Clustering

COMP5318 Machine Learning and Data Mining

Semester 2, 2024, week 10

Nguyen H. Tran (slides prepared by Irena Koprinska)

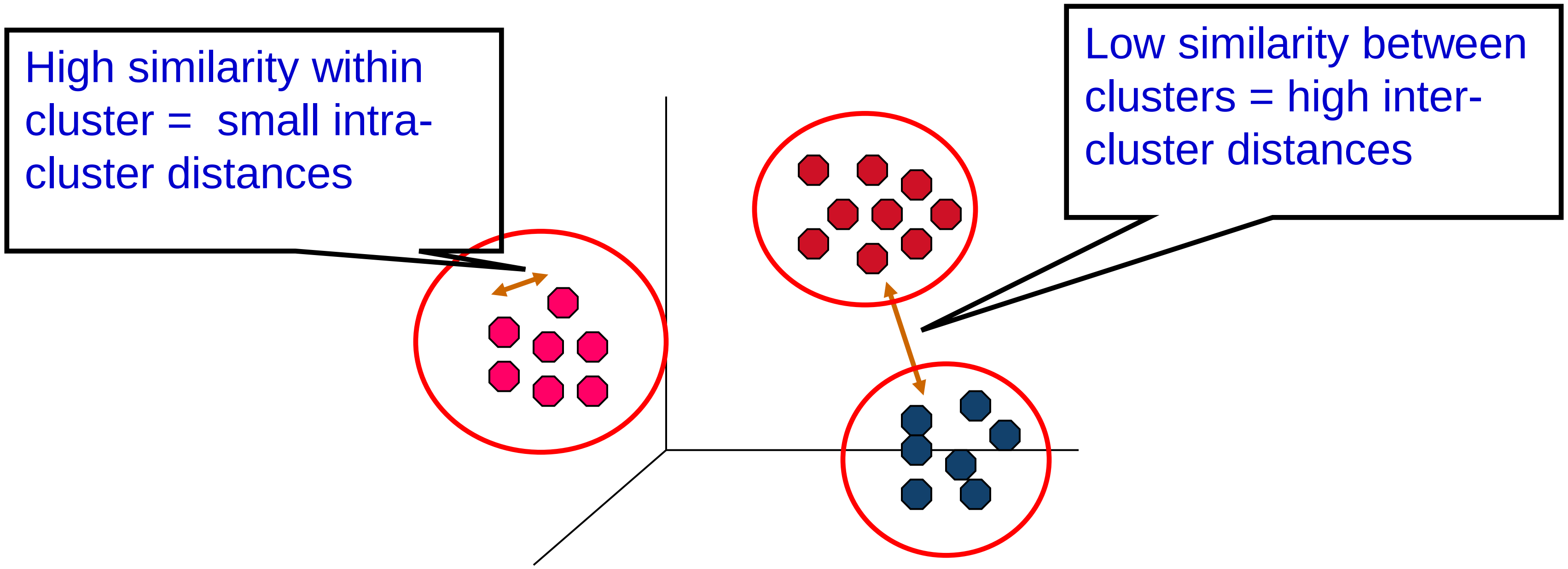
Reference: Tan ch.7.4-7.5, 8.3, Witten ch.4.8, Müller & Guido: ch.3.5, Geron: ch.9

Outline

- Introduction
  - Definition and applications
  - Distance measures and distance between clusters
  - Taxonomy
- Partitional clustering: K-means algorithm
- Model-based clustering: GMM
- Hierarchical clustering: agglomerative and divisive
- DBSCAN
- Evaluation

What is clustering?

- The process of dividing the data objects (items, examples) into groups (called clusters) so that the objects from the same group are:
  - similar to each other within the cluster
  - dissimilar to the objects in other clusters
- The similarity is computed using a distance measure


High similarity within cluster = small intra-cluster distances
Low similarity between clusters = high inter-cluster distances

Clustering is unsupervised learning

- Clustering is unsupervised learning: no labels
- Given:
  - A set of unlabeled examples (input vectors)
  - k – desired number of clusters (may not be given)
- Task: Cluster (group) the examples into k clusters (groups)

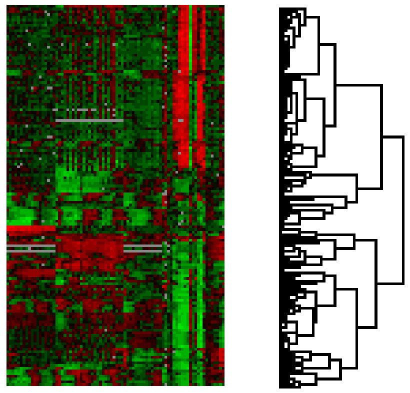
- Supervised vs unsupervised learning
  - Supervised – the class labels are given; the goal is to build a classifier that can be used to predict the class of new (unlabelled) examples
  - Unsupervised – there are no class labels; the goal is to group similar examples together.

Applications of clustering

- Clustering is used
  - As a stand-alone tool to group data
  - As a building block for other algorithms - e.g. a pre-processing tool for dimensionality reduction – using the cluster center to represent all data points in the cluster

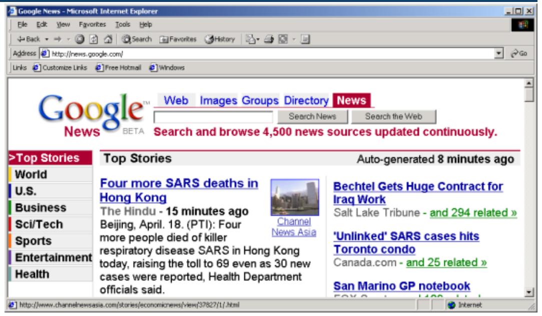
Clustering applications

- Ex.1: Targeted marketing
  - Segment customers into groups with distinct characteristics and use this knowledge to develop targeted marketing campaigns
  - Data – purchase history, browsing history, demographic data
  - Targeted campaigns are cheaper than mass-campaigns
- Ex. 2: Customer loyalty
  - Analyse customer behavior and find groups of customer who are likely to defect, e.g. to another medical insurance, electricity or phone company
- Ex. 3: Gene clustering
  - Find genes with similar structure and functionality – important for understanding diseases and finding effective treatments
  - Data: microarray – from thousands of genes, analysed simultaneously



Clustering applications (2)

- Ex. 3: Document clustering
  - Find groups of documents that are similar to each other based on their content
  - Applications:
    - Patent documents assessment: group similar patent documents to make the evaluation of a new patent document easier
    - Personalized news recommendations


Google News - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://news.google.com/
Go
Links Customize Links Free Hotmail Windows
Google News BETA
Web Images Groups Directory News
Search News
Search the Web
>Top Stories
World
U.S.
Business
Sci/Tech
Sports
Entertainment
Health
Channel News Asia
http://www.channelnewsasia.com/stories/economicnews/view/37827/1/.html
Internet

Search and browse 4,500 news sources updated continuously.

Top Stories Auto-generated 8 minutes ago

Four more SARS deaths in Hong Kong
The Hindu - 15 minutes ago
Beijing, April. 18. (PTI): Four more people died of killer respiratory disease SARS in Hong Kong today, raising the toll to 69 even as 30 new cases were reported, Health Department officials said.

Bechtel Gets Huge Contract for Iraq Work
Salt Lake Tribune - and 294 related »

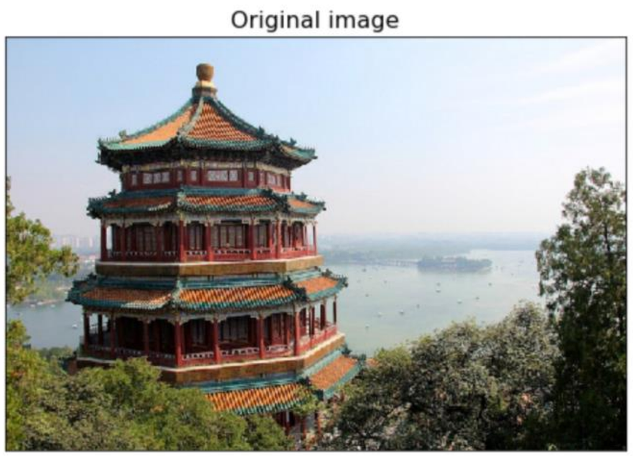
'Unlinked' SARS cases hits Toronto condo
Canada.com - and 25 related »

San Marino GP notebook

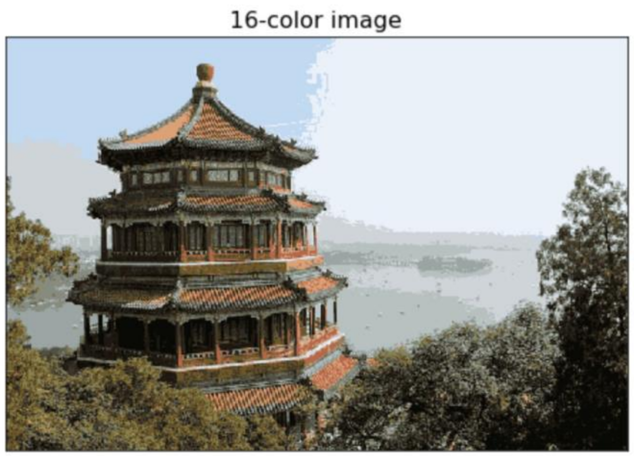
- Ex. 4: Clustering for understanding eating habits and dietary patterns of a particular cohorts (e.g. of young Australians)
  - Group 1: People who skip breakfast, care about weight, do not exercise regularly; eat high protein, low fat and high sugar diet; eat out because they enjoy the social aspect; snack after dinner
  - Group 2: …
  - Use this knowledge to promote good eating habits and changes in government policies

Clustering applications (3)

- Color compression
  - Reduce the number of colors in an image – smaller storage requirements
- Cluster the pixels and then replace them with the color of their cluster centroids – smaller number of colors used, smaller storage requirements
- See the tutorial exercises



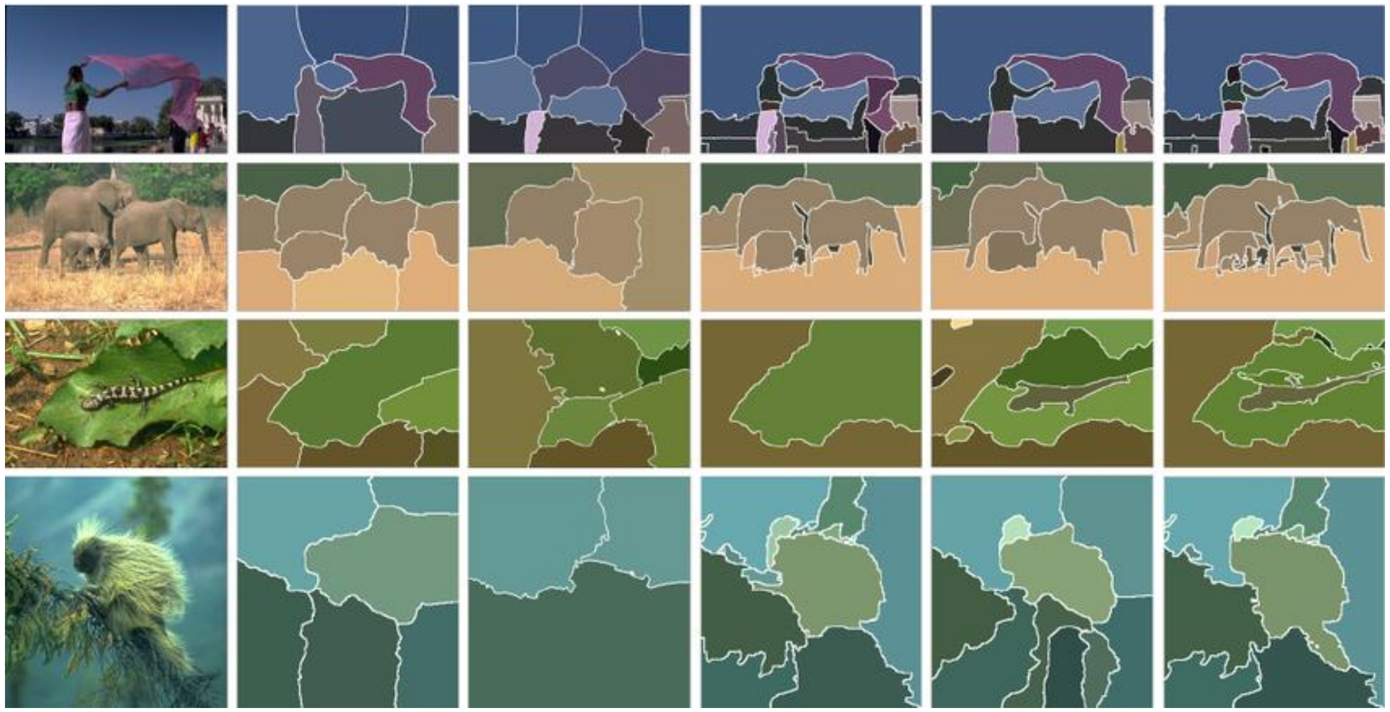
16 million colors



16 colors

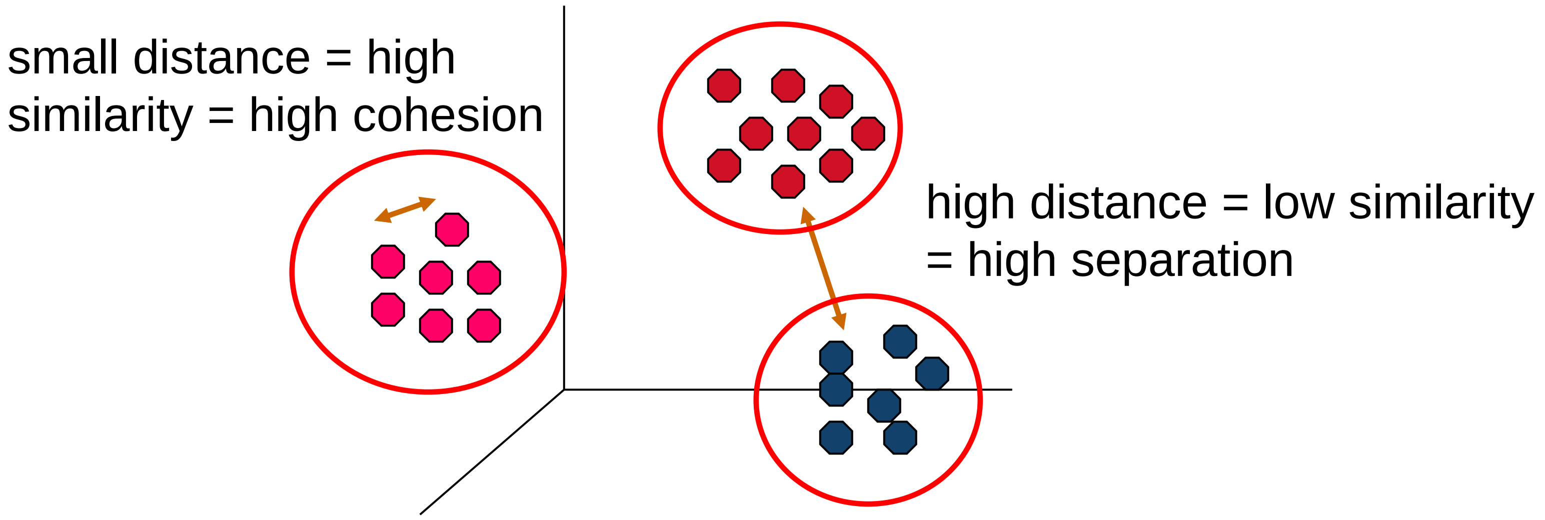
Clustering applications (4)

- Image segmentation
  - Partition an image into segments based on similar colors



Good clustering

- A good clustering will produce clusters with
  - High cohesion (i.e. high similarity within the cluster)
  - High separation (i.e. low similarity between the clusters)
- How to evaluate the quality of the clustering – next week


small distance = high similarity = high cohesion
high distance = low similarity = high separation

Measuring similarity between data points

- Similarity between 2 data points A and B is measured using a distance measure
  - If the distance D(A,B) is high -> A and B are dissimilar
  - If the distance D(A,B) is low -> A and B are similar
- Various similarity measures – see lecture 1b
  - Euclidean and Manhattan distance
  - Cosine similarity
  - Many others

Euclidean and Manhattan distance - revision

- 2 examples (data points): A = [$a_1, a_2, \ldots, a_n$], B = [$b_1, b_2, \ldots, b_n$]
- e.g. A= [1, 3, 5], B=[1, 6, 9]

- Euclidean distance

$$D(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

$$D(A,B) = \sqrt{(1-1)^2 + (3-6)^2 + (5-9)^2} = 5$$

- Manhattan distance

$$D(A,B) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

D(A,B)=|1-1|+|3-6|+|5-9|=7

Cosine distance - revision

- Cosine similarity

$$\cos(A, B) = \frac{A.B}{\|A\|\ \|B\|} = \frac{\sum_i^n a_i b_i}{sqrt(\sum_i^n {a_i}^2)\ sqrt(\sum_i^n {b_i}^2)}$$

- Geometric representation: measures the angle between A and B
  - Cosine similarity = 1 => angle(A,B)=0º
  - Cosine similarity = 0 => angle (A,B)=90º

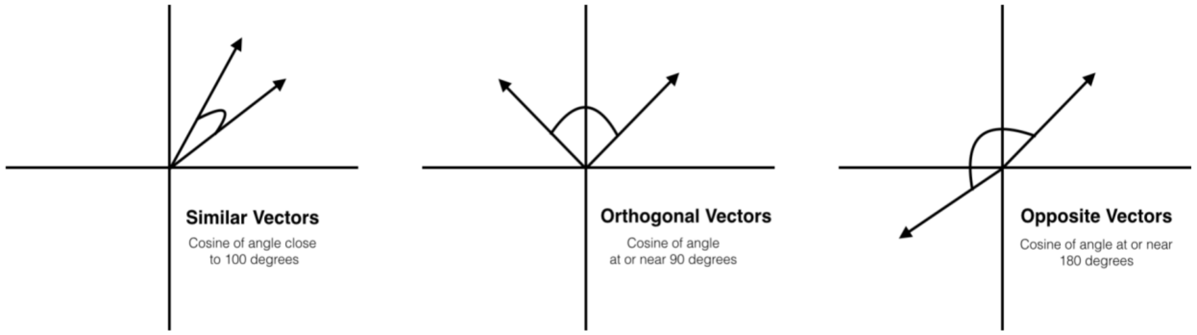

Similar Vectors
Cosine of angle close to 100 degrees
Orthogonal Vectors
Cosine of angle at or near 90 degrees
Opposite Vectors
Cosine of angle at or near 180 degrees

Image from https://deepai.org/machine-learning-glossary-and-terms/cosine-similarity

Cosine distance – revision (2)

