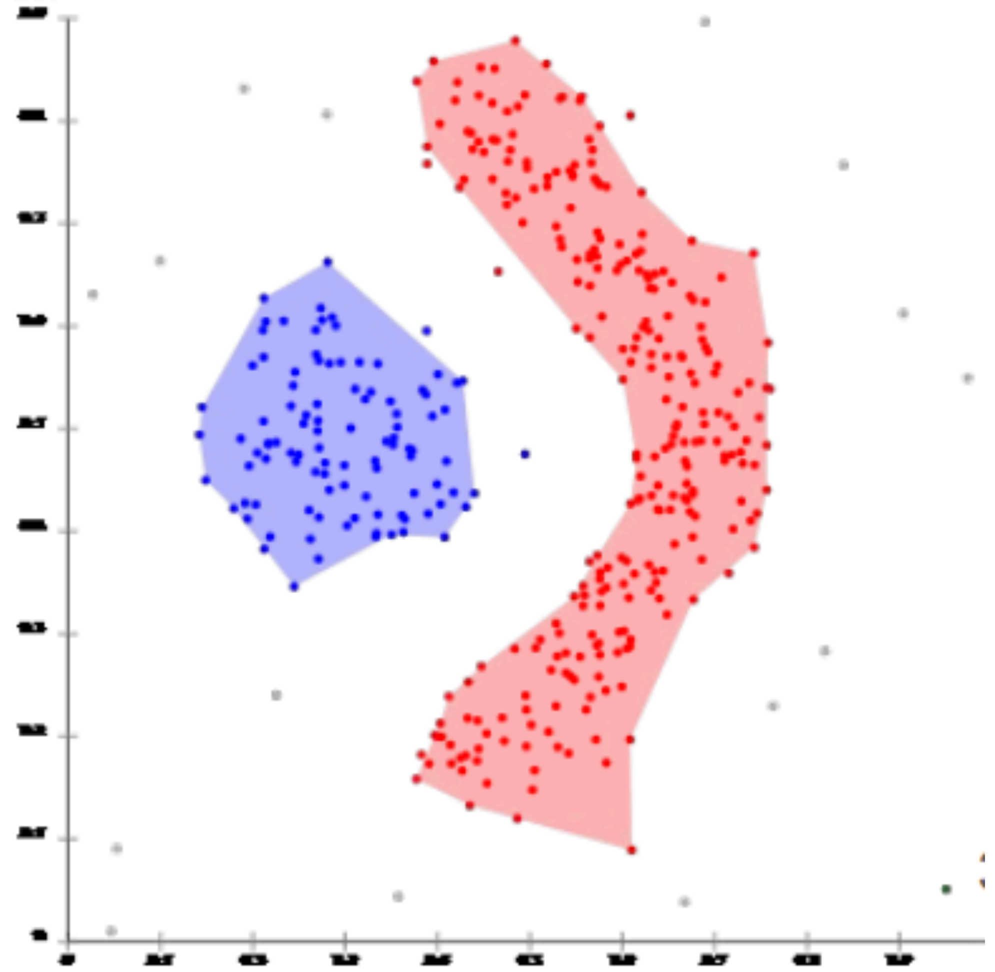
Elements are split into 3 types:

