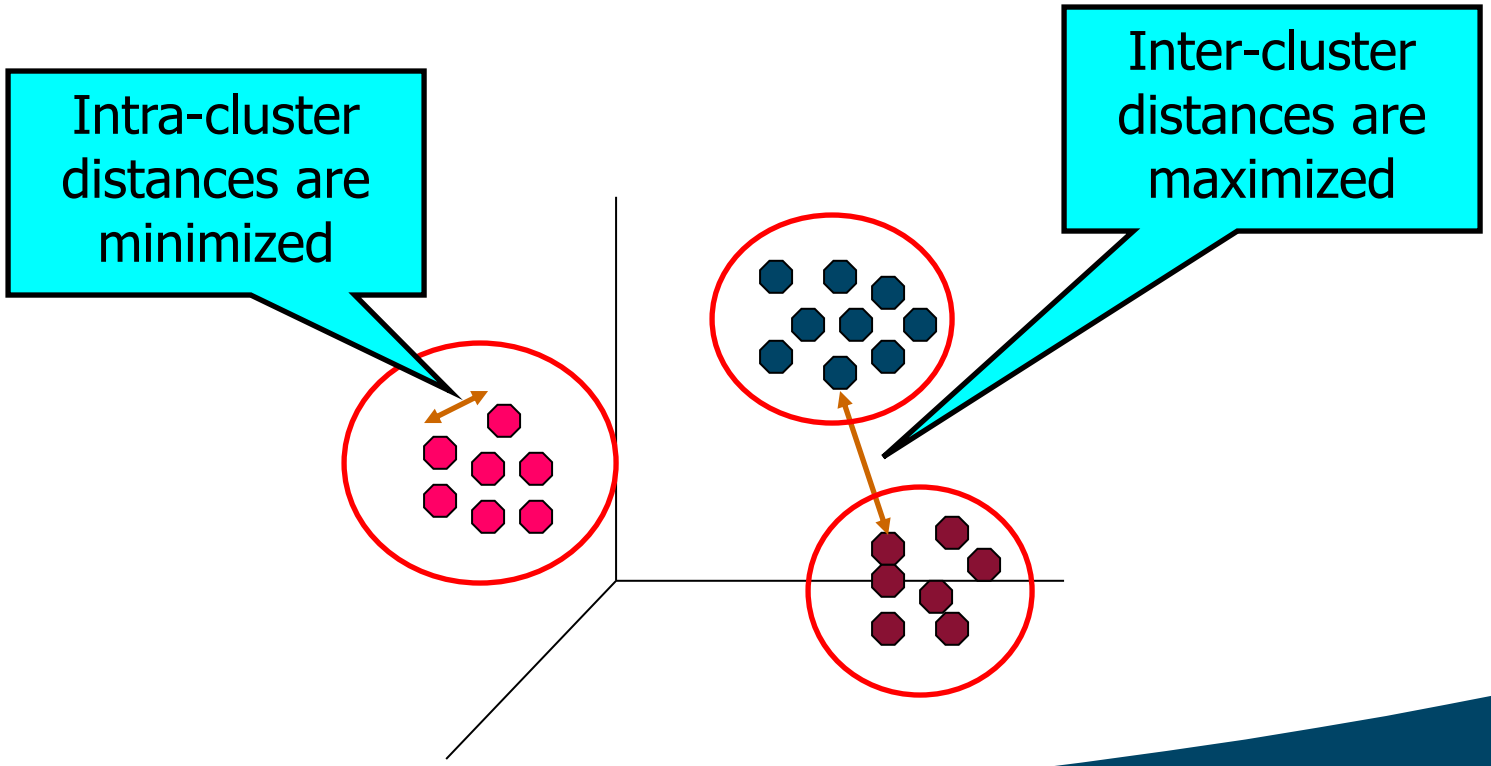
Data Mining: Clustering

Toon Calders toon.calders@uantwerpen.be
Bart Goethals bart.goethals@uantwerpen.be

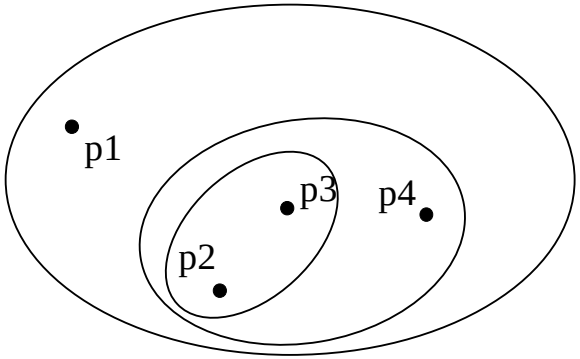
University of Antwerp

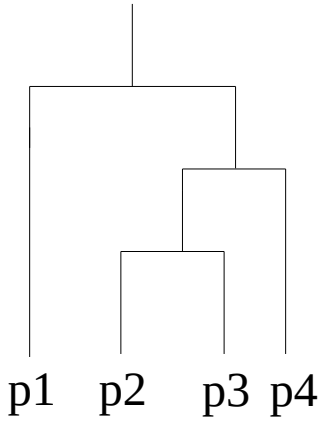
What is Cluster Analysis?

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups

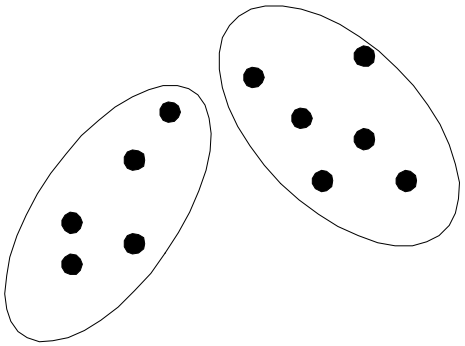


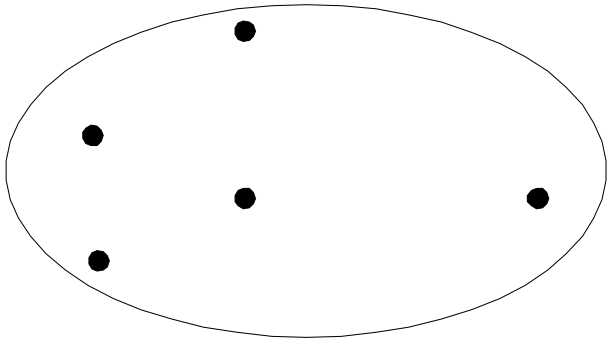
Types of Clustering





Hierarchical Clustering





Partitional Clustering

Objective Function

Clusters Defined by an Objective Function

- Finds clusters that minimize or maximize an objective function.
- These problems very quickly become NP Hard
- Can have global or local objectives.
  - Hierarchical clustering algorithms typically have local objectives
  - Partitional algorithms typically have global objectives

- A variation of the global objective function approach is to fit the data to a parameterized model.
  - Parameters for the model are determined from the data.
  - Mixture models assume that the data is a ‘mixture' of a number of statistical distributions.

NP-Completeness of Cannot-Link constraints

- Popular type of constraints in clustering:
  - must-link(a,b): a and b have to be in the same cluster
  - cannot-link(a,b): a and b must be in a different cluster
- Can be considered *semi-supervised* learning

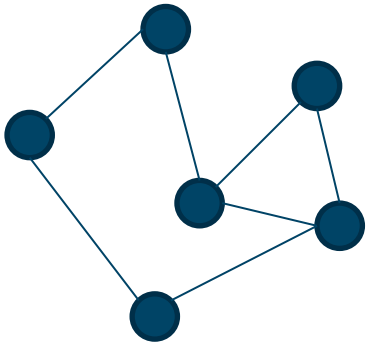
Following problem CLUS is NP-hard:

Given a set of cannot-link constraints and a parameter k, does there exist a clustering satisfying the cannot-link constraints that has at most k clusters?

Proof of NP-hardness

Reduction from 3COL:

Given a graph, can we color the vertices using at most 3 colors such that no two adjecent vertices have the same color?



Every vertex becomes a data point, every edge a cannot-link constraint; k=3

If we can find a clustering C1, C2, C3: color all points in C1 blue, all in C2 red, and all-in C3 green.
Since each vertex is a cannot-link constraint there are no edges between nodes of the same cluster.

Other variants that are NP-hard

Can we cover D with k balls of size R?

Consider a cluster. If we use the cluster center c as the representative of the cluster, then for each point x in the cluster we make an "error" d(c,x).

Given a dataset, can we find a clustering into k clusters such that the total/average/maximal (sum of squared) error is E?

For reasonable distance measures these problems are provably hard.

Therefore mainly heuristic solutions

Outline

Partitional Clustering

- Distance-based
  - K-means, K-medoids, Bisecting K-means
- Density-based
  - DBSCAN
- Expectation-Maximization

Hierarchical Clustering

Cluster validity

K-means Clustering

Partitional clustering approach

Each cluster is associated with a centroid (center point)

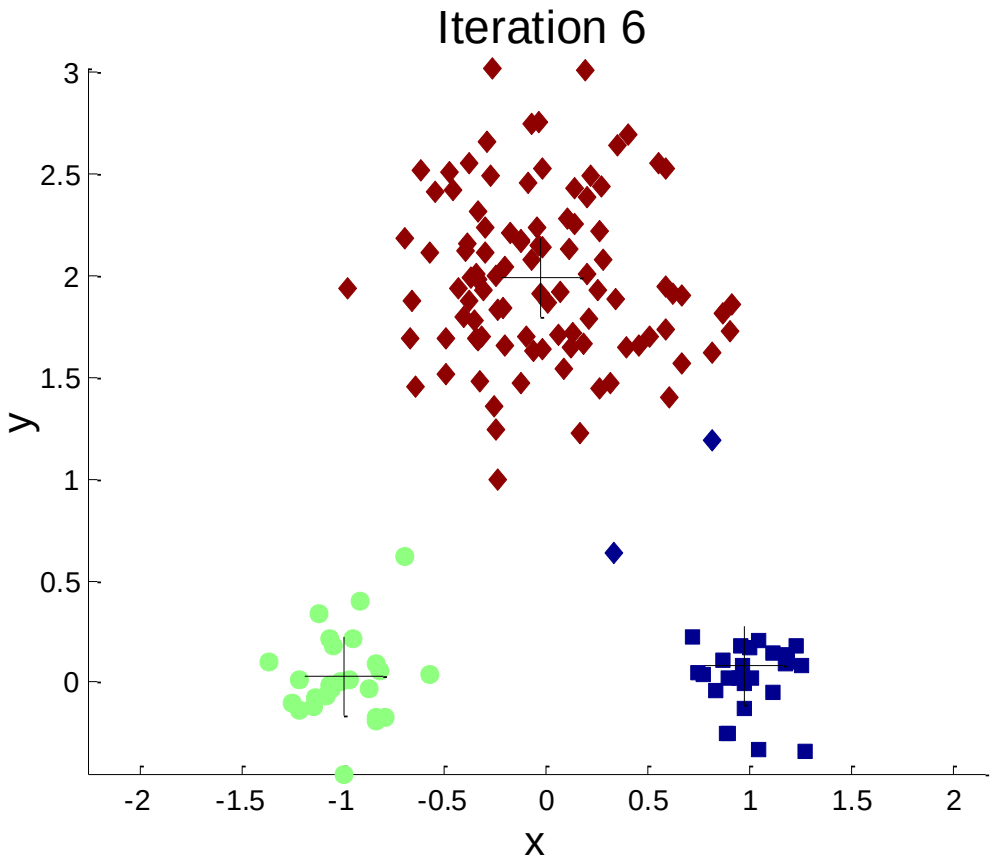
Each point is assigned to the cluster with the closest centroid

Number of clusters, K, must be specified

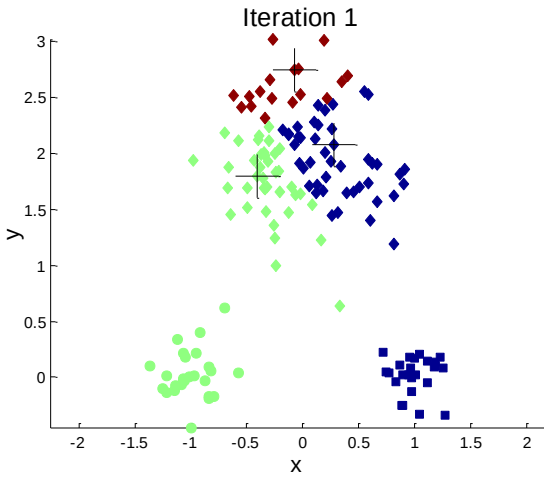
The basic algorithm is very simple

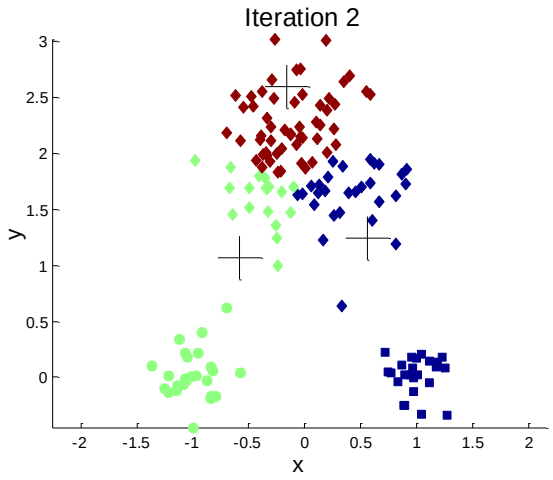
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

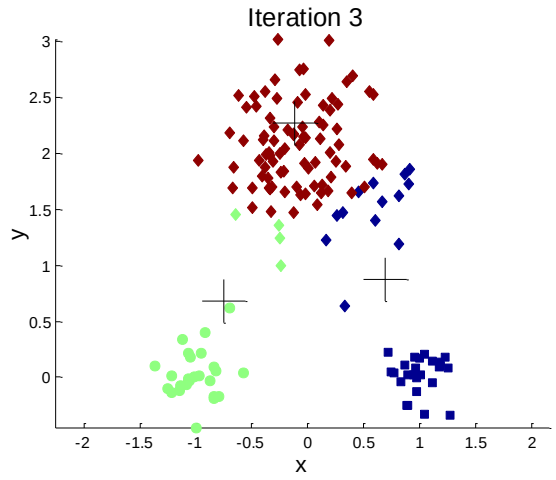
K-Means: Example Run

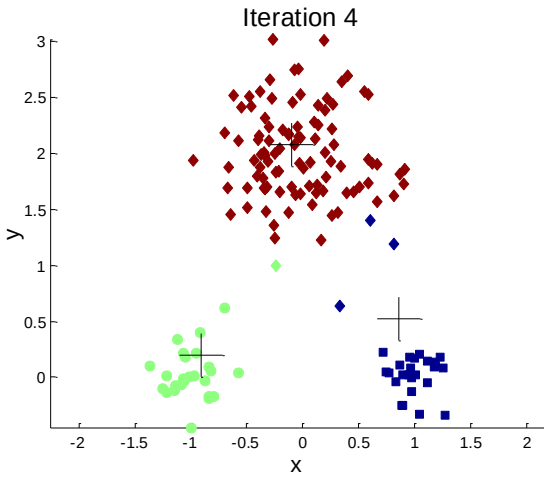


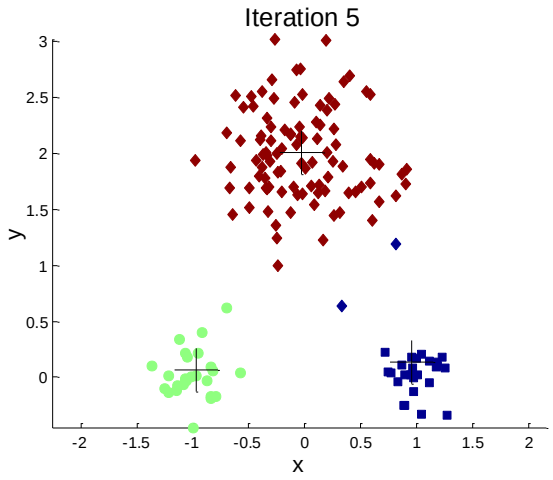
K-Means: Example Run

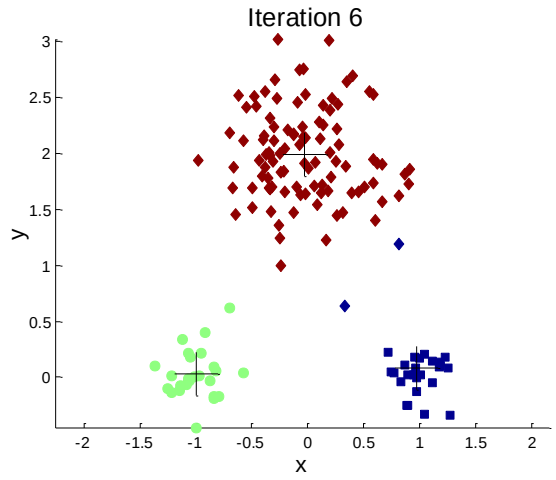












Some Questions

1. How can we evaluate a clustering?

Common measure: Sum of Squared Error (SSE)

- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
- Given two clusterings, we can choose the one with the smallest error

Some Questions

2. Will K-means always converge?

For some combinations of distance measure and cluster center we can prove that K-means will always converge:

- Finite number of potential centers
- SSE *always* becomes smaller

E.g.: Euclidian distance and SSE → mean

Cosine similarity and sum of similarities → norm. mean

Manhattan distance → median

Other combinations: convergence not guaranteed

Illustration: guaranteed combinations

See blackboard.

- Determine function to optimize
  - Points are constant, centre coordinates are parameters
- Take partial derivatives, equate to 0
  - Lagrange multipliers for cosine similarity
- Solve for the centre coordinates

Some Questions

3. What if we cannot compute the mean?

Happens often

- Only matrix of pairwise distances has been given
- Data is non-numerical
  - E.g. graph data with geodesic distance between nodes

Solution (K-medoids):

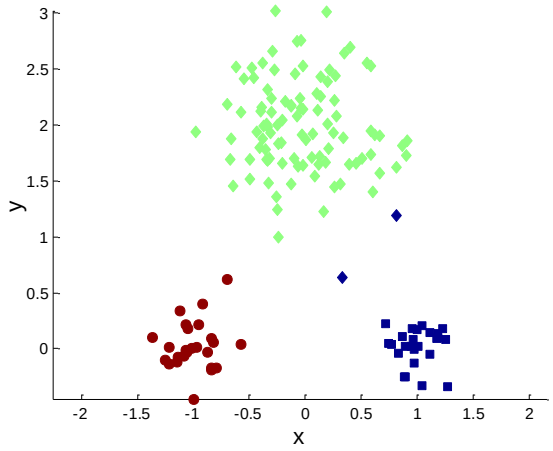
- For every point in the cluster: add up all distances to the other points in the cluster
- Point in the cluster for which this distance is the smallest becomes the new center

Some Questions

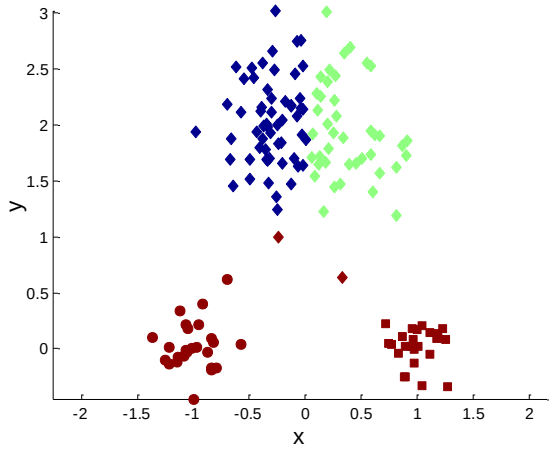
- 4. Will it always converge to the same solution?

No! Depends highly on the initial cluster centers.

- Every “equilibrium” is a potential outcome



Optimal Clustering



Sub-optimal Clustering

Problems with Selecting Initial Points

If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.

- Chance is relatively small when K is large
- If clusters are the same size, n, then

Pick point 1 in cluster 1 : n/K
x … x
Pick point k in cluster 1 : n/K
x
any permutation of these: K!

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

Pick point 1 : n
x … x
Pick point k : n
(all permutations are already there)

- For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
- Sometimes the initial centroids will re-adjust themselves in the 'right' way, and sometimes they don't

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Merge clusters that are close (cluster the clusters)

- Bisecting K-means

Bisecting K-means

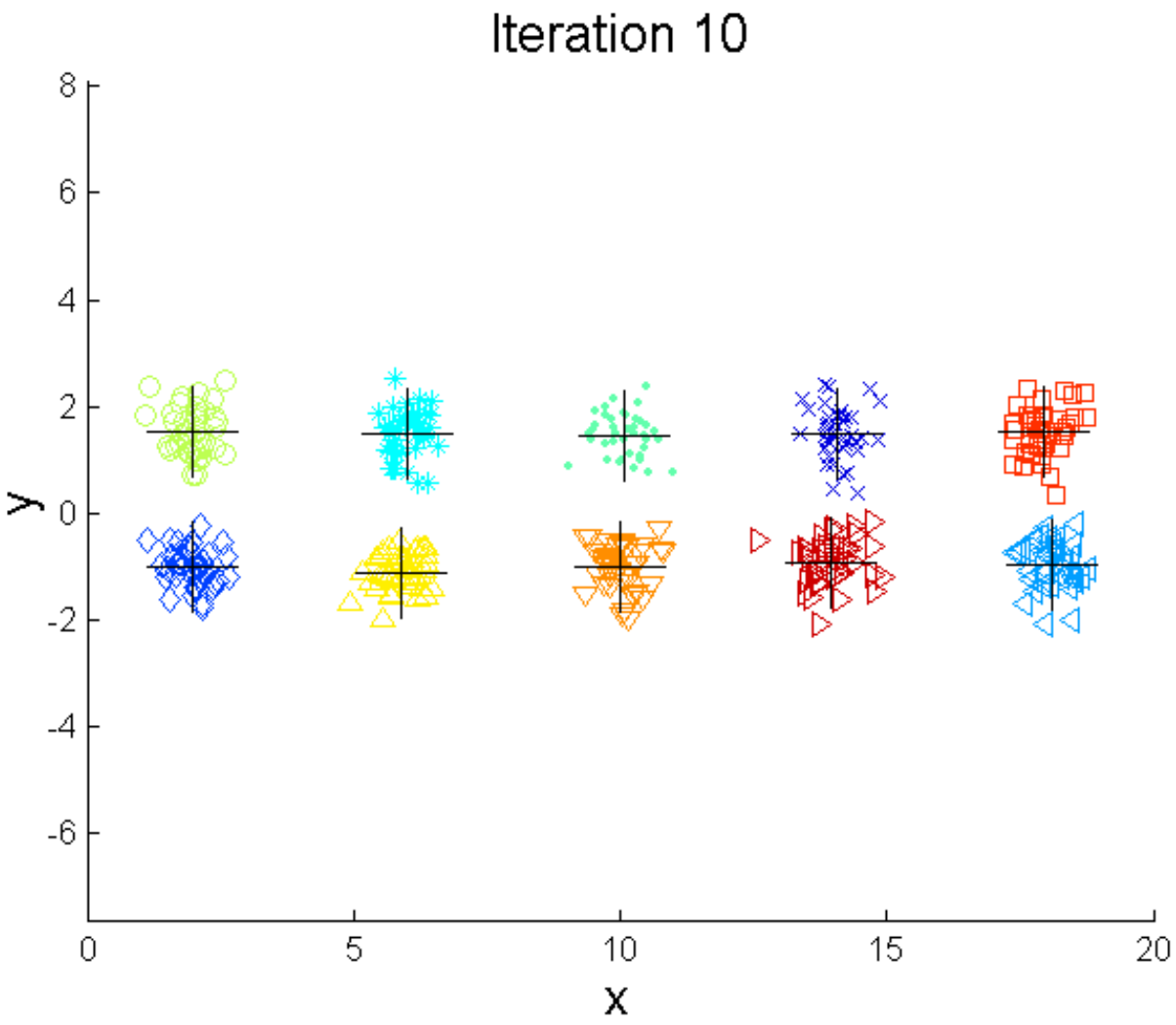
Bisecting K-means algorithm

- Variant of K-means that can produce a partitional or a hierarchical clustering

---

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

Bisecting K-means Example



Limitations of K-means

K-means has problems when clusters are of different

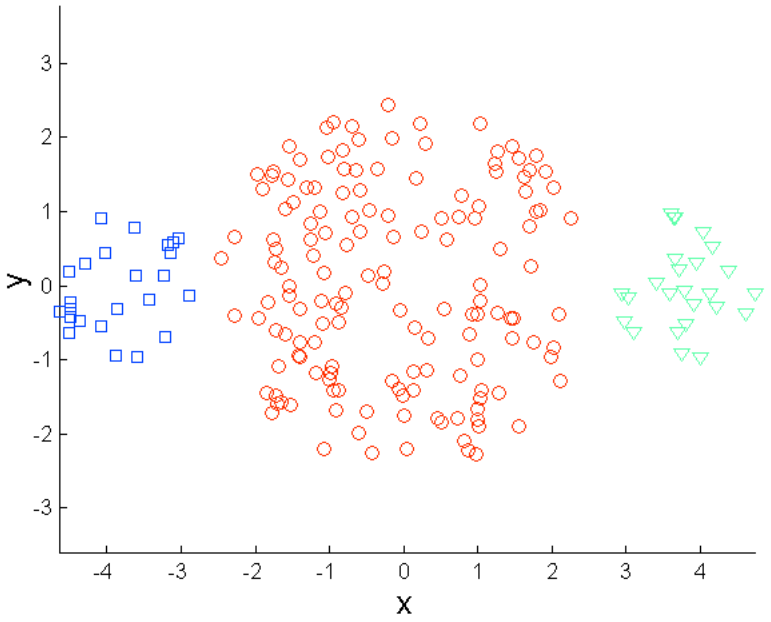
- size
- density

Or have non-globular shapes

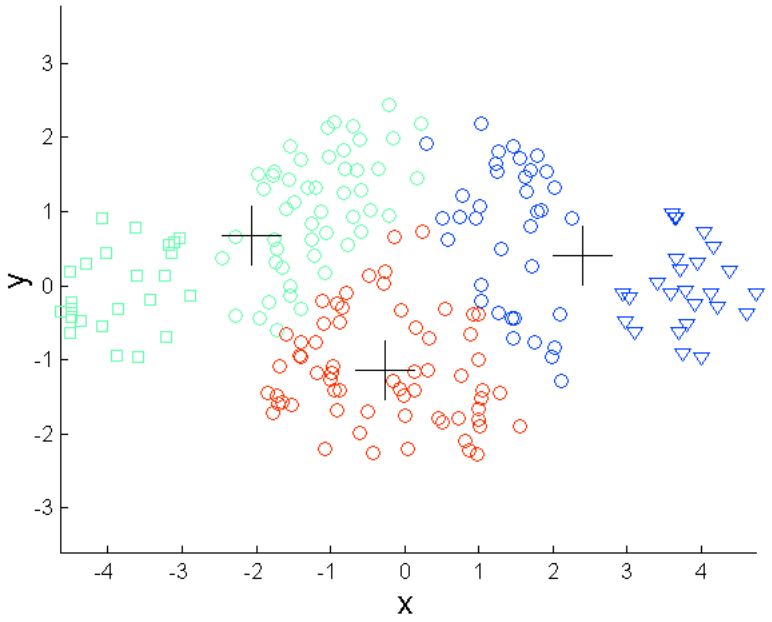
K-means has problems when the data contains outliers