- A= [3,2,0,5,0,0,0,2,0,0]
- B = [1,0,0,0,0,0,0,1,0,2]

$$A.B = 3*1+2*0+0*0+5*0+0*0+0*0+0*0+2*1+0*0+0*2=5$$

$$||A|| = \sqrt{3^2+2^2+0^2+5^2+0^2+0^2+0^2+2^2+0^2+0^2} = \sqrt{42} = 6.481$$

$$||B|| = \sqrt{1^2+0^2+0^2+0^2+0^2+0^2+0^2+1^2+0^2+2^2} = \sqrt{6} = 2.245$$

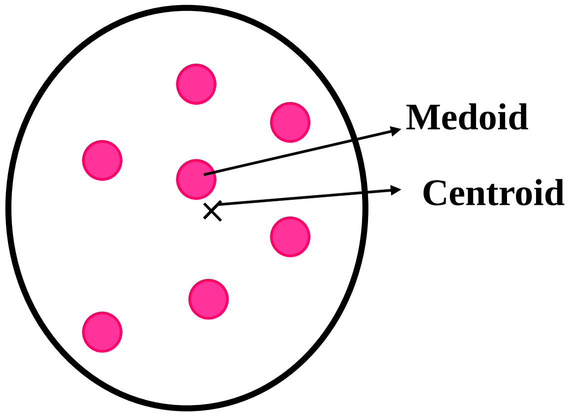
=> cos(*A*,*B*) = 0.3150

Centroid and medoid of a cluster

- Consider a cluster with N points {p1,..,pN}
- Centroid C – the "middle" of the cluster
  - Typically not an actual data point in the cluster

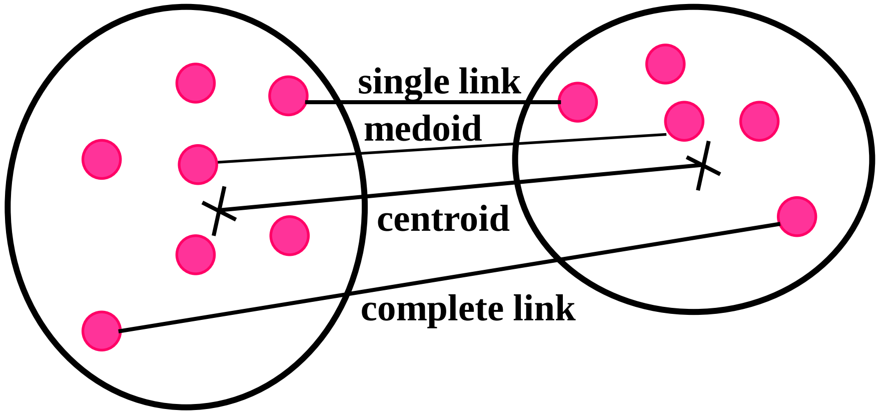
$$C = \frac{\sum_{i=1}^{N} p_i}{N}$$

- Medoid M – the centrally located data point in the cluster



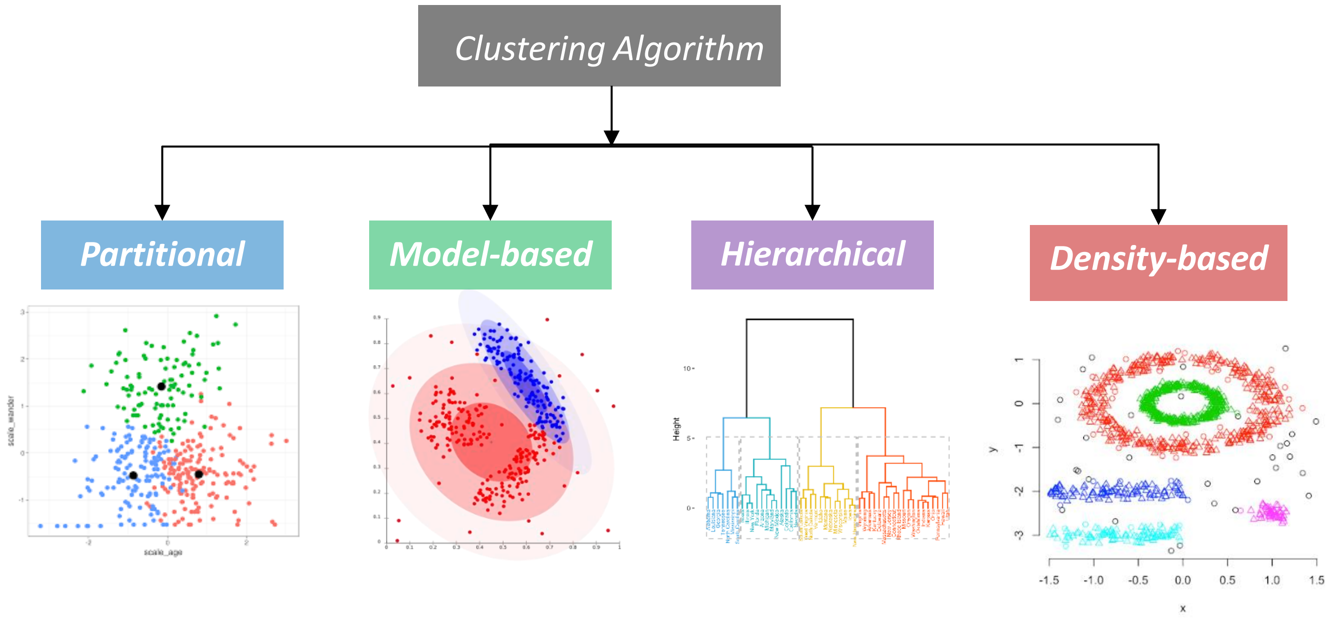
Distance between clusters

- Centroid – the distance between the centroids
- Medoid – the distance between the medoids
- Single link (MIN) – The smallest pairwise distance between elements from each cluster
- Complete link (MAX) – the largest pairwise distance between elements from each cluster
- Average link – the average pairwise distance between elements from each cluster



Taxonomy of clustering algorithms

- Partitional – k-means, k-medoids; create 1 set of clusters
- Model-based – GMM
- Hierarchical - agglomerative and divisive; create a nested set of clusters

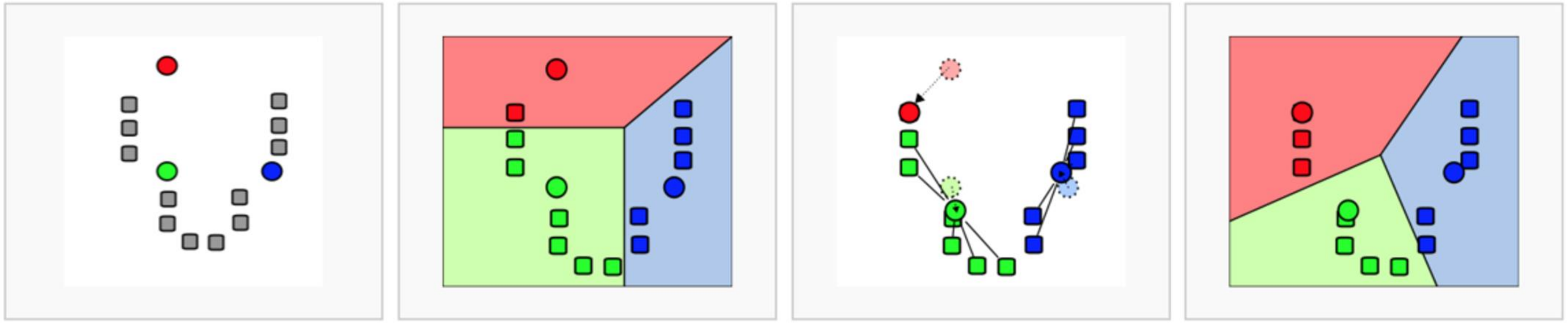


K-Means clustering algorithm

- Partitional clustering algorithm
- Very popular and widely used
- Requires the number of clusters k to be specified
- 3 main steps:
  - Choose k examples as the initial centroids of the clusters
  - Form k clusters by assigning each example to the closest centroid
  - At the end of each epoch:
    - Re-compute the centroid of the clusters
    - Check if the stopping condition is satisfied: centroids do not change. If yes – stop; otherwise, repeat steps 2 and 3 using the new centroids.

Pseudo code and example

```
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
```



Step 1: Choose initial centroids

Step 2: Assign data points to the cluster of the closest centroid

Step 3: End of epoch:
-Recompute centroids
-Check if stopping condition is satisfied – yes, stop; no - repeat from step 2

Step 4: Assign data points to the cluster of the closest centroid

K-means - more details

- The initial centroids are typically chosen randomly
- The algorithm is sensitive to the initial centroids – different clusters will be produced – see next slides
- “Closeness” is measured by a distance measure
- Most of the convergence happens in the first few epochs
- Often the stopping condition is changed from “Until centroids do not change” to ‘Until relatively few points change clusters’
- Complexity is O(n*k*i*d)
  - n - number of points, k - number of clusters, i - number of iterations, d - number of attributes

K-Means task

- Given: 5 items with the distance between them
- Task: Use the K-means algorithm to cluster them into 2 clusters. The initial centroids are A and B. Show the clusters after the first epoch.

	A	B	C	D	E
A	0	2	7	10	1
B	2	0	3	4	6
C	7	3	0	5	9
D	10	4	5	0	1
E	1	6	9	1	0

```
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
```

Solution

- Initial centroids: A and B
- cluster1 – the cluster of A
- cluster2 – the cluster of B

	A	B	C	D	E
A	0	2	7	10	1
B	2	0	3	4	6
C	7	3	0	5	9
D	10	4	5	0	1
E	1	6	9	1	0

Epoch1 – start:

- A is assigned to cluster1 (centroid)
- B is assigned to cluster2 (centroid)

- C:
  d(C, A)=7, d(C, B)=3
  => C is assigned to cluster2

- D:
  d(D, A)=10, d(D, B)=4
  => D is assigned to cluster2

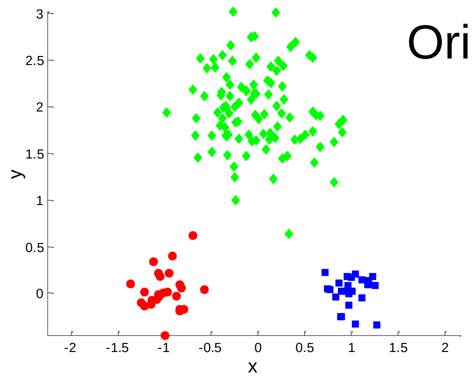
- E:
  d(E, A)=1, d(E, B)=6
  => E is assigned to cluster1

End of epoch 1, clusters are:

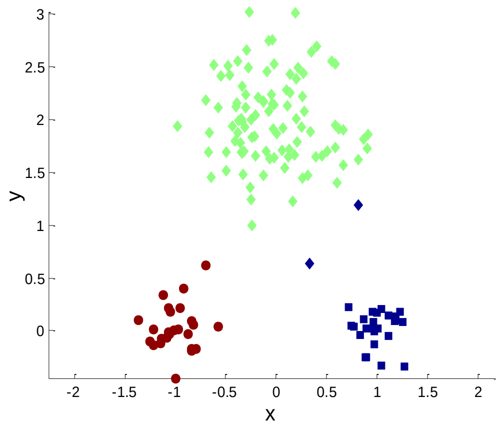
- {A, E} and {B, C, D}

Sensitive to initial centroids

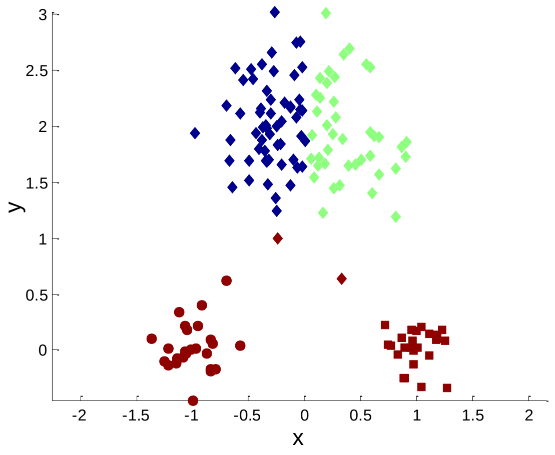
- Different random initialisation -> different result



Original data

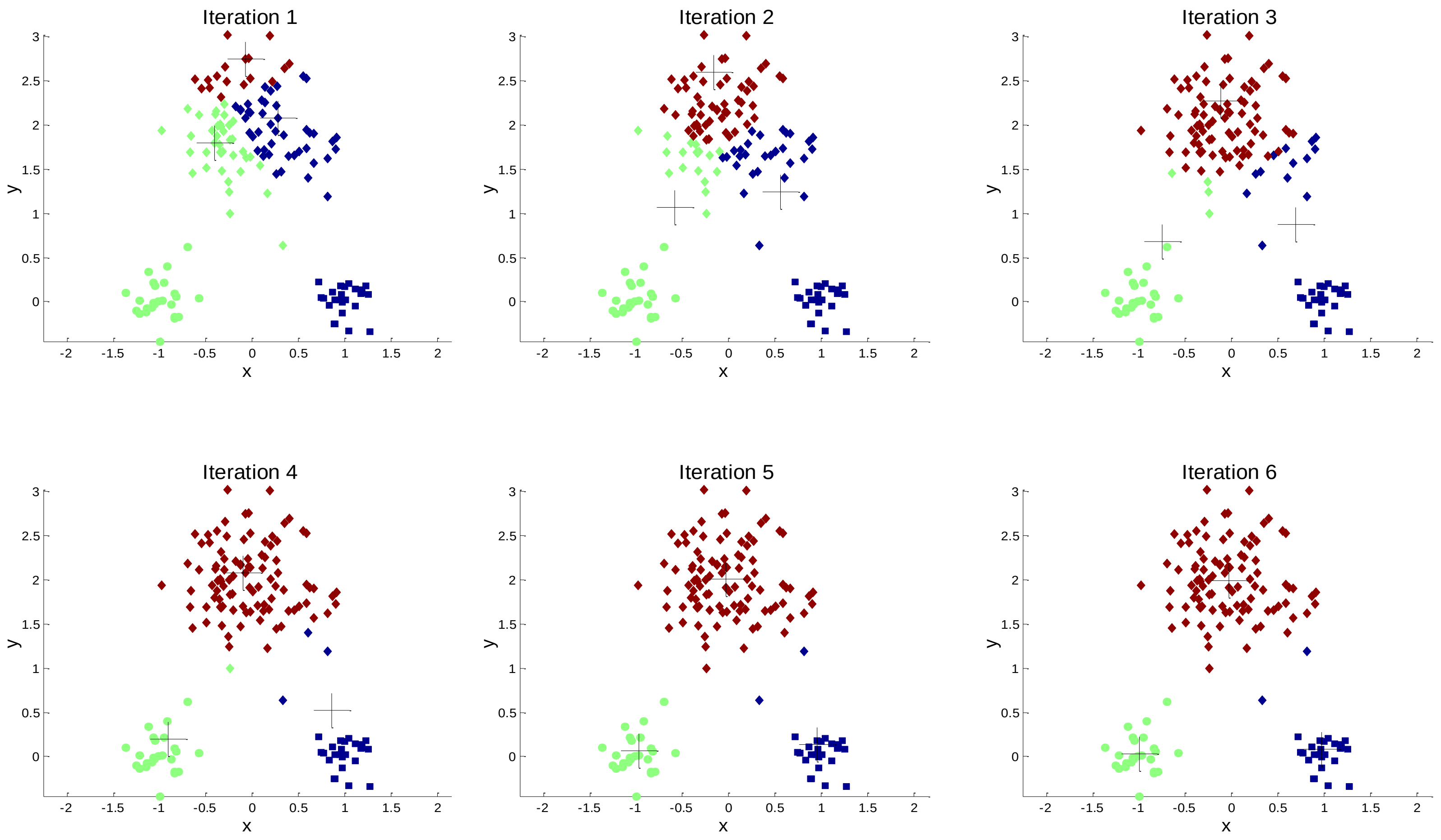


Initialisation 1 - good clustering

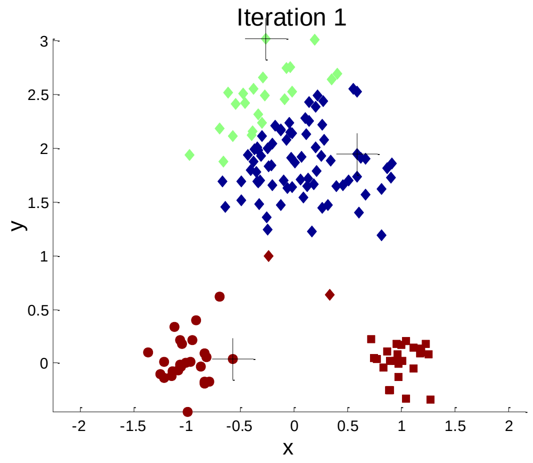


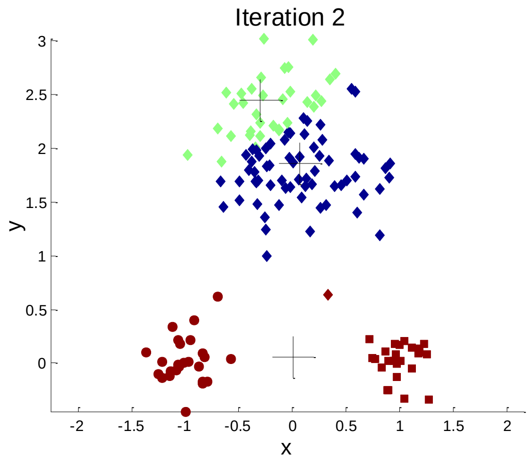
Initialisation 2 - bad clustering

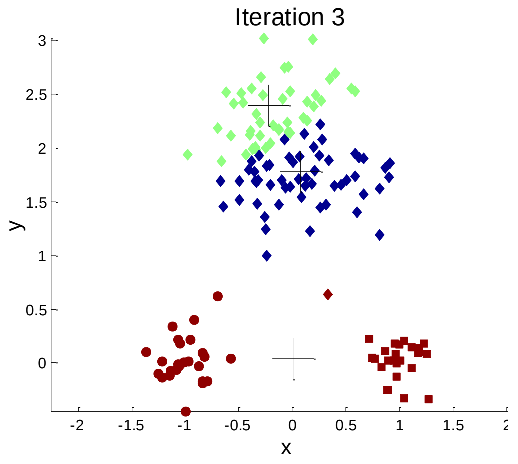
Good initial centroids

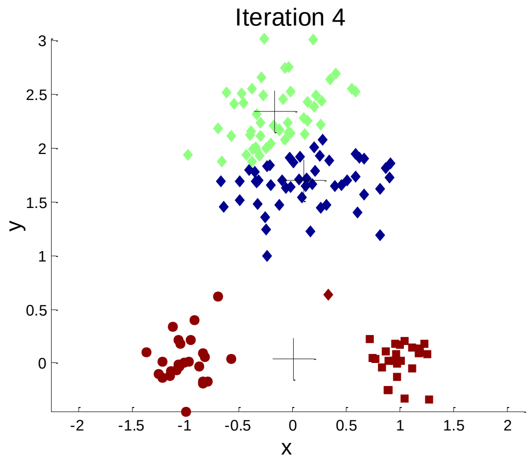


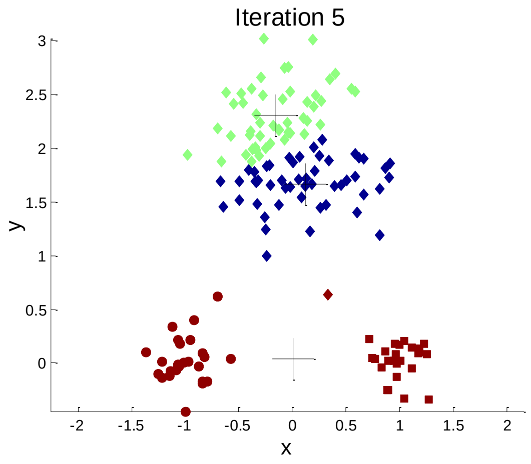
Poor initial centroids











Selecting good initial centroids

- Method 1: Run K-means several times with different randomly selected initial centroids; evaluate each clustering using SSE, select the clustering with the smallest Sum of Squared Error (SSE)
- SSE:
  - For each point, the error is the distance to the closest centroid
  - To get SSE, we square these errors and sum them