1. Core points: elements having at least k other elements in their ε-neighbourhood

2. Border points: elements having at least b element in their ε-neighbourhood

3. Noise points: other elements

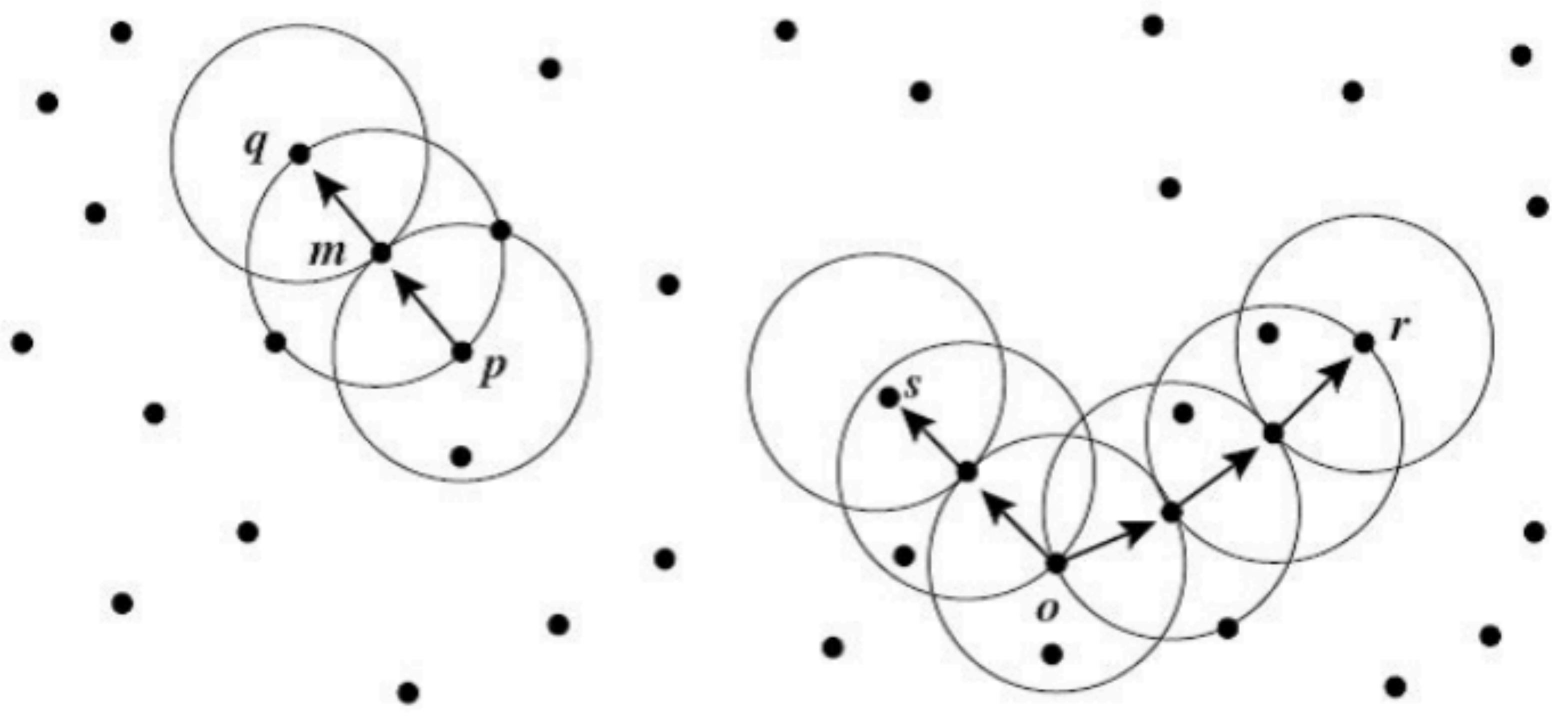


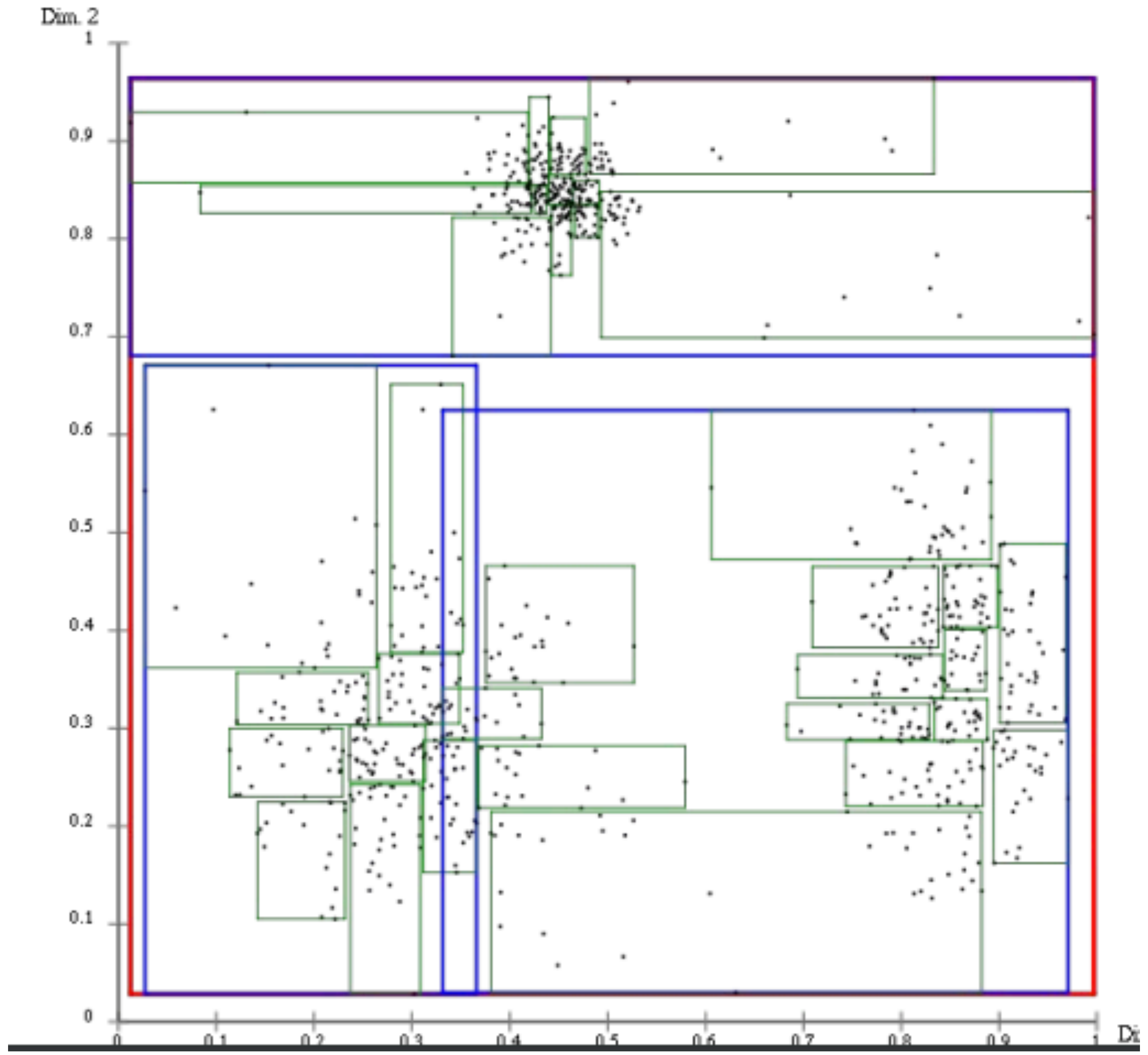


Density-based: DBSCAN

Algorithm

1) Mark elements with those three types
2) Build a graph, connecting core points that are no farther than ε from each other
3) Determine connected components
4) Link every border point to the closest connected component