- Use median instead of mean

Limitations of K-means: Differing Sizes

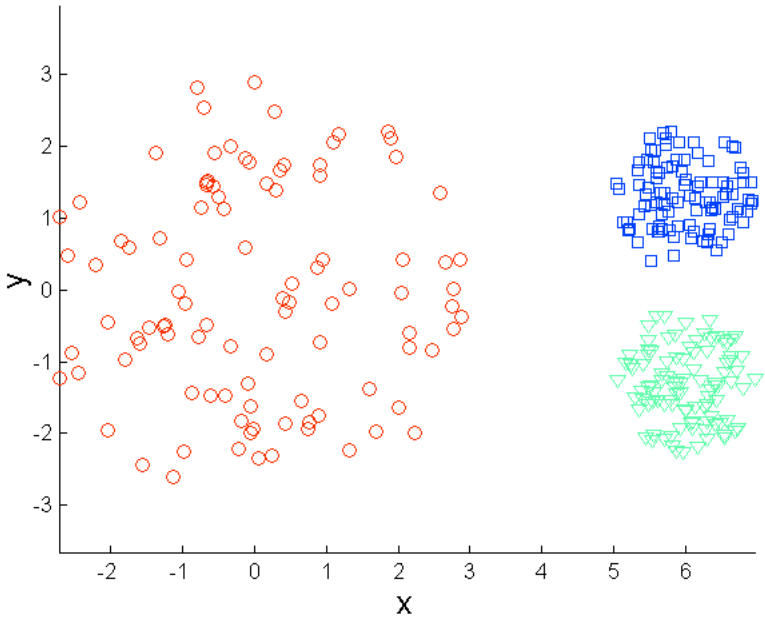


Original Points

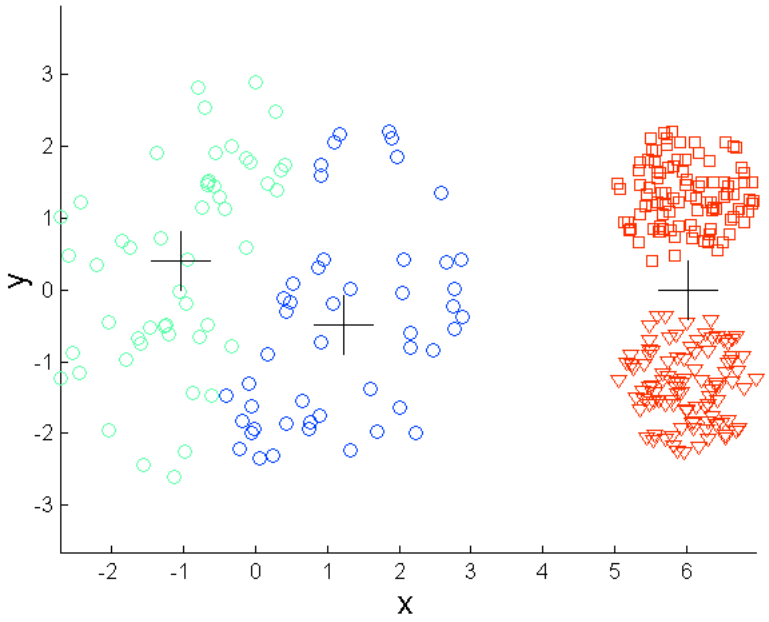


K-means (3 Clusters)

Limitations of K-means: Differing Density

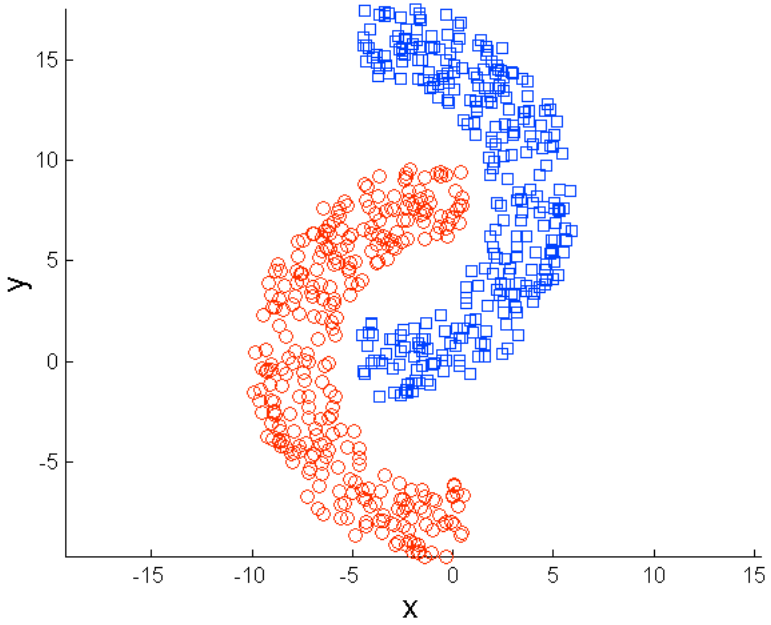


Original Points

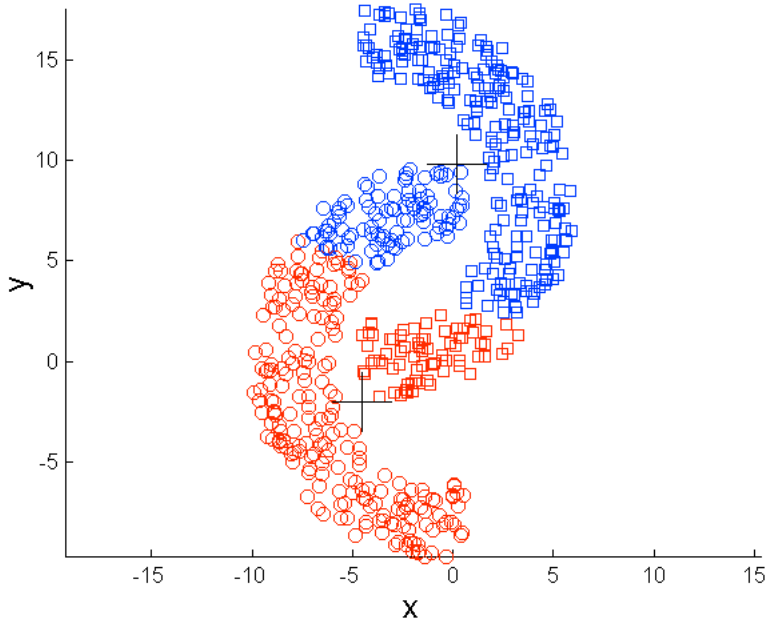


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

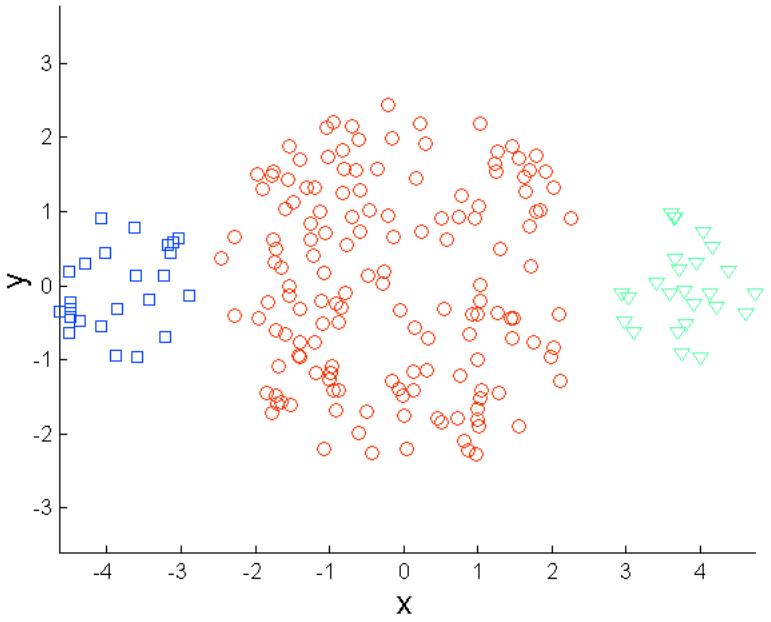


Original Points

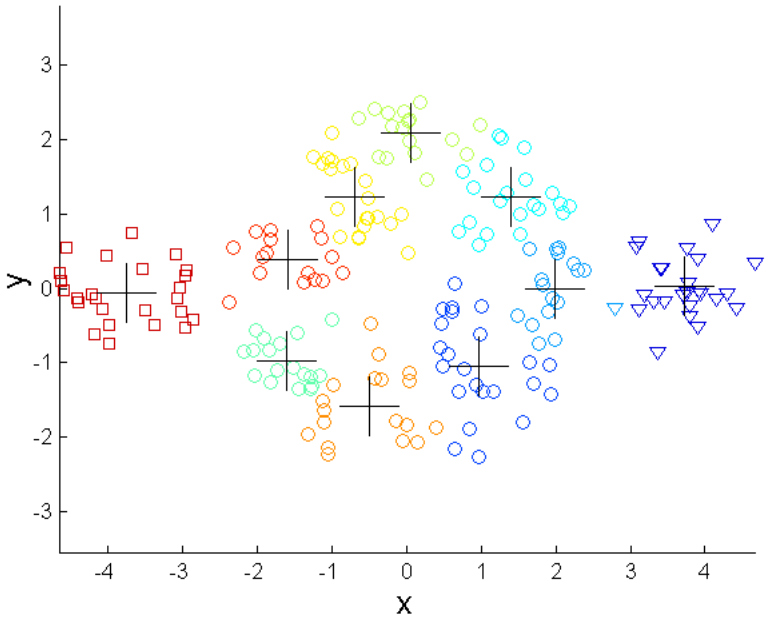


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

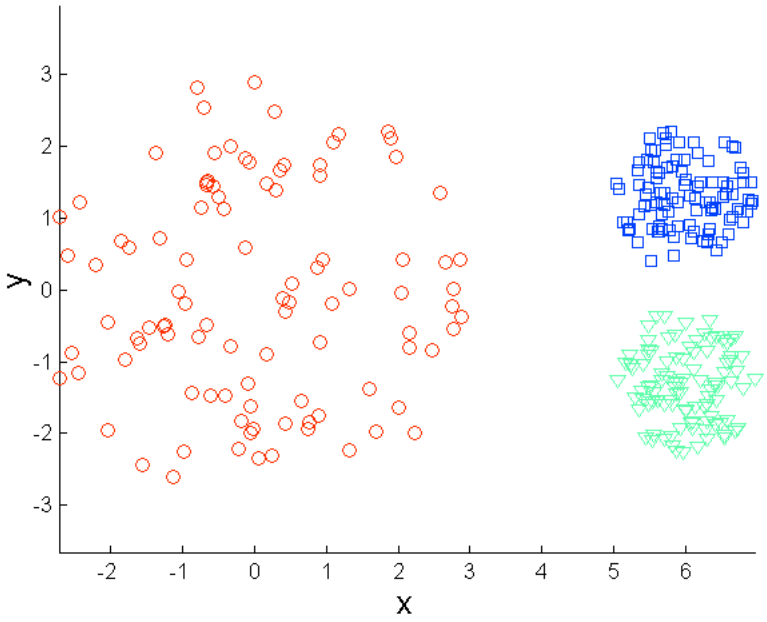


K-means Clusters

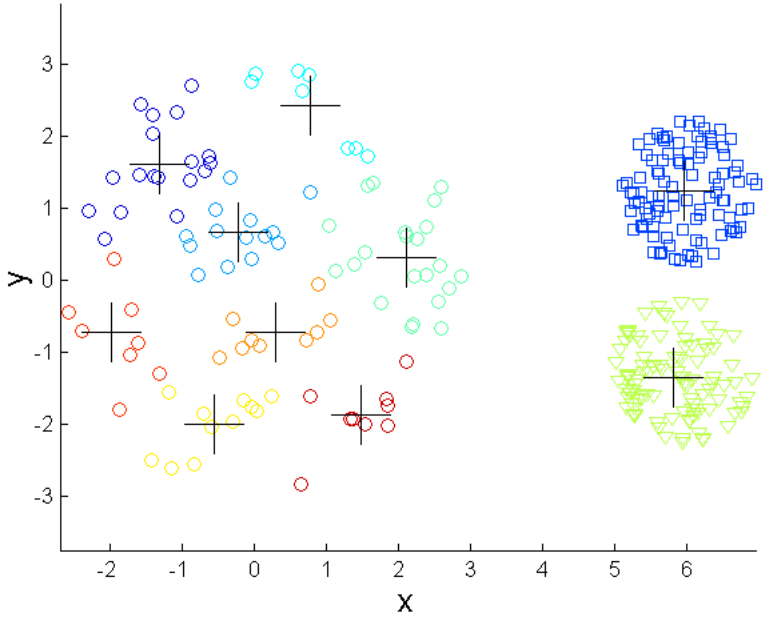
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

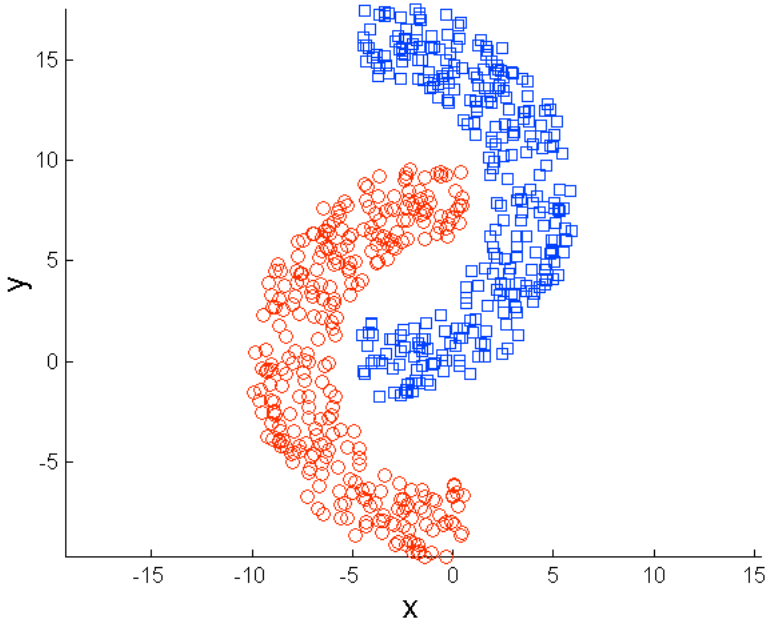


Original Points

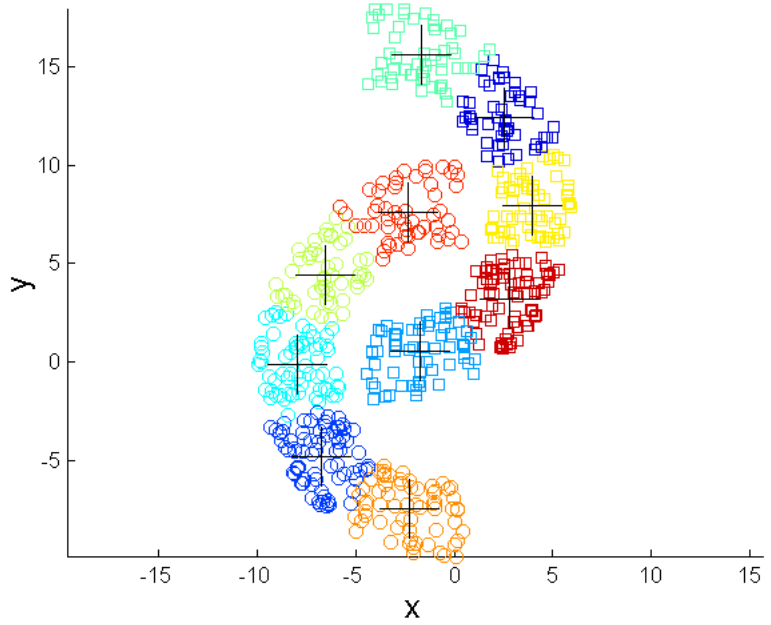


K-means Clusters

Overcoming K-means Limitations

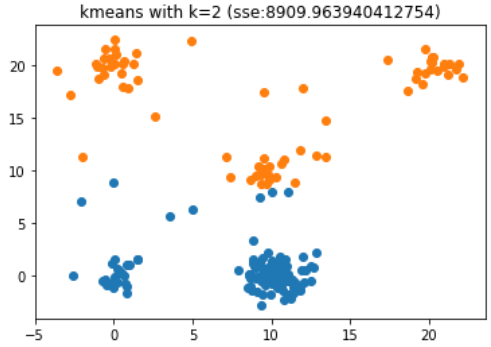


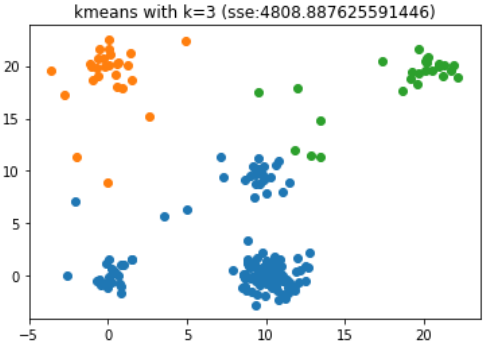
Original Points

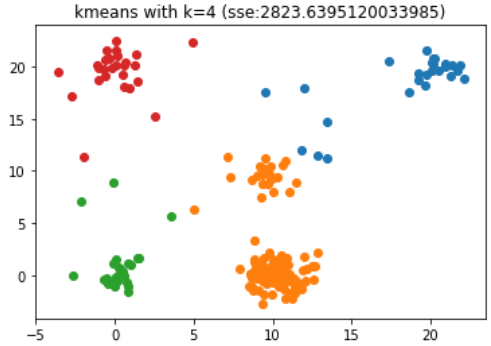


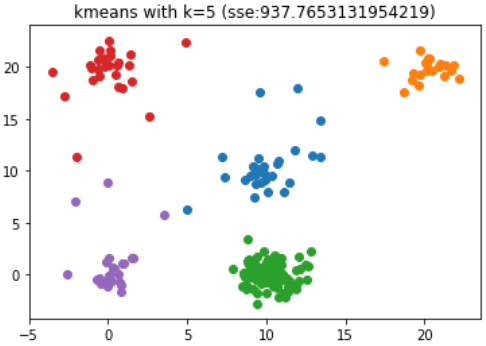
K-means Clusters

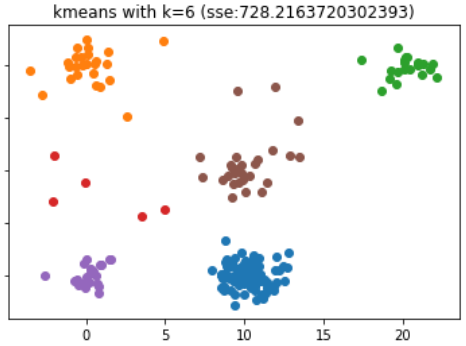
Determining k for k-means

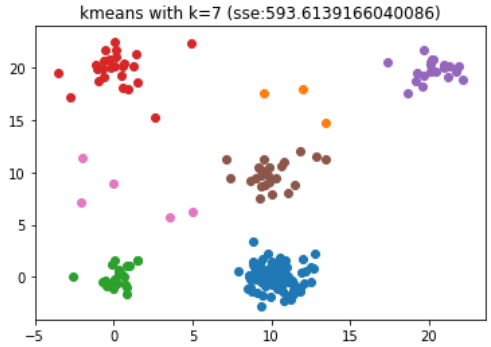


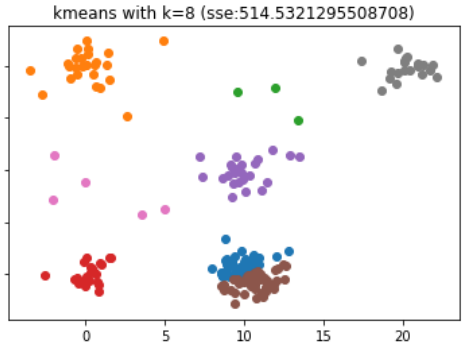


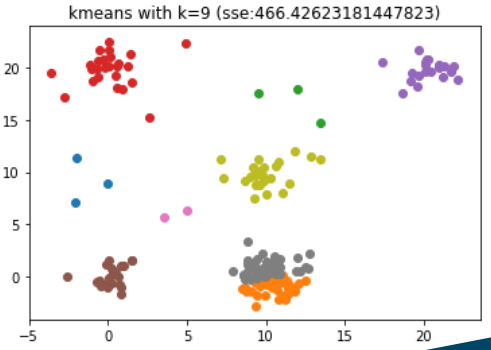






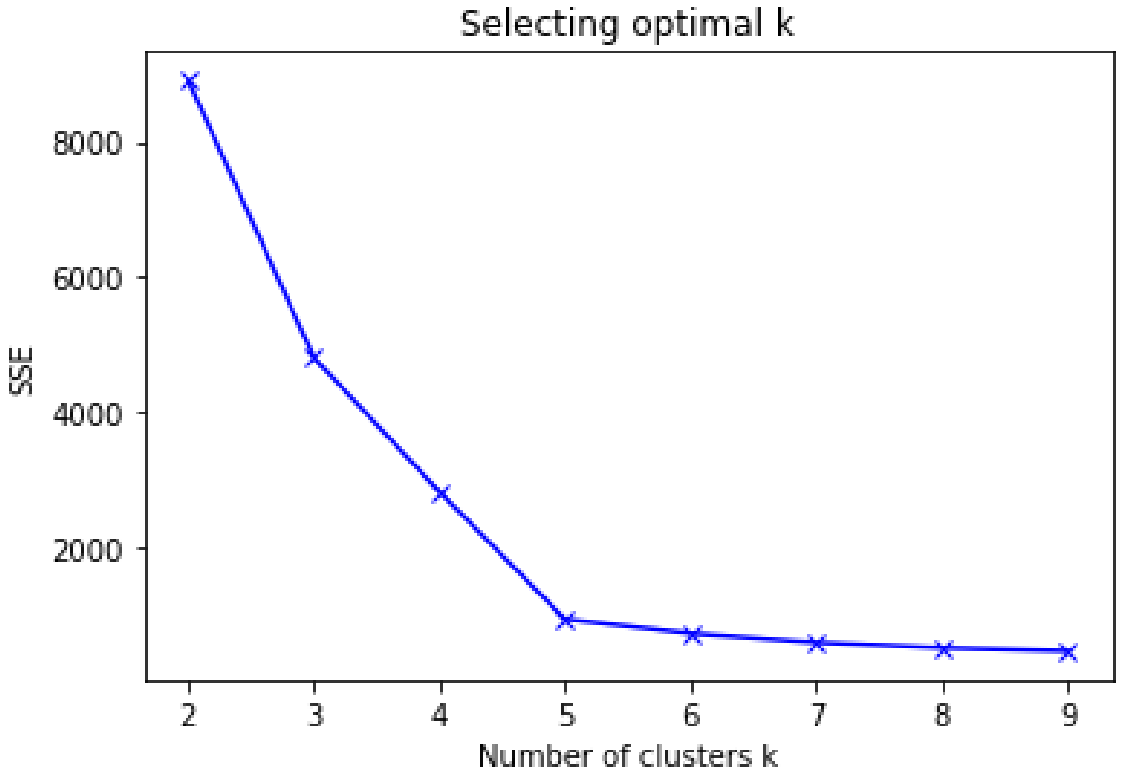






Determining k for k-means

After the optimal number of clusters is reached, the SSE will still decrease, but the decrease will be less outspoken



Outline

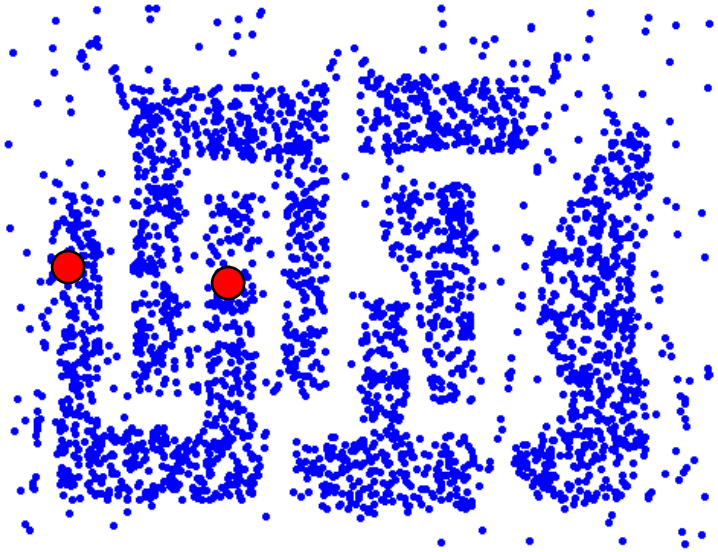
Partitional Clustering

- Distance-based
  - K-means, K-medoids, Bisecting K-means
- Density-based
  - DBSCAN
- Expectation-Maximization

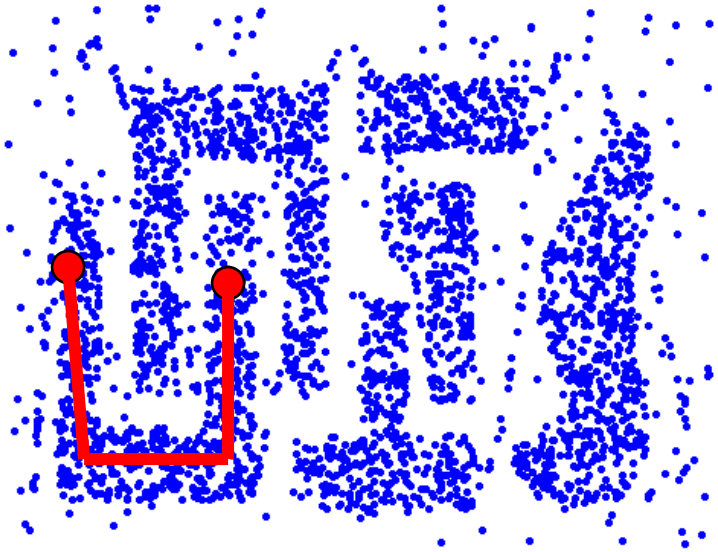
Hierarchical Clustering

Cluster validity

Density-based clustering



Density-based clustering

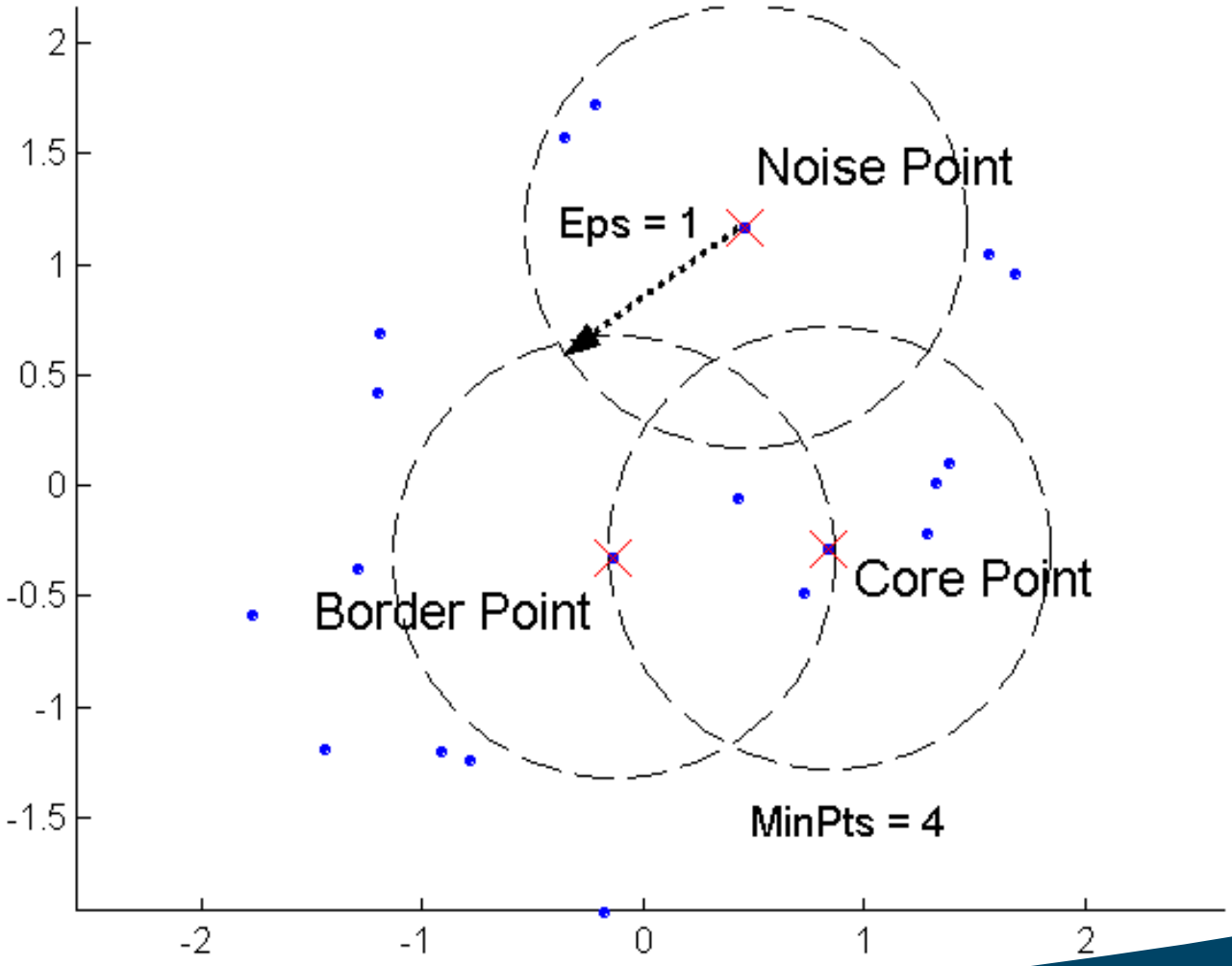


DBSCAN

DBSCAN is a density-based algorithm.