$$SSE = \sum_{i=1}^{k} \sum_{\mathrm{X} \in K_i} d(c_i, \mathbf{x})^2$$

c_i are the centroids, k - number of clusters, x – data points

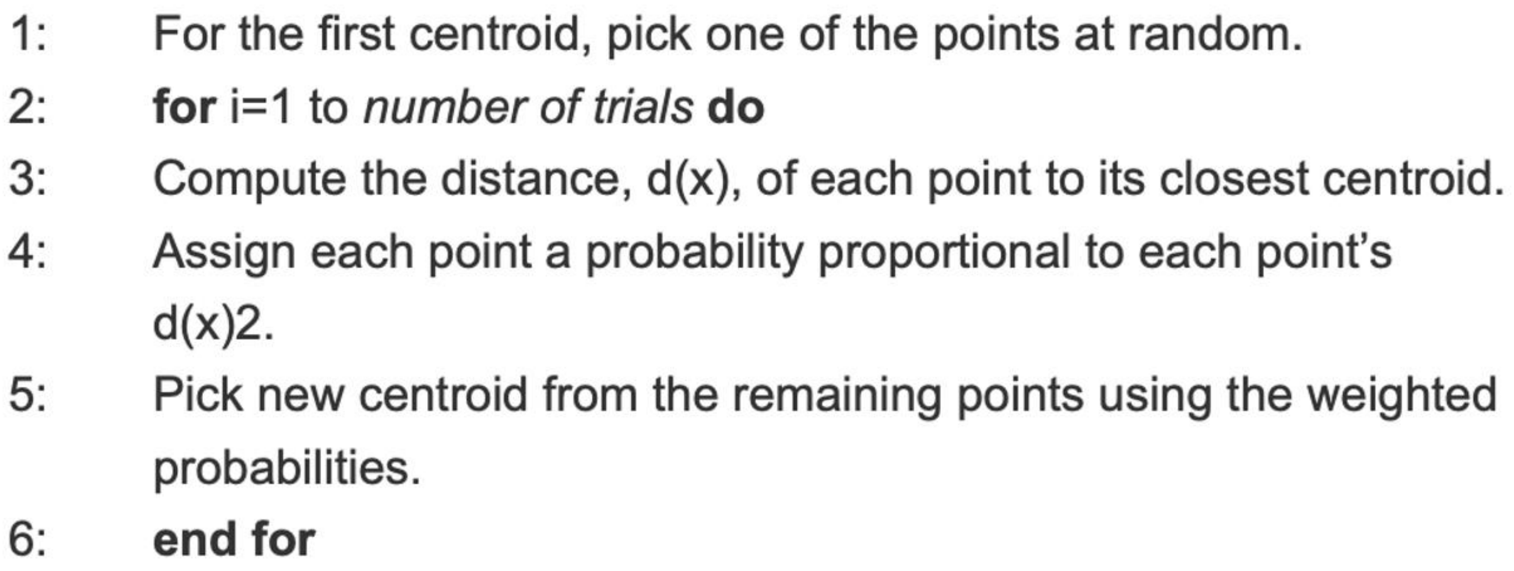
- Method 2: K-means++

K-means++

- K-means++ is a variation of K-means which uses a new method for selecting the initial centroids; the rest is the same as the standard K-means
- Centroid selection:
  - Select centroids incrementally, until k centroids have been selected
  - At every step, each point has a probability to be selected as a centroid that is proportional to the square of its distance to its closest centroid
  - => Selects points that are farthest away from the current centroids – selects well-separated points
  - Can select outliers, but outliers are rare by definition
- Works well in practice

K-means++ pseudocode for centroid selection

- Selection of initial centroids in K-means++:



```
1: For the first centroid, pick one of the points at random.
2: **for** i=1 to *number of trials* **do**
3:     Compute the distance, d(x), of each point to its closest centroid.
4:     Assign each point a probability proportional to each point's
       d(x)2.
5:     Pick new centroid from the remaining points using the weighted
       probabilities.
6: **end for**
```

K-means issues: Handling empty clusters

- K-means can yield empty clusters – no points are allocated to a cluster during the assignment step – the cluster consists only of the initial centroid
- Solution: choose different initial centroid – strategies:
  - Choose the point that is farthest away from any current centroid
  - Use the K-means++ approach (similar idea)
  - Choose a point from the cluster that has the highest SSE
    - This will typically split the cluster and reduce the overall SSE of the clustering
- If there are several empty clusters, the above can be repeated several times

K-means issues: Outliers

- When there are outliers, the resulting cluster centroids are less representative and the SSE is higher
- Solution: remove outliers before clustering
- Caution: for some clustering applications outliers are important and should not be removed, e.g.:
  - Data compression: all points need to be clustered
  - Financial analysis – unusually profitable customers can be the most interesting
- Alternatively, remove outliers as a post-processing step after clustering – keep track of SSE contributed by each point, eliminate points with unusually high contributions, especially over multiple runs

- Various techniques for identifying outliers - out scope for this course, see Tan ch.9

Variations: Bisecting K-means

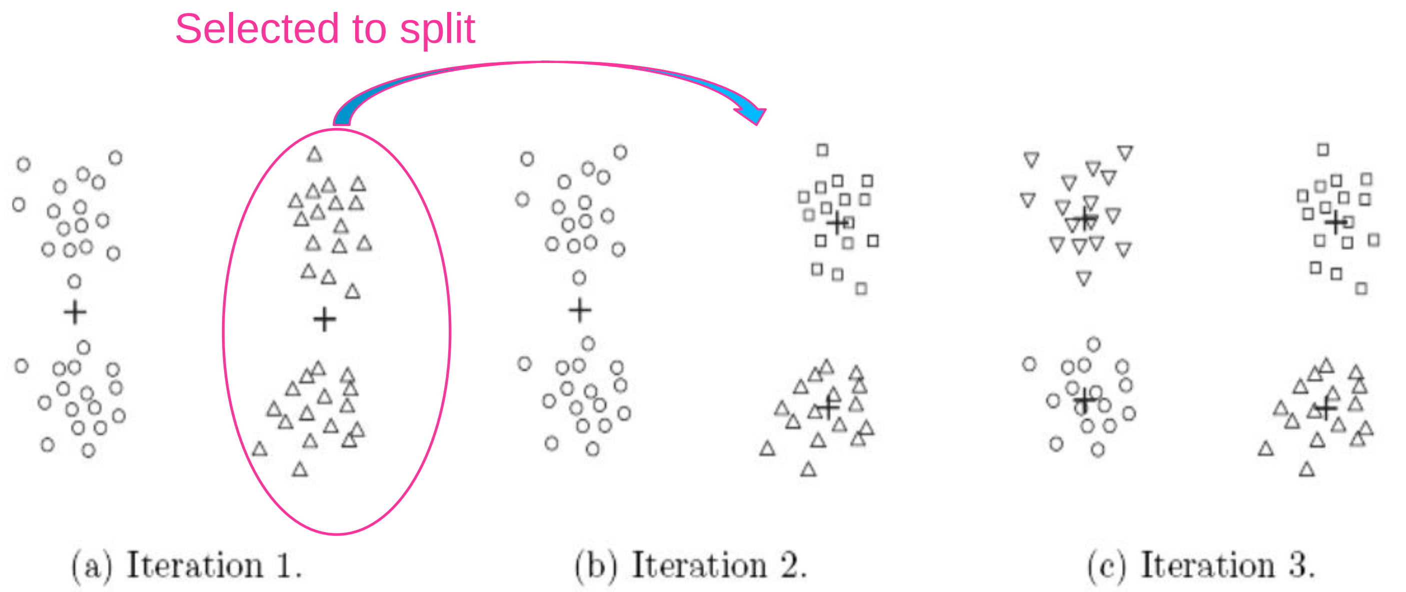
- Extension of k-means
- Main idea: to obtain k clusters, split the data into 2 clusters, select one of these clusters to split further and so one until k clusters are obtained

```
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
```

Different ways: the largest cluster, the cluster with the largest SSE, or based on both size and SSE

Bisecting K-means - Example

- Less sensitive to initialization problems



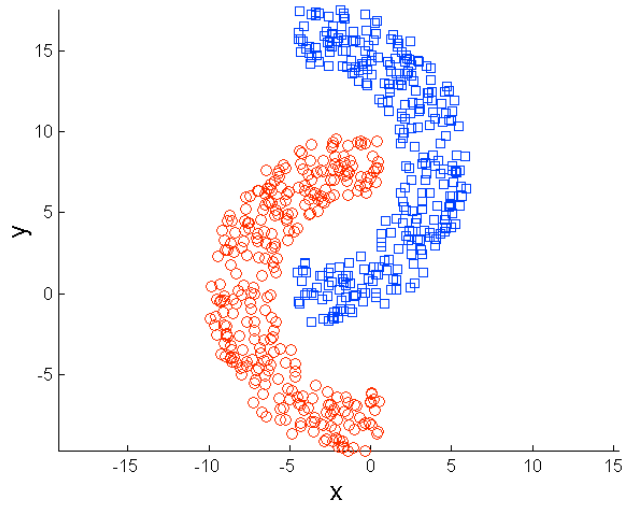
K-means and different types of clusters

- K-means works well if the clusters are spherical, of equal density, equal size and are well separated

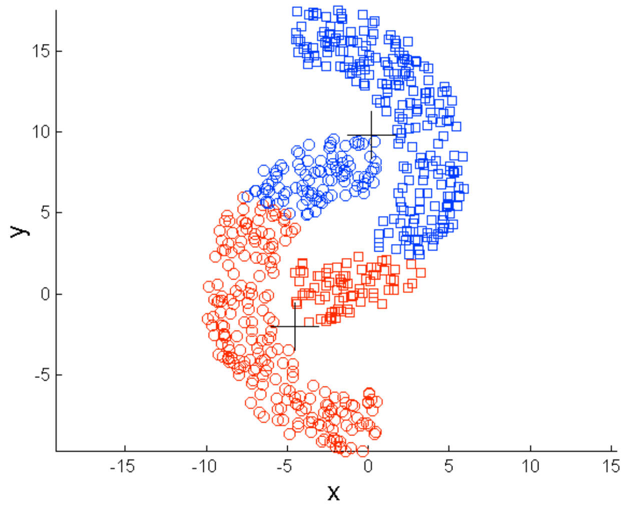
THE UNIVERSITY OF SYDNEY

K-means on clusters with non-spherical shapes

- Doesn’t work well for natural clusters with complex (non-spherical) shape



Original points

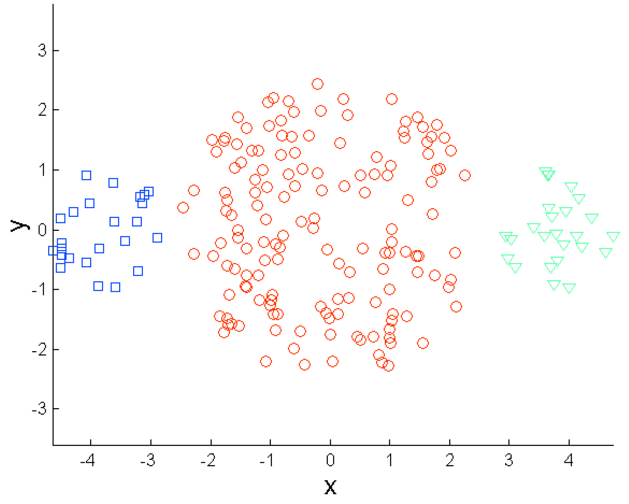


K-means (2 Clusters)

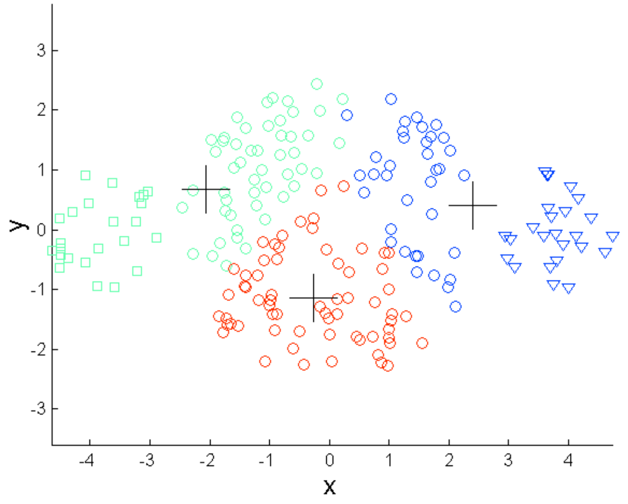
THE UNIVERSITY OF SYDNEY

K-means on clusters with different size

- Doesn’t work well for natural clusters with vastly different size
- 3 natural clusters, the second one is much bigger than the first and third
- K-Means cannot find the natural clusters – it splits the largest



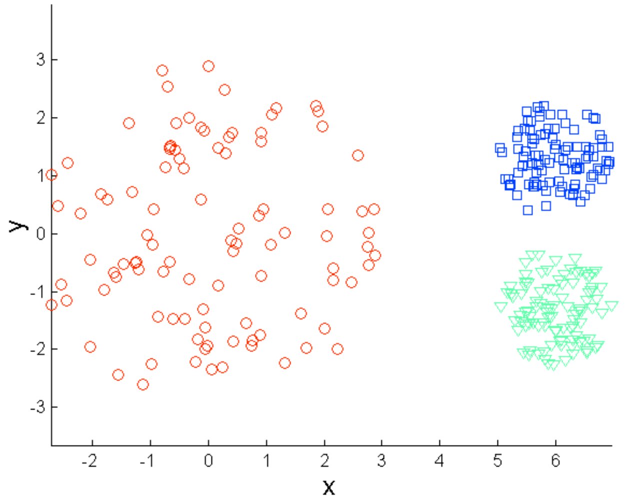
Original points



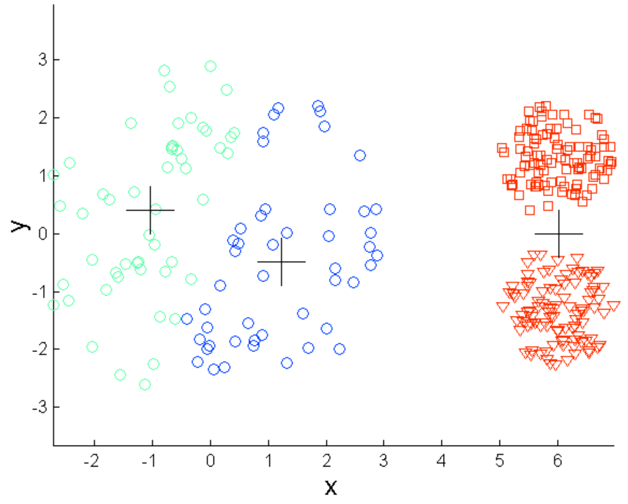
K-means (3 Clusters)

K-means on clusters with different density

- Doesn’t work well for natural clusters with different density
- 3 natural clusters, 2 of them are much denser than the other
- K-Means cannot find the natural clusters – it splits the large cluster



Original points

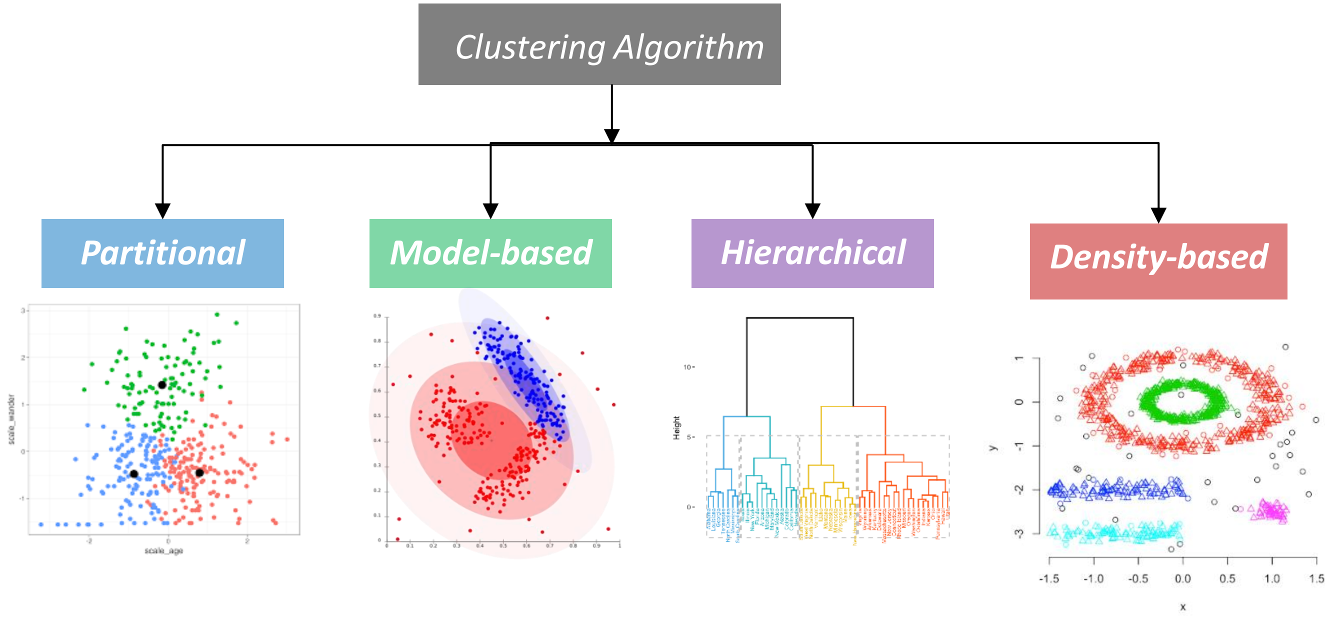


K-means (3 clusters)

K-Means - strengths and weaknesses

- Simple and very popular
- Relatively efficient, even though multiple runs are required
- Sensitive to centroid initialization
- Not sensitive to the order in which the input examples are applied
- Does not work well for clusters with non-spherical shape, different sizes and different density
- Does not work well for data containing outliers
  - Pre-processing is needed - outlier detection and removal
- Different variations – e.g. bisecting K-means and K-means++
  - Both reduce sensitivity to initialization (choice of initial centroids)

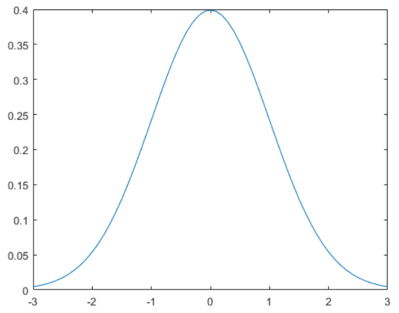
Model-based: GMM



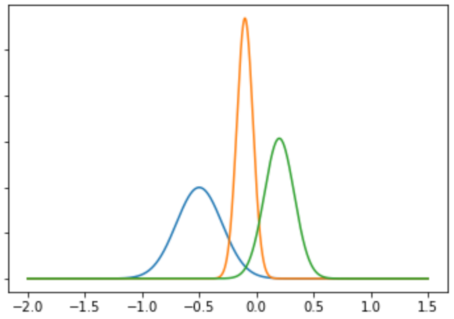
Gaussian mixture model clustering

- Gaussian Mixture Model (GMM) clustering is a probabilistic clustering
- It assumes that the data is generated by a mixture of normal (Gaussian) distributions

Normal distribution



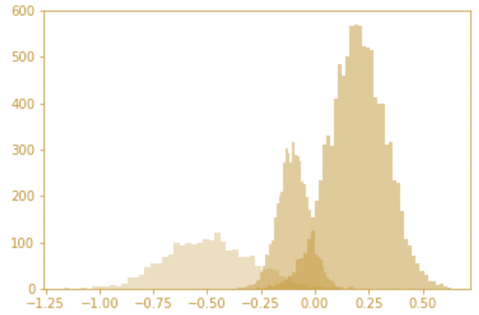
3 normal distributions



ND = normal distribution

- ND1: μ = 0.5, σ = 0.2
- ND2: μ= -0.1, σ = 0.07
- ND3: μ = 0.2, σ = 0.13

Randomly generated data for these NDs



- *2000 data samples for ND1*
- *5000 data samples for ND2*
- *10000 data samples for ND3*

- Probability density function for a normal distribution with mean μ and standard deviation σ:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

THE UNIVERSITY OF SYDNEY

GMM Algorithm

- We assume that the data is generated by a mixture of *k* Gaussian (normal) distributions. Each distribution has 2 parameters: μ and σ.
- One distribution corresponds to 1 cluster
- We don't know the parameters μ and σ of the distributions; starting from random initial values we iteratively estimate them from the data
- After each estimation, we compute the probability of each example to belong to each distribution (cluster)
- Using the probabilities, we re-compute the parameters, until they don't change

GMM Algorithm

Algorithm 9.2 EM algorithm.