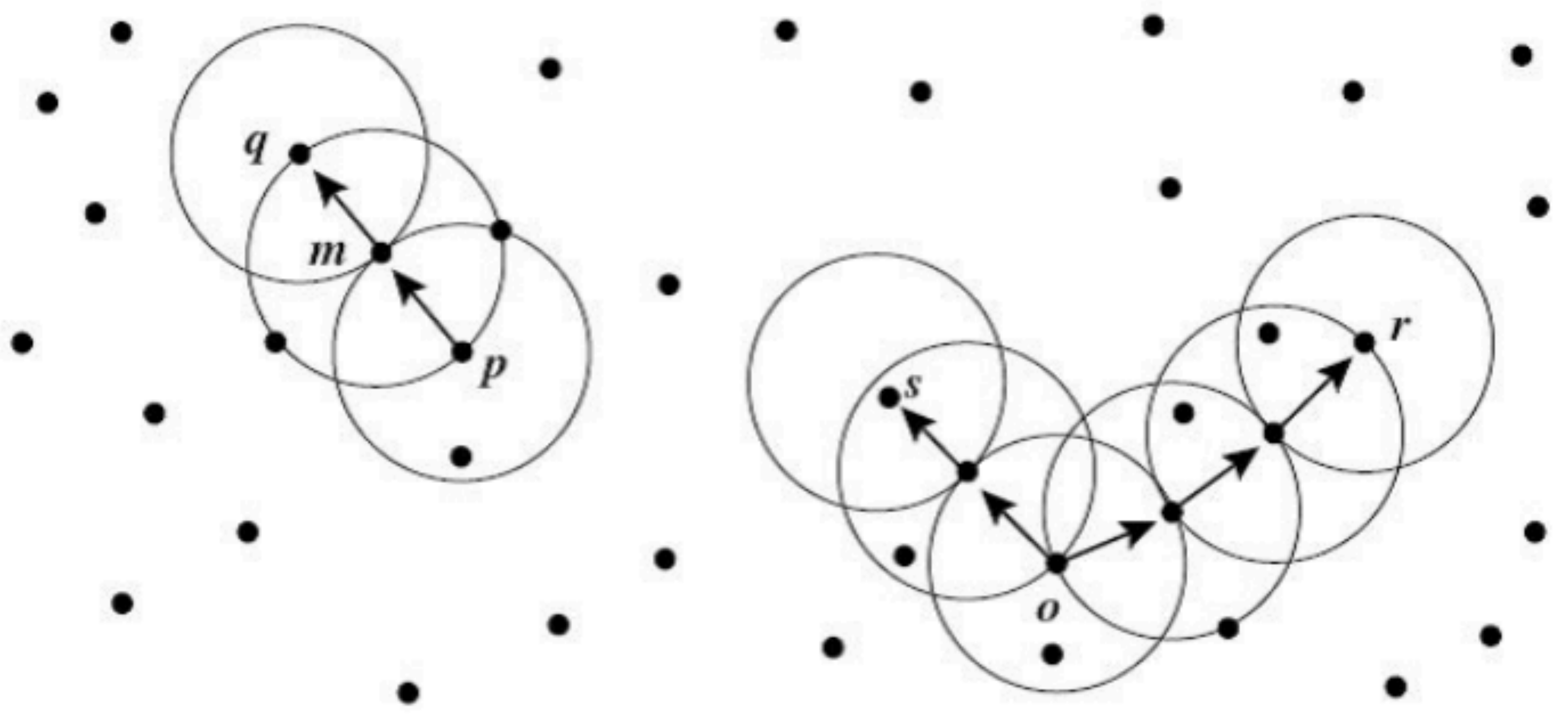


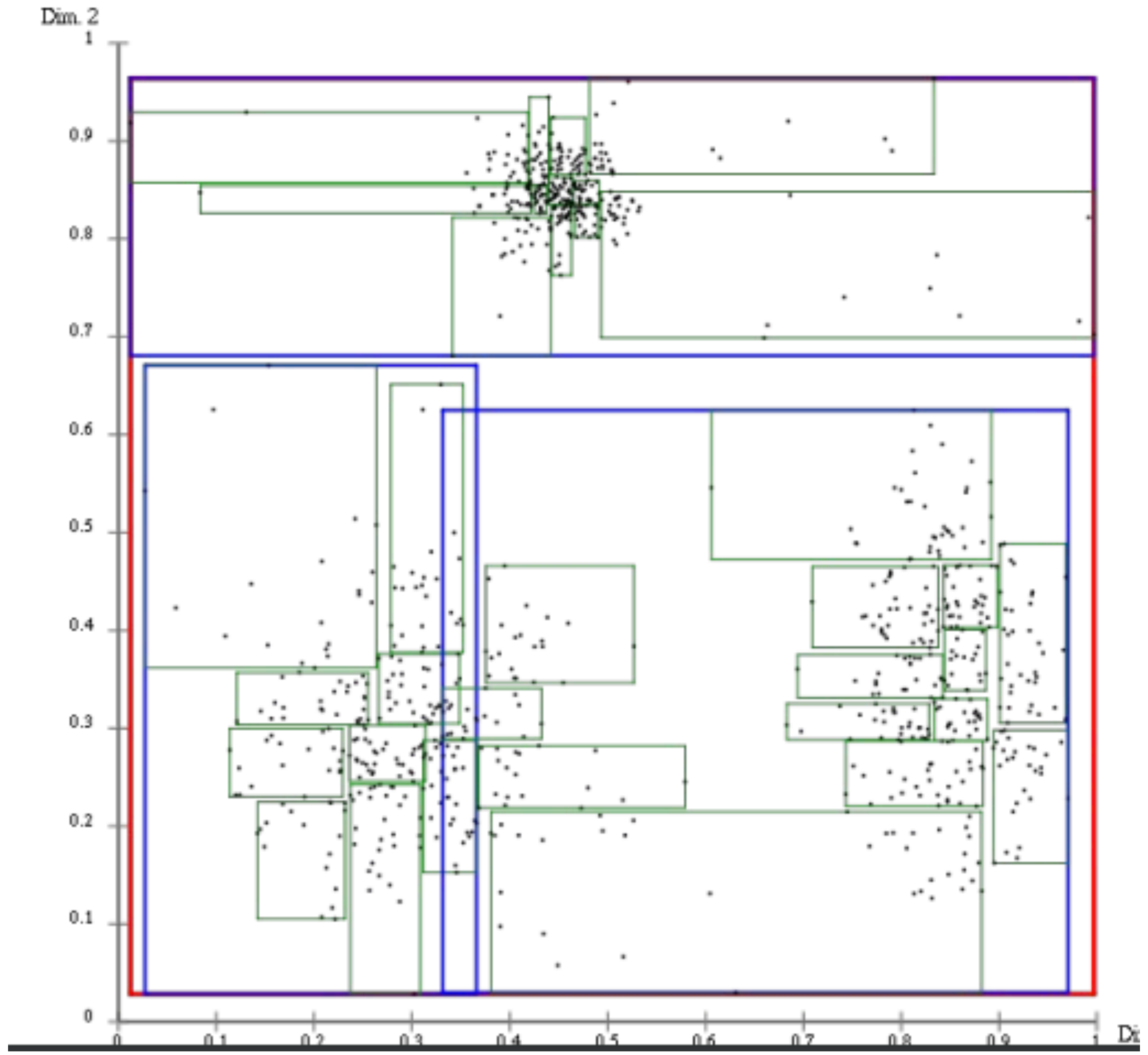


Density-based: DBSCAN

Discussion:
+ determines the number of clusters
+ robust to outliers and noise
+ detects clusters of arbitrary shape and form

- is slow
- fails at determining clusters of different density
- tuning parameters may be a challenge