- Density = number of points within a specified radius (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points



Density-based clustering: DBSCAN

A point y is *(ε, μ)-density-reachable* from point x if there exists a sequence $c_1, \ldots, c_k$ of *(ε,μ)*-core points such that:

- $d(x,c_1) \leq \varepsilon$
- $\forall i \in \{1, \ldots, k-1\} : d(c_i,c_{i+1}) \leq \varepsilon$
- $d(c_k,y) \leq \varepsilon$

Remark: computing all pairs (x,y) such that y is *(ε, μ)-density-reachable* from core point x = computing transitive closure.

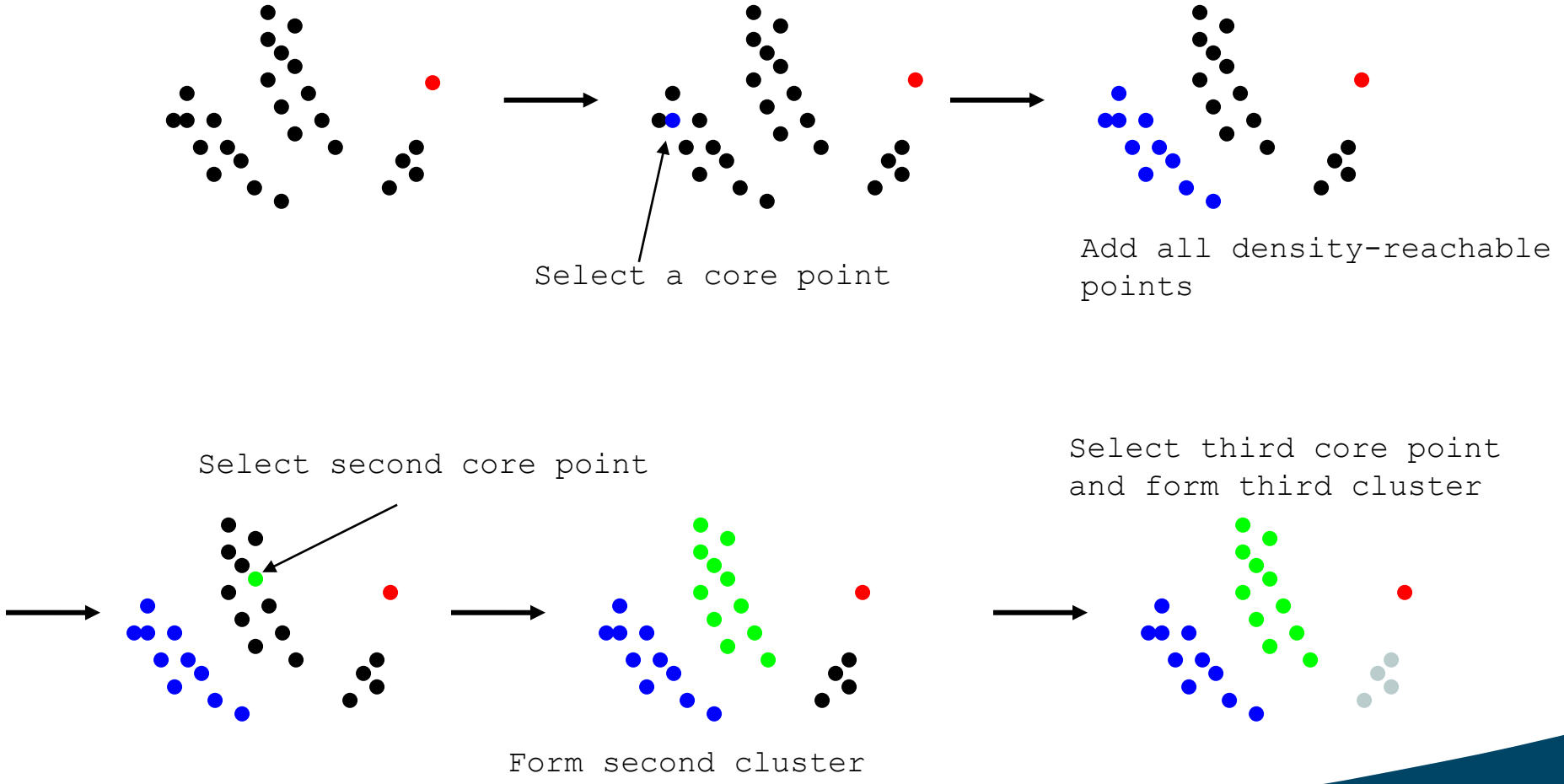
DBSCAN Algorithm

Eliminate noise points

Perform clustering on the remaining points

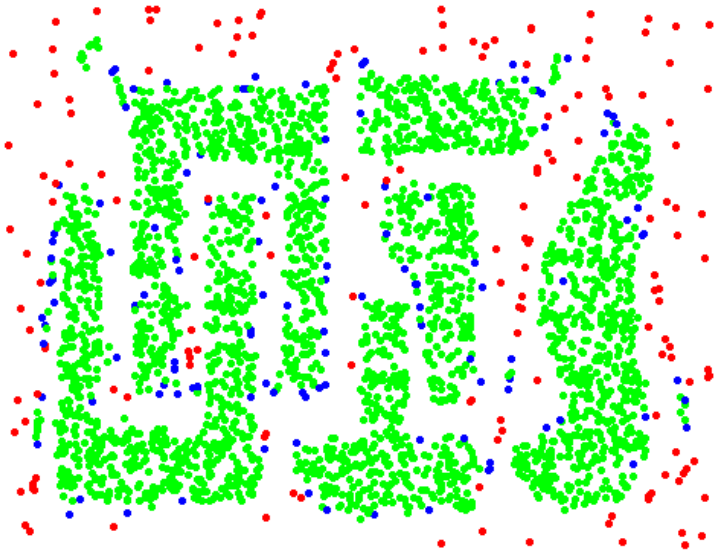
```
current_cluster_label ← 1
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    end if
    for all points in the Eps-neighborhood, except i^th the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
        end if
    end for
end for
```

Voorbeeld: DBSCAN



DBSCAN: Core, Border and Noise Points

Original Points

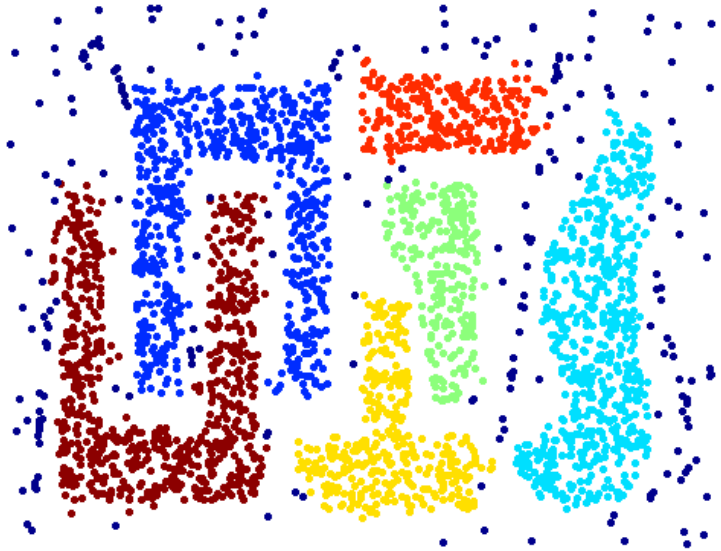


Point types: core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

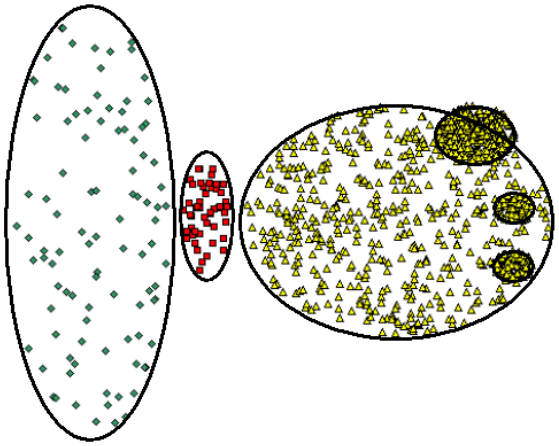
Original Points



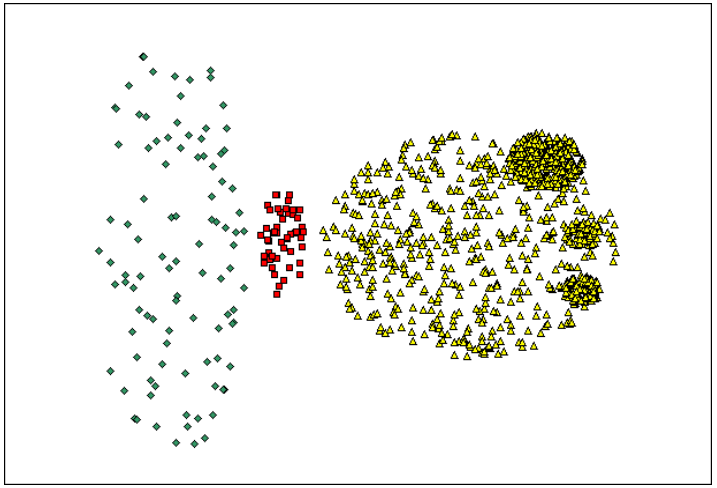
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

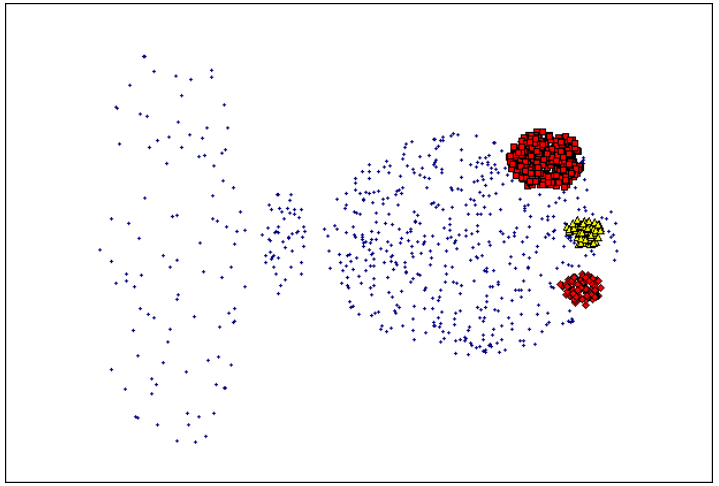
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92)



(MinPts=4, Eps=9.75).

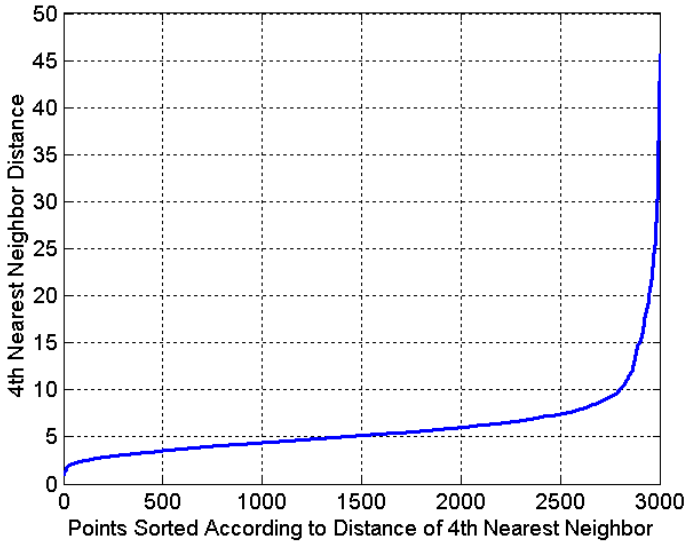
- Varying densities
- High-dimensional data

DBSCAN: Determining EPS and MinPts

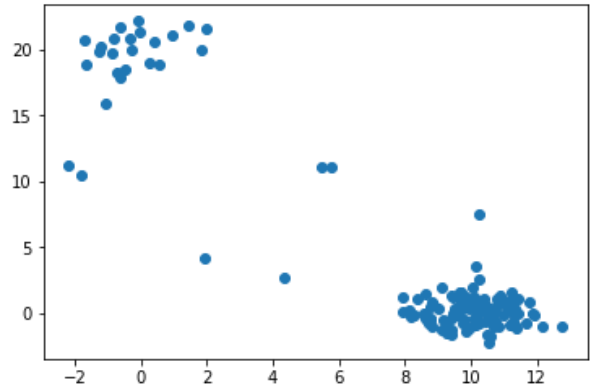
Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance

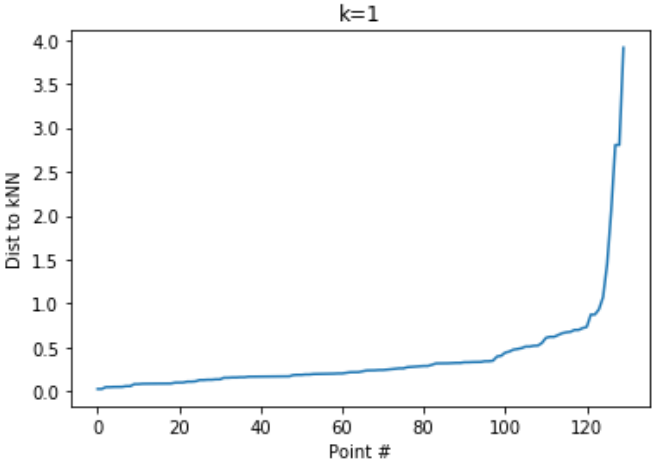
Noise points have the k^{th} nearest neighbor at farther distance

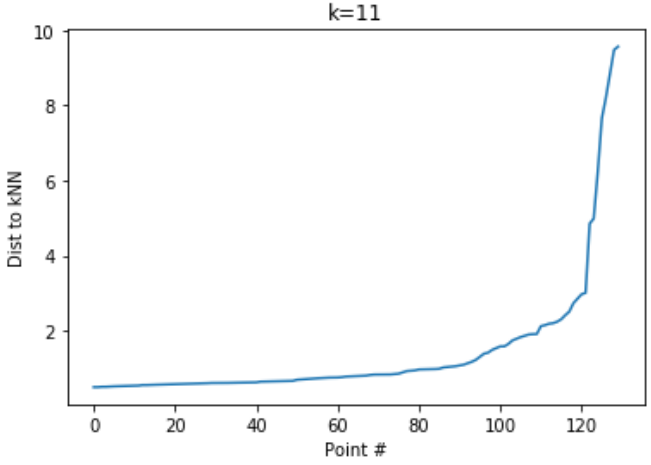
So, plot sorted distance of every point to its k^{th} nearest neighbor