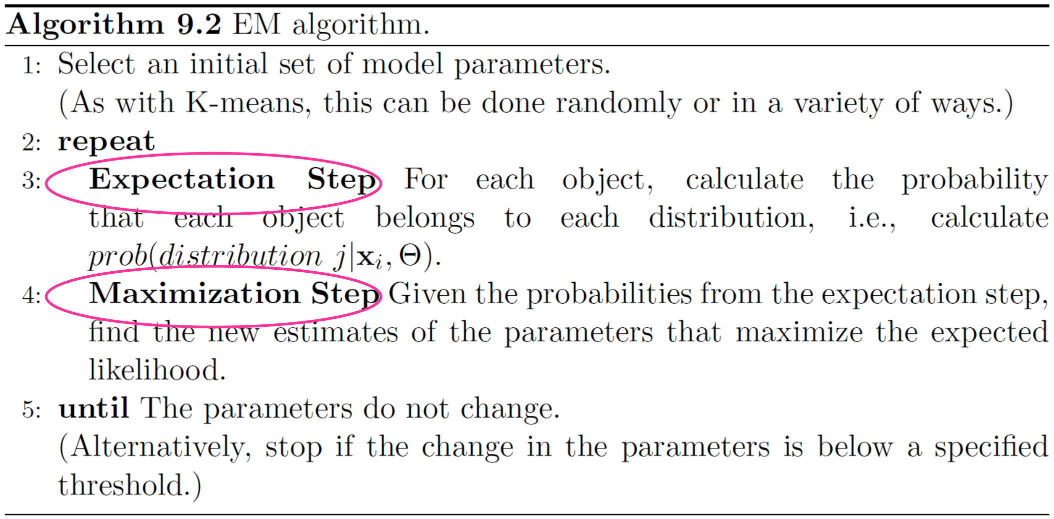
```
1: Select an initial set of model parameters.
   (As with K-means, this can be done randomly or in a variety of ways.)
2: repeat
3:    Expectation Step For each object, calculate the probability
      that each object belongs to each distribution, i.e., calculate
      $prob(distribution\ j|\mathbf{x}_i, \Theta)$.
4:    Maximization Step Given the probabilities from the expectation step,
      find the new estimates of the parameters that maximize the expected
      likelihood.
5: until The parameters do not change.
   (Alternatively, stop if the change in the parameters is below a specified
   threshold.)
```

THE UNIVERSITY OF SYDNEY

GMM and K-Means correspondence

- Expectation step in GMM -> Assigning each object to a cluster in K-means
  - In K-means – crisp assignment, in GMM – probabilistic; each object is assigned to each cluster with a certain probability

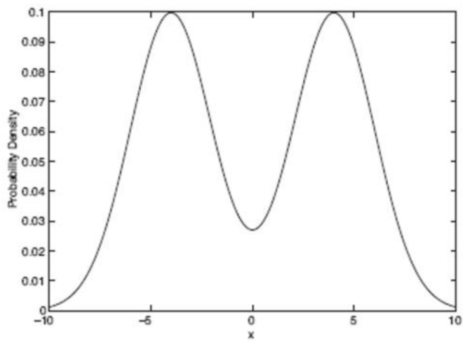
- Maximization step in GMM -> Computing the cluster centroid in K-Means
  - In GMM we compute the parameters of the distributions



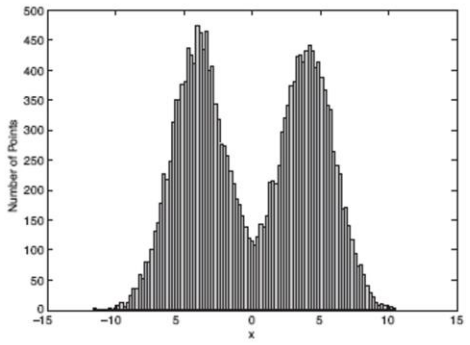
Algorithm 9.2 EM algorithm.

```
1: Select an initial set of model parameters.
   (As with K-means, this can be done randomly or in a variety of ways.)
2: repeat
3:    Expectation Step For each object, calculate the probability
      that each object belongs to each distribution, i.e., calculate
      $prob(distribution\ j|\mathbf{x}_i, \Theta)$.
4:    Maximization Step Given the probabilities from the expectation step,
      find the new estimates of the parameters that maximize the expected
      likelihood.
5: until The parameters do not change.
   (Alternatively, stop if the change in the parameters is below a specified
   threshold.)
```

Example



(a) Probability density function for the mixture model.



(b) 20,000 points generated from the mixture model.

- 1-dim data x (20 000 points), generated by 2 Gaussian distributions: distribution 1 and distribution 2
- For simplicity, let's assume that we know the standard deviations σ_1 and σ_2 and that both are the same: $\sigma_1 = \sigma_2 = 2$

- Step 1: Initial guesses for μ_1 and μ_2 : $\mu_1 = -2$, $\mu_2 = 3$
- => initial parameters of the 2 distributions: θ_1=(-2,2), θ_2=(3,2)
- Set of parameters for the entire mixture model: $\theta = (\theta_1, \theta_2)$

Example (2)

- Step 2: Expectation step
  - Compute the probability that a point *x*$_i$ came from each distribution *j* (j=1,2)

$$P(distribution\ j|x_i, \theta) = \frac{w_j\, P(x_i|\theta_j)}{w_1\, P(x_i|\theta_1) + w_2\, P(x_i|\theta_2)}$$

  w_j are the weights of each distribution (the probability for distribution j to generate the example); all weights should sum to 1, i.e. $w_1+w_2=1$.

  For our example we assume that $w_1=w_2=0.5$:

$$P(distribution\ j|x_i, \theta) = \frac{0.5\, P(x_i|\theta_j)}{0.5\, P(x_i|\theta_1) + 0.5\, P(x_i|\theta_2)}$$

  - E.g. for point *x*$_i$ = 0: $P(x_i|\theta_1) = 0.12$, $P(x_i|\theta_2) = 0.06$ (calculated using the probability density function of normal distribution)
  - => $P(distribution\ 1|\mathrm{x} = 0, \theta) = \frac{0.12}{0.12+0.06}$ = 0.66

    $P(distribution\ 2|\mathrm{x} = 0, \theta) = \frac{0.06}{0.12+0.06}$ = 0.33
  - We compute these probabilities for all n = 20 000 points

Example (3)

- Step 3: Maximization step – compute new estimates for μ_1 and μ_2

$$\mu_1 = \sum_{i=1}^{n} x_i \frac{P(distribution\ 1|x_i, \theta)}{\sum_{i=1}^{n} P(distribution\ 1|x_i, \theta)}, \quad \mu_2 = \sum_{i=1}^{n} x_i \frac{P(distribution\ 2|x_i, \theta)}{\sum_{i=1}^{n} P(distribution\ 2|x_i, \theta)}$$

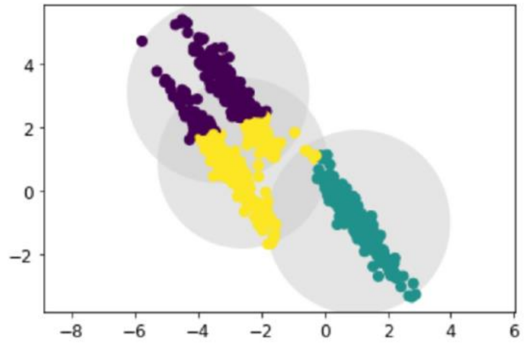
- Notice that the new estimate for the mean μ is the weighted average of the points, where the weights are the probabilities that the points belong to the distribution

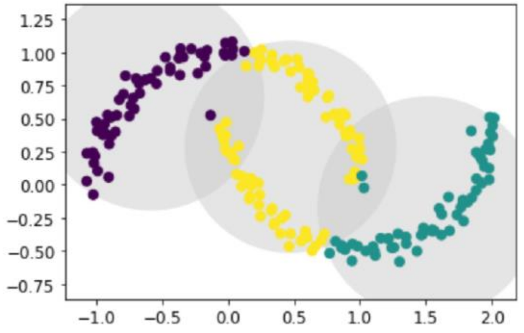
- Repeat the expectation and maximization steps until the estimates for μ_1 and μ_2 don't change or change very little

- After convergence, each point is assigned to the cluster with the highest probability

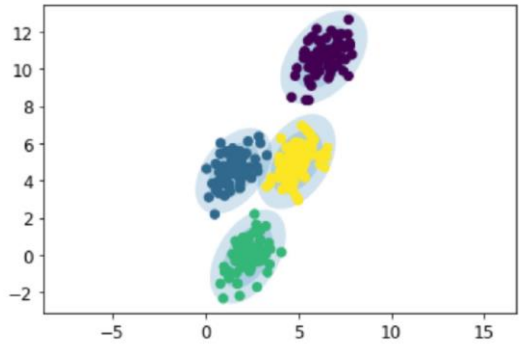
GMM vs K-means

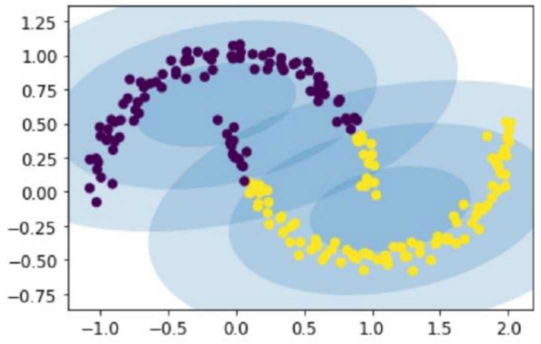
- GMM can be seen as a generalization of K-means
- It is more flexible – it allows for elliptical clusters rather than circular and for probabilistic assignment to each cluster rather than crisp – see the tutorial notes
  - K-means:





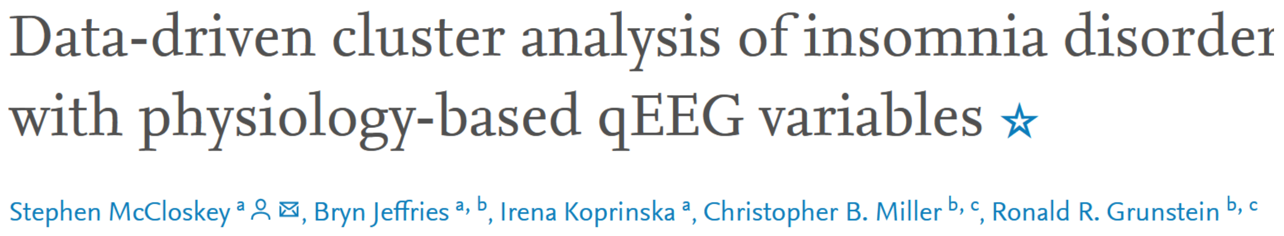
  - GMM:





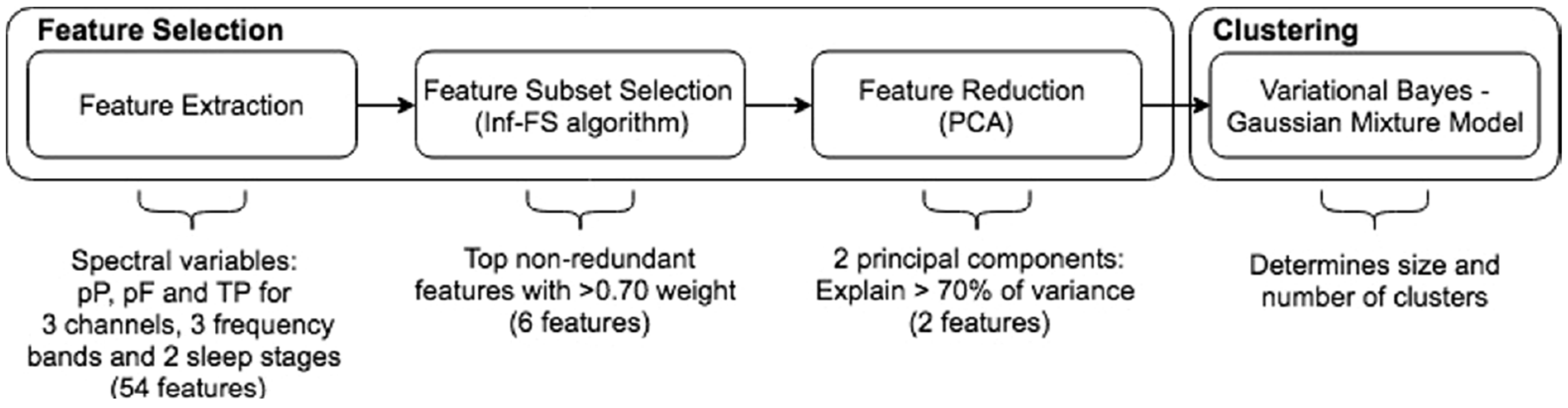
GMM example

- Clustering of people with insomnia based on EEG data
- https://www.sciencedirect.com/science/article/abs/pii/S0950705119303387



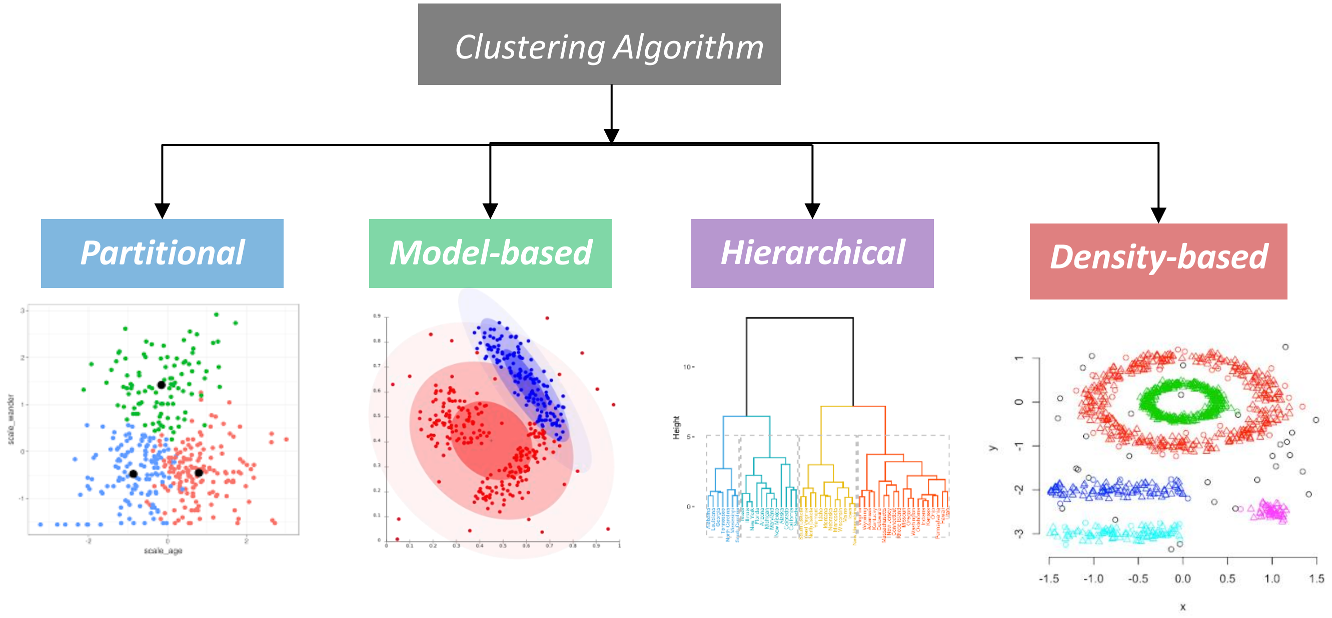
Data-driven cluster analysis of insomnia disorder with physiology-based qEEG variables ☆

Stephen McCloskey [a], Bryn Jeffries [a, b], Irena Koprinska [a], Christopher B. Miller [b, c], Ronald R. Grunstein [b, c]



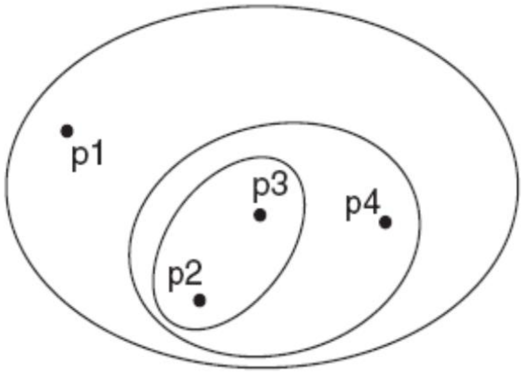
Feature Selection
Feature Extraction
Feature Subset Selection (Inf-FS algorithm)
Feature Reduction (PCA)
Clustering
Variational Bayes - Gaussian Mixture Model
Spectral variables: pP, pF and TP for 3 channels, 3 frequency bands and 2 sleep stages (54 features)
Top non-redundant features with >0.70 weight (6 features)
2 principal components: Explain > 70% of variance (2 features)
Determines size and number of clusters

Hierarchical clustering

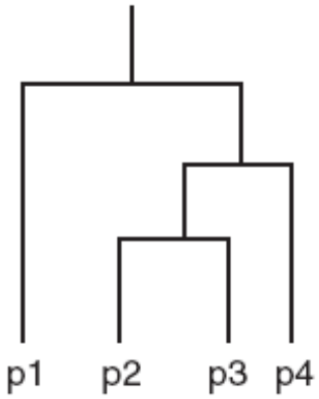


Hierarchical clustering

- 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



Nested clusters



Dendrogram

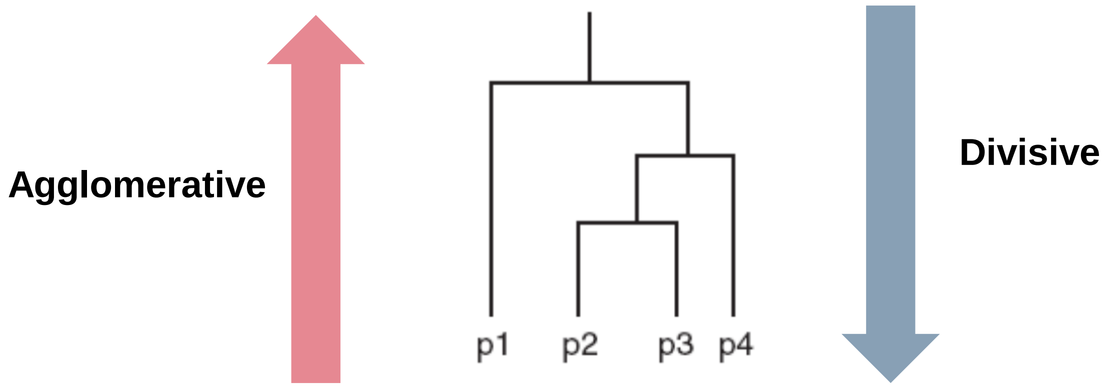
Strengths of hierarchical clustering

- No need to specify the number of clusters in advance
- A desired number of clusters can be obtained by ‘cutting’ the dendrogram at different levels

