TÉCNICO+
FORMAÇÃO AVANÇADA

Clustering (2/2)

Clustering: advanced aspects

DASH: Data Science e Análise Não Supervisionada
Rui Henriques, rmch@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa

Outline

- Distances: advanced notes
- Clustering approaches
  - *k*-means
  - model-based (EM)
  - deep learning
- Evaluation recap
- Advanced aspects

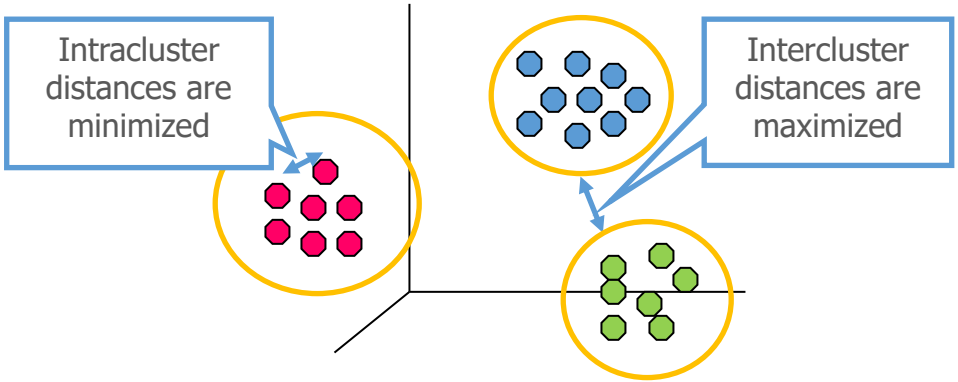
Outline

- **Distances: advanced notes**
- Clustering approaches
  - *k*-means
  - model-based (EM)
  - deep learning
- Evaluation recap
- Advanced aspects

Recall: clustering

Cluster: group of observations

Cluster analysis: group observations into clusters according to their (dis)similarity: observations in the same cluster are more similar than those in different clusters

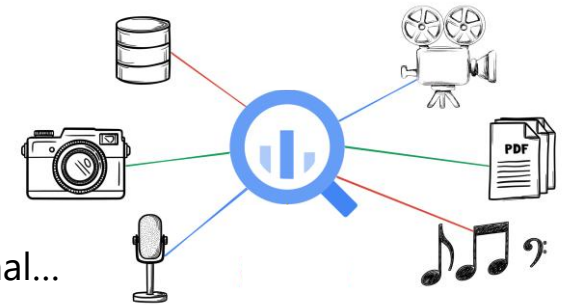


Main ingredients for clustering: ***distance + approach***

Clustering: distance and approach

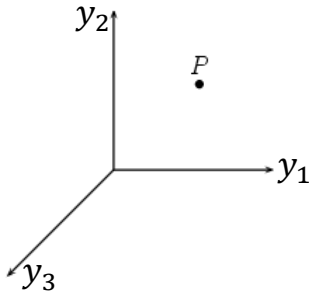
Distance metrics depend on the:

- type of input variables:
  - *numeric* distances, e.g. Euclidean, $d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{j=1}^{m}(a_j - b_j)^2}$
  - *nominal* distances, e.g. Hamming
  - *ordinal* encodings, how?
  - *non-iid attributes,* how?
- data structures
  - (multivariate) time series
  - image
  - text
  - others: spatiotemporal, events, relational…

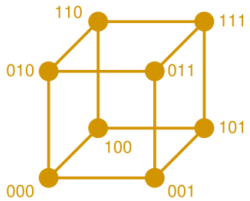


Categorical features

How do we represent categorical features in a multivariate space?