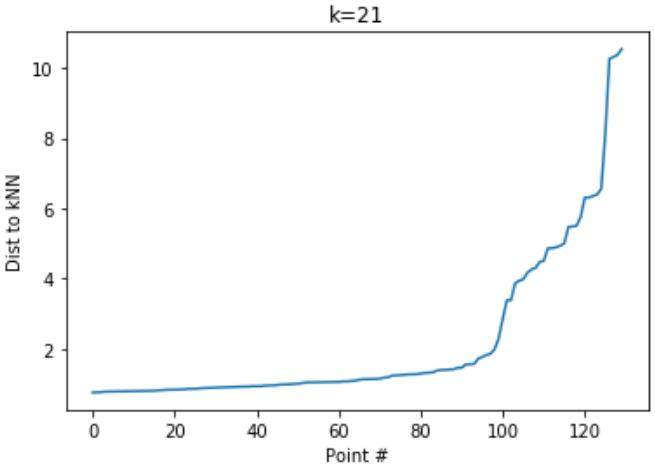


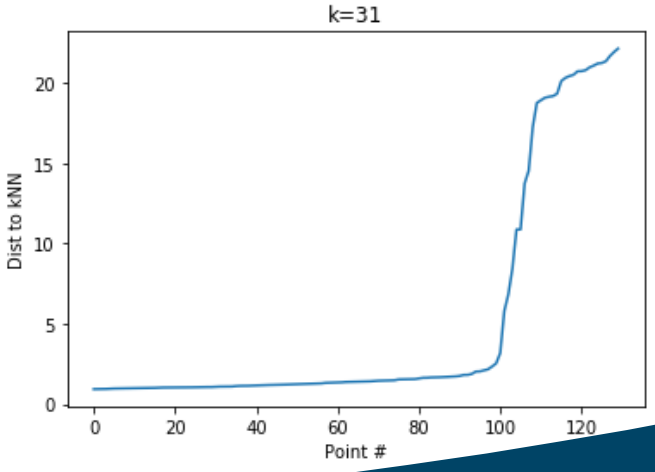
Parameters for DBScan



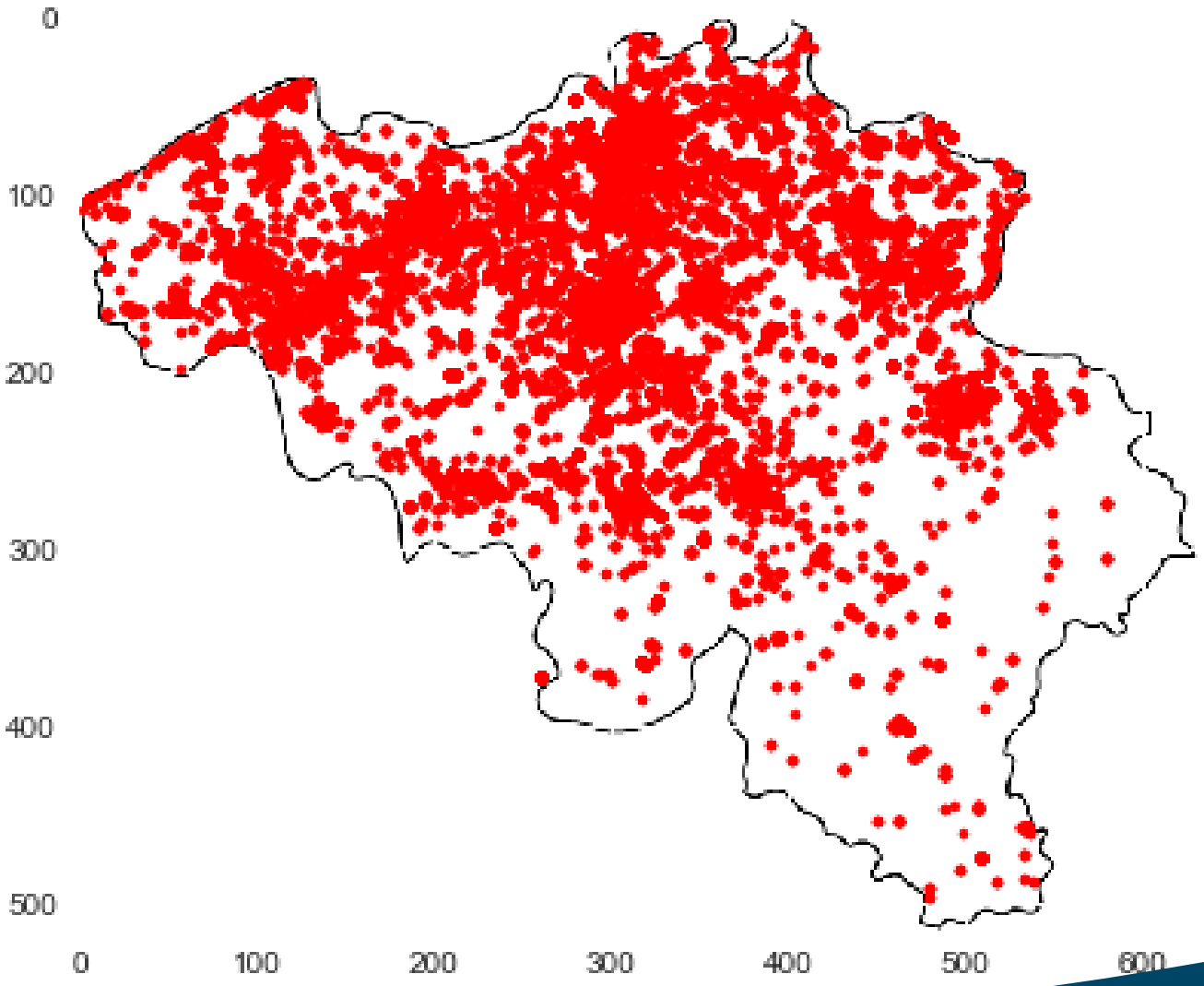




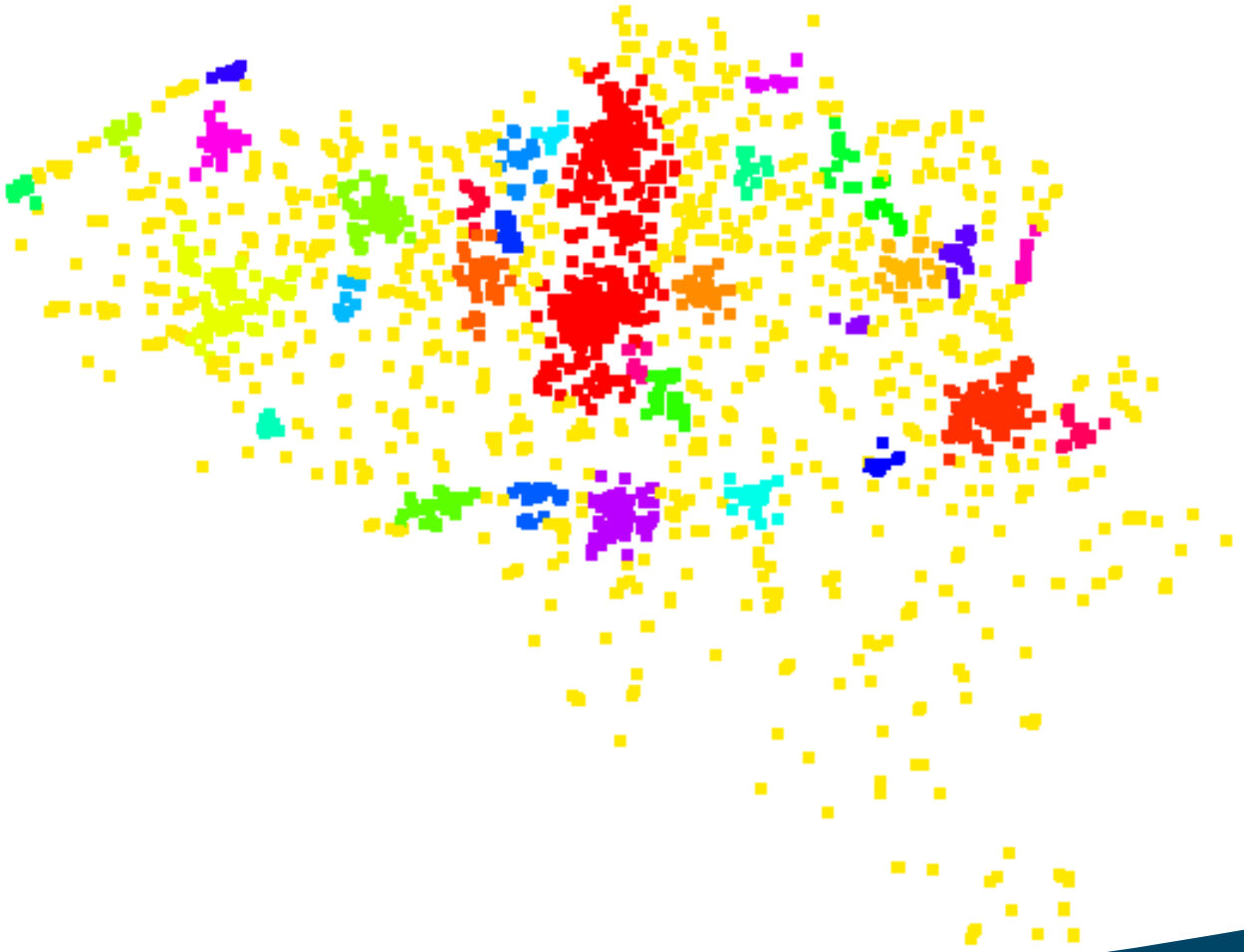




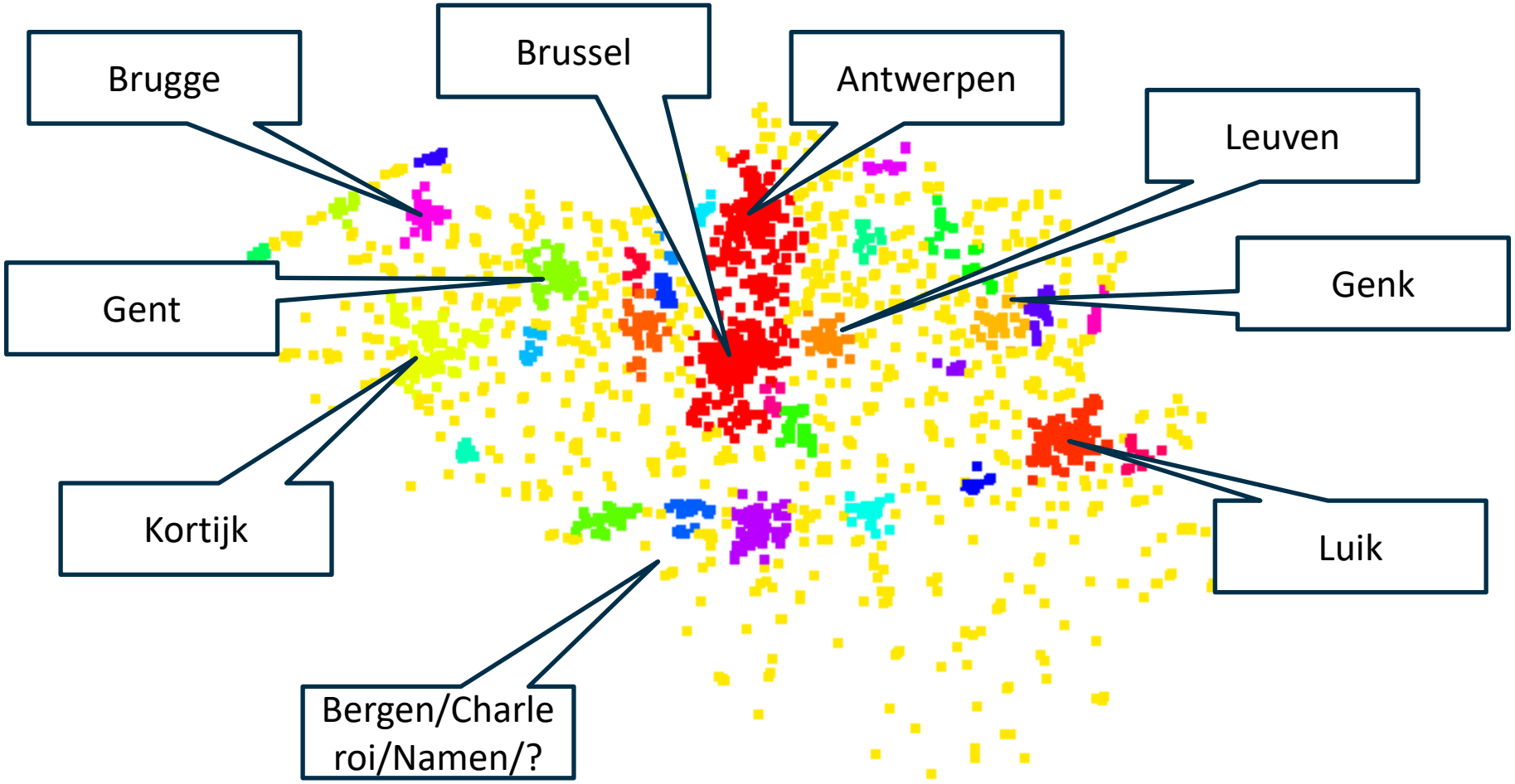
Customer location – Electronic Meal Vouchers



Clustering by DBScan



Interpretation of the Clusters



Outline

Partitional Clustering

- Distance-based
  - K-means, K-medoids, Bisecting K-means
- Density-based
  - DBSCAN
- Expectation-Maximization

Hierarchical Clustering

Cluster validity

Now: Another Approach; EM

Assume we know the process that generated the clusters

- E.g., mixture of multi-variate normal distributions

But we do not know the parameters of the model

- E.g., covariance matrix, relative weights of the distributions

Mixture modeling = finding the best parameters of the assumed model

Expectation Maximization (EM) = an optimization technique to find these parameters

Example: Mixture Model

Consider a mixture of three uni-variate normal distributions:

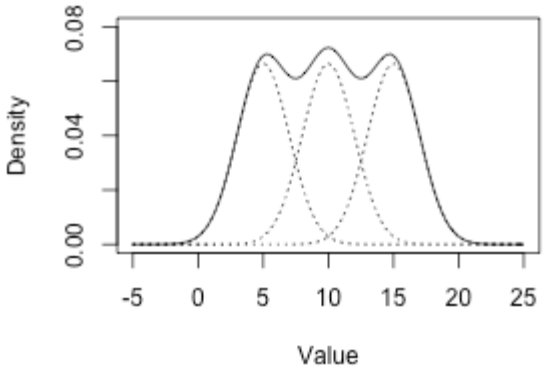


Figure source: Wikipedia

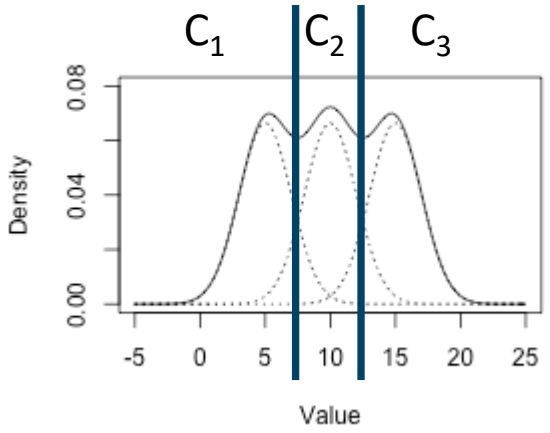
Our population consists of three clusters

- The dotted lines are the pdf's of the distributions in the clusters
- The full line is the density we observe from the (unlabelled) data = pdf of the

Example: Mixture Model

If we know the three distributions:

- Assign every object to its most likely class
- If $P(x \in C_1) > P(x \in C_2)$ and $> P(x \in C_3)$ Then assign x to C_1



Hence, clustering = finding parameters of the model

- Mean and variance of the three distributions + their relative weights

Mixture modeling in this example = finding the parameters

$w_1, w_2, w_3, \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3$

Mixture Modeling

Central question: how measure quality of parameter setting?

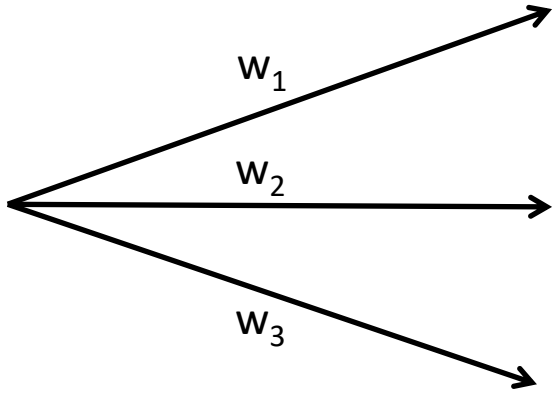
Answer: Maximum Likelihood

- The more probable the data given the model, the more likely is the model
  - Likelihood of a model = probability of the data given the model

Maximum Likelihood

Probability of a point x given parameters

$\Theta = (w_1, w_2, w_3, \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3)$



$N(\mu_1, \sigma_1) \rightarrow P(x \mid C_1)$

$N(\mu_2, \sigma_2) \rightarrow P(x \mid C_2)$

$N(\mu_3, \sigma_3) \rightarrow P(x \mid C_3)$

$$P(x \mid \Theta) = w_1 P(x \mid C_1) + w_2 P(x \mid C_2) + w_3 P(x \mid C_3)$$

Maximum Likelihood

Assumption: tuples are i.i.d.

Hence, $P(D \mid \Theta) = \prod_{x \in D} P(x \mid \Theta)$

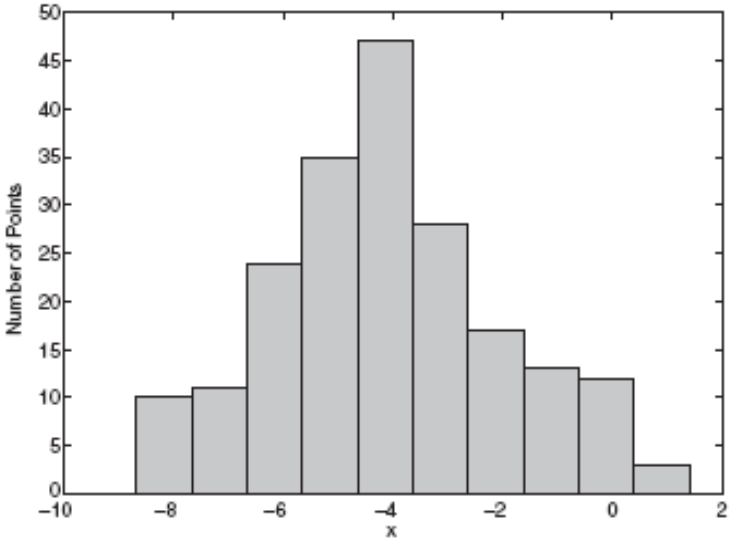
Likelihood of Θ given D:

$L(\Theta \mid D) := P(D \mid \Theta)$

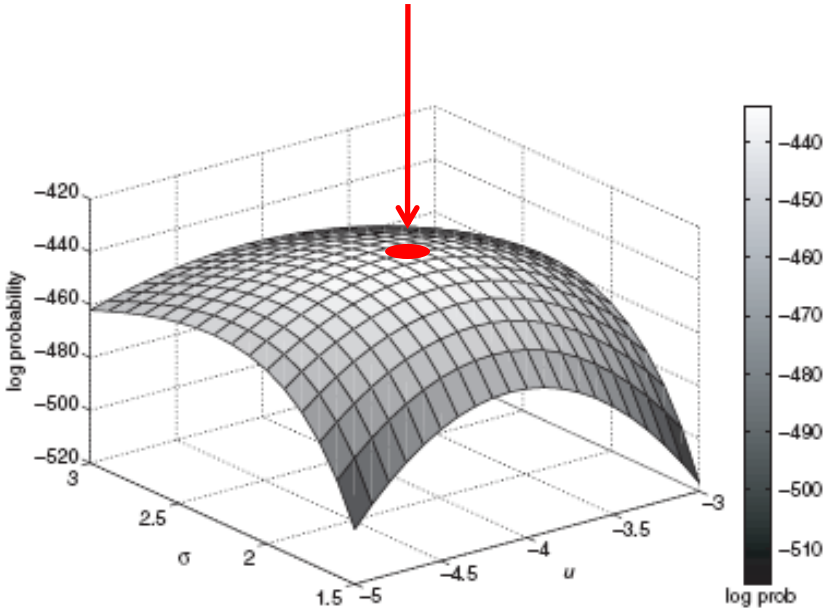
For numerical reasons: log-likelihood

$L^*(\Theta \mid D) := \log L(\Theta \mid D)$

Maximum Likelihood: Illustration



(a) Histogram of 200 points from a Gaussian distribution.

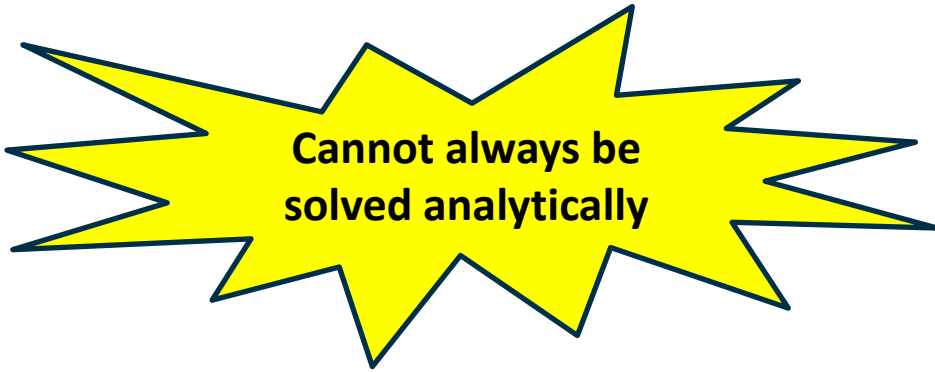


(b) Log likelihood plot of the 200 points for different values of the mean and standard deviation.

Figure 9.3. 200 points from a Gaussian distribution and their log probability for different parameter values.

Figure source: Tan, Steinbach, Kumar. Introduction to data mining.

Are We There Yet?



The ingredients:

- Model class and data are given
- Likelihood to score the parameters

→ We have turned our clustering problem into an optimization problem!

→ Open up our mathematics toolbox

→ take partial derivatives

→ equate to zero

→ solve system

Expectation – maximization is an approximation method

Expectation-Maximization

EM is based on

- If we know Θ, it is easy to find the clusters
- If we know the clusters, it is easy to find Θ

EM is an iterative procedure:

- Starts with a random initialization of the parameters
- It is now easy to find to which cluster and with what probability objects belong
- Based on this cluster assignment, update the parameters (easy; we can do it for every cluster separately)
- Repeat …

Expectation - Maximization Algorithm

Random initialization of Θ

Repeat

E(xpectation)-Step

For every training example x compute the probability that it was generated by distribution i:

$$P(x \text{ generated by } D_i \mid \Theta)$$

Update the parameters such that Θ maximizes likelihood given the probabilities in step (2)

M(aximization)-Step

Until no changes

EM converges towards a local maximum of the likelihood.

Illustration: Clusters Found By EM

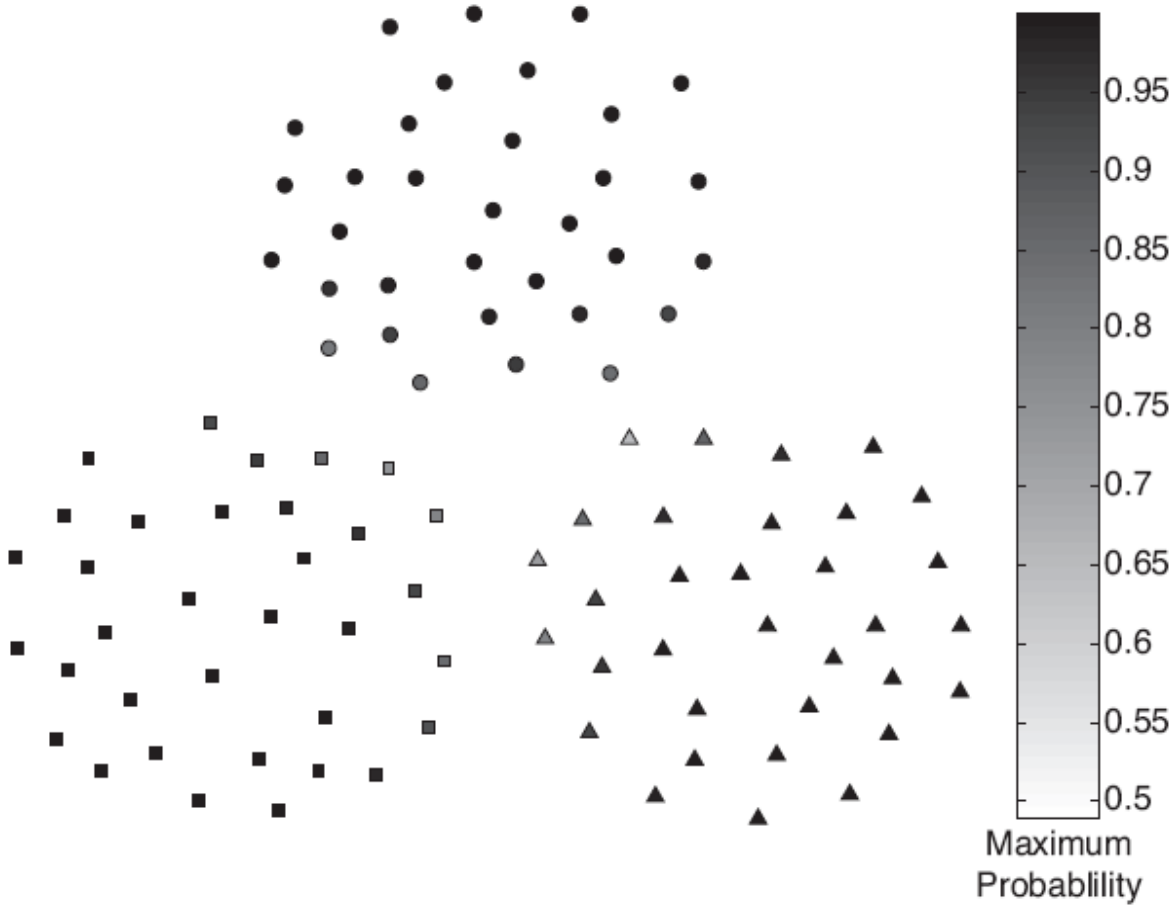


Figure 9.4. EM clustering of a two-dimensional point set with three clusters.

Figure source: Tan, Steinbach, Kumar. Introduction to data mining.

Illustration: Clusters Found By EM

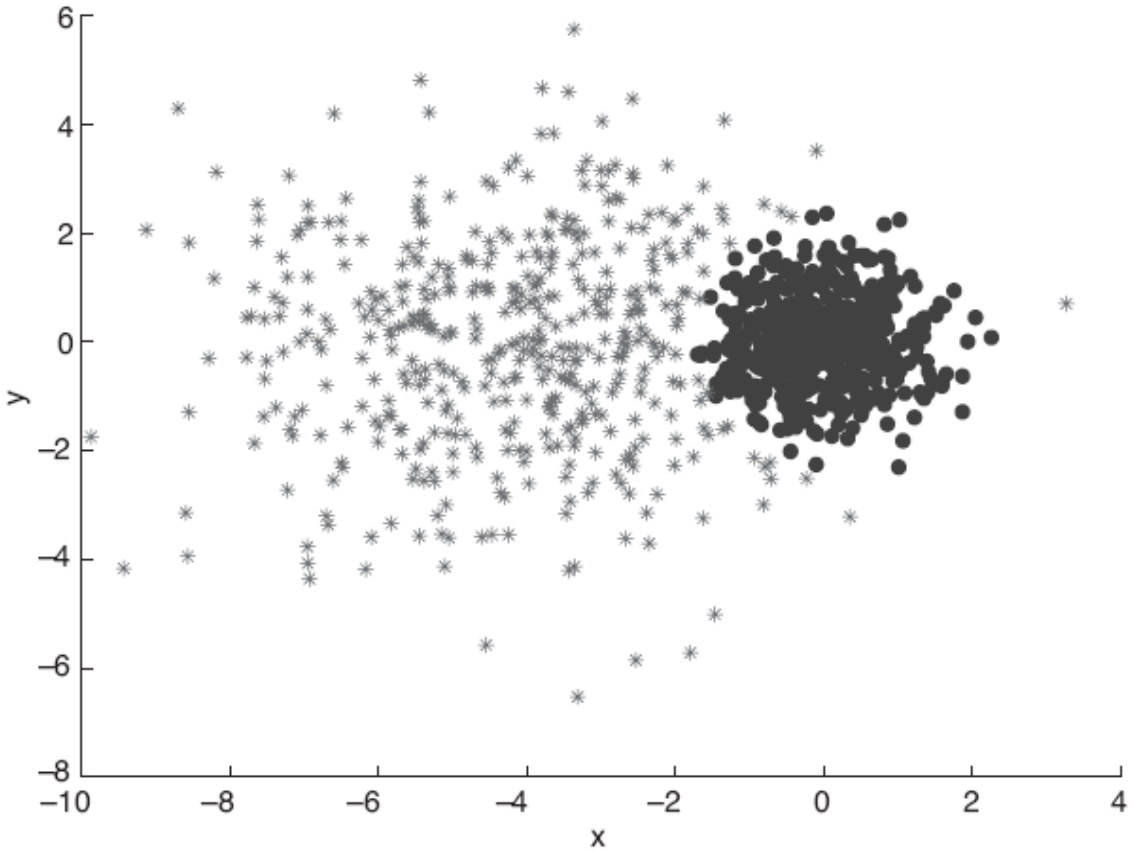
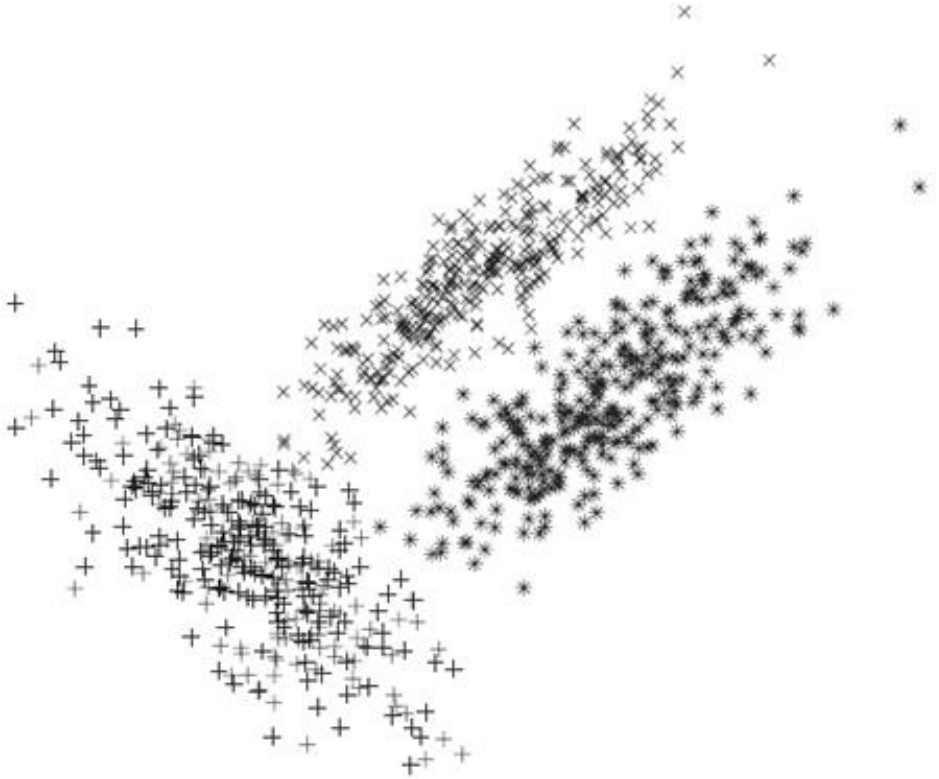


Figure 9.5. EM clustering of a two-dimensional point set with two clusters of differing density.

Figure source: Tan, Steinbach, Kumar. Introduction to data mining.

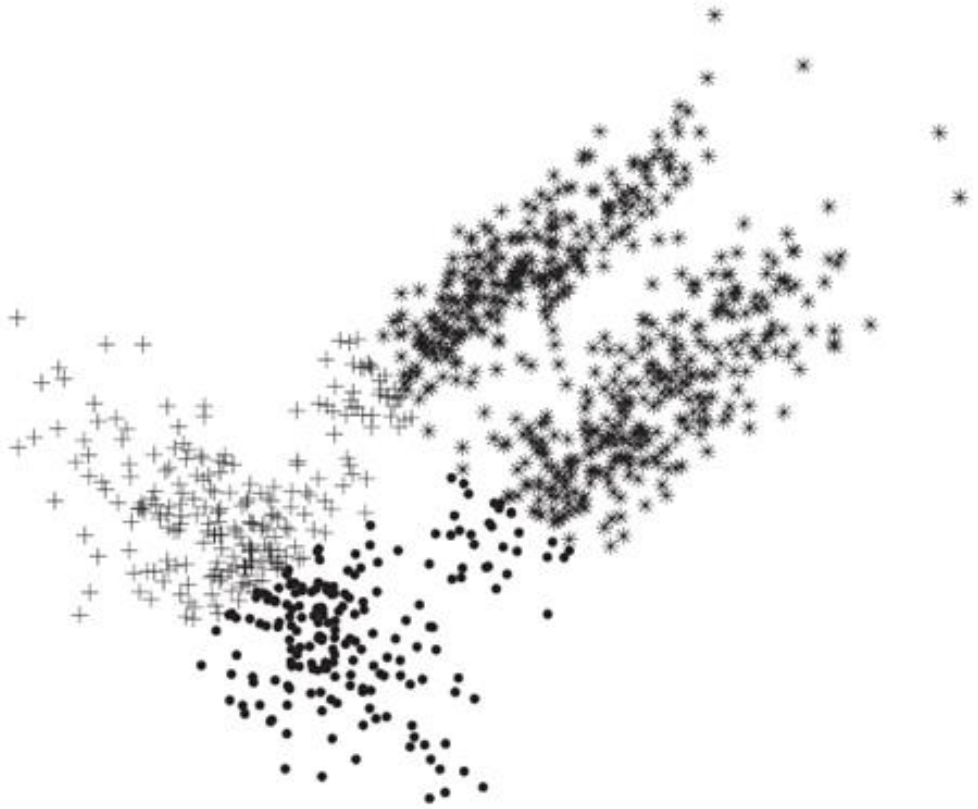
Illustration: Comparison EM vs k-means (1/2)



(a) Clusters produced by mixture model clustering.

Figure source: Tan, Steinbach, Kumar. Introduction to data mining.

Illustration: Comparison EM vs k-means (2/2)



(b) Clusters produced by K-means clustering.

Figure source: Tan, Steinbach, Kumar. Introduction to data mining.

Mixture Modeling & EM: Summary

Based on the assumptions:

- We know the processes that generated the data
- But we do not know the exact parameters

Clustering = separating the data according to the process they were generated by

- Easy if we know the parameters

Maximum likelihood as selection criterion for parameters

- The more probable the data is given the model, the more the model is

Mixture Modeling & EM: Summary

Finding the parameters that optimize the likelihood is not always easy

Expectation-Maximization is an approximation algorithm that finds a local maximum of the likelihood

- Start with random assignment
- Iteratively fix parameters and estimate clusters followed by fix clusters and change parameters

K-means is a special case of EM

Assumptions:

- Mixture of Gaussian distributions

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

  - Same weight
  - Covariance matrix = identity matrix

Density of such distribution is monotone in d(**x**,**μ**).