- The dendrogram provides a useful visualization – an interpretable description of the clustering process
- The dendrogram may reveal a meaningful taxonomy

Two approaches: agglomerative and divisive

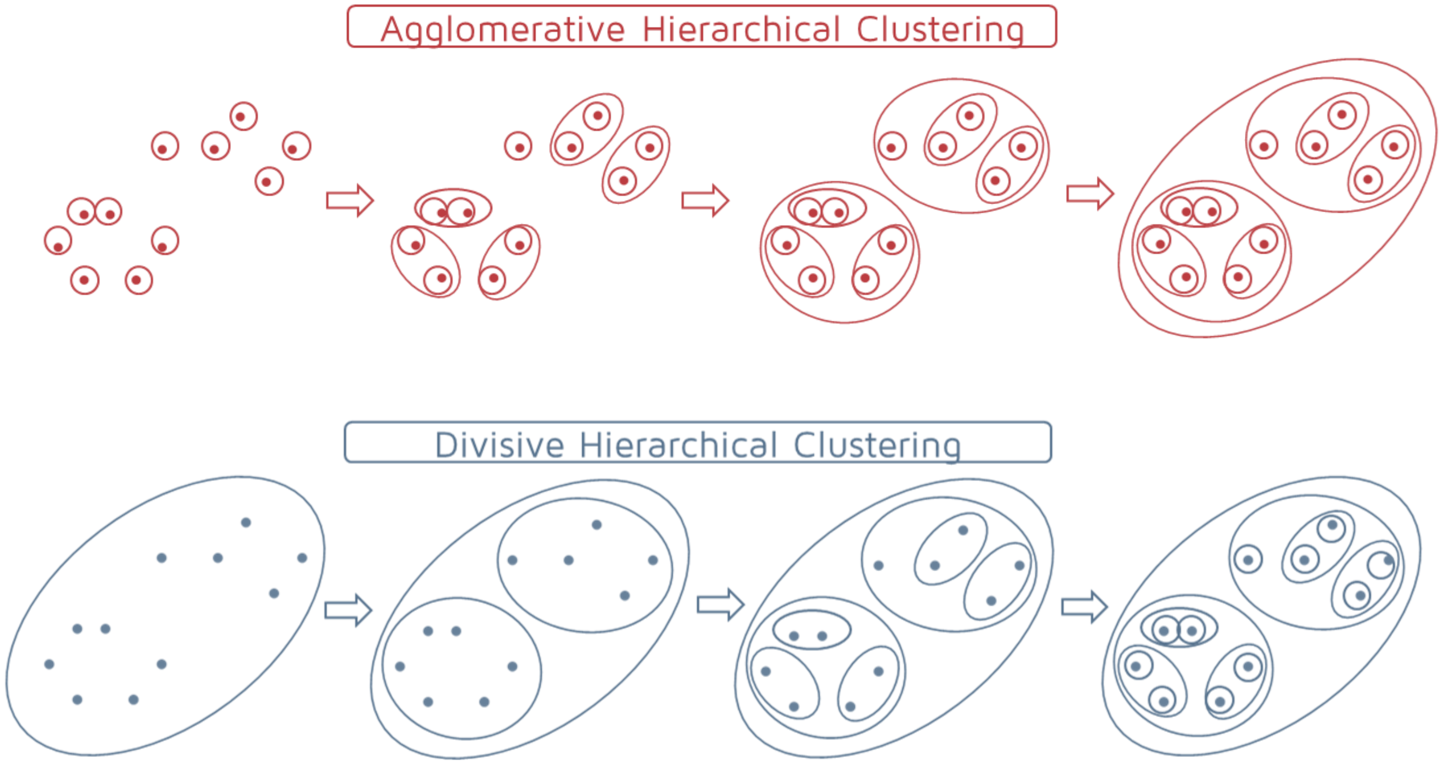
- Agglomerative (bottom-up) – merges clusters iteratively
  - Start with each item in its own cluster; iteratively merge clusters until all items belong to one cluster
- Divisive (top-down) – splits a cluster iteratively
  - Place all items in one cluster; iteratively split clusters in two until all items are in their own cluster
- Divisive is less popular than agglomerative; we will focus on agglomerative



THE UNIVERSITY OF SYDNEY

Agglomerative and divisive clustering - diagrams

- Agglomerative (bottom-up) – merges clusters iteratively
- Divisive (top-down) – splits a cluster iteratively


Agglomerative Hierarchical Clustering
Divisive Hierarchical Clustering

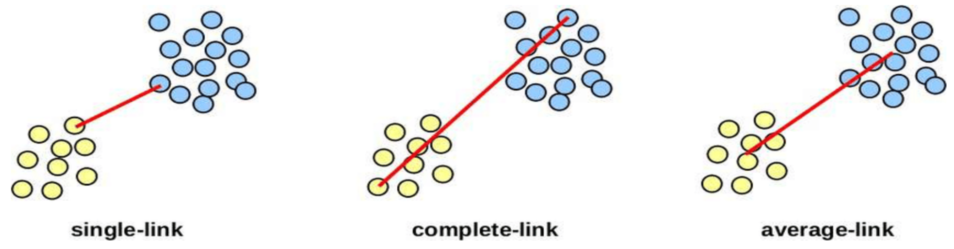
Agglomerative hierarchical clustering

- Agglomerative is the most popular hierarchical clustering algorithm
- The key operation is computing the distance between two clusters (the proximity matrix)
- There are different versions of how the clusters are merged at each step; we will use the version that merges the 2 closest clusters

```
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
```

Distance between clusters

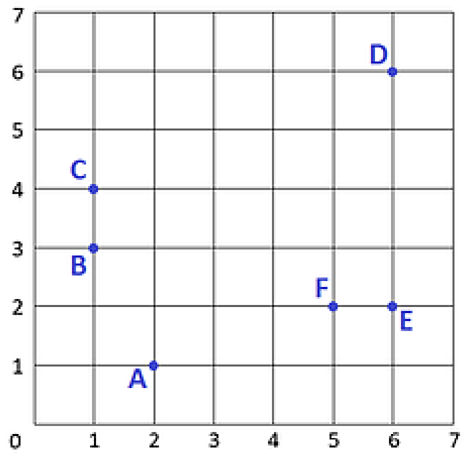
- Hierarchical clustering typically uses the following distance measures:
  - Single link (MIN)
  - Complete link (MAX)
  - Average link
  - Ward’s method – the distance between 2 clusters is the increase in SSE that results when the 2 clusters are merged



	The distance between 2 clusters is:
Single link (MIN)	the **smallest** distance between an element in one cluster and an element in the other
Complete link (MAX)	the **largest** distance between an element in one cluster and an element in the other
Average link	the **average** distance between each element in one cluster and each element in the other

Agglomerative clustering - example

- Given are 6 data points. Apply agglomerative hierarchical clustering to cluster them using the single-link distance between clusters and the Manhattan distance between data points.



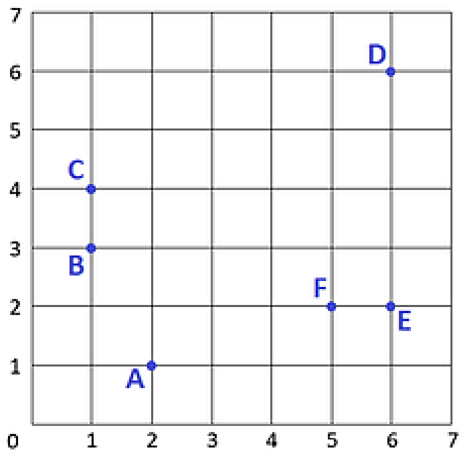
Data points

Algorithm:

```
Compute the distance matrix
Let each data point be a cluster
Repeat
        Merge the two closest clusters
        Update the proximity matrix
Until only a single cluster remains
```

Step 1

- Step 1: Compute the distance matrix using the Manhattan distance



Data points

	A	B	C	D	E	F
A	x	3	4	9	5	4
B		x	1	8	6	5
C			x	7	7	6
D				x	4	5
E					x	1
F						x

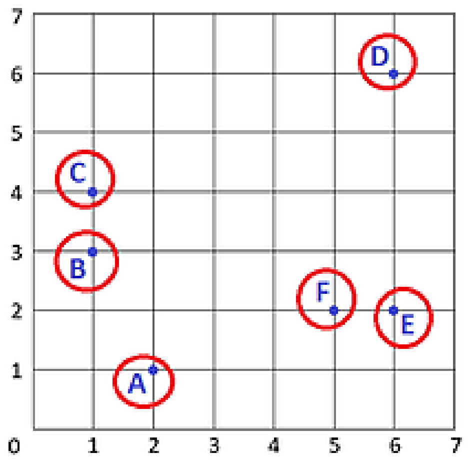
Distance matrix

Algorithm:

```
Compute the distance matrix
Let each data point be a cluster
Repeat
    Merge the two closest clusters
    Update the proximity matrix
Until only a single cluster remains
```

Step 2

- Step 2: Let each data point be a cluster



Data points

	A	B	C	D	E	F
A	x	3	4	9	5	4
B		x	1	8	6	5
C			x	7	7	6
D				x	4	5
E					x	1
F						x

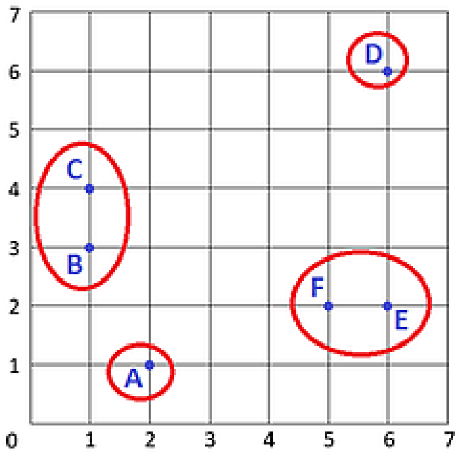
Distance matrix

Algorithm:

```
Compute the distance matrix
Let each data point be a cluster
Repeat
    Merge the two closest clusters
    Update the proximity matrix
Until only a single cluster remains
```

Step 3 part 1

- Step 3: Merge the 2 closest clusters and update the distance matrix
  - There are 2 pairs of clusters with the smallest distance of 1: (B and C) and (E and F) – we can merge them



Data points

	A	B	C	D	E	F
A	x	3	4	9	5	4
B		x	1	8	6	5
C			x	7	7	6
D				x	4	5
E					x	1
F						x

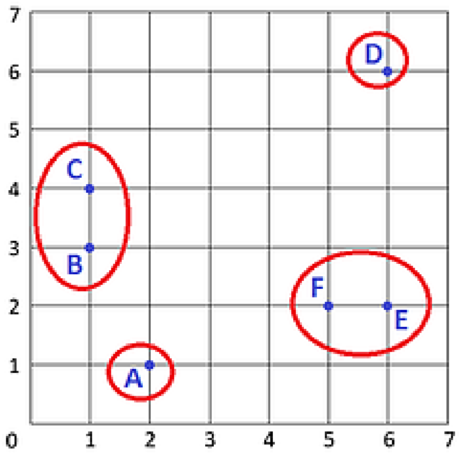
Distance matrix (not updated)

Algorithm:

```
Compute the distance matrix
Let each data point be a cluster
Repeat
        Merge the two closest clusters
        Update the proximity matrix
Until only a single cluster remains
```

Step 3 part 2

- Step 3: Merge the 2 closest clusters and update the distance matrix
  - B and C are merged; E and F are merged
  - Now we can update the distance matrix using the single-link distance



Data points

	A	B, C	D	E, F
A	x	3	9	4
B, C		x	7	5
D			x	4
E, F				x

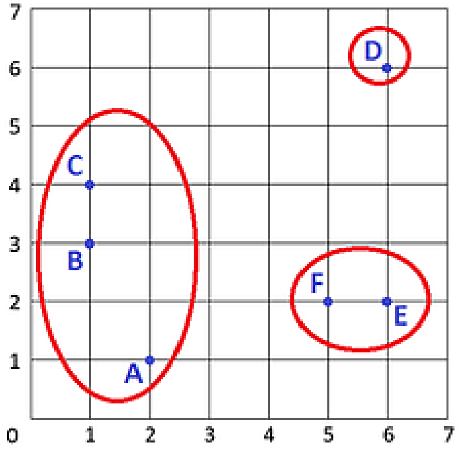
Distance matrix (updated)

Algorithm:

```
Compute the distance matrix
Let each data point be a cluster
Repeat
    Merge the two closest clusters
    Update the proximity matrix
Until only a single cluster remains
```

Step 4

- Step 4: Merge the 2 closest clusters and update the distance matrix (repeat until we have only 1 cluster)



Data points

	A	B, C	D	E, F
A	x	3	9	4
B, C		x	7	5
D			x	4
E, F				x

Distance matrix (not updated)

- Merge {A} with {B,C} (smallest distance)

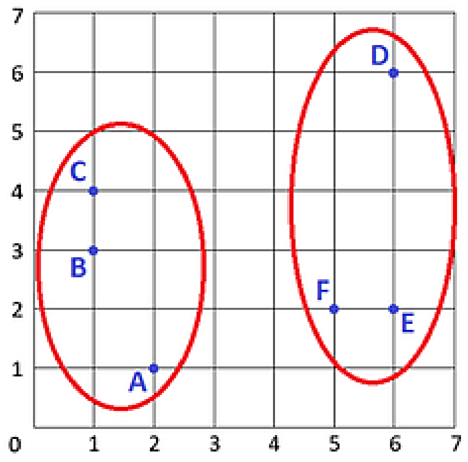
	A, B, C	D	E, F
A, B, C	x	7	4
D		x	4
E, F			x

Distance matrix (updated)

- Update the distance matrix

Step 5

- Step 5: Merge the 2 closest clusters and update the distance matrix (repeat until we have only 1 cluster)



Data points

	A, B, C	D	E, F
A, B, C	x	7	4
D		x	4
E, F			x

Distance matrix (not updated)

	A, B, C	D, E, F
A, B, C	x	4
D, E, F		x

Distance matrix (updated)

- The smallest distance is 4 in 2 cases:
  - 1) {A,B,C} and {E,F}
  - 2) {D} and {E,F}
- There is an overlap – both involve {E,F} – which one to merge first? We need a rule to resolve ties; assume random choice – we select 2)
- Merge {D} with {E,F} and update the distance matrix

Step 6 - finish

- Step 6: Merge {A,B,C} with {D,E,F}
- Finish – all items belong to the same cluster
- Draw the dendrogram

Algorithm:

Compute the distance matrix

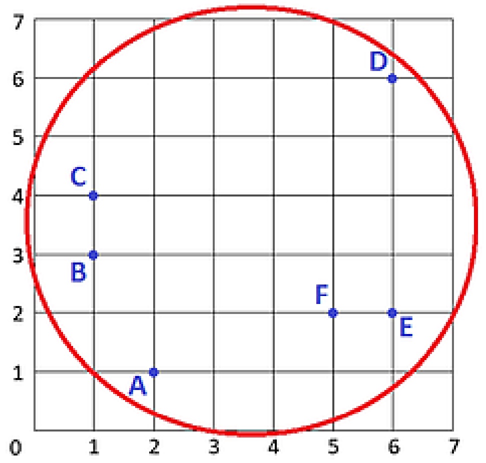
Let each data point be a cluster

Repeat

    Merge the two closest clusters

    Update the proximity matrix

Until only a single cluster remains



Data points

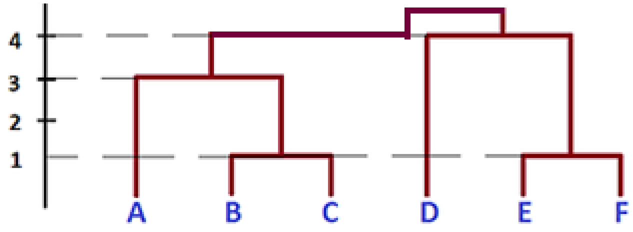
	A, B, C	D, E, F
A, B, C	x	4
D, E, F		x

Distance matrix (not updated)

	A, B, C, D, E, F
A, B, C, D, E, F	x

Distance matrix (updated)

- Note: The last 2 merges are both at distance 4



Dendrogram

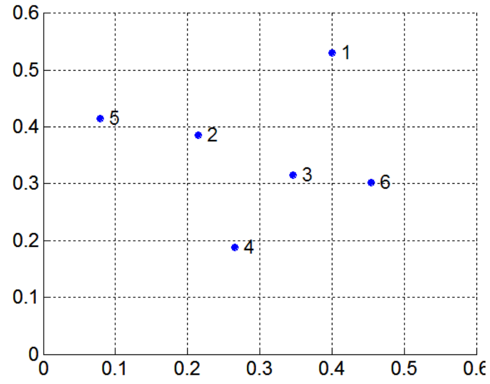
THE UNIVERSITY OF SYDNEY

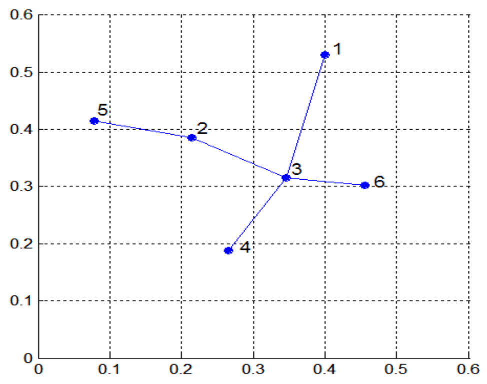
Divisive hierarchical clustering

- Less popular
- Can be implemented based on computing the minimum spanning tree

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

```
1: Compute a minimum spanning tree for the proximity graph.
2: repeat
3:     Create a new cluster by breaking the link corresponding to the largest distance
       (smallest similarity).
4: until Only singleton clusters remain
```



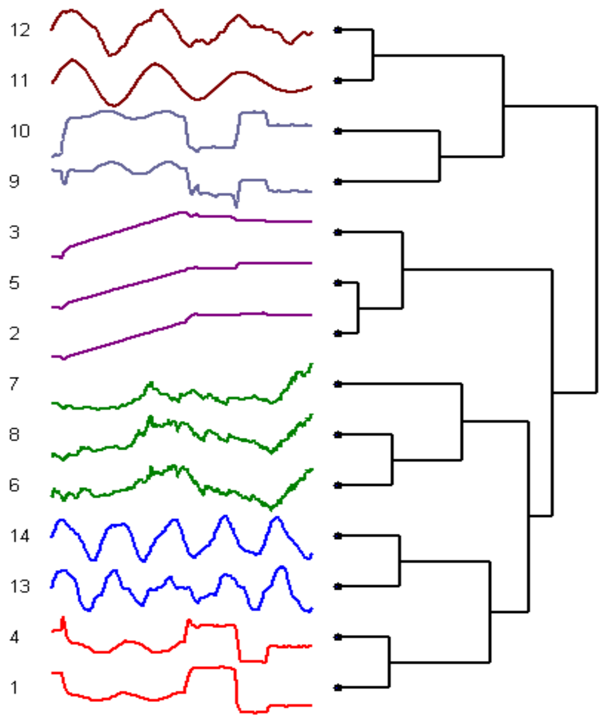


Minimum spanning tree

THE UNIVERSITY OF SYDNEY

Hierarchical clustering – example

- Using hierarchical clustering to cluster time series
- The similarity between two time series can be calculated using correlation or other measures