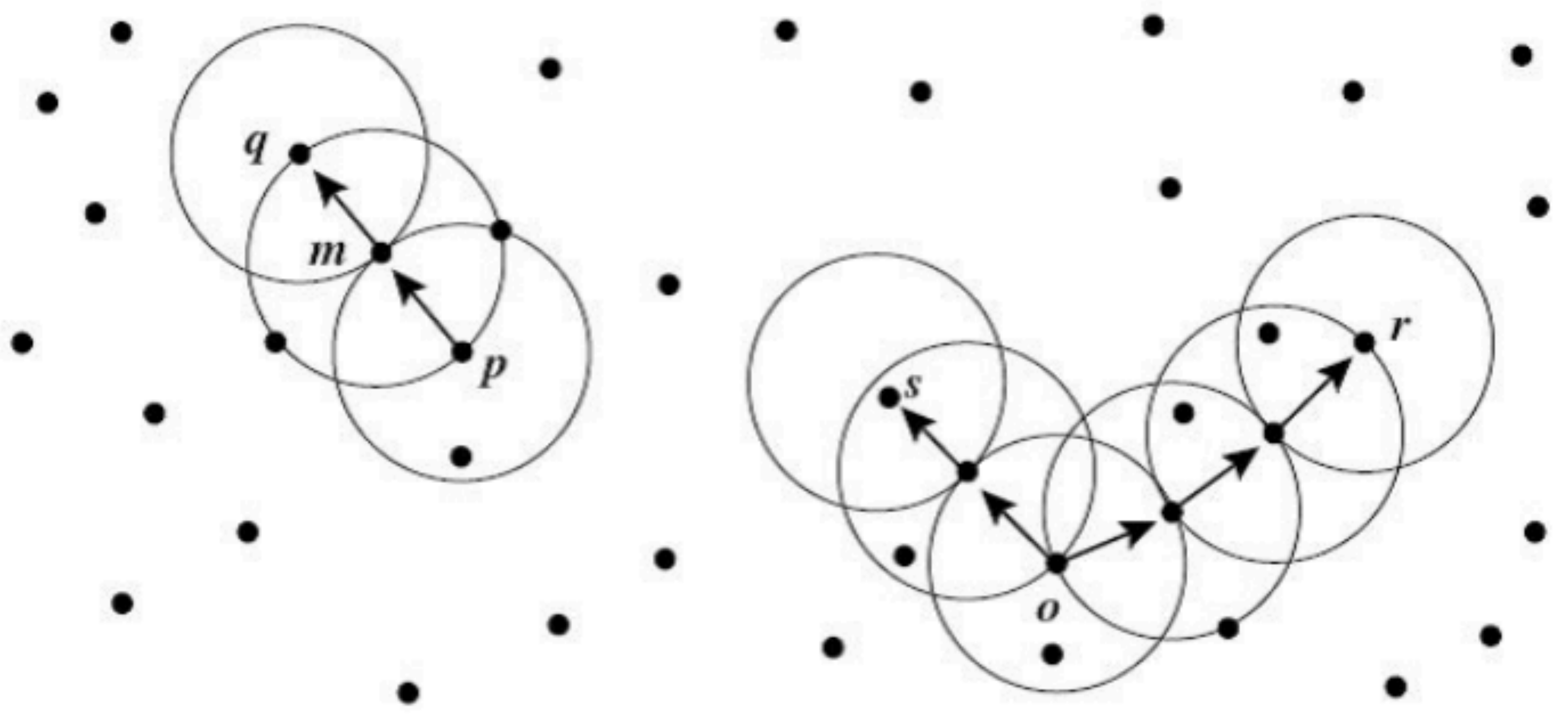
Other guys:

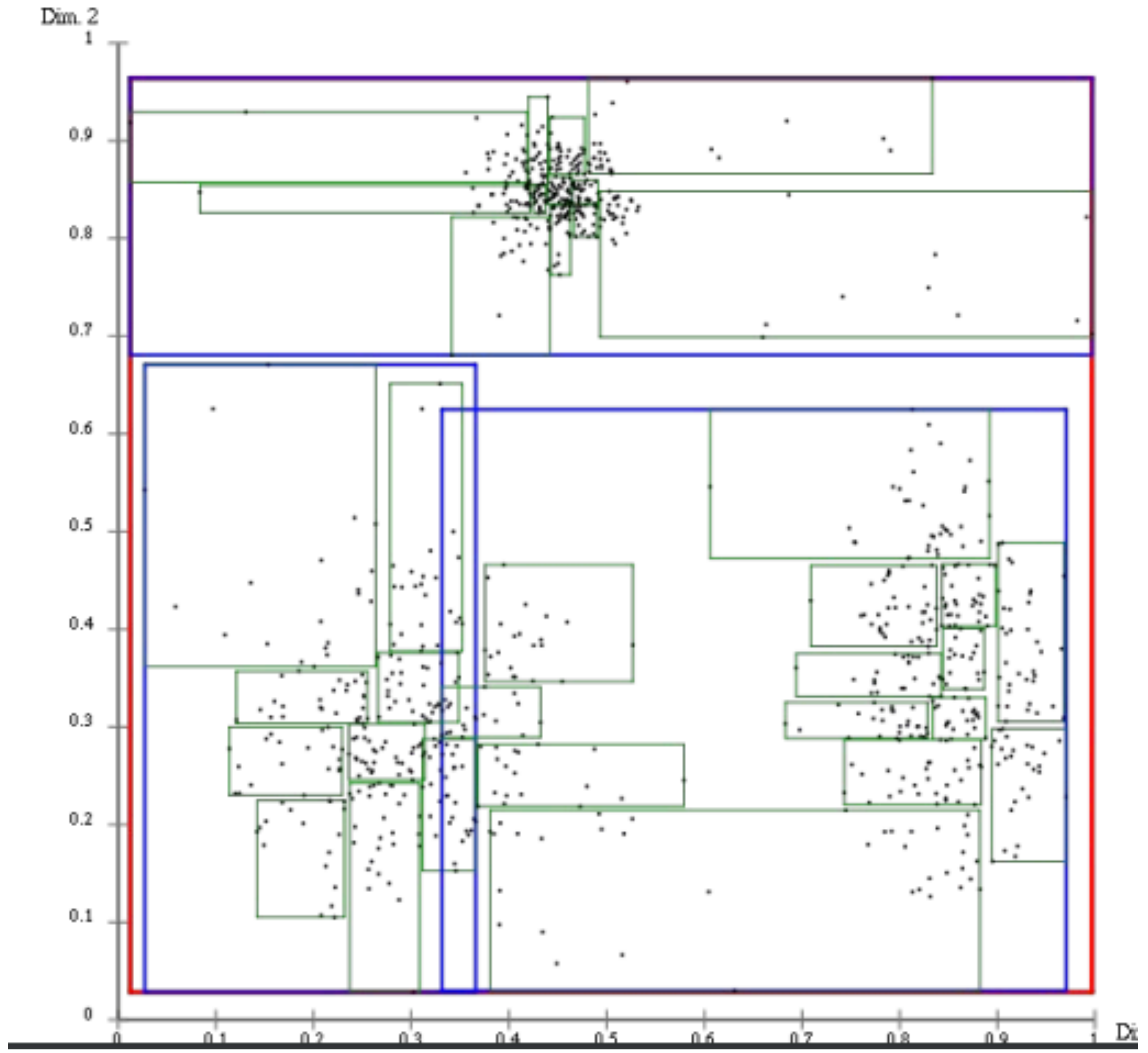
CURE (Clustering Using REpresentatives: hybrid of hierarchical and flat clustering; keeping several representative data points for each cluster)

BIRCH (hierarchical, designed for large datasets, we build a tree of subclusters, preserving certain constraints, SIGMOD 10y test time award)

OPTICS and other DBSCAN modifications (DBSCAN taking density into account)

Community and graphs





Tools & Refs

Mainstream instruments allowing to try different approaches
- scipy.cluster
- sklearn.cluster
- custom libraries, e.g. pyclustering

1. CSC 2014 course
2. Mining of Massive Datasets Jure Leskovec, Anand Rajaraman, Jeff Ullman
3. Scikit-learn docs
4. MSU course slides and other materials MSU course slides and other materials
5. EM-algorithm @ ml.ru
6. Wikipedia (English)