Outline

Partitional Clustering

- Distance-based
  - K-means, K-medoids, Bisecting K-means
- Density-based
  - DBSCAN
- Expectation-Maximization

Hierarchical Clustering

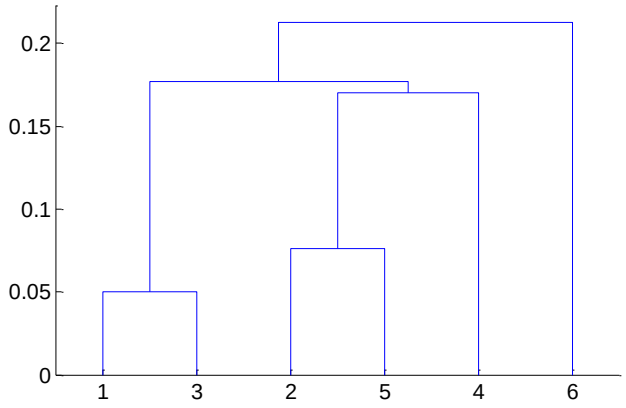
Cluster validity

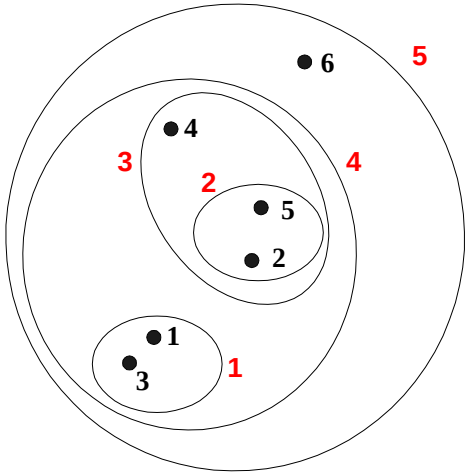
Hierarchical Clustering

Produces a set of nested clusters organized as a hierarchical tree

Can be visualized as a dendrogram

- A tree like diagram that records the sequences of merges or splits





No need to set a predefined number of clusters

Agglomerative Clustering Algorithm

More popular hierarchical clustering technique

Basic algorithm is straightforward

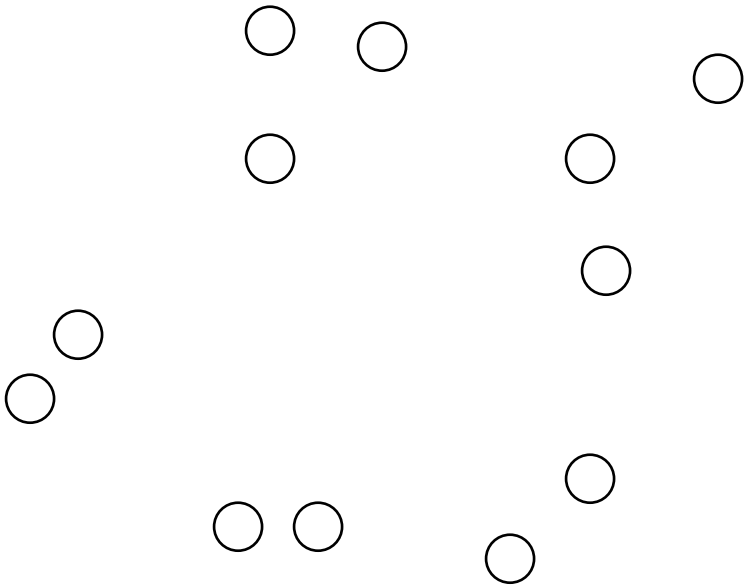
1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4. Merge the two closest clusters
5. Update the proximity matrix
6. Until only a single cluster remains

Key operation is the computation of the proximity of two

- Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

Start with clusters of individual points and a proximity matrix

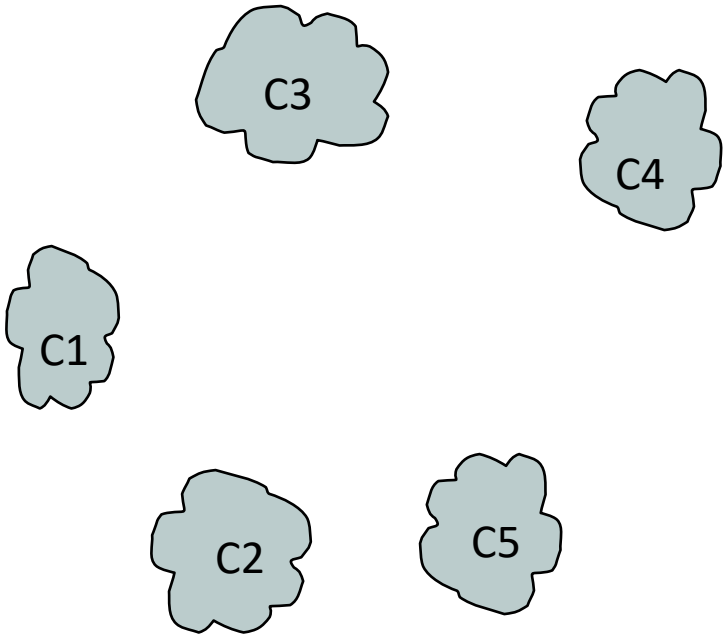


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

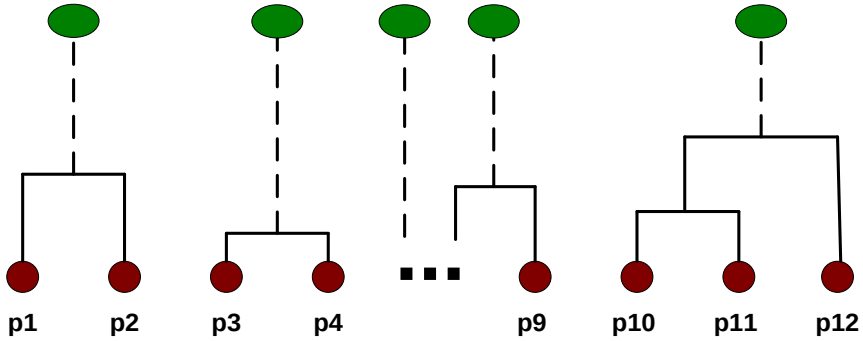
Intermediate Situation

After some merging steps, we have some clusters



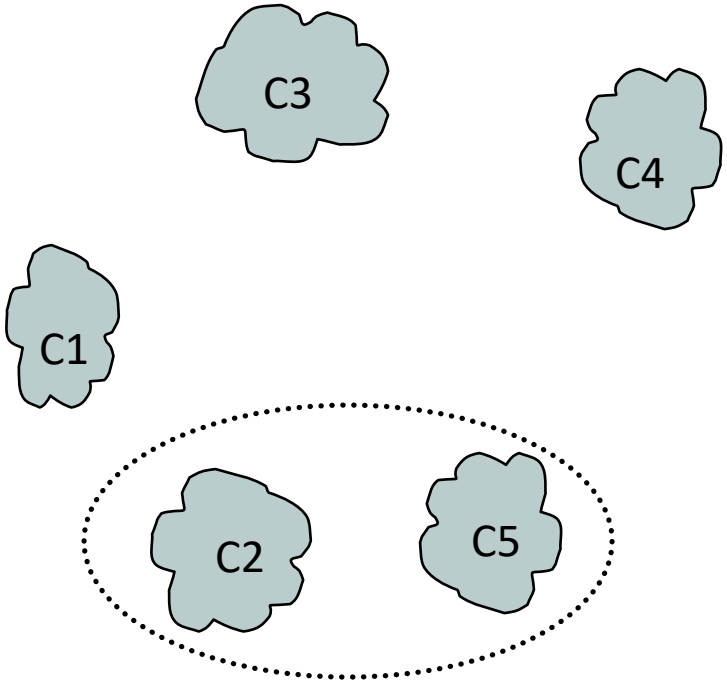
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



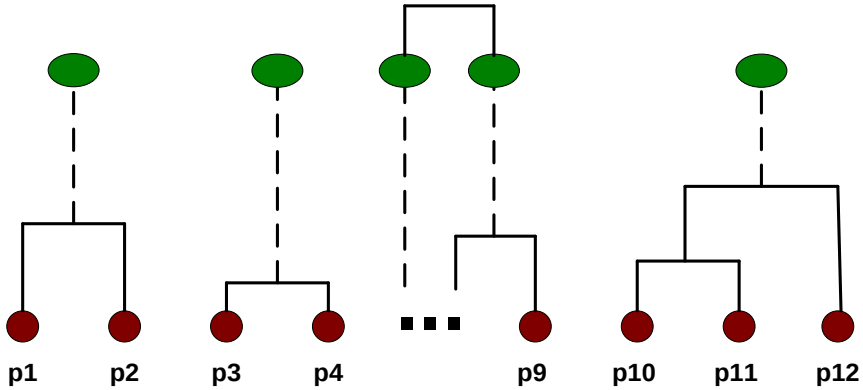
Intermediate Situation

We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



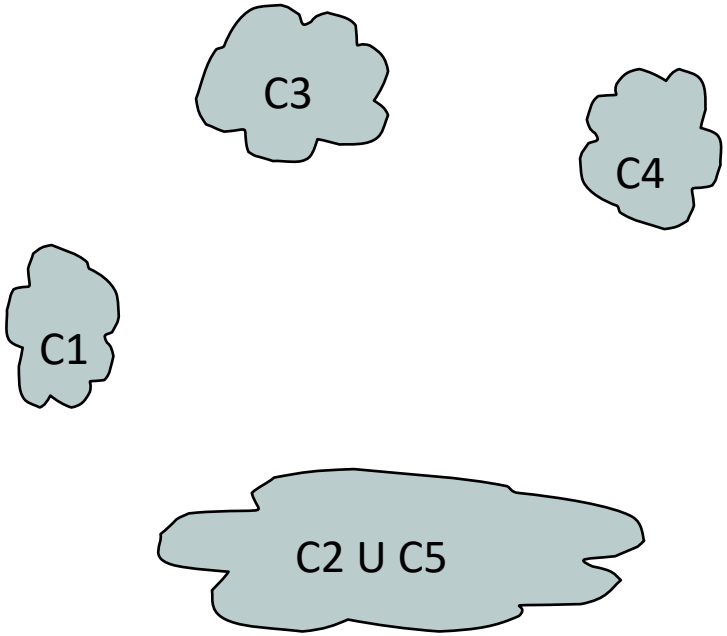
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



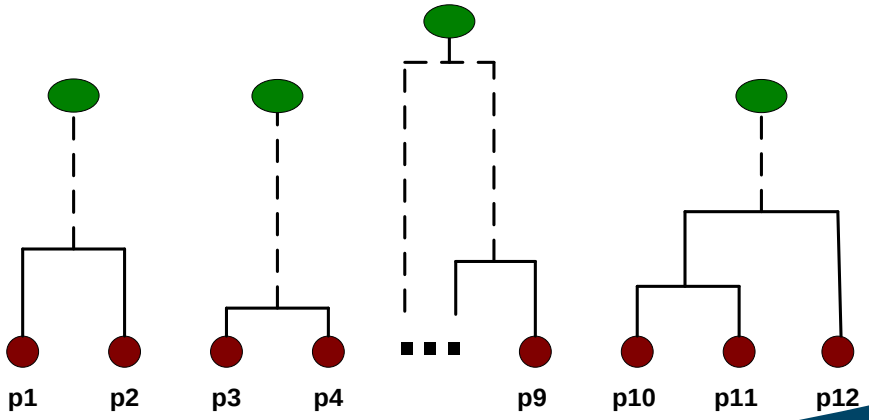
After Merging

The question is “How do we update the proximity matrix?”

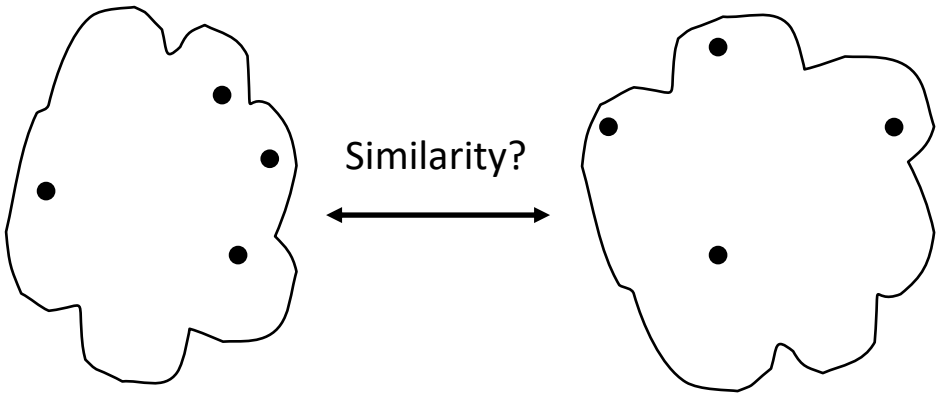


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

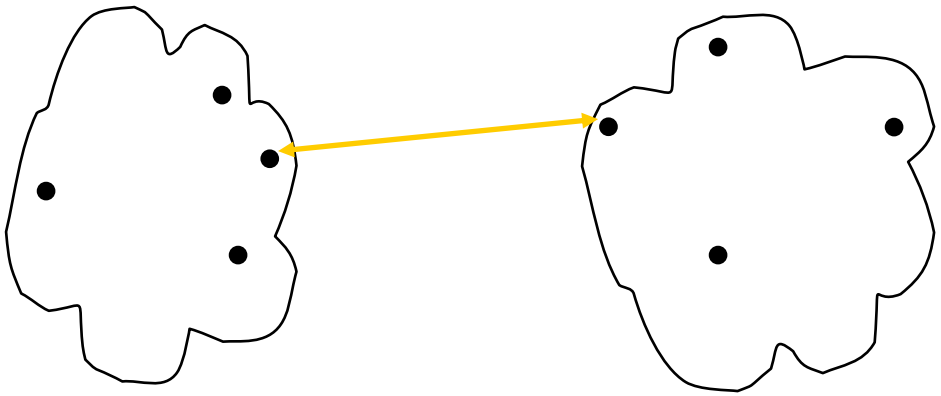


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Single Link

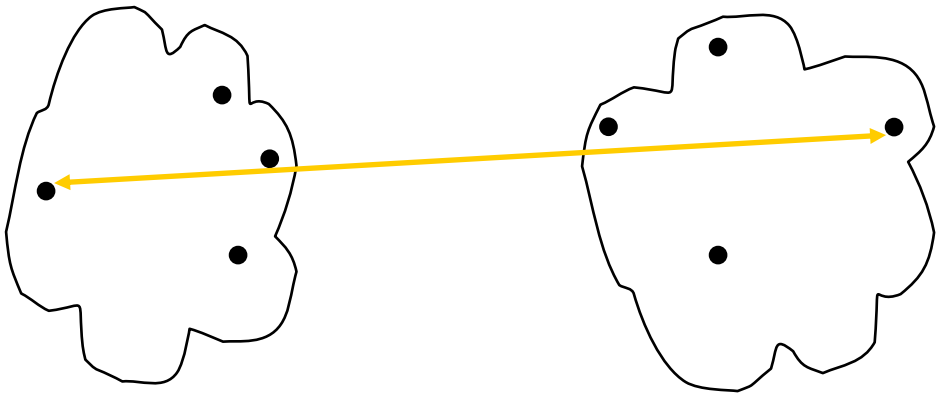


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Complete Link

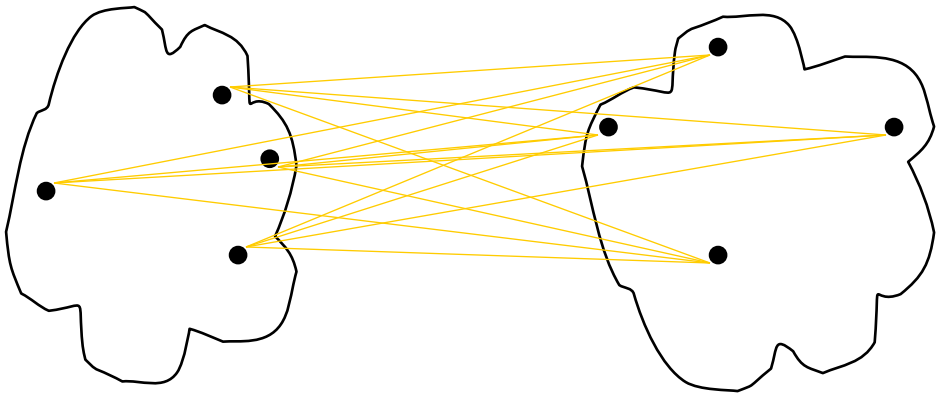


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Average Link

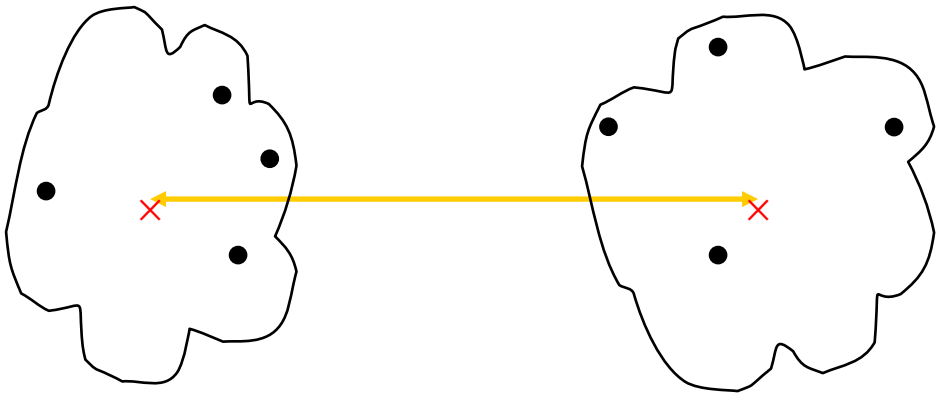


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Cluster Similarity: Ward's Method

Similarity of two clusters is based on the increase in squared error when two clusters are merged

- Similar to group average if distance between points is distance squared

Less susceptible to noise and outliers

Biased towards globular clusters

Hierarchical analogue of K-means

- Can be used to initialize K-means

Hierarchical Clustering: Time and Space requirements

$O(N^2)$ space since it uses the proximity matrix.

- N is the number of points.

$O(N^3)$ time in many cases

- Two bottlenecks:
  - The distance of the new cluster to O(N) other clusters needs to be determined
    - E.g., Ward's Criterion requires $O(|C_1||C_2|) = O(N^2)$ time
  - Out of the $O(N^2)$ cluster pairs, the one with smallest distance needs to be selected
    - We can use a heap: N deletes + N inserts require N log(N) time

Clustering by Compression

Distance/similarity measures can take many different forms

For instance, Cilibrasi and Vitanyi proposed a distance based on *compression*:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

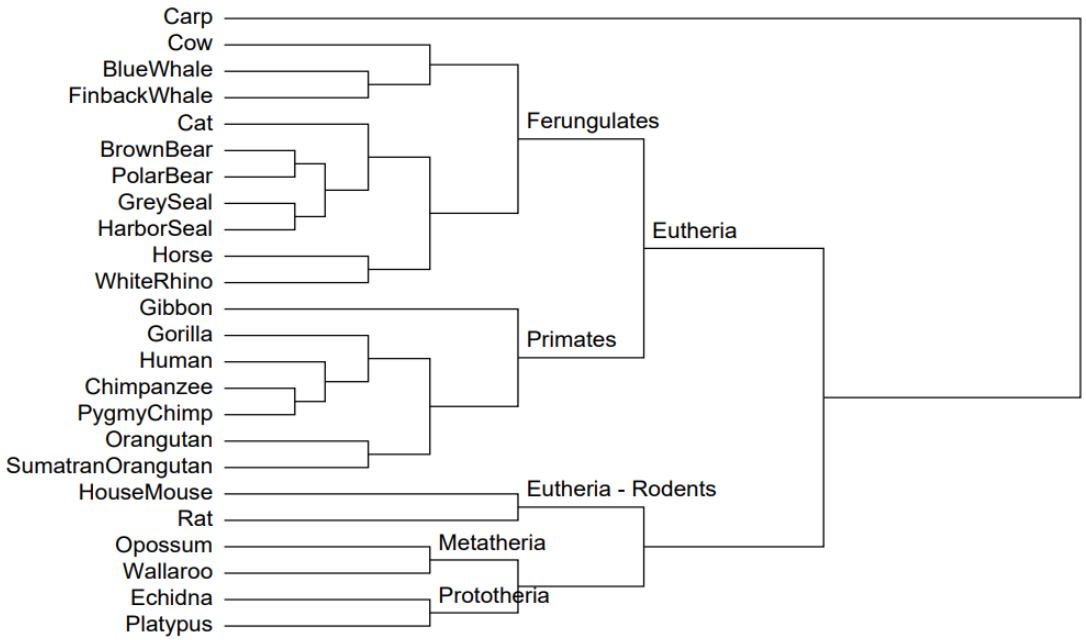
C(x) = size of x after compression

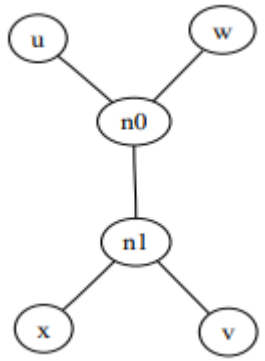
xy = concatenation of x and y

Cilibrasi, R., & Vitányi, P. M. (2005). Clustering by compression. *IEEE Transactions on Information theory*, *51*(4), 1523-1545.

Clustering by Compression

Validation of the distance measure:



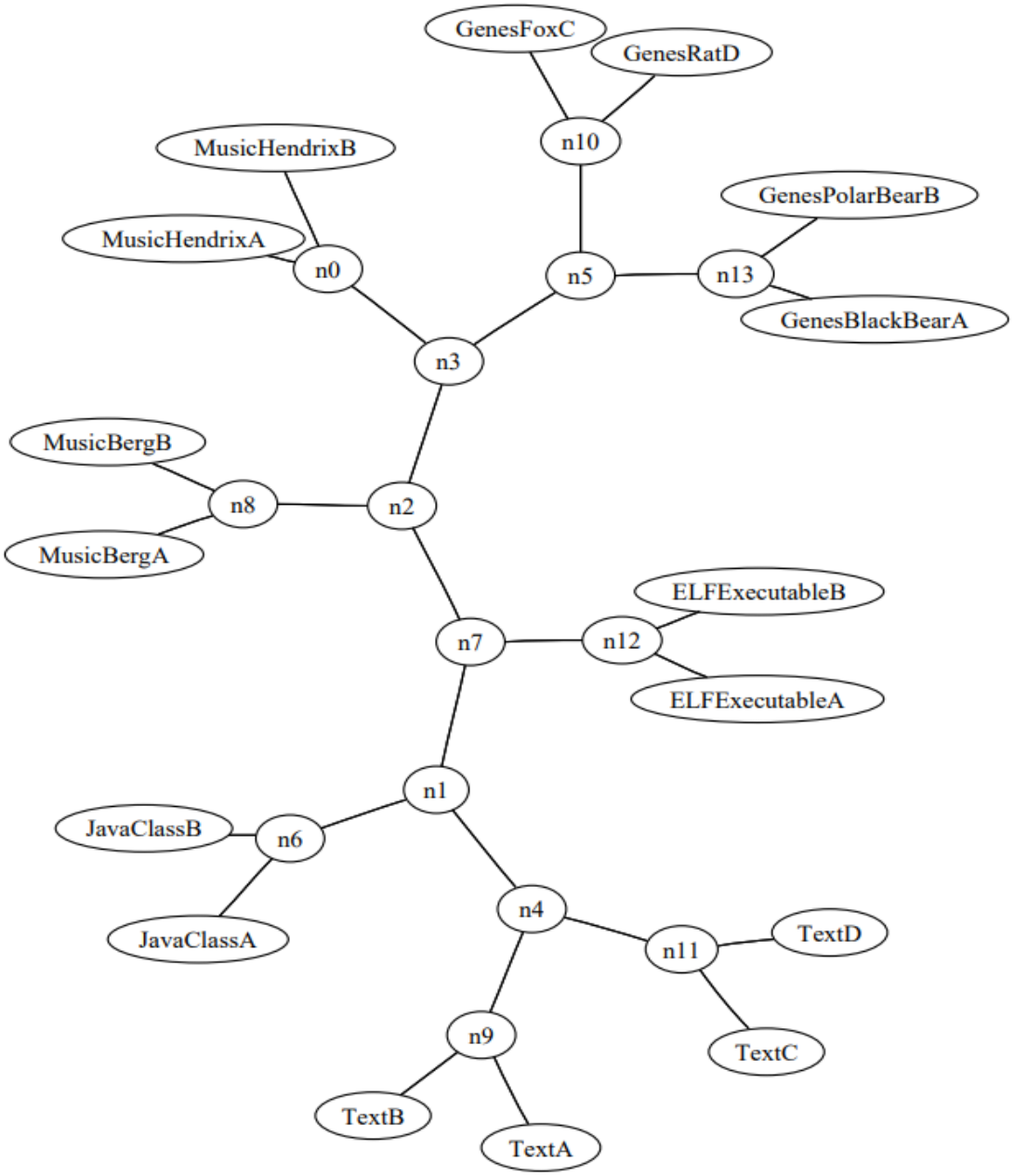


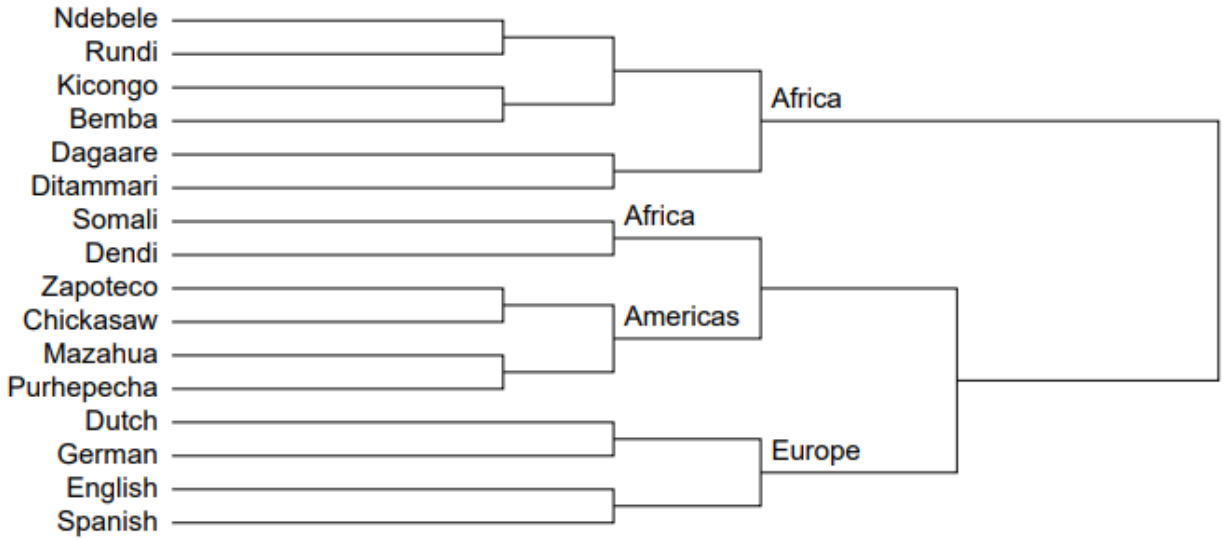
Quartet topology

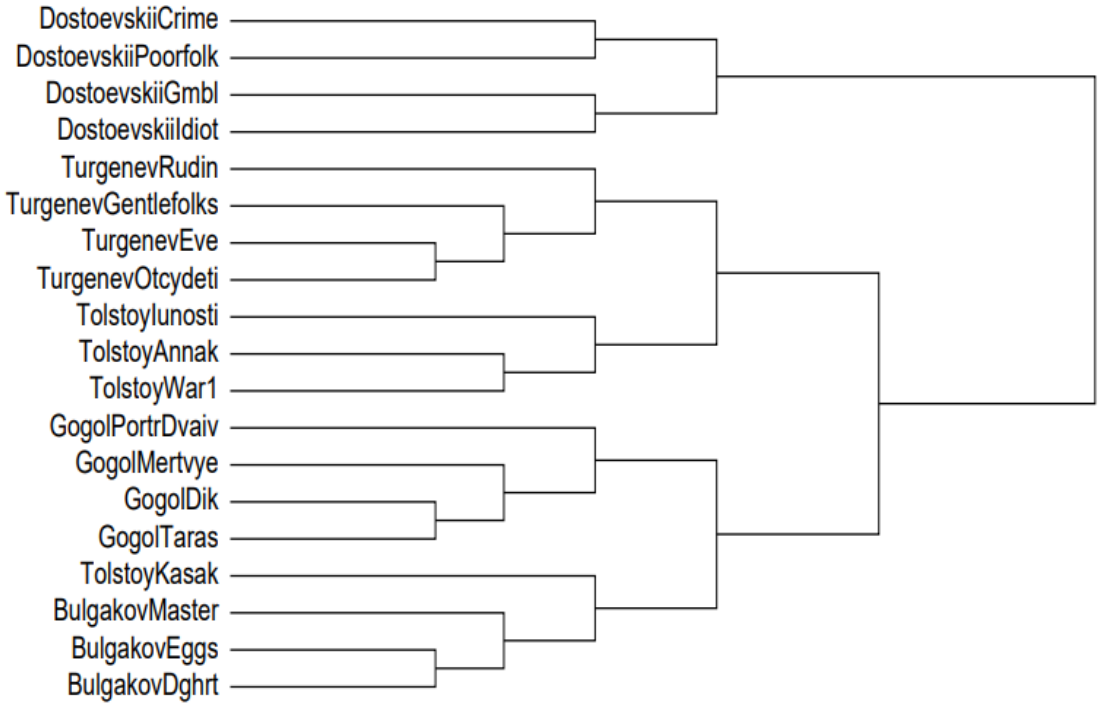
Known hierarchy; distance based on DNA

Consider al quarters and add up the distances of neighbours

This measure should be minimized


GenesFoxC
GenesRatD
n10
MusicHendrixB
GenesPolarBearB
MusicHendrixA
n0
n5
n13
GenesBlackBearA
n3
MusicBergB
n8
n2
MusicBergA
ELFExecutableB
n7
n12
ELFExecutableA
n1
JavaClassB
n6
n4
JavaClassA
n11
TextD
n9
TextC
TextB
TextA