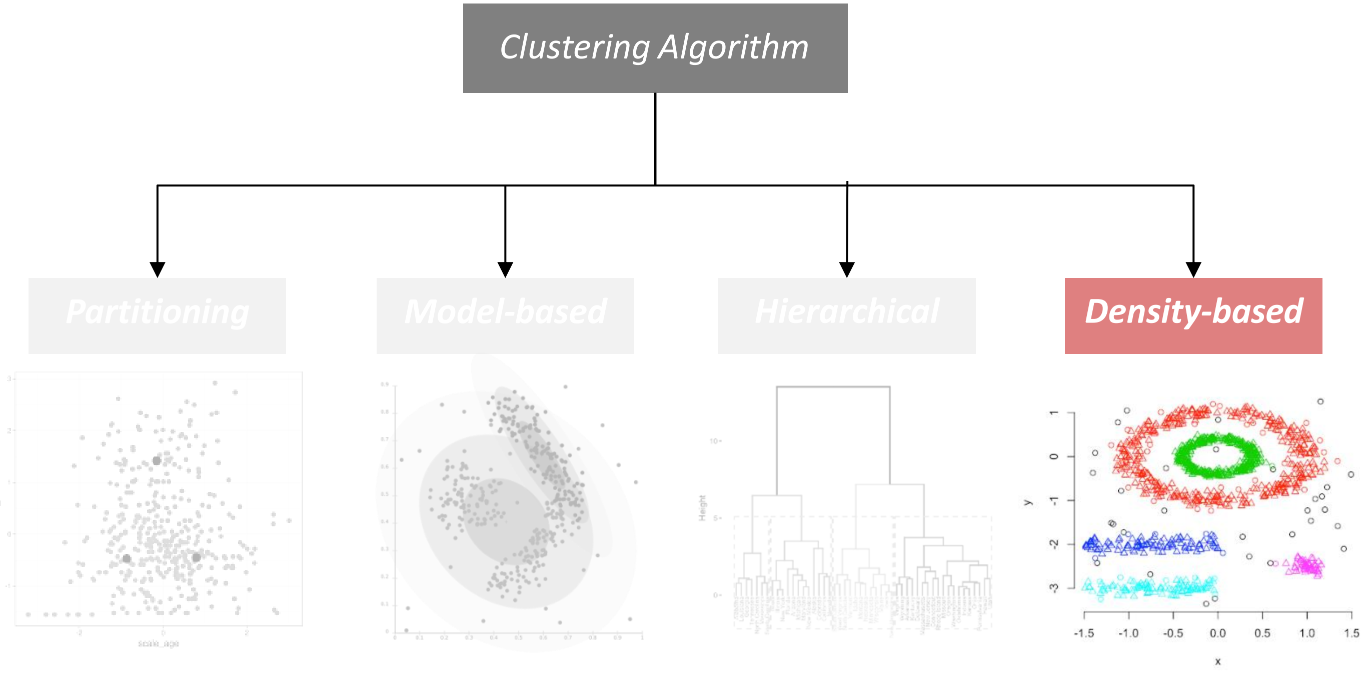


Hierarchical clustering – strengths and limitations

- Especially suitable for tasks with natural nesting relationships between clusters (taxonomies, hierarchies)
- Does not require the number of clusters to be specified in advance
- Computationally expensive which limits its applicability to high dimensional data
  - Space complexity: $O(n^2)$, n - number of examples (storing the distance matrix and dendrogram)
  - Time complexity: $O(n^3)$ – n levels; at each of them n^2 distance matrices must be searched and updated
- Not incremental – assumes all data is present
- Sensitive to noise and outliers
  - Outliers are more problematic for the Ward’s method as they increase the SSE and less problematic for the single, complete and average link
  - Outliers tend to form single clusters that do not merge with any other clusters until later in the process; they can be removed by discarding small clusters

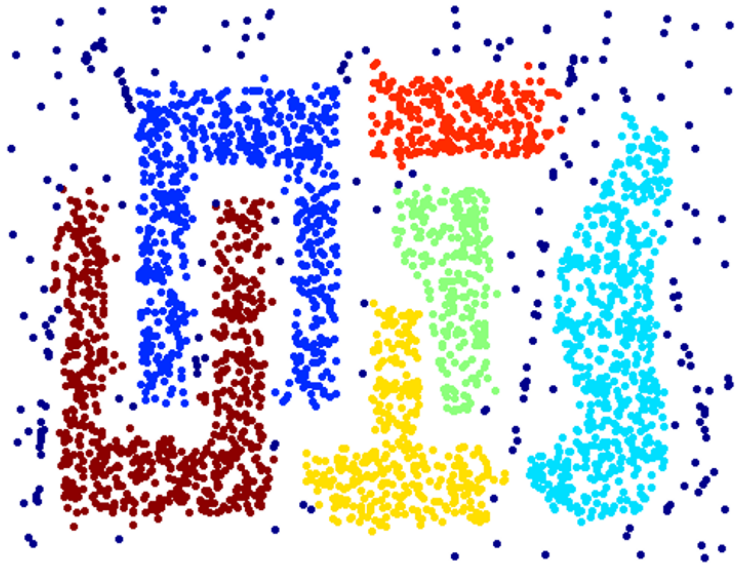
Clustering

Density-based: DBSCAN



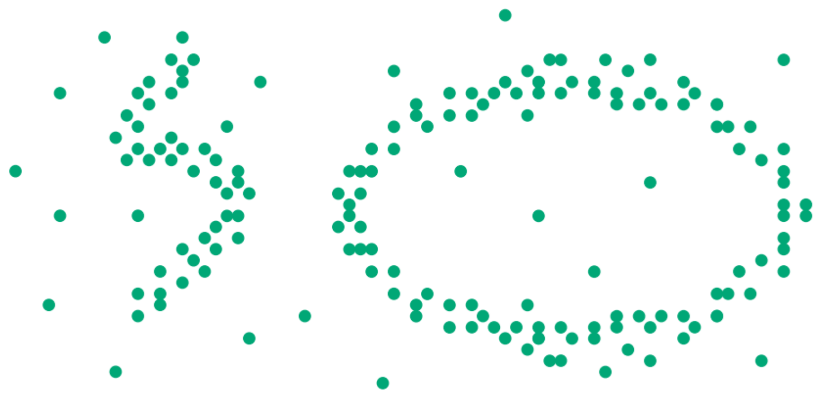
DBSCAN

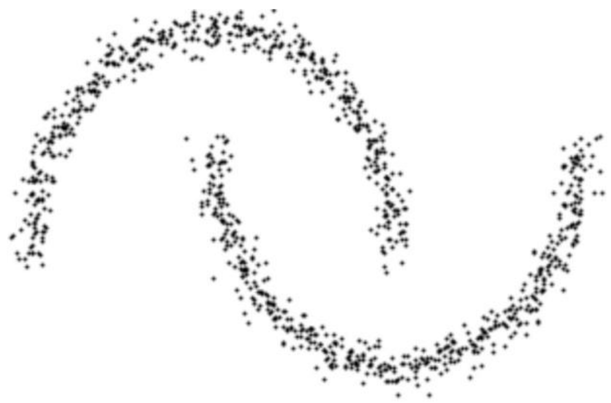
- DBSCAN – **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Clusters are regions of high density, separated from one another by regions with low density



DBSCAN

- In contrast to K-Means which finds circular clusters, DBSCAN can find clusters with arbitrary and complex shape, e.g. S shape, ovals, half circles



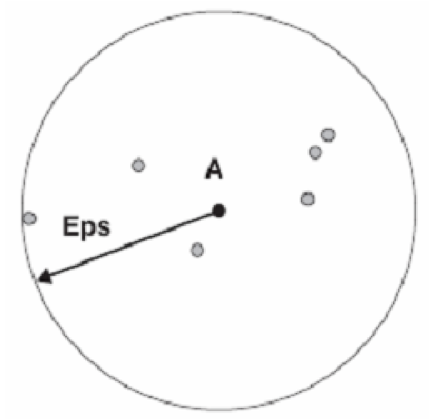


We will apply DBSCAN to the moons dataset during the tutorial!

Main idea

- If point A is in a cluster, the density of the points around it should be higher than a threshold
    - We need to define density and neighborhood of a point

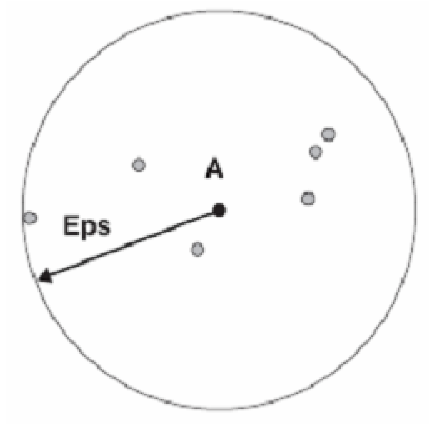
- Neighborhood of a point A = the area within a radius Eps
- Density of a point A = the number of points in the neighborhood of A, including the point A
- Density threshold MinPts – the minimum number of points in the Eps neighborhood of a point



What is the density of A?

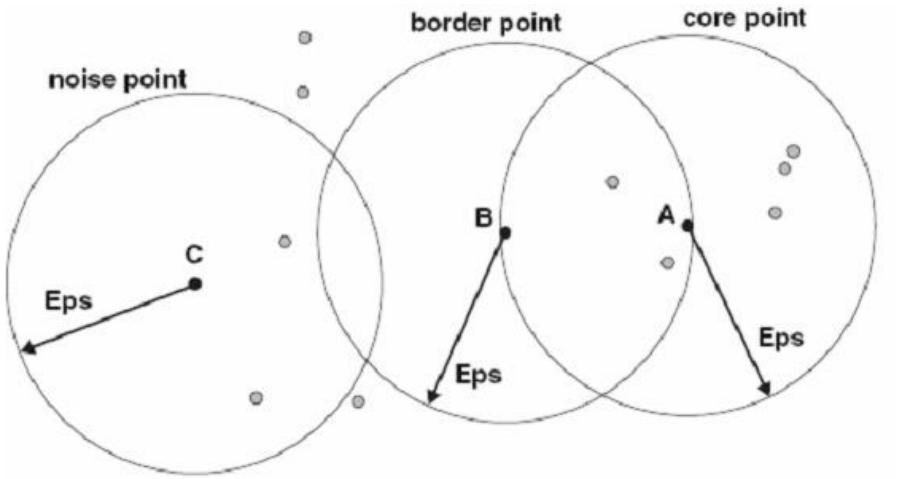
Density depends on Eps

- The density of a point depends on the neighborhood Eps
  - Eps too big - all points will have a density of *m* (*m* =number of points in the dataset)
  - Eps is too small – all points will have a density of 1



Three types of points in DBSCAN

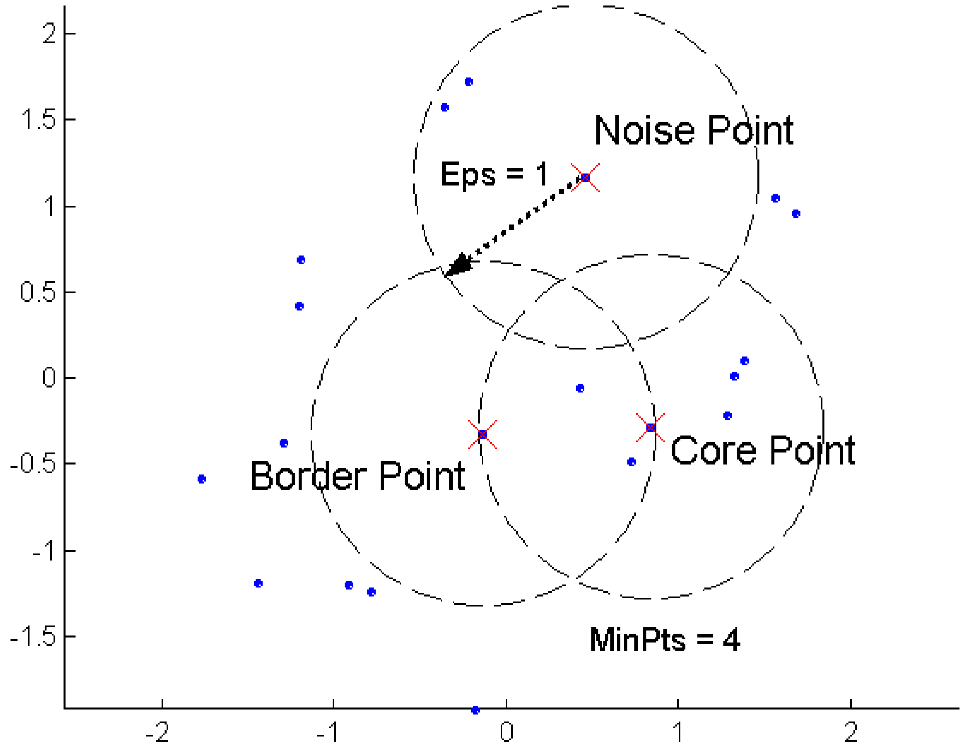
- There are 3 types of points in DBSCAN: core, border and noise
- Given Eps and MinPts, a point X is a:
  - Core point – if its Eps neighborhood contains at least MinPts points. Core points are in the interior of a density-based cluster.
  - Border point – if it is not a core point but falls within the neighborhood of a core point. A border point may fall in the neighborhood of several core points.
  - Noise point – if it is not a core or a border point



For Eps and MinPts =7:
A – core
B – border
C – noise

Core, border and noise points - example

- Another example showing the 3 types of points:



DBSCAN algorithm

1. Label points as core, border and noise
2. Discard noise points
3. Cluster the remaining points as follows
   - Any 2 core points within Eps of each other are put in the same cluster
   - Any border point from the neighborhood of a core point is put in the same cluster as the core point
   - Ties need to be resolved when a border point belongs to the neighborhood of more than 1 core point

Example

- Given are 5 items and their distance matrix. What clusters will DBSCAN find for Eps=2 and MinPts=3?

	A	B	C	D	E
A	0	1	4	5	6
B	1	0	2	6	7
C	4	2	0	3	4
D	5	6	3	0	1
E	6	7	4	1	0

Hint:

1. Find neighborhoods using Eps<=2
2. Label points using MinPts>=3
3. Find clusters

Solution

- Given are 5 items and their distance matrix. What clusters will DBSCAN find for Eps=2 and MinPts=3?

	A	B	C	D	E
A	0	1	4	5	6
B	1	0	2	6	7
C	4	2	0	3	4
D	5	6	3	0	1
E	6	7	4	1	0

The neighborhood of A contains 2 points: A and B; don't forget to include the point itself

1. Find the neighborhoods for Eps<=2:

N(A)={A, B}

N(B)={B, A, C}

N(C)={C, B}

N(D)={D, E}

N(E)={E, D}

2. Label points as core, border and noise

MinPts>=3: core, border or noise?

A: <u>border</u> or noise?

B: core

C: <u>border</u> or noise?

D: border or <u>noise</u>?

E: border or <u>noise</u>?

3. Find clusters

{A,B,C} - 1 cluster only

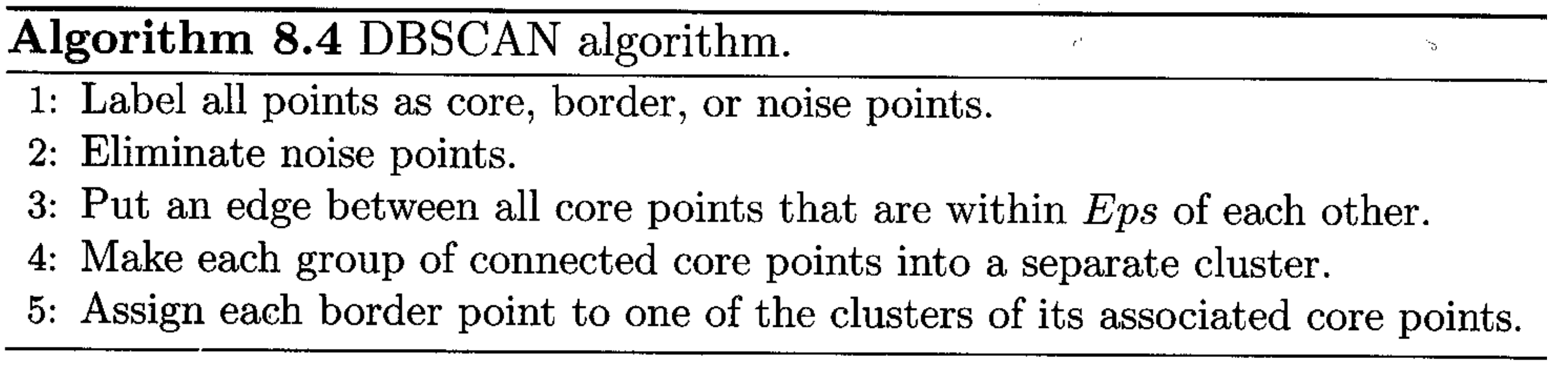
Example 2

- Given are 5 items and their distance matrix. What clusters will DBSCAN find for Eps=1 and MinPts=2?

	A	B	C	D	E
A	0	1	4	5	6
B	1	0	2	6	7
C	4	2	0	3	4
D	5	6	3	0	1
E	6	7	4	1	0

Hint:

1. Find neighborhoods using Eps<=1
2. Label points using MinPts>=2
3. Find clusters



Algorithm 8.4 DBSCAN algorithm.

```
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within $Eps$ of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.
```

Solution

- Given are 5 items and their distance matrix. What clusters will DBSCAN find for Eps=1 and MinPts=2?

	A	B	C	D	E
A	0	1	4	5	6
B	1	0	2	6	7
C	4	2	0	3	4
D	5	6	3	0	1
E	6	7	4	1	0

1. Find the neighborhoods for Eps<=1:

N(A)={A, B}

N(B)={B, A}

N(C)={C}

N(D)={D, E}

N(E)={E, D}

2. Label points as core, border and noise

MinPts>=2: core, border or noise?

A: core

B: core

C: border or noise?

D: core

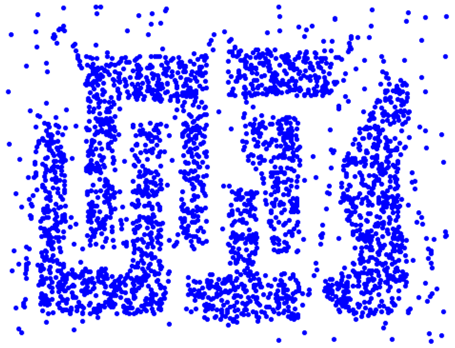
E: core

3. Find clusters

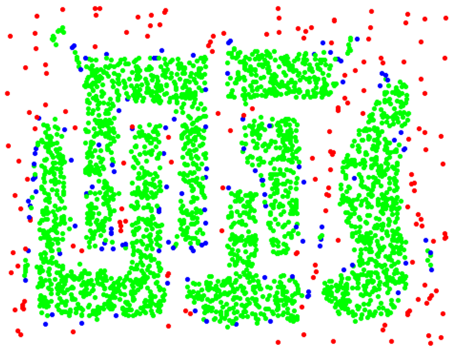
{A,B}, {D,E} – 2 clusters

How many clusters?