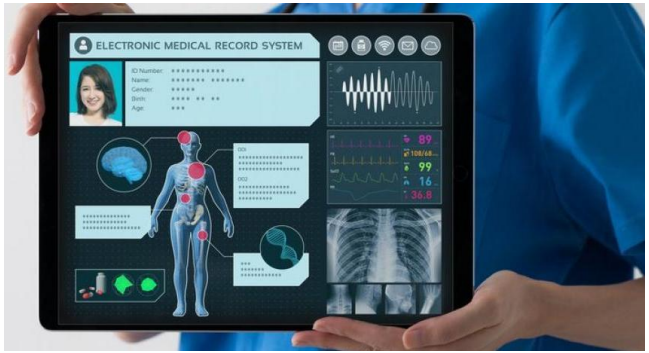
- Let us start with **binary variables** (e.g. gender)
  - a binary feature can be mapped into one axis in a m-dimensional space where values are constrained to 0 or 1
- **Ordinal variables** (e.g. high-moderate-low)
  - we can find a numeric encoding f for an ordinal variable
    - e.g. $f(high) = 5$, $f(moderate) = 1$, $f(low) = 0$
  - in the absence of an encoding, one can assume equally spaced integers (e.g. 0, 1, 2…). Problems?
- Let us finally consider **nominal variables** with cardinality higher than 2 (e.g. Europe, Africa, America)
  - challenge? As there is no order, we cannot position values along a single axis/dimension
  - solution? Create p axes for a nominal variable with cardinality p
    - only one of the p axes is active (1), while remaining are inactive (0)
    - this operation is called **dummification**
    - some machine learning methods depend on dummification



Mixed data

- A data space can be composed by **non-iid numeric variables**
  - *i.i.d.* stands for **i**ndependent and **i**dentically **d**istributed
  - two numeric variables may be correlated, i.e. not independent
  - two numeric variables may have distinct distributions (e.g. $U(0,100)$ and $N(3,5)$), i.e. non-identically distributed

- Observations can have numeric, nominal and ordinal features
  - we informally term such data as **mixed data**
  - examples?



Distances for mixed data

- First concern: two non-identically distributed variables – e.g. $Y_1 \sim U(0,1)$ and $Y_2 \sim U(0,100)$
  - *Problem*? In the given example, distances are most affected by variable Y_2
    - yet why should this variable have higher weight than Y_1!
  - *Possible solution*? Normalize variables
    - in the given example, $Y_2 \sim U'(0,1)$ using for instance min-max scaling

- Second concern: how to deal with simultaneous categoric and numeric variables?
  - distance can be a composition: $d(\mathbf{x}_1, \mathbf{x}_2) = \alpha\ d_{numeric}(\mathbf{x}_1, \mathbf{x}_2) + \beta\ d_{categoric}(\mathbf{x}_1, \mathbf{x}_2)$
    where α and β are parameters that reveal the relevance of each component, generally $\alpha + \beta = 1$
    - parameters can be fixed based on the number of variables or available domain knowledge