Ndebele
Rundi
Kicongo
Bemba
Dagaare
Ditammari
Somali
Dendi
Zapoteco
Chickasaw
Mazahua
Purhepecha
Dutch
German
English
Spanish
Africa
Africa
Americas
Europe


DostoevskiiCrime
DostoevskiiPoorfolk
DostoevskiiGmbl
DostoevskiiIdiot
TurgenevRudin
TurgenevGentlefolks
TurgenevEve
TurgenevOtcydeti
Tolstoylunosti
TolstoyAnnak
TolstoyWar1
GogolPortrDvaiv
GogolMertvye
GogolDik
GogolTaras
TolstoyKasak
BulgakovMaster
BulgakovEggs
BulgakovDghrt

Outline

Partitional Clustering

- Distance-based
- Density-based
- Model-based (Expectation-Maximization)

Hierarchical Clustering

Clustering and visualizing simultaneously

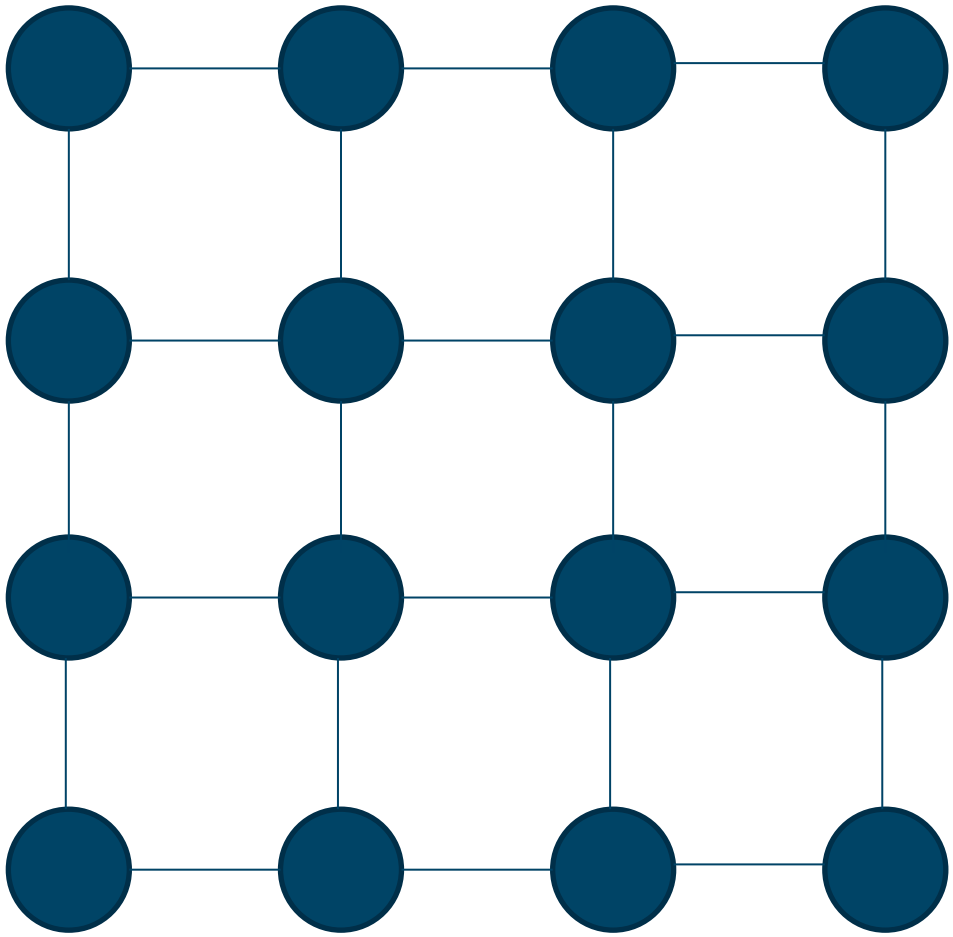
- Self-Organizing Maps
- tSNE and Multi-Dimensional Scaling

Cluster validity

Self-Organizing Maps

At the same time cluster, and assign to a map

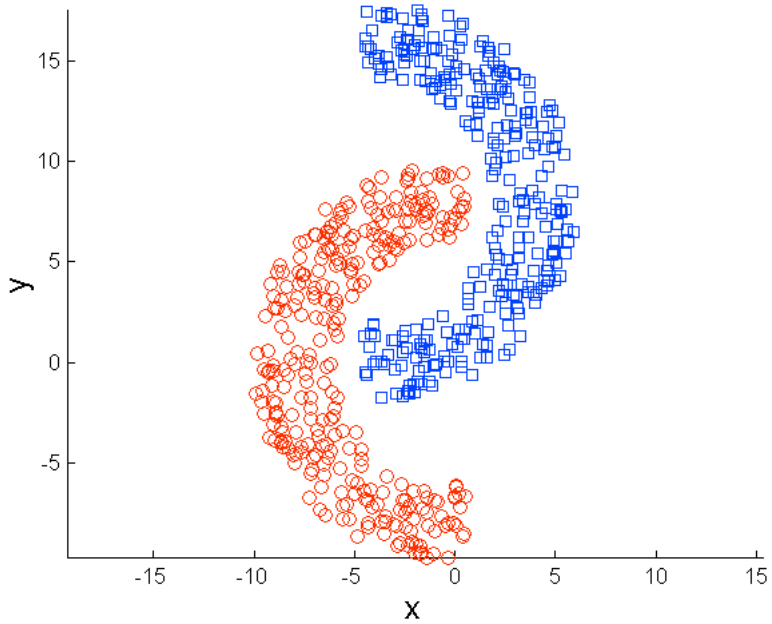
Similar points should map to similar regions in the map

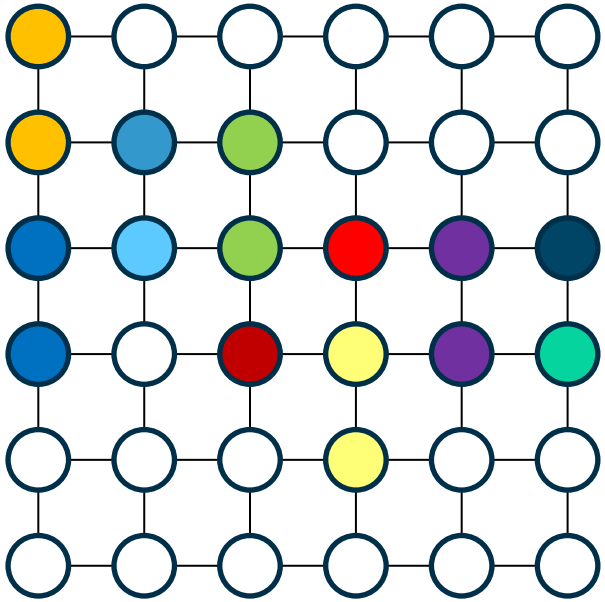


Each node represents a cluster and is defined by a centre much like k-means.

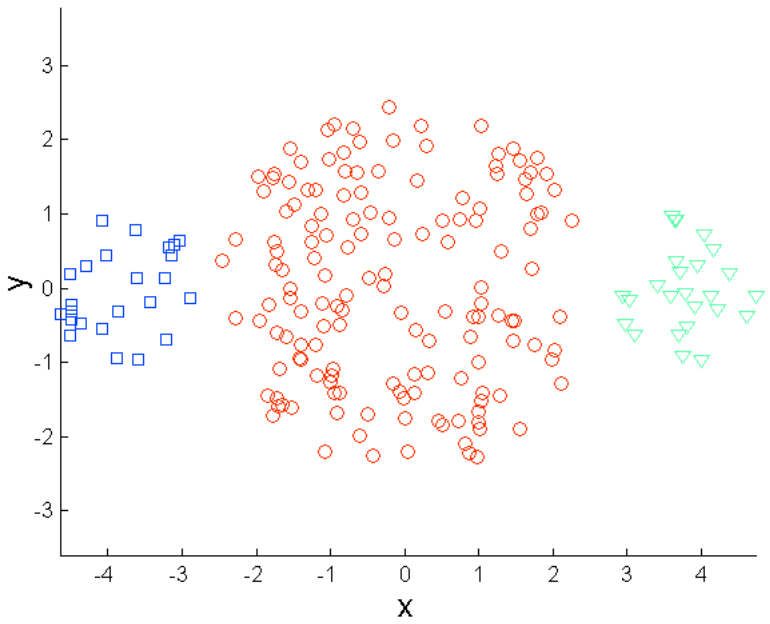
Nearby cluster centres should be similar.

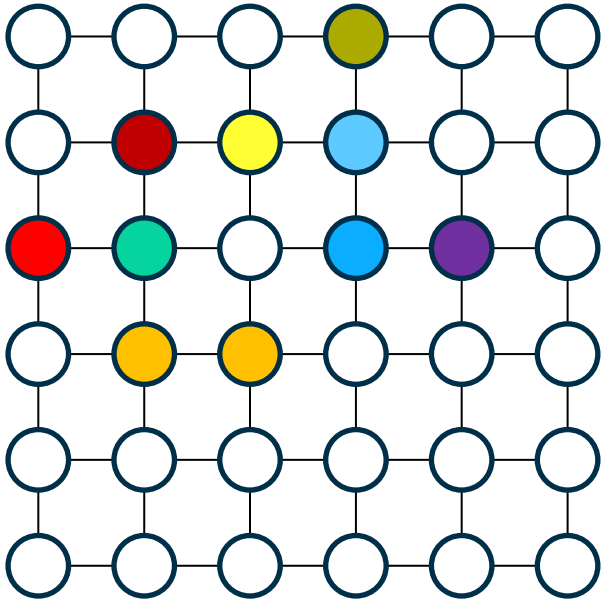
Clusters and Their Relations





Clusters and Their Relations





SOM: High-level overview

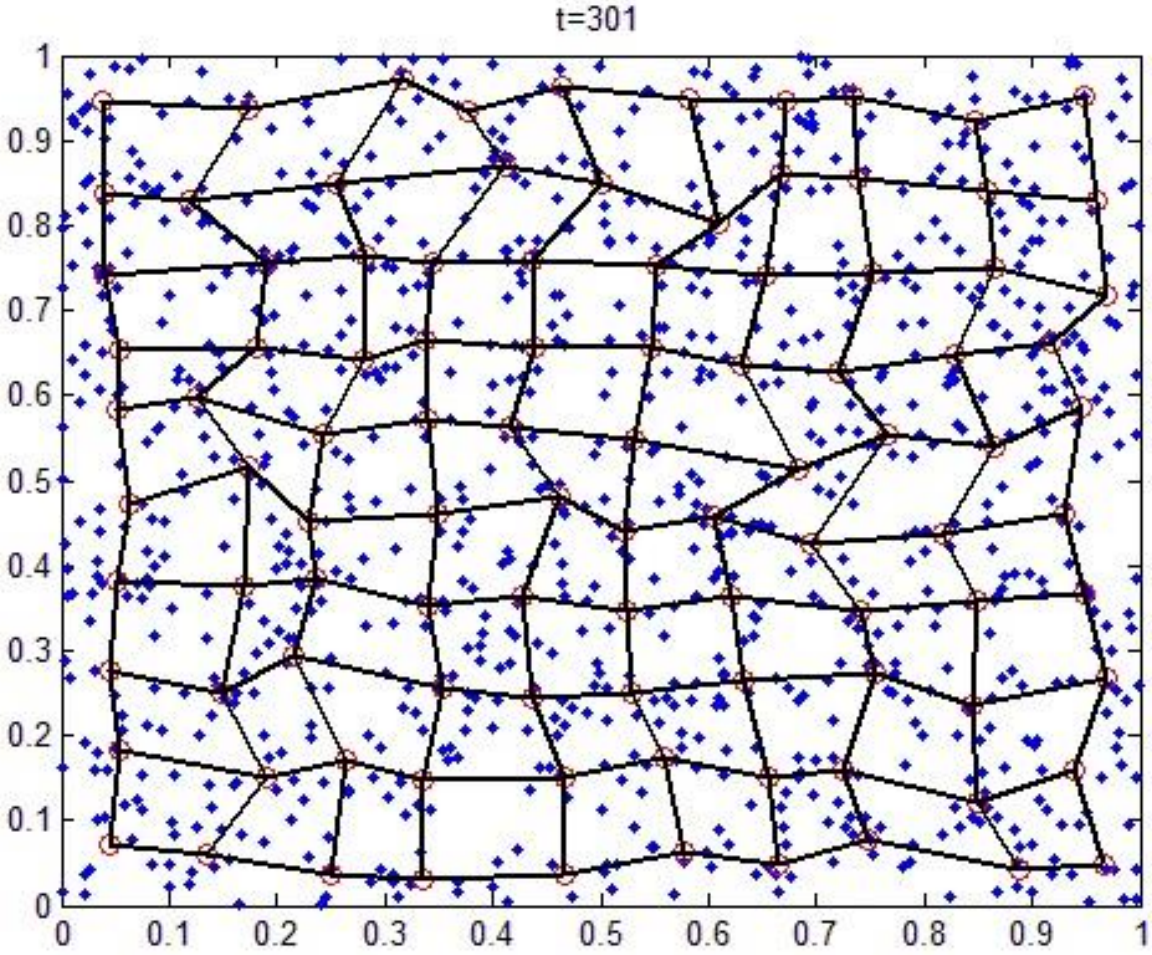
Assign initial weights to each “neuron”

Repeat

- Randomly pick next example from the dataset
- Select closest centre
- Update centre and its neighbours

Until convergence / maximal number of epochs

SOM: Illustration



https://www.youtube.com/watch?v=QvI6L-KqsT4

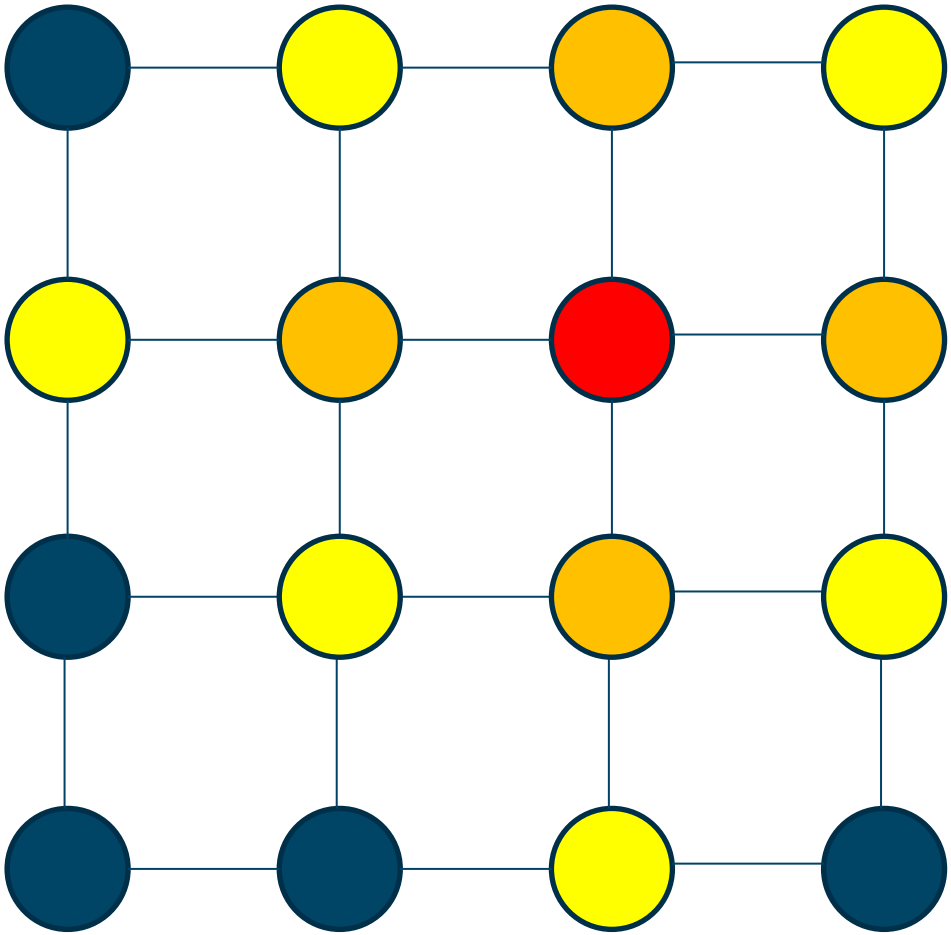
SOM: Initialize Weights

Assume the data is k-dimensional and we have an n x n grid

- Option 1: Randomly initialize centres or randomly select n^2 points to serve as centres
- Option 2: Apply PCA, and assign the centres uniformly over the 2D space spanned by the first 2 principle components

SOM: Select Neighbours

Gradually decrease neighbourhood



Red = closest centre

During first steps:

Update red, orange, yellow

Later:

Update red, orange

At the end:

Only red

SOM: Update Weights

Move the selected centres towards the data point:

$$W_v(s+1) = W_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (D(t) - W_v(s))$$

$W_v(s)$: weights of node v at t

u : Closest centre

D(t) : Randomly selected training example

Θ(u,v,s) : Similarity between u and v at time s

α(s) : Learning rate at time s

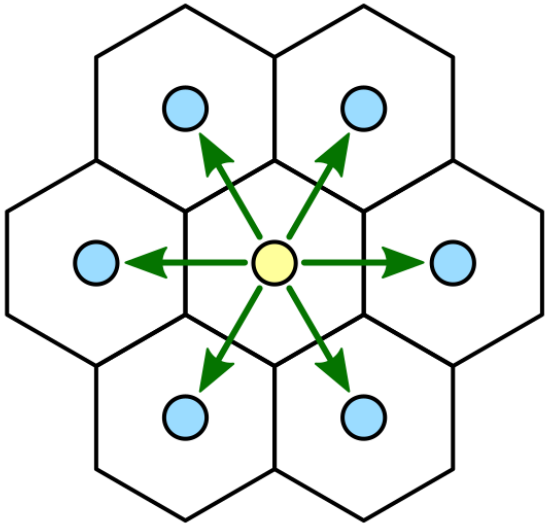
Decreases as s increases
Can be 0-1 (neighbour or not)
Or e.g., Gaussian

Decreases as s increases; so changes fade out as time progresses

Different topologies

Grid (examples above)

Hexagon



Ring: https://www.youtube.com/watch?v=ddbBlWAlzxE

Visualizing Resulting SOM

Use topology of the SOM

Use colours to represent properties of the centres

- e.g., distribution of variables, ratios

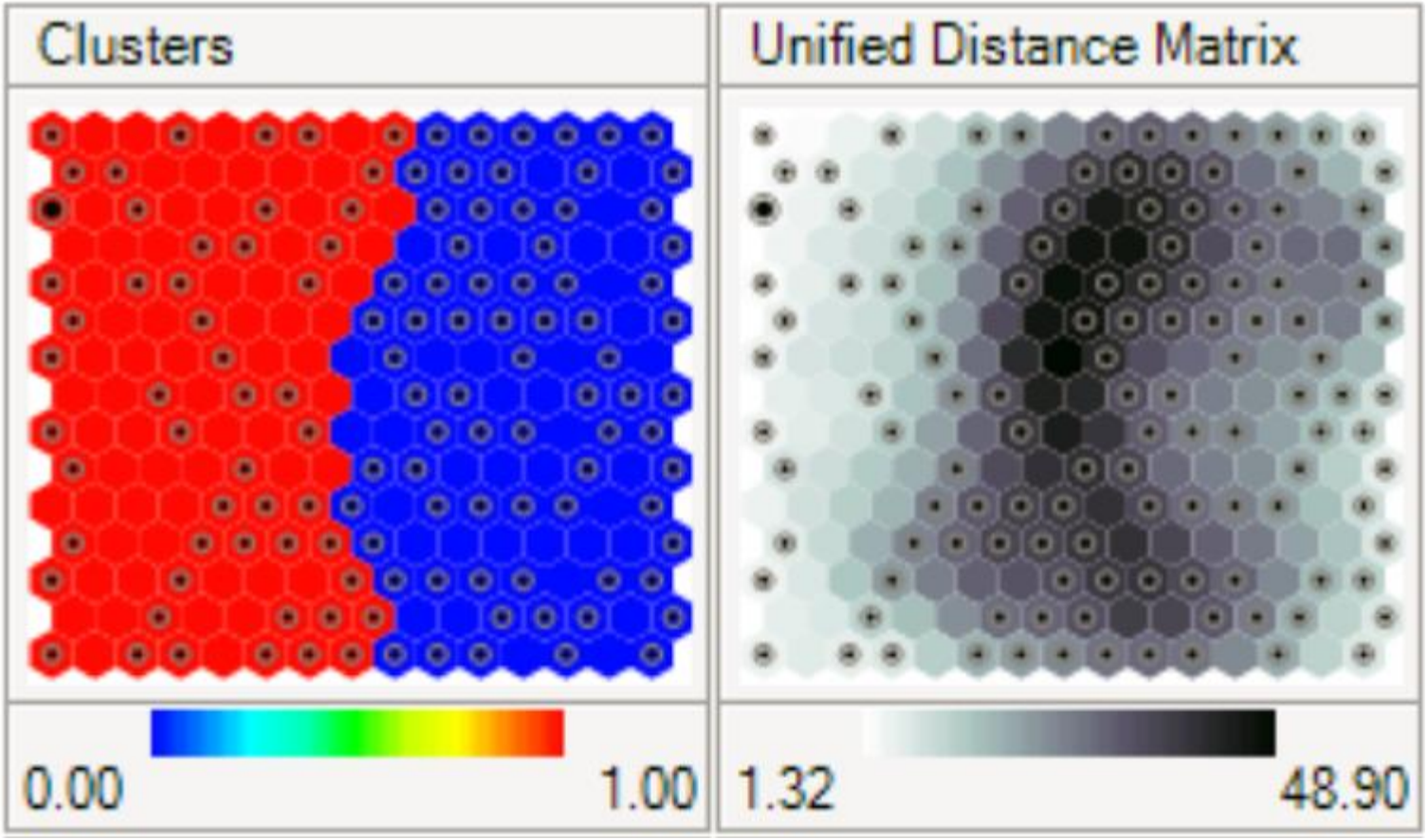
Specific examples can be associated to their centre

U-map: same topology as SOM

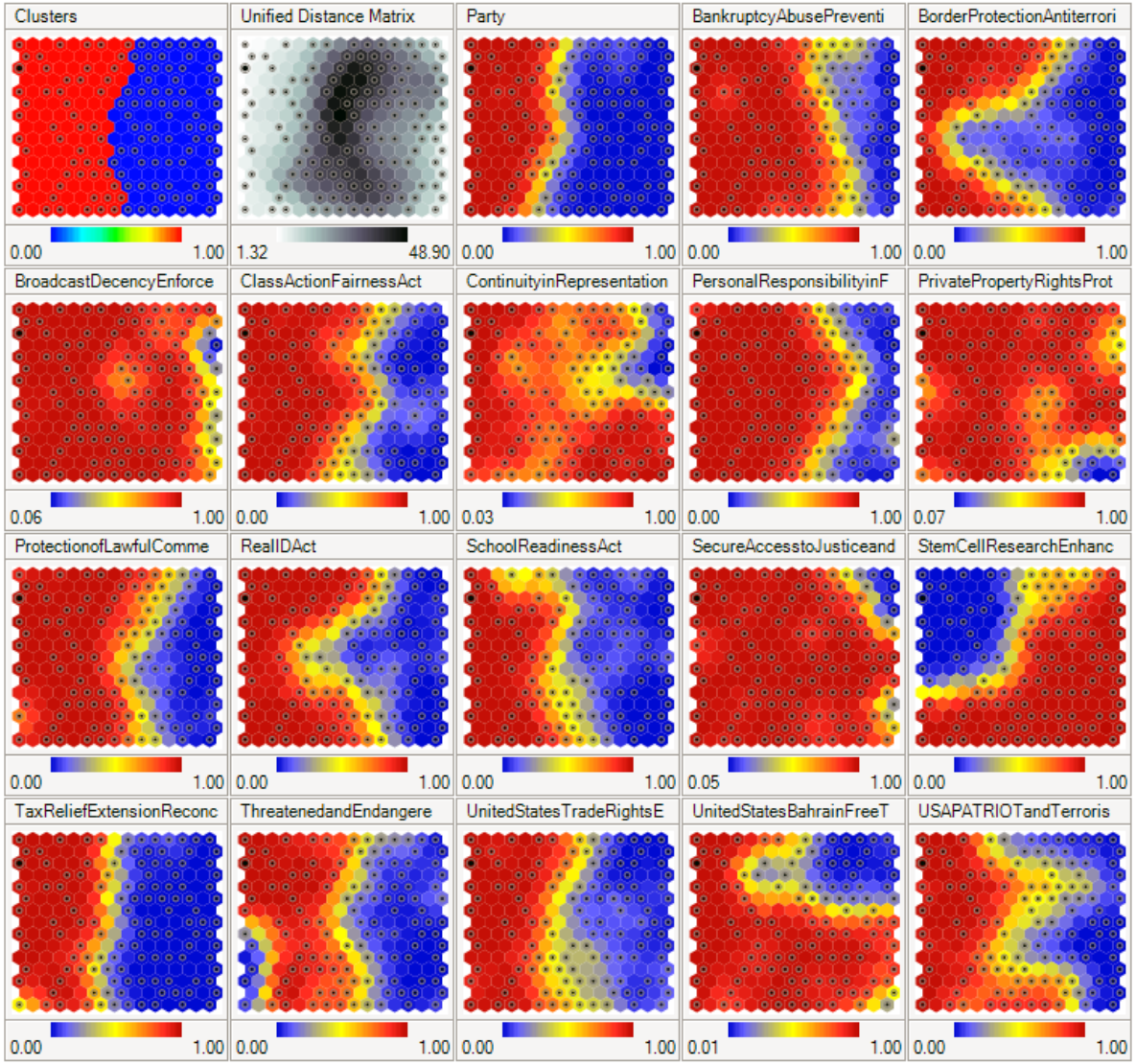
Plots the average distance to the neighbors