- Different number of clusters depending on Eps and MinPts

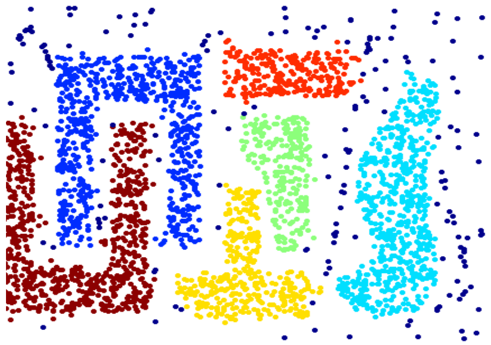


Original points

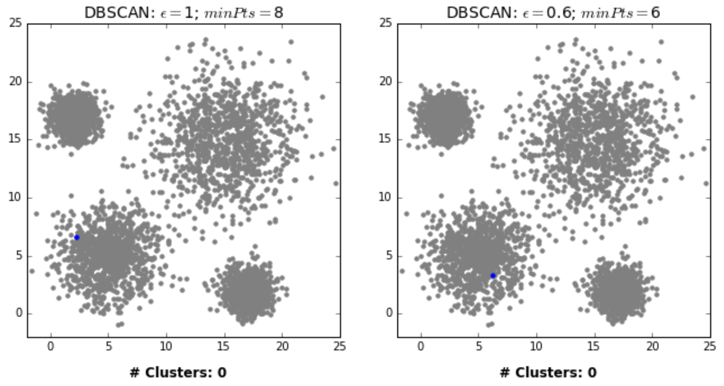


1 cluster

core, border and noise points



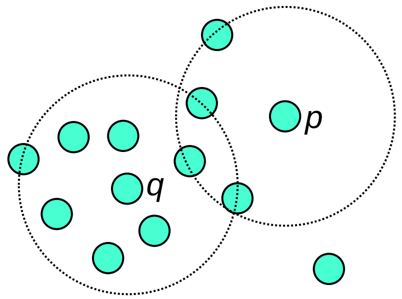
6 clusters



https://dashee87.github.io/images/DBSCAN_search.gif

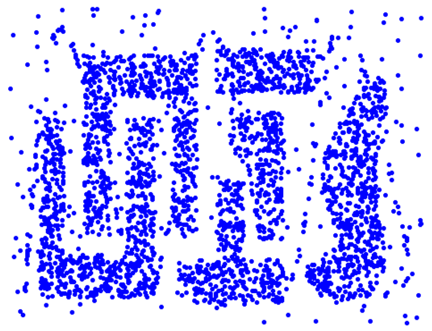
How to select Eps and MinPts?

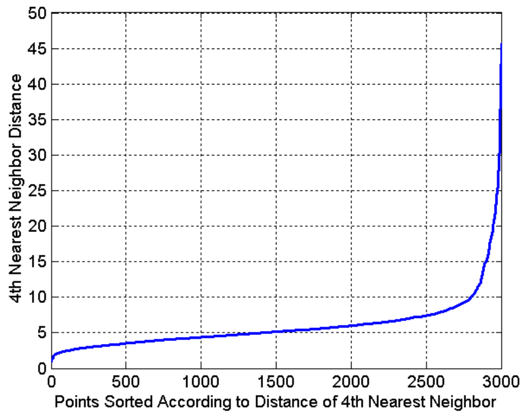
- If Eps is too big, then all points will be core points
- If Eps is too small, then all points will be noise



Method for selecting Eps and MinPts

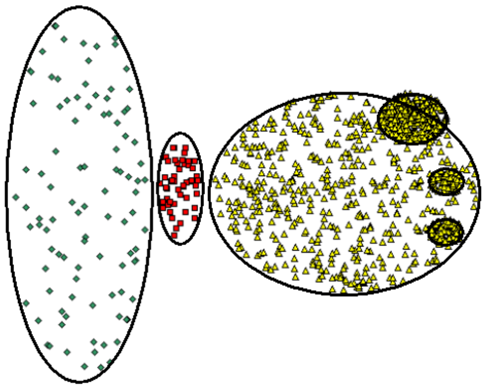
- Examine the distance from a point to its k-th nearest neighbor (k-dist)
  - For points that belong to some cluster, k-dist will be small (if k is not larger than the cluster size)
  - For points that are not in a cluster, such as noise points, k-dist will be large
- Hence, we can compute the k-dist for all data points for some k (e.g. for k=4), sort them in increasing order and plot the graph. A sharp increase in k-dist indicates a suitable distance value for Eps and MinPts can be set to k.
- If we do this, the points for which k-dist is less than Eps will be labelled as core points, while the others will be labelled as noise or border points
- However, this method requires k

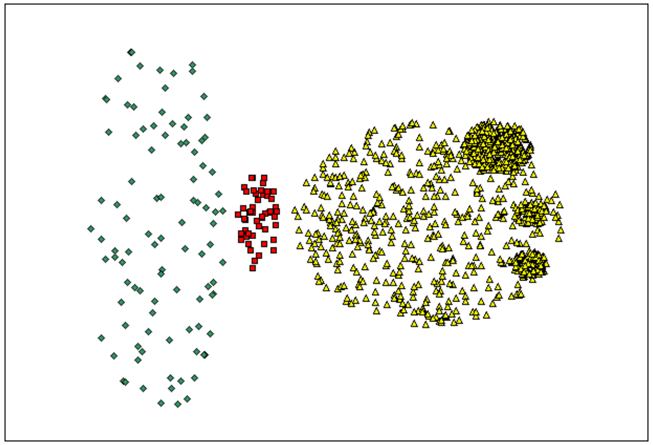




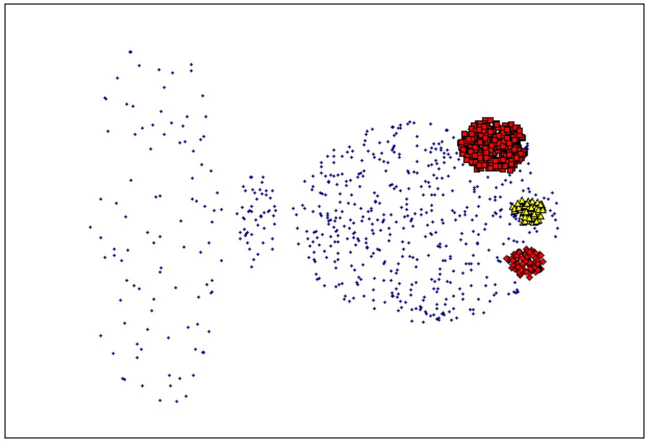
DBSCAN with clusters of varying density

- Cannot handle well clusters with widely varying density
- Different results depending on the parameters Eps and MinPts


Original points


DBSCAN

MinPts=4, Eps=9.75

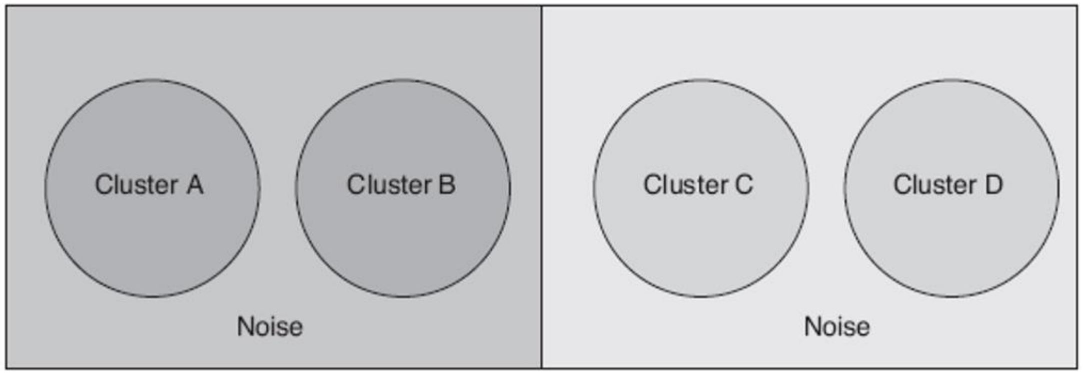

DBSCAN

MinPts=4, Eps=9.92

DBSCAN with clusters of varying density (2)

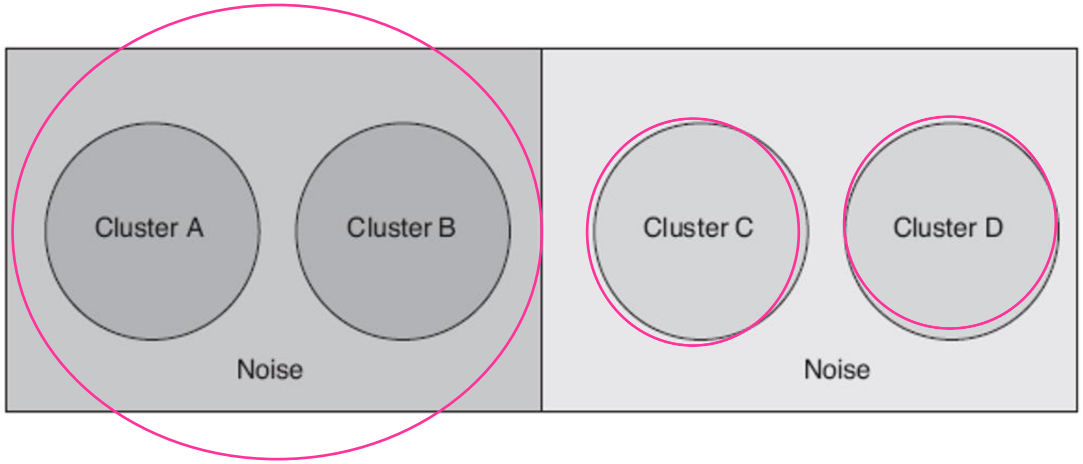
- Another example
  - 4 clusters: A, B, C and D, embedded in noise
  - The darker the color, the higher the density
  - The noise around A and B has the same density as clusters C and D

- How many clusters will DBSCAN find?



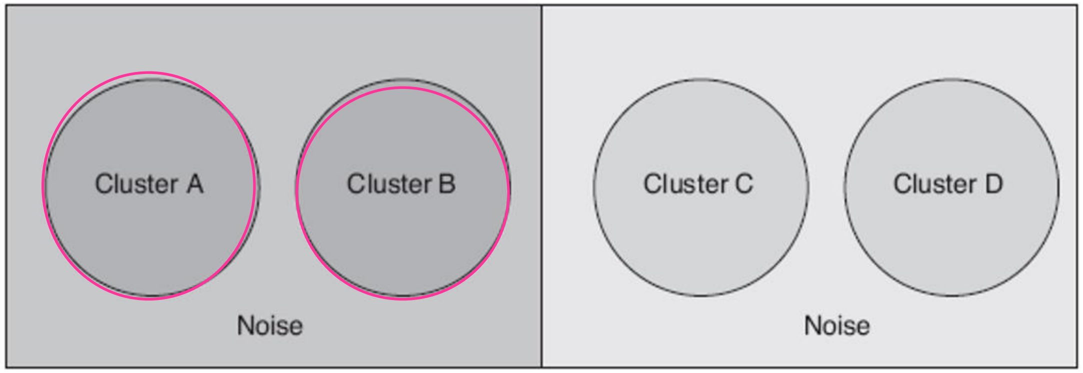
How many clusters?

- For a fixed MinPts, if Eps is high (big enough) so that C and D are located as separate clusters and the points around them as noise, then A, B and the points around them will become 1 cluster
- => There will be 3 clusters: {C}, {D}, {A,B, noise around them}



How many clusters? (2)

- For a fixed MinPts, if Eps is low (small enough) so that A and B can be located as separate clusters and the points around them are marked as noise, then C, D and the points around them will be marked as noise
- => There will be 2 clusters: {A}, {B}



DBSCAN – time and space complexity

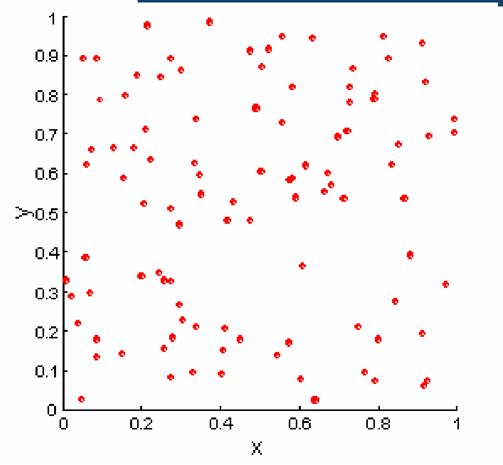
- Time – $O(n^2)$, n - number of points
  - Can be reduced to $O(n \log n)$ in low dimensional spaces using kd-trees
- Space – $O(n)$
  - Requires to keep a small amount of information for each point – its cluster and type: core, border or noise

DBSCAN – strengths and weaknesses

- Strengths:
  - Can form clusters of arbitrary shapes and different sizes => can find clusters that K-means cannot
  - Does not require the number of clusters to be pre-specified (but this is done indirectly by specifying Eps and MinPts)
  - Resistant to noise as it uses a density-based definition of a cluster and labels points as noise
- Weaknesses:
  - Does not work well for clusters with widely varying density
  - Does not work well for high dimensional data – more difficult to define density
  - Sensitive to the input parameters Eps and MinPts
  - Eps and MinPts may be difficult to determine – heuristic approaches to determine their values

Evaluating clustering results

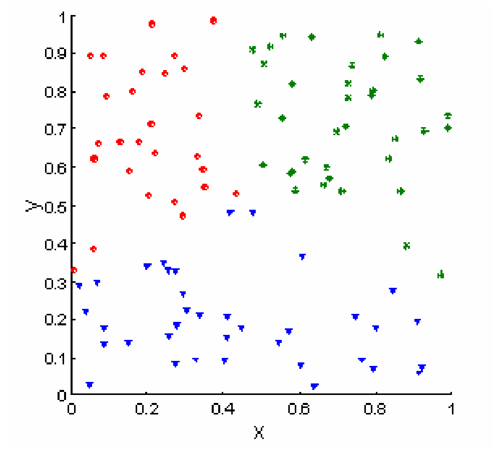
Clusters found in random data



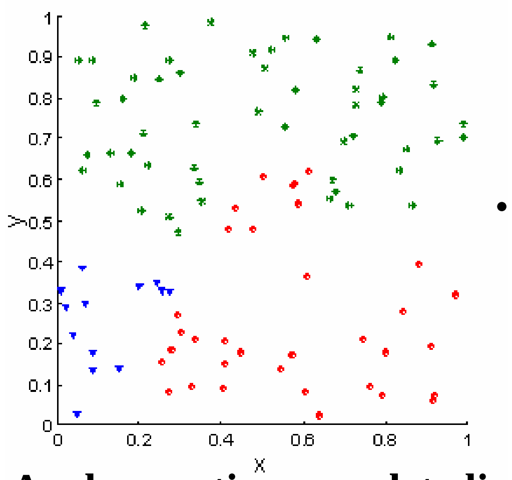
Random points

- 1000 data points, randomly generated
- We set K-means and agglomerative also to find 3 clusters

- All clustering algorithms will find clusters even if data is random and has no natural clusters
- We need to evaluate the resulting clusters as we don't want to find patterns in noise



K-means



Agglomerative complete link

How good is our clustering?

- 1) Evaluating clustering quality using unsupervised measures
  - High similarity within a cluster, low similarity between clusters
  - These measures are also called internal
- 2) Evaluating clustering quality using supervised measures
  - Comparing the clustering results with “ground truth” (correct cluster labels provided by an expert
  - These measures are also called external

Related topics:

- 3) Determining the correct number of clusters

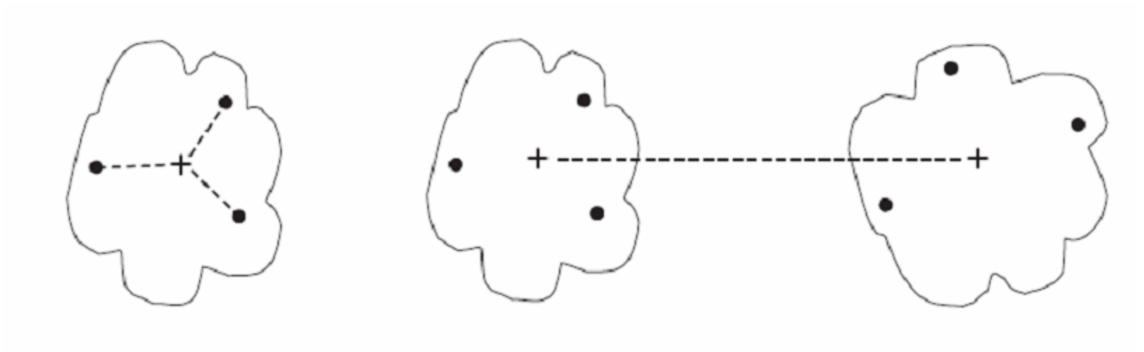
1) Evaluating clustering quality using unsupervised measures

- Method 1: Measuring cohesion and separation
  - Cohesion and separation alone
  - Combining cohesion and separation: Silhouette coefficient
- Method 2: Correlation between 2 similarity matrices
  - matrix 1: derived from the distance matrix
  - matrix 2: derived from the clustering results
  - Are the similar items in the same cluster?
- Method 3: Visual inspection of sorted similarity matrix

Unsupervised method 1: Cohesion and separation

Good clustering

- A good clustering produces clusters with
  - high cohesion = high similarity within the cluster
  - high separation = low similarity between the clusters



(a) Cohesion. (b) Separation.