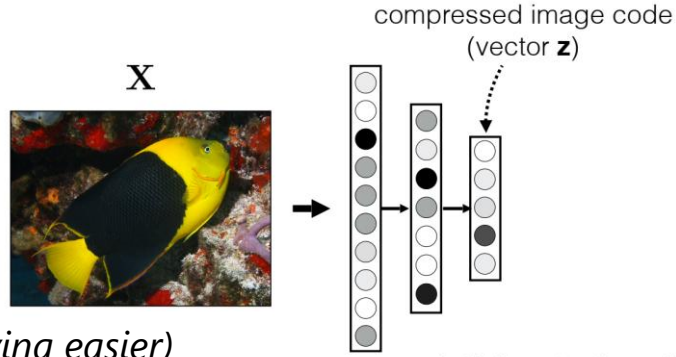
Handling complex data structures

Two major paradigms

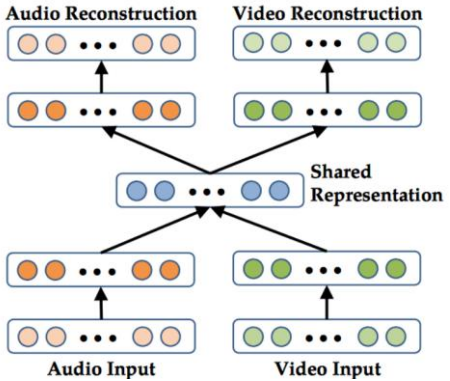
- dedicated distances for series/image/text/events (see previous class!)
- feature extraction to map complex data structures into tabular data
  - *manual extraction* often incomplete and error-prone
  - *automated extraction* susceptible to diverse idiosyncrasies
    - examples: statistics on a sliding window for *time series*, structure–texture–color cues for *images*, term frequencies for *text*, geometric features for *trajectories*…
  - **problem**: *feature extraction* and subsequent *engineering* entail 90% of overall ML effort!
  - **solution**: **learn representations**
    - "*A good representation is one that makes subsequent learning task easier*" Goodfellow et al. 2016 (including clustering!)

Numeric encodings of complex data

- A good **representation** is:
  - compact (*minimal*)
  - explanatory (*sufficient*)
  - disentangled (*independent factors*)
  - informative *(make subsequent problem solving easier)*

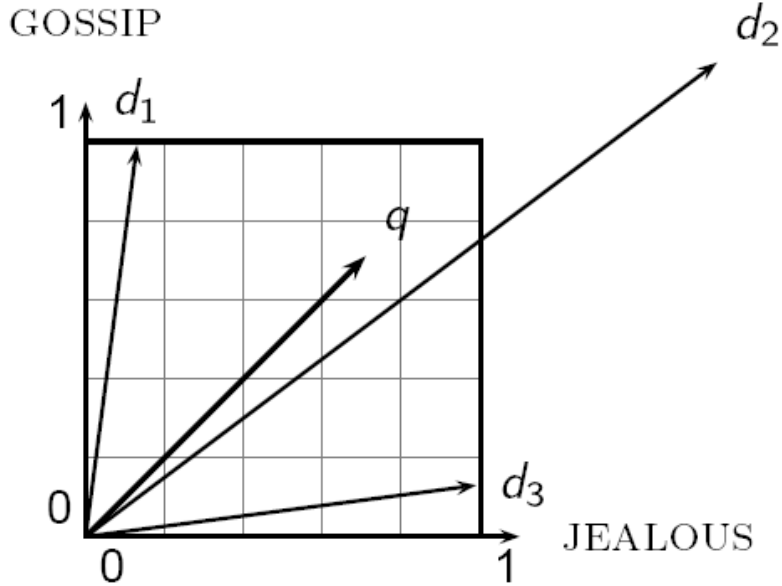


- How to learn numeric representations from complex data?
  - **neural networks** ⟹ check our class on ***representation learning***!
- Implications?
  - classic distances can be applied
  - yet given the above properties: **cosine distance** recommended



Distances *vs* correlation

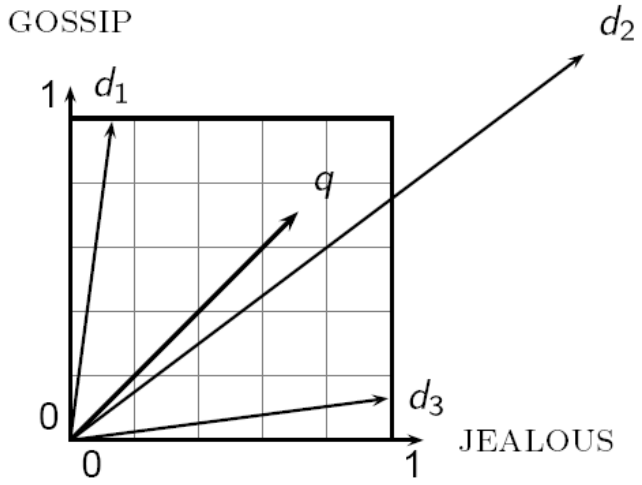
- Consider the scenario where you have three documents (observations): d_1, d_2 and d_3
  - each document is characterized by the frequency of terms in the text (features)
- Now consider we enter a query q to retrieve documents similar to q
- Euclidean distance is… a bad idea as it is large for vectors of different lengths
  - the Euclidean distance between q and d_2 is large even though the distribution of terms in the query q and d_2 are very similar