- Can be used to reveal cluster information; core points have smaller distance than border points

SOM: Illustration

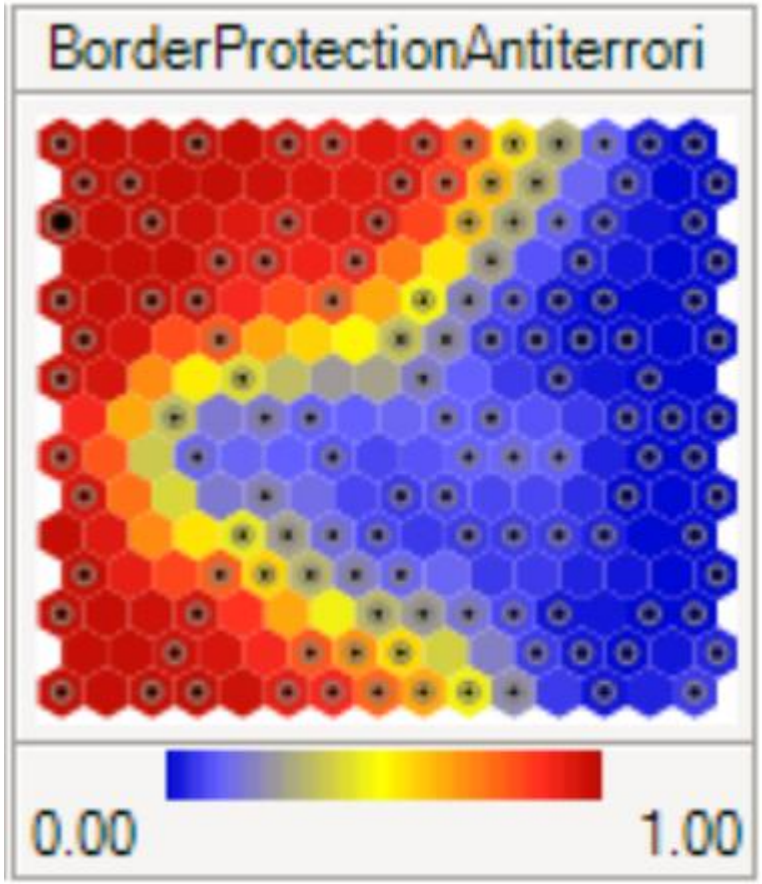


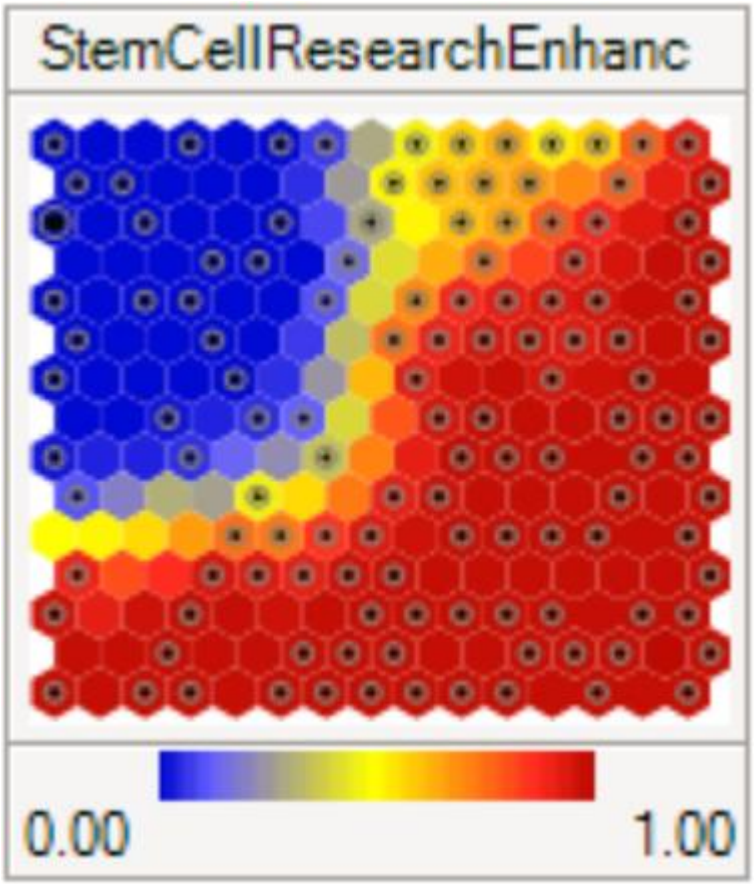
SOM: Illustration



Self-Organizing Map showing US congress voting results. Screenshot from Peltarion Synapse. Source: https://en.wikipedia.org/wiki/Self-organizing_map

SOM: Illustration





Similar Idea: MDS, tSNE

Embed data points in a lower dimensional space

Distances should be preserved somehow

MDS: Multi-dimensional scaling

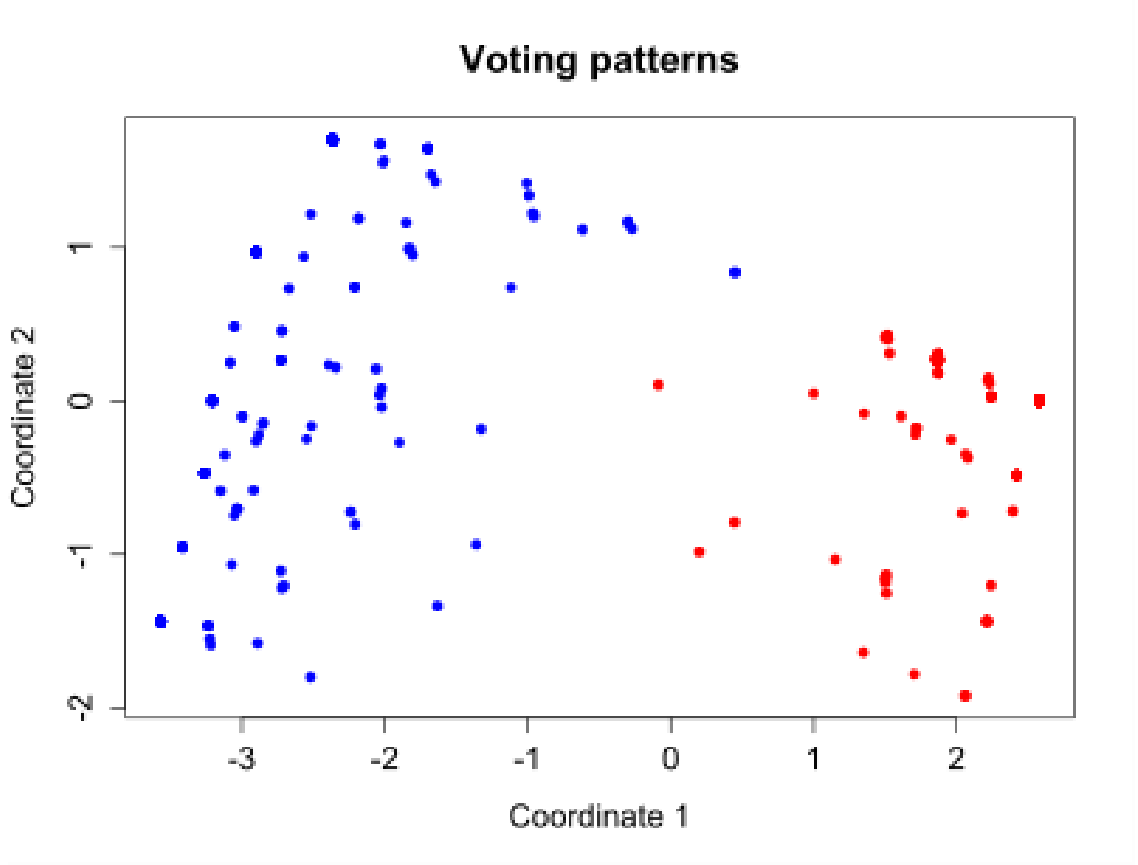
Given a dissimilarity matrix:

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix}$$

Find coordinates $x_1, \ldots, x_I$ that lead to similar distances:

$$\min_{x_1,\ldots,x_I} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2$$

MDS: Example



Outline

Partitional Clustering

- Distance-based
  - K-means, K-medoids, Bisecting K-means
- Density-based
  - DBSCAN
- Expectation-Maximization

Hierarchical Clustering

Cluster validity

Cluster Validity

For supervised classification we have a variety of measures to evaluate how good our model is

- Accuracy, precision, recall

For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?

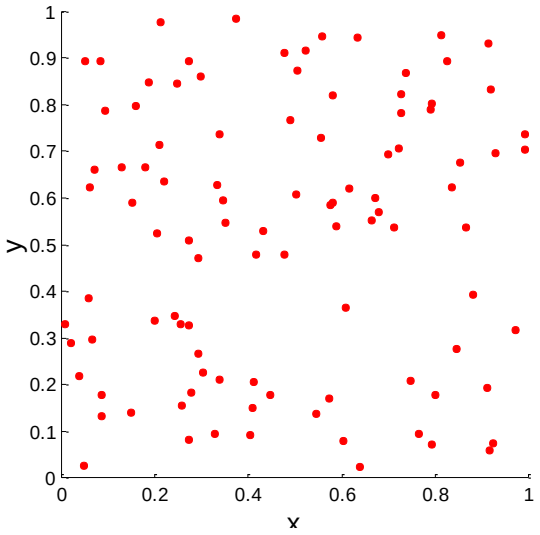
But “clusters are in the eye of the beholder”!

Then why do we want to evaluate them?

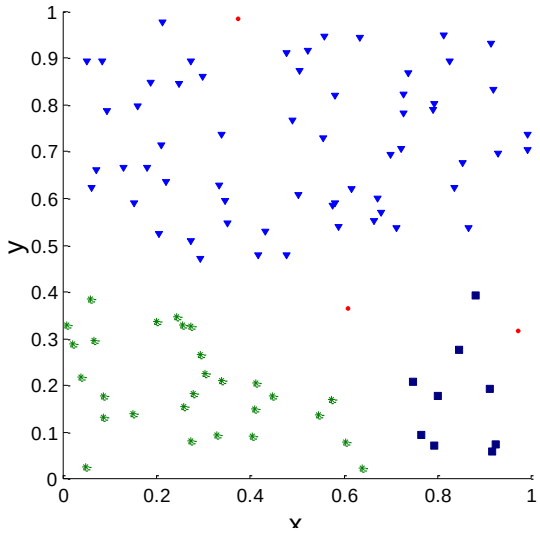
- To avoid finding patterns in noise
- To compare clustering algorithms
- To compare two sets of clusters
- To compare two clusters

Clusters found in Random Data

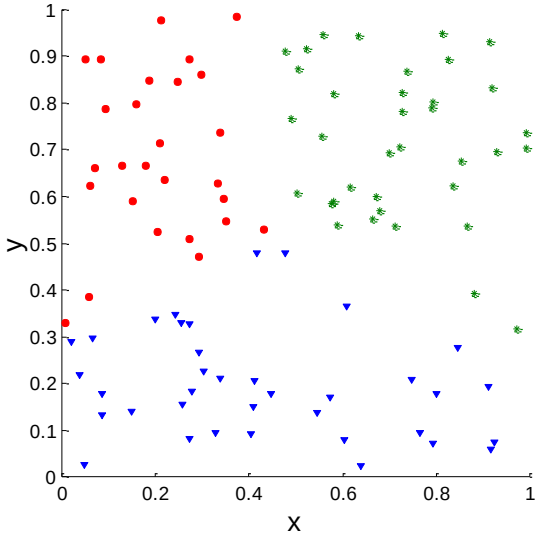
Random Points



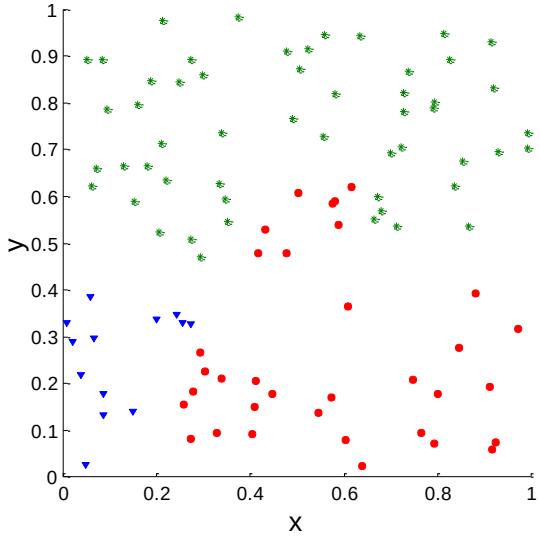
DBSCAN



K-means

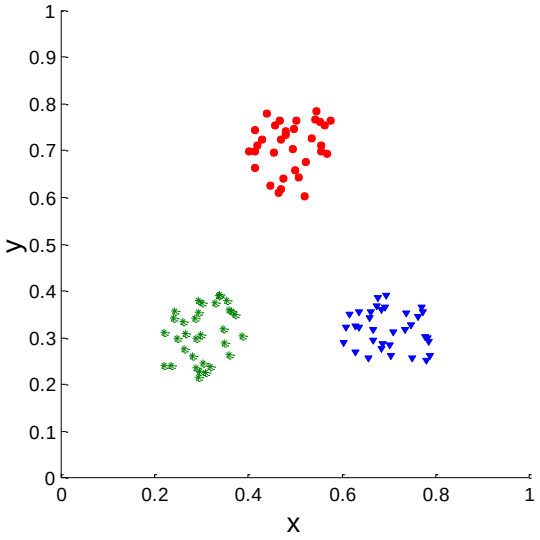


Complete Link

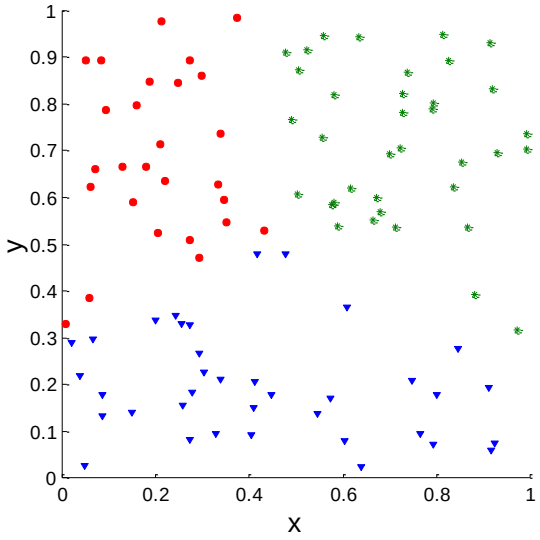


Measuring Cluster Validity Via Correlation

Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.



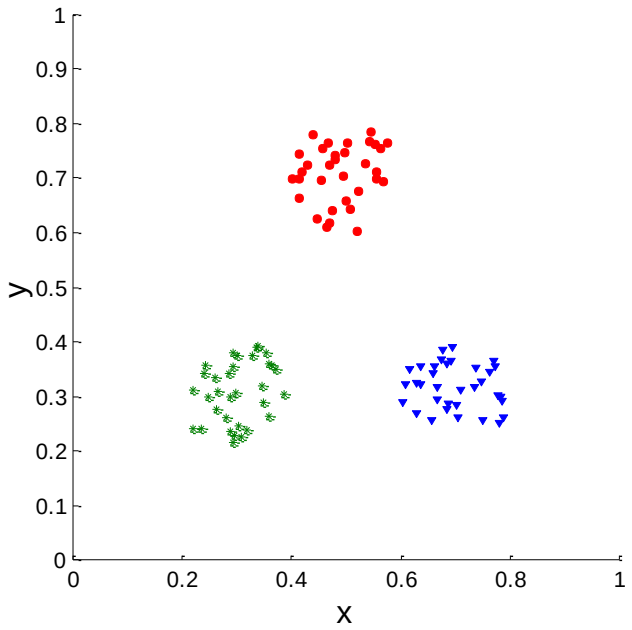
Corr = -0.9235

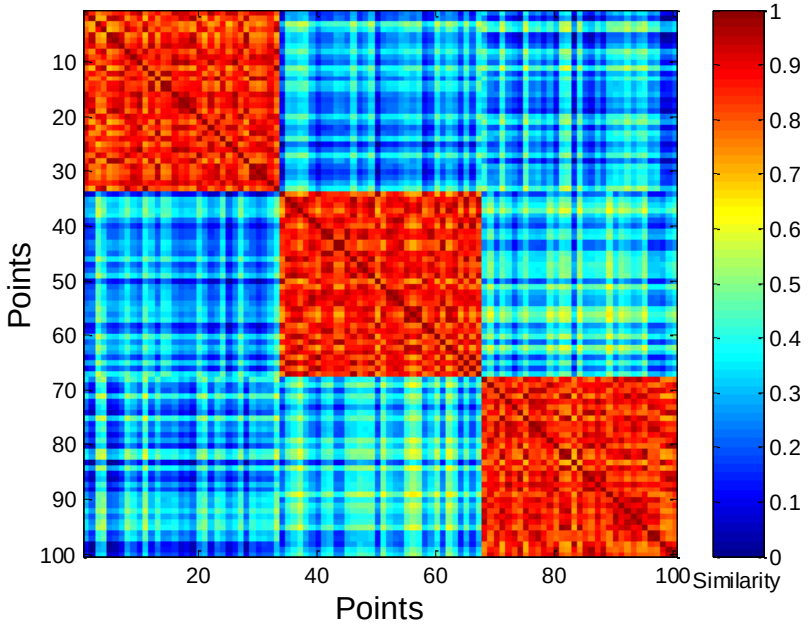


Corr = -0.5810

Using Similarity Matrix for Cluster Validation

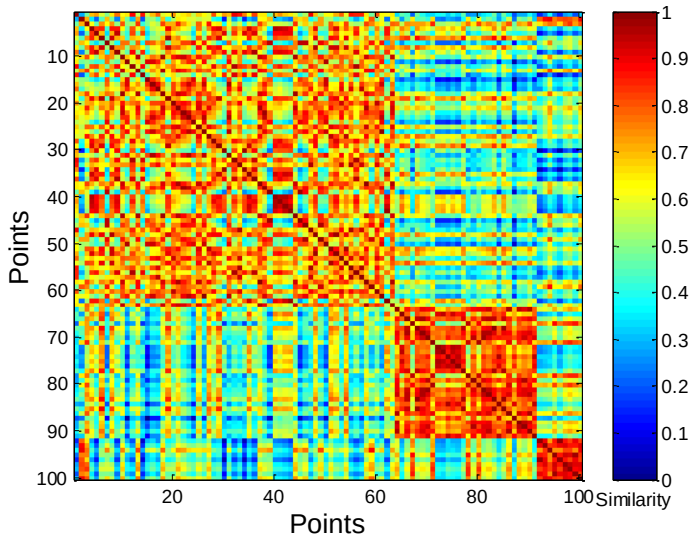
Order the similarity matrix with respect to cluster labels and inspect visually.

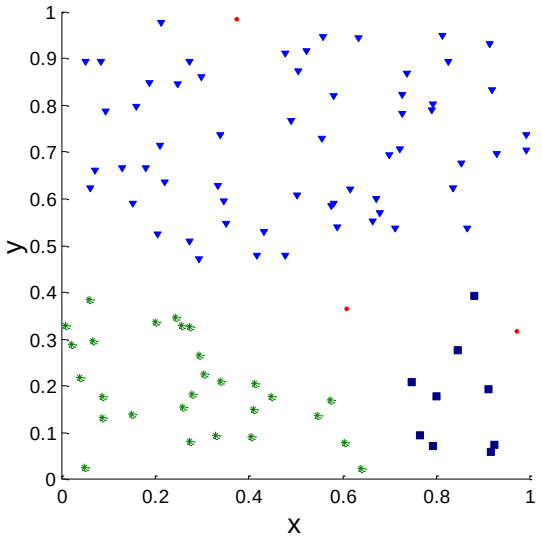




Using Similarity Matrix for Cluster Validation

Clusters in random data are not so crisp

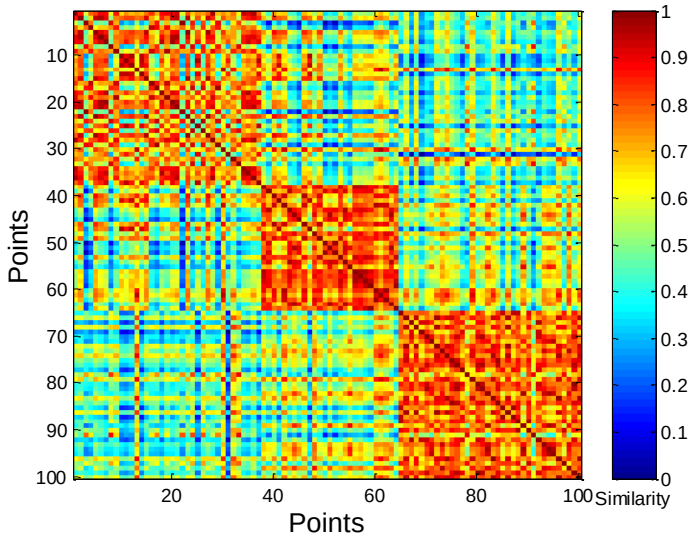


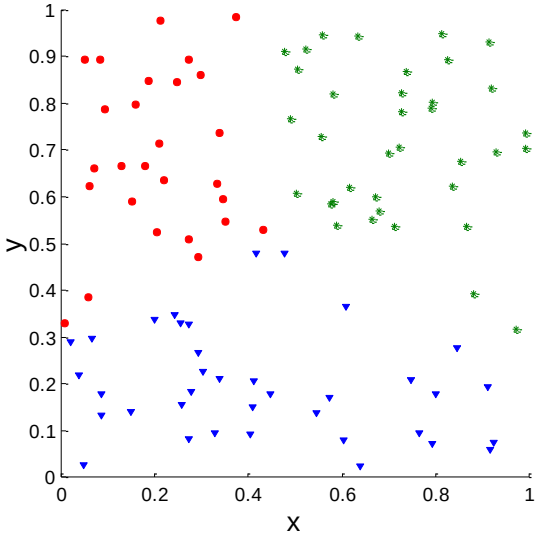


DBSCAN

Using Similarity Matrix for Cluster Validation

Clusters in random data are not so crisp

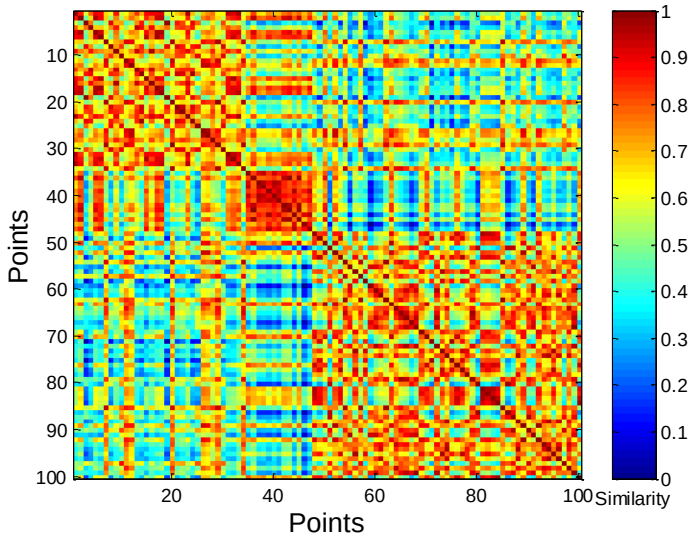


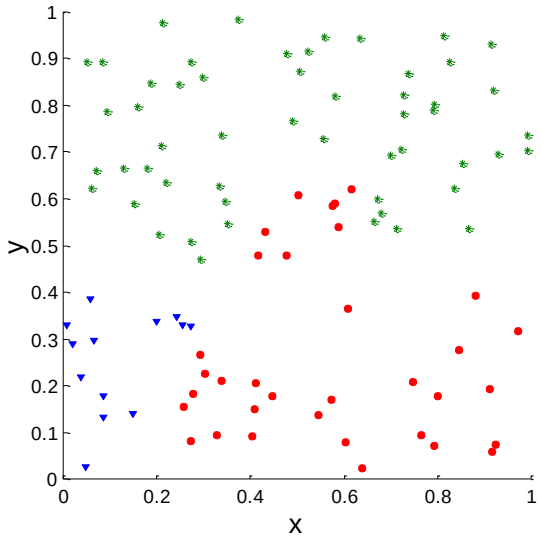


K-means

Using Similarity Matrix for Cluster Validation

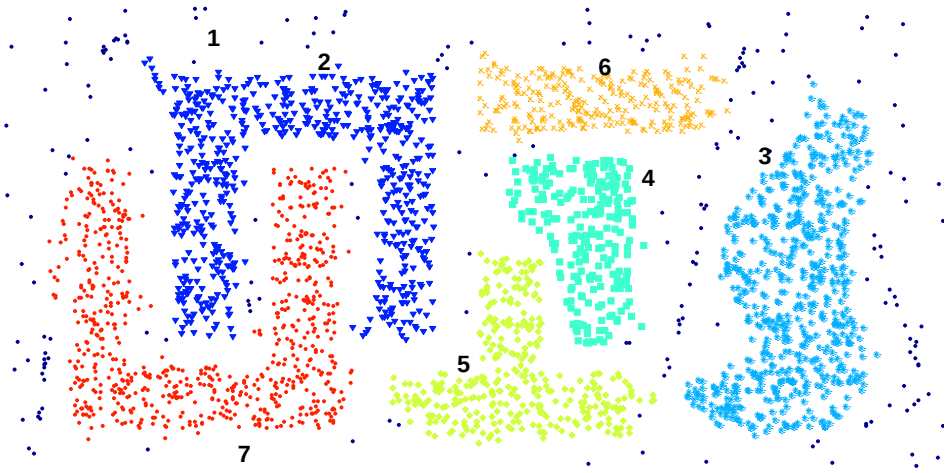
Clusters in random data are not so crisp

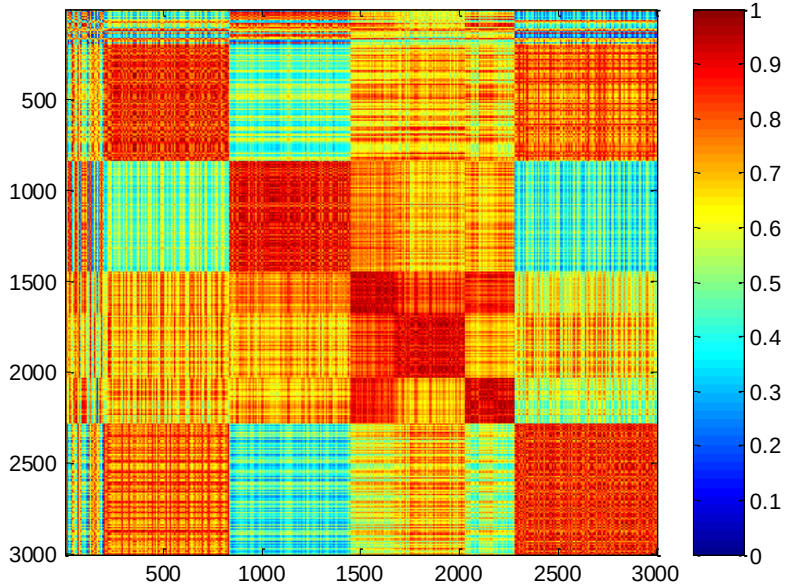




Complete Link

Using Similarity Matrix for Cluster Validation





DBSCAN

Generate reference value for SSE

SSE is a good measure to compare clusterings

But how to know if we found true structure in the data?