- Cohesion and separation are measured using distance measures

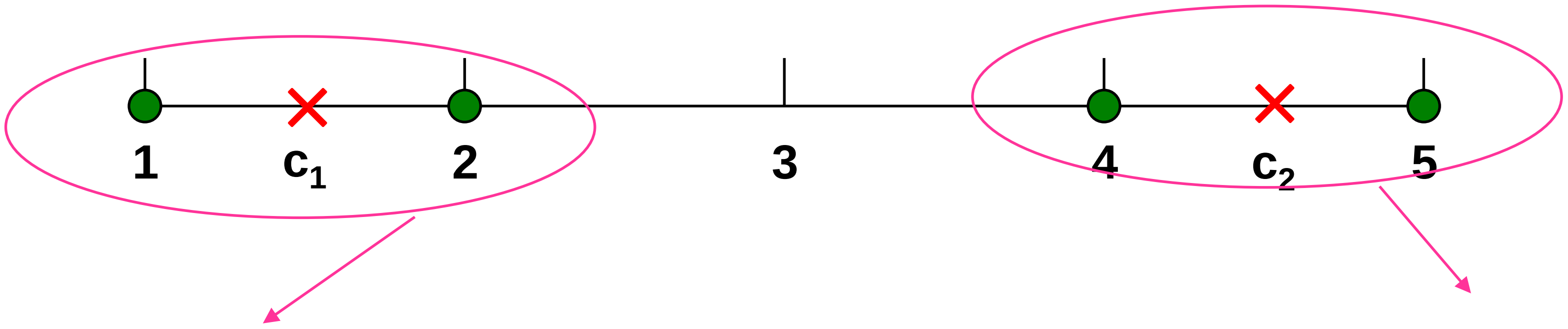
Cohesion

- Cohesion of a cluster K_i:

$$cohesion(K_i) = \sum_{x \in K_i} dist(x, c_i)$$

c_i is the cluster centroid, x are all the objects in this cluster

- Example: 2 clusters K1 and K2:



Using the Manhattan distance:
cohesion(K1) =|1-1.5|+|2-1.5|=1

cohesion(K2)=|4-4.5|+|5-4.5|=1

- Cohesion of the whole clustering - overall cohesion over all clusters K_i:

cohesion =cohesion(K1)+cohesion(K2)=2

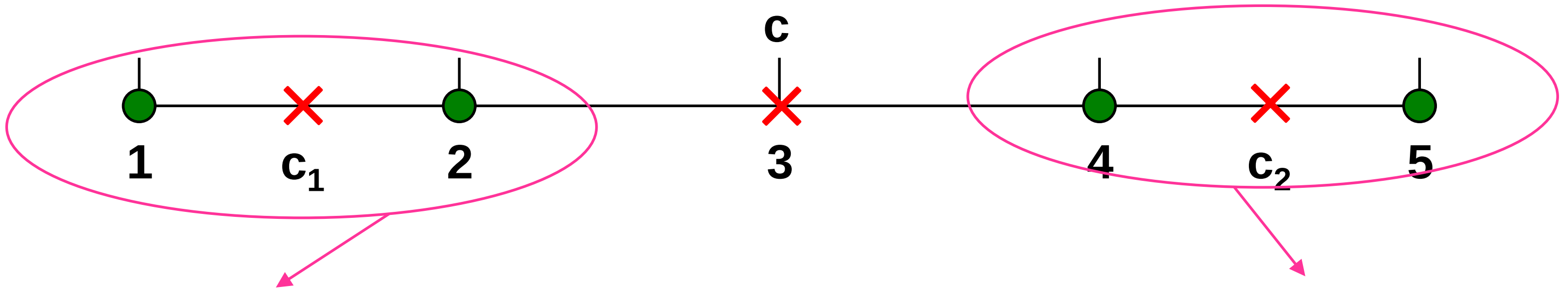
Separation

- Separation of a cluster K_i from the other clusters:

$$separation(K_i) = dist(c_i, c)$$

c is the overall centroid (all points considered)

c_i are the cluster centroids, i=1,2



separation(K1) =|1.5-3|=1.5

separation(K2)=|4.5-3|=1.5

- Separation of the clustering - overall separation weighted by the size of each cluster $|K_i|$:

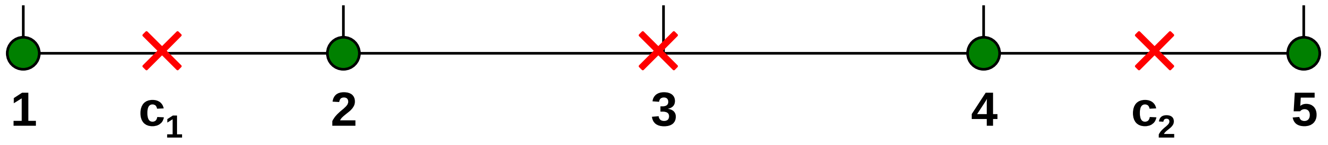
$$separation = \sum_{i=1}^{k} |K_i| dist(c_i, c)$$

- Sum of the separations of the individual clusters, weighed by their size

separation =2*|1.5-3|+2*|4.5-3|=6

Cohesion and separation using squared distance

- We can use squared distances to express cohesion and separation:
  - SSE = distance within a cluster $$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in K_i} (c_i, \mathbf{x})^2$$
  - BSE = distance between clusters $$BSE = \sum_{i=1}^{k} |K_i| (c_i, c)^2$$



k=2 clusters

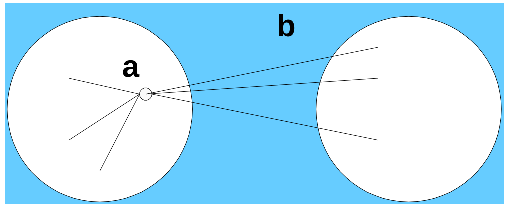
({1,2}, {3,4})

SSE = $(1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$

BSE = $2*(3-1.5)^2 + 2*(4.5-3)^2 = 9$

Silhouette coefficient

- Combines cohesion and separation
- Calculated for a point, cluster and clustering



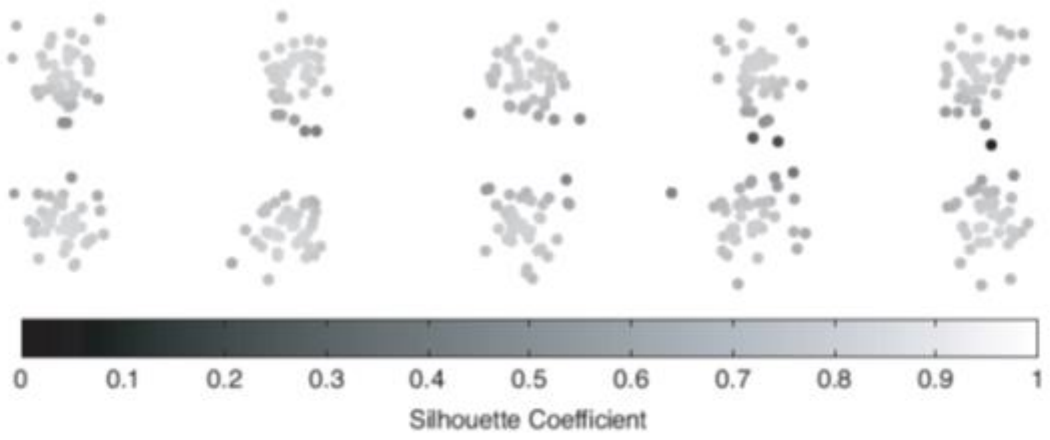
- Silhouette coefficient for a point i:
  - a_i = the average distance from i to all points in its cluster (cohesion)
  - b_i = the average distance from i to all points in another cluster (separation). Find the minimum value with respect to all clusters. (minimum average distance to points in another cluster)
  - Silhouette coefficient $s_i = (b_i - a_i)/\max(a_i, b_i)$

- Values between -1 and 1; the higher the better
  - max value=1 when $a_i=0$ and $b_i>a_i$ ($s_i=b_i/b_i$, high cohesion & high separation)
  - negative values: $b_i<a_i$ – bad quality clustering (separation<cohesion)

Silhouette coefficient (2)

- Silhouette coefficient for a cluster
  - average of the Silhouette coefficients for all points in the cluster

- Silhouette coefficient for a clustering
  - average of the Silhouette coefficients for all clusters



- Example: Silhouette coefficient calculated for all points in the 10 clusters
- Darker color = lower Silhouette coefficient

Unsupervised method 2: Correlation between two similarity matrices

Correlation between similarity matrices

- Finds if the similar items (the items which are close to each other) are in the same cluster
- Example:
  - 4 items were clustered into 2 clusters {P1, P2} and {P3, P4}
  - Similarity matrix:

	P1	P2	P3	P4
P1	1	0.8	0.65	0.55
P2		1	0.7	0.6
P3			1	0.9
P4				1

- Task: Evaluate the clustering quality using correlation
- Idea: Compute the correlation between the given similarity matrix and the similarity matrix defined by the clustering (value =0, if 2 items are not in the same cluster, value =1, if they are in the same cluster)

	P1	P2	P3	P4
P1	1	1	0	0
P2		1	0	0
P3			1	1
P4				1

- High correlation: good clustering (similar items are in the same cluster)
- Low correlation: poor clustering (similar items are not in the same cluster)

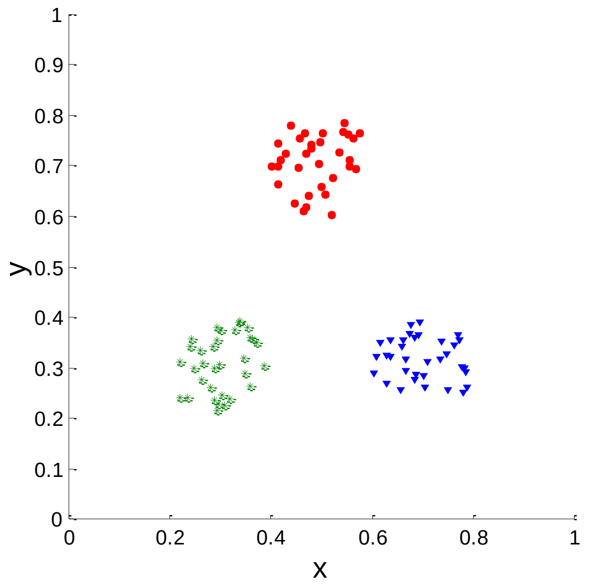
Correlation method - summary

- Finds the correlation between 2 similarity matrices
  - matrix1 computed from the distance matrix
  - matrix2 computed from the clustering results
- The higher the correlation, the better the quality of the clustering
- Matrix1 - computed from the distance matrix
  - different ways to define similarity; in general the higher the distance, the smaller the similarity, e.g.:
  - sim=1 - (d - d_min)/(d_max - d_min)
- Matrix2 – based on the cluster labels obtained by the clustering
  - values of 0 and 1 only
  - ij entry = 0 if items i and j are from different clusters
  - ij entry = 1 if items i and j are from the same cluster
- Revision: correlation measures a linear relationship between numeric attributes, see lecture 1b; range = [-1, 1], -1: perfect negative correlation, +1: perfect positive, 0- no correlation

Evaluating Clustering Quality Based on Correlation – Example

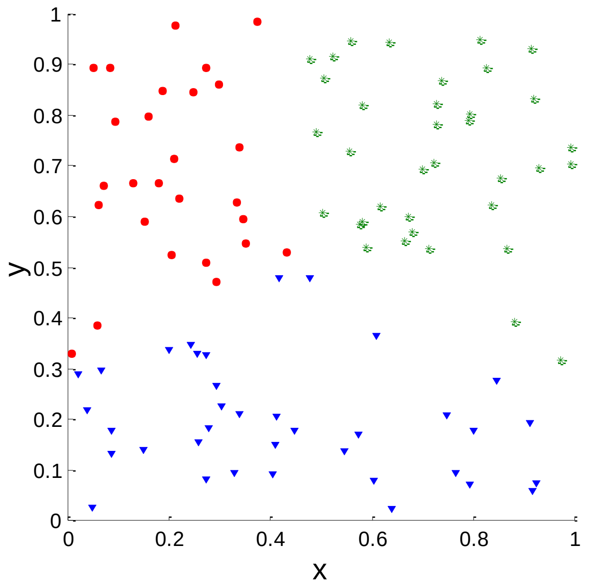
- Example for K-means clustering

data with 3 well separated clusters



Corr = -0.9235

random data – not well separated clusters

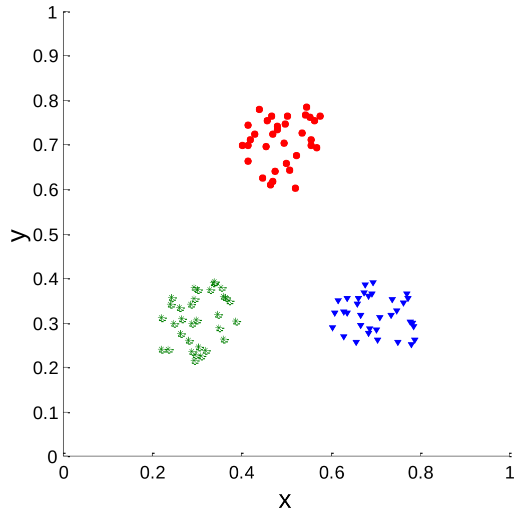


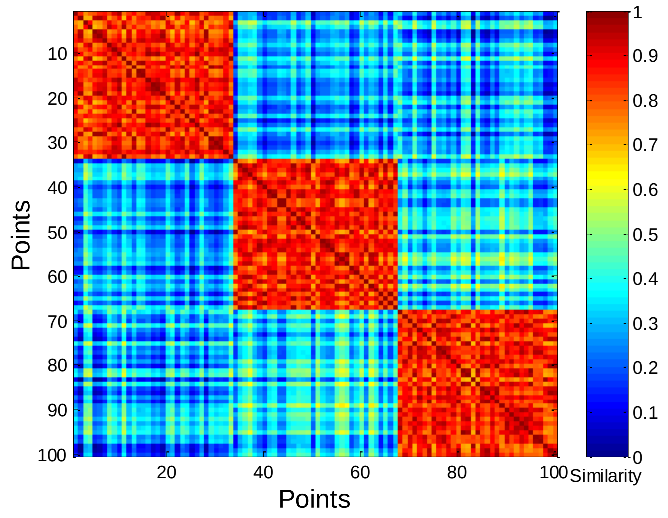
Corr = -0.5810

Unsupervised method 3: Visual inspection of similarity matrix

Visual inspection of the similarity matrix

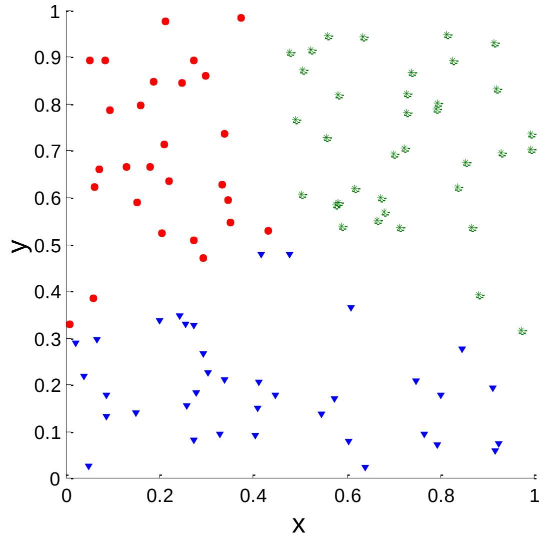
- Order the points based on their cluster
  - Points from cluster 1, points from cluster 2, etc.
- Plot the similarity matrix using coloring based on the similarity
- Well defined blocks along the main diagonal indicate good clustering
  - I.e. items from the same cluster are similar to each other

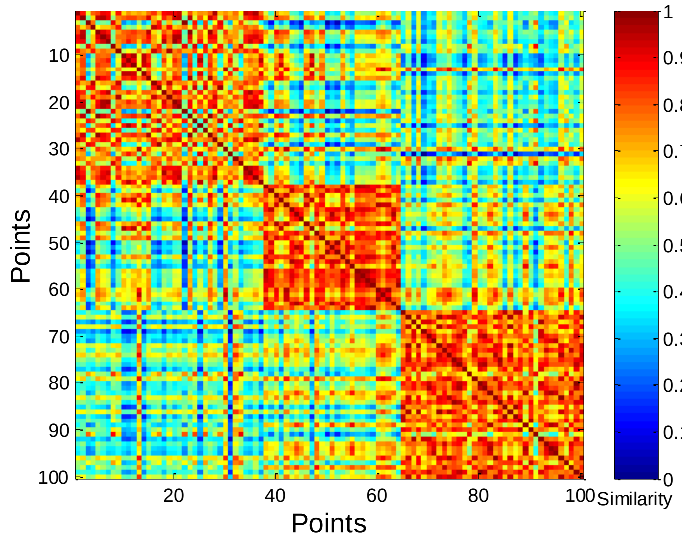




Example: K-means

- Clustering random data: weak block diagonal patterns

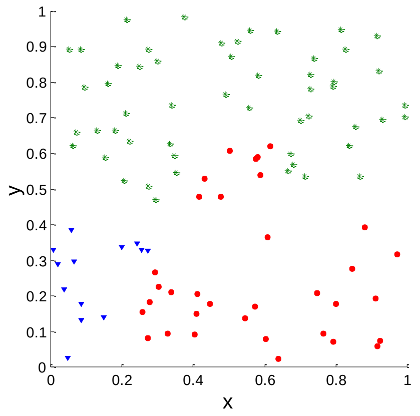


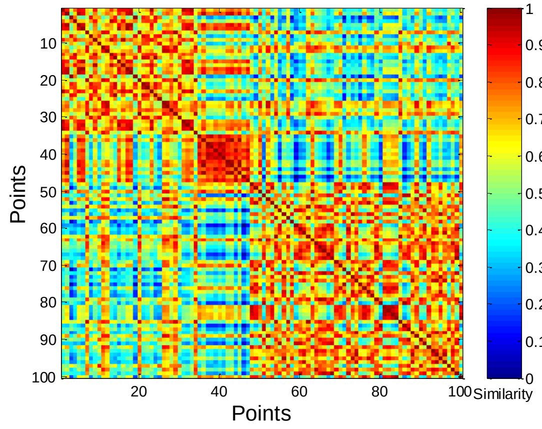


K-means

Example: agglomerative

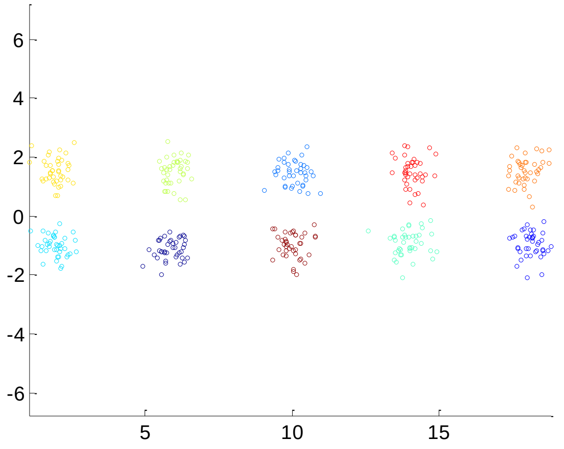
- The same: weak block diagonal patterns when clustering random data





Agglomerative
complete link

3) Determining the Number of Clusters



- How to determine a good number of clusters?

- Elbow method
  - Run the clustering algorithm for several k, plot SSE or other unsupervised measure vs number of clusters
  - Look for distinct knee (drop) or peak = good number of clusters

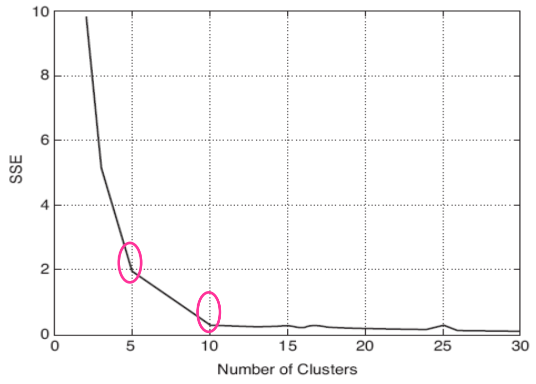


Figure 8.32. SSE versus number of clusters for the data of Figure 8.29.

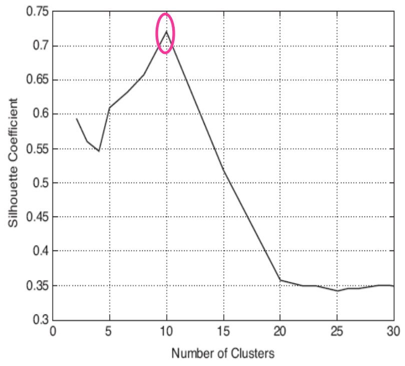


Figure 8.33. Average silhouette coefficient versus number of clusters for the data of Figure 8.29.

- See also the tutorial exercises!

Clustering evaluation - summary

- How good is our clustering?
- Unsupervised evaluation
  - Cohesion and separation alone or combined (e.g. Silhouette coefficient)
  - Correlation between 2 similarity matrices - 1) derived from the distance matrix and 2) from the clustering results
  - Visual inspection of similarity matrix (sorted)
- Determining the number of clusters – Elbow method