We need a reference SSE value!

We could rely on *permutation test*:

- Randomize the data while maintaining certain properties
  - For instance: permute column-wise; uni-variate distributions of the individual features are maintained, but joint distribution is distorted
- Cluster the randomized data to generate a reference SSE
  - Repeat several times

Case Study

- Selected 12 web-pages
  - 3 news paper front pages (standard, nieuwsblad, standaard)
  - 1 website with football news (voetbalprimeur)
  - 1 website for fans of football team Antwerp
  - 3 web pages of Python documentation
  - 2 web pages of C++ documentation
  - 2 former home pages of myself (@TU/e and @ULB)

Preprocessing

HTML → Text extraction → remove all non-alphanumeric characters → split on white-space → lower case

<h1>RAFC Kampioen 2016-2017 in 1b</h2>

→

{rafc, kampioen, in, b}

Hence, every document corresponds to a set of tokens.

Can also straightforwardly be represented as 0-1 bitvectors

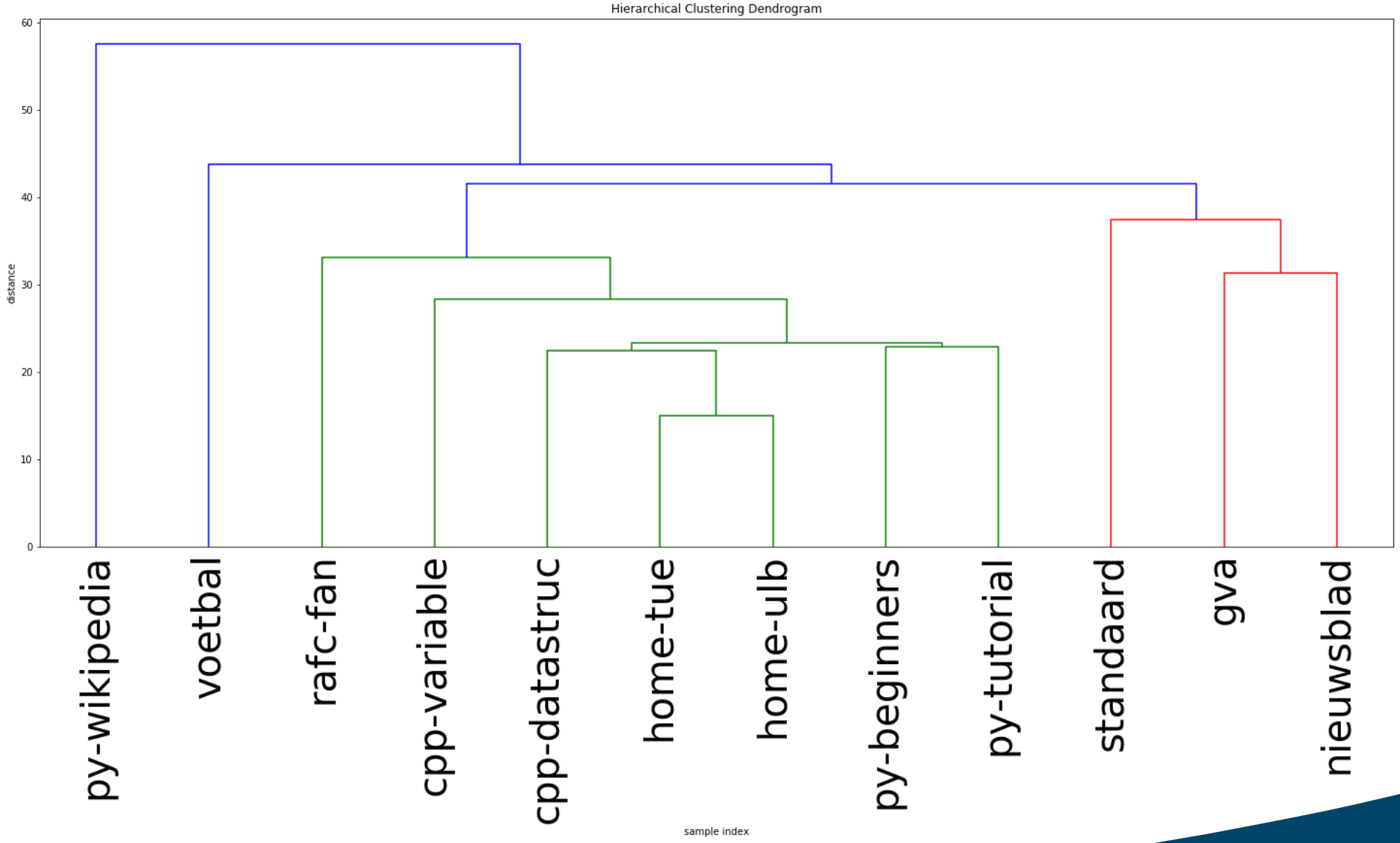
Distance measures

The documents were clustered using hierarchical clustering with the following 3 distance measures:

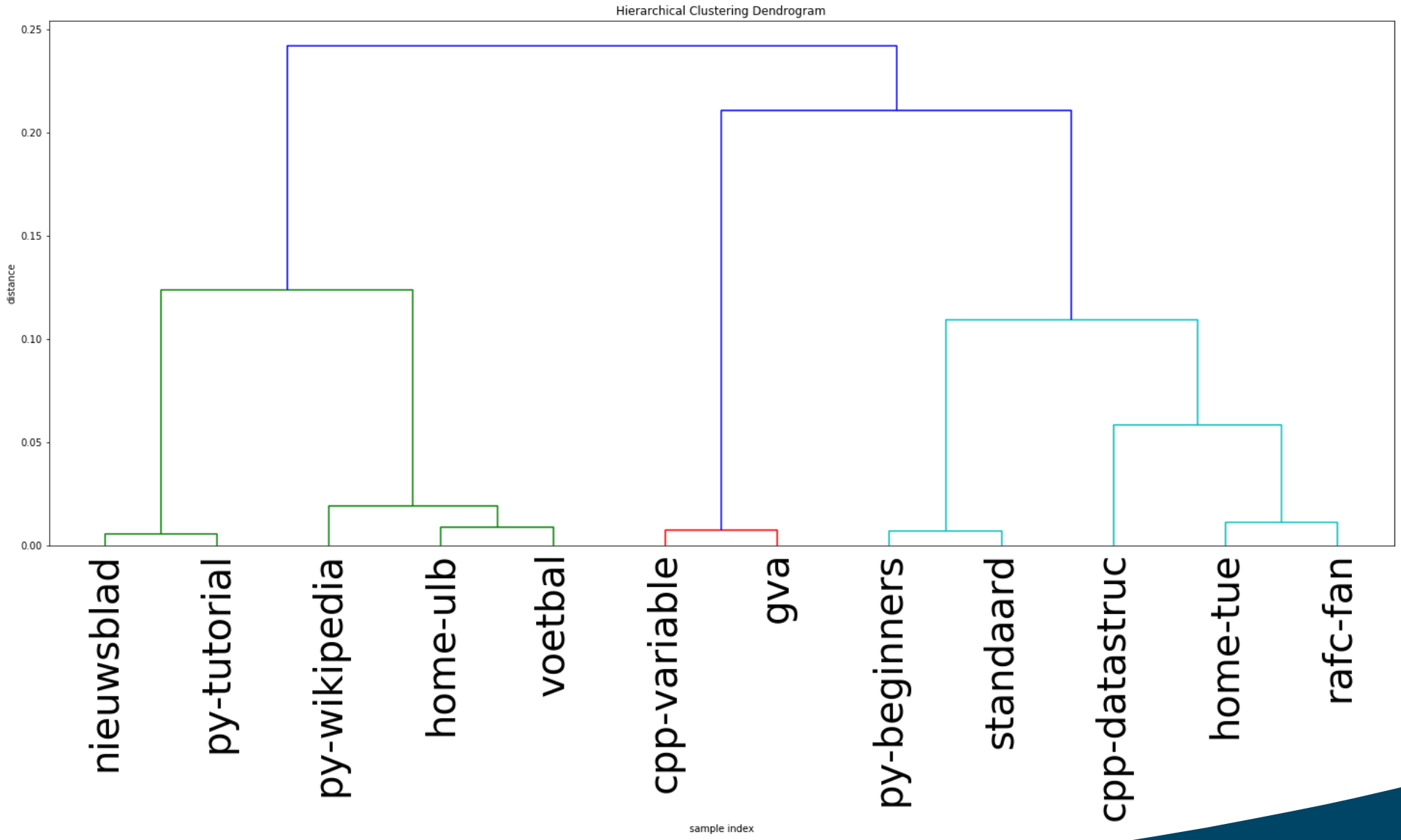
- Hamming distance
- Euclidian distance
- 1 - Cosine similarity

Goal is to illustrate the importance of selecting the right distance measure

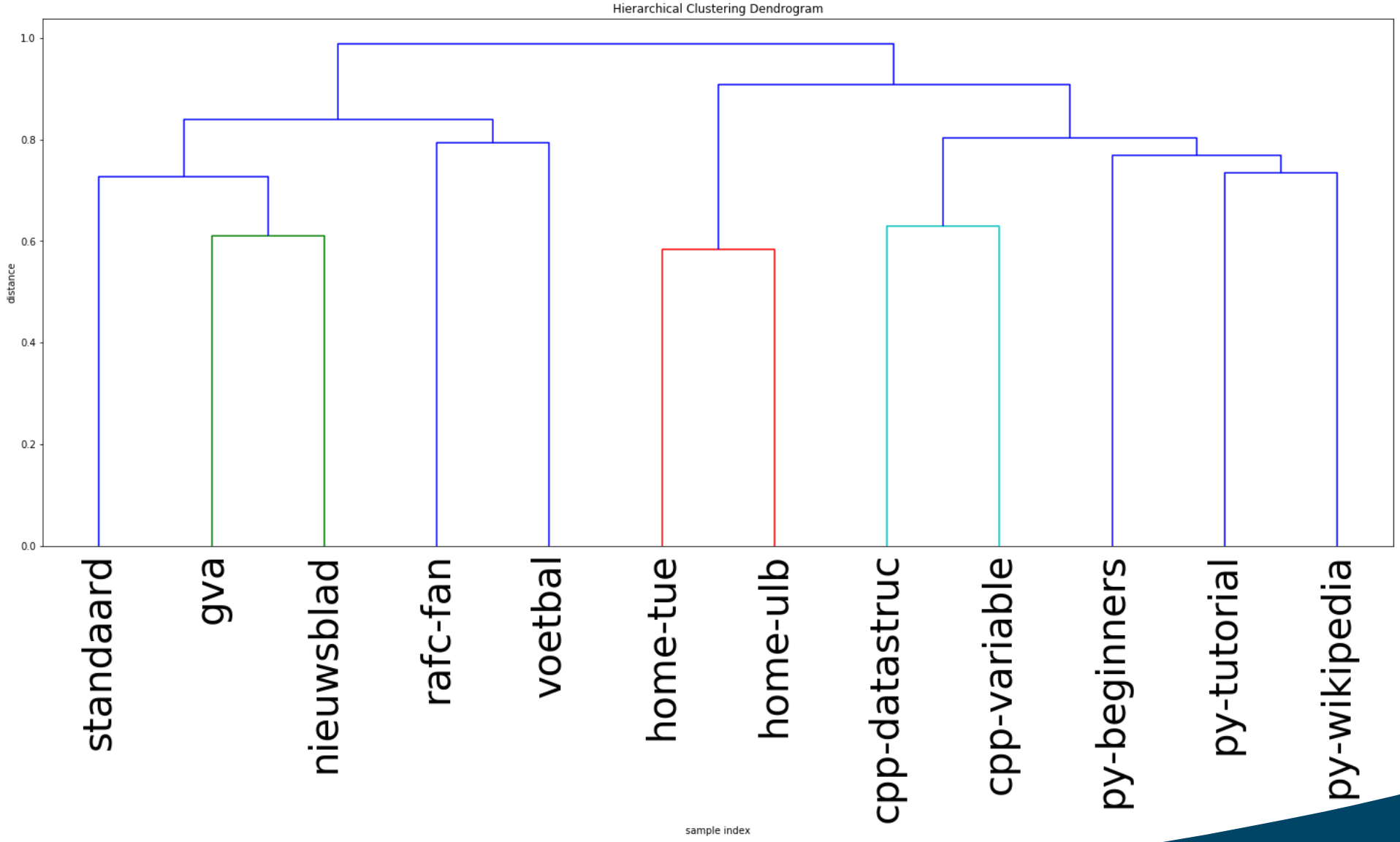
Dendrogram for Euclidian Distance



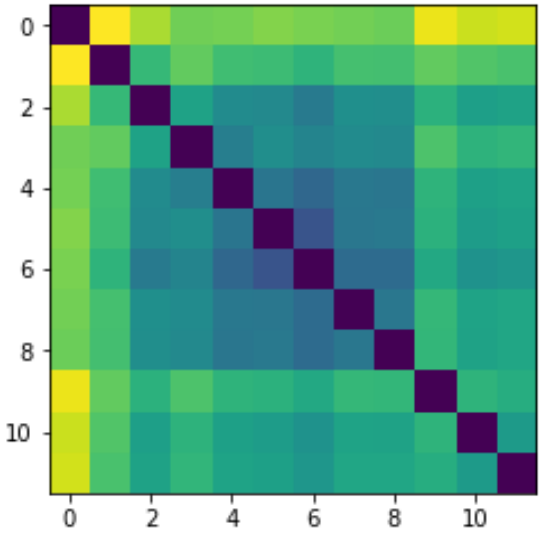
Dendrogram for Hamming Distance



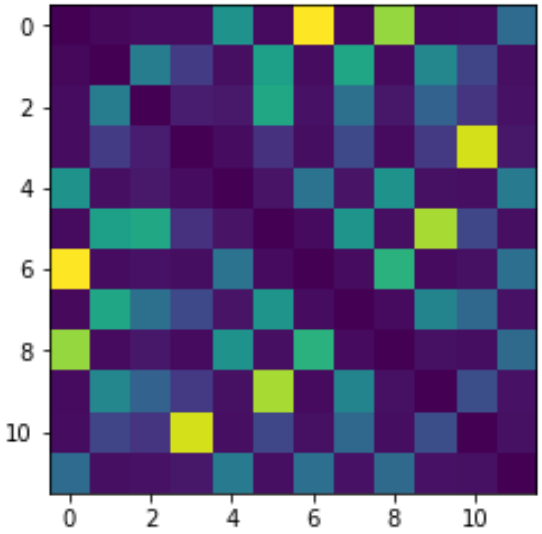
Dendrogram for Cosine Distance



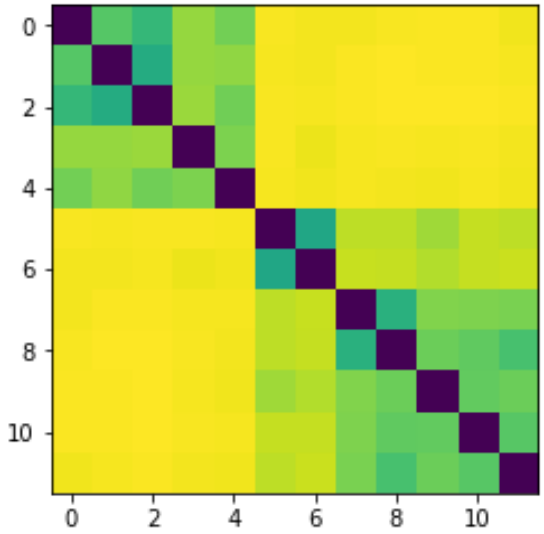
Similarity Matrices



Euclidian



Hamming



Cosine

Case Study: Observations

- Way of computing cluster distance of little influence
  - Min, max, avg produce exactly the same results
- Count produces slightly better results than Boolean

- Euclidian distance & SMC perform extremely poorly
- Cosine measure and Jaccard produce (almost) perfect result

Case Study: Take-Away Message

- There are three important points when performing clustering:
  - Choose a suitable distance measure
  - Choose a suitable distance measure
  - Choose a suitable distance measure (really!)

- Much more important than choosing the right algorithm …

A note on outlier detection

Outlier-detection is an important by-product of clustering

Outlier or anomaly detection has many applications

- Errors in the data
- Credit card fraud
- Network intrusion

Distinguish two big classes:

- Model-based
- Distance-based

We usually assume the existence of an (almost) anomaly-free training set

A note on outlier detection

Based on a similar principle as in *statistical hypothesis testing*

- Select a statistic
- Estimate distribution of the statistic in the training set
- When a new instance comes: see how extreme the observation is
- If probability of the statistic is low: reject hypothesis that the new example is generated by the process that generated the training data

Statistical test

Statistical test:

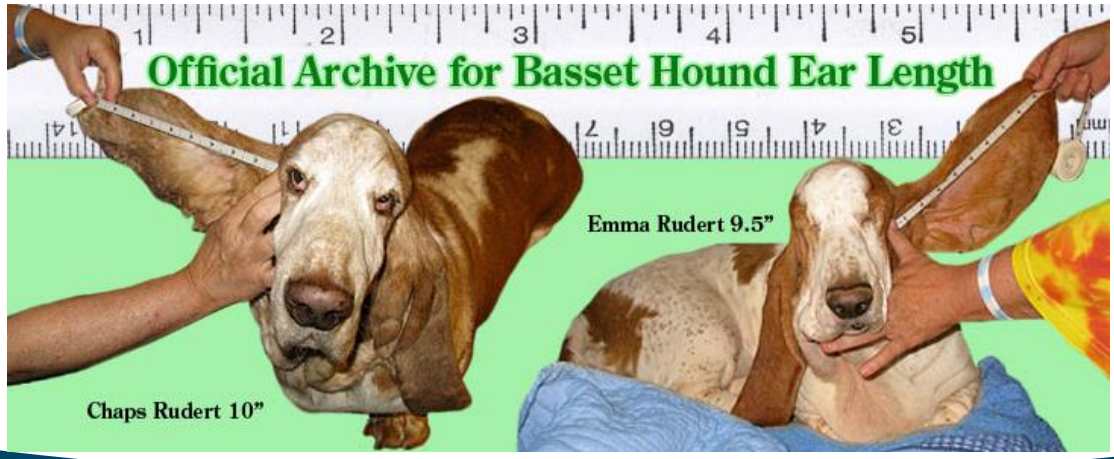
1. Formulate hypothesis
2. Collect data
3. Test hypothesis on the data

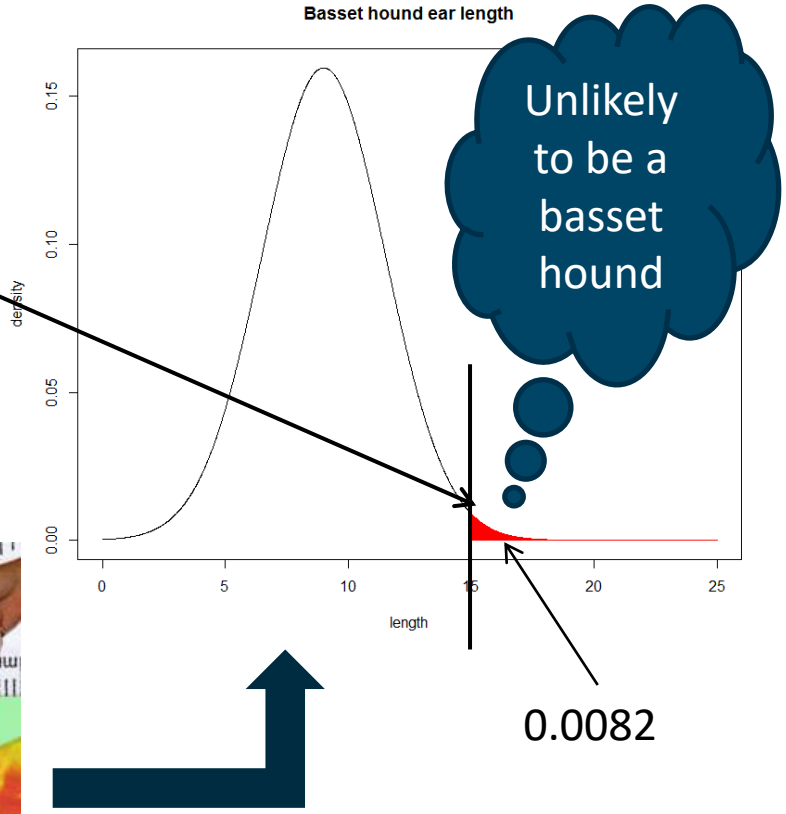
p-value expresses *how extreme* a finding is

- “the chance of getting the observed outcome is p”
- If p is very low: reject hypothesis

p-Value Illustrated

- **H0: the following animal is a Basset Hound**

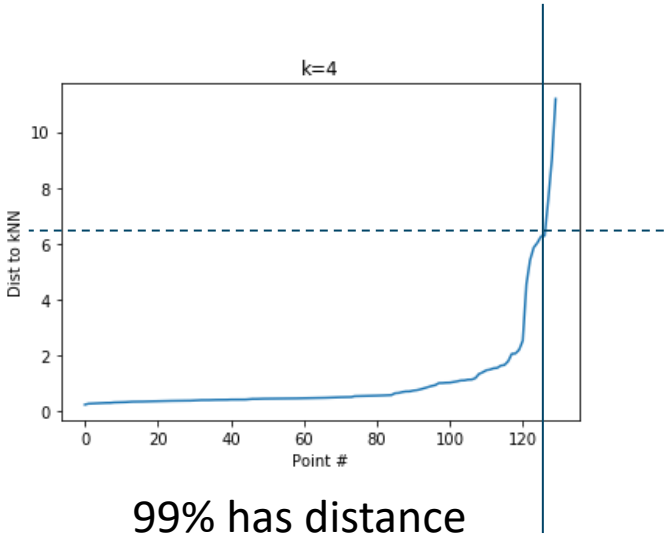




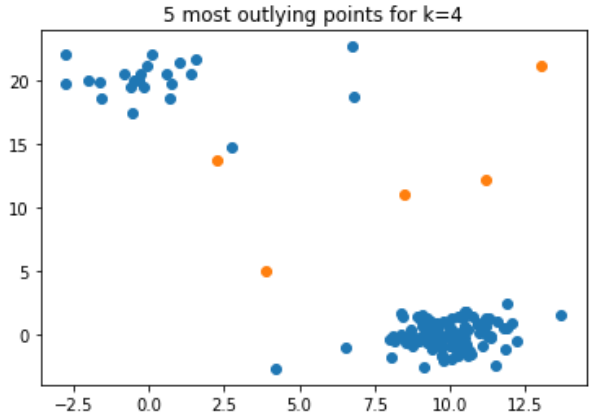
Anomaly detection

Statistic:

- Probability density of a data point
- Distance to the closest cluster center
- Distance to kth nearest neighbor



99% has distance
< 6.5 from 4NN



A note on selecting a distance measure

- Often we have a few examples of matches that we can exploit → semi-supervised learning
  - Distance measure should be such that the known pairs are close to each other
- We may have redundant features
  - Use feature set 1 to generate certain matches ; discard any uncertain match
  - Learn a distance measure on feature set 2
  - Repeat with set 1 and set 2 interchanged
  - Combine both distance measures
- Often we can *generate* such examples

A note on selecting a distance measure

Generate pairs:

- Cluster songs: split song into 10s intervals; a good distance measure will map intervals from the same song close to each other
- EEG/ECG from patients: record on different days/situations; signal from same patient should be close
- Classify images
  - Depending on domain: split into parts; parts should be close to each other
  - Images with similar tags should be close

Distance Metric Learning with Side Information (Eric Xing's method)

Must-link constraint set: $\mathcal{M}$

each element is a pair of data that come from the same class or cluster

Cannot-link constraint set: $\mathcal{C}$

each element is a pair of data that come from different classes or clusters

$$\max_{\mathbf{M}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} (\mathbf{x}_i-\mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i-\mathbf{x}_j)$$

$$s.t. \sum_{(\mathbf{x}_u,\mathbf{x}_v)\in\mathcal{M}} (\mathbf{x}_u-\mathbf{x}_v)^\top \mathbf{M}(\mathbf{x}_u-\mathbf{x}_v) \leqslant 1$$

$$\mathbf{M} \succeq 0$$

Matrix M positive semi-definite

E.P. Xing, A.Y. Ng, M.I. Jordan and S. Russell, Distance Metric Learning, with application to Clustering with side-information. NIPS 16. 2002.

Conclusions

- Partitional algorithms
  - K-means and its variations
    - Restricted to spherical clusters
    - Number of clusters needs to be set by user
  - DBSCAN
    - Any shape of cluster
    - Works only if density is uniform
    - Two parameters to set …
  - Expectation - Maximization

- Hierarchical
  - Merging versus splitting (bisecting K-means)
  - Different criteria to determine distances between clusters

Final Comment on Cluster Validity

Cluster validity is very hard to check!

Without a good validation, clustering is potentially just random.

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.
Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data,
Jain and Dubes