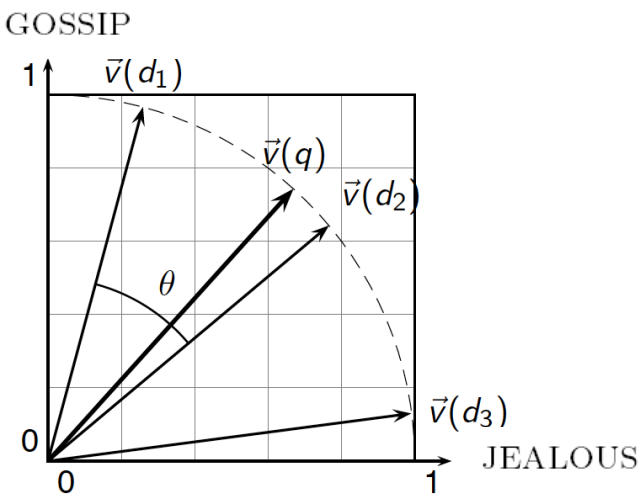


Vector norm

The normalization of an observation, should not be mislead with *variable normalization*

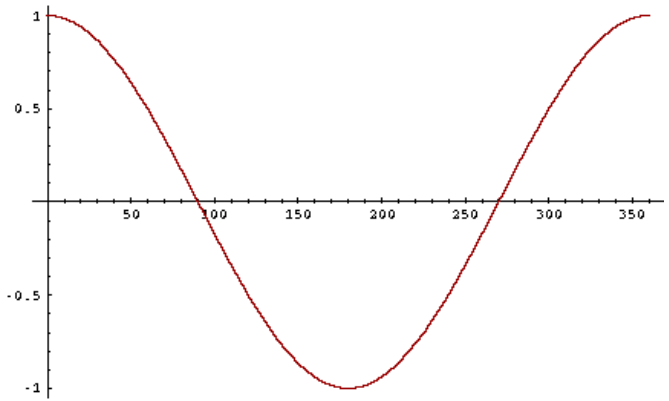
- variable normalization guarantees that the measurements of a given variable yield statistical properties of interest, e.g. [0,1] or N(0,1)





From angles to cosines

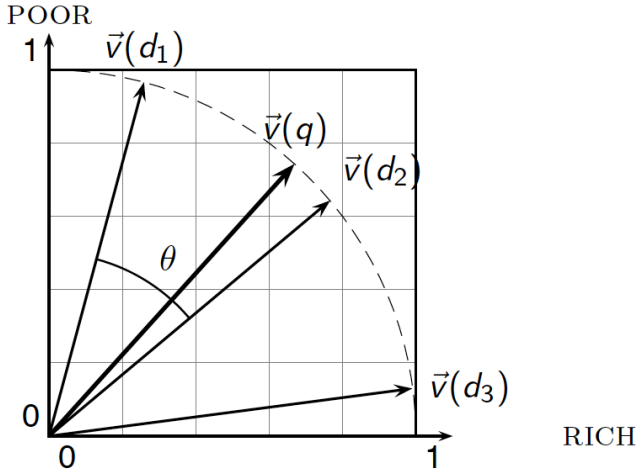
- the following two notions are equivalent:
  - *distance* of the angle between two vectors
  - *similarity* of the cosine between two vectors
- cosine is a monotonically decreasing function for the interval [0º, 180º]



Cosine similarity

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\|\|\mathbf{x}_2\|} = \frac{\sum_i x_{1i}\, x_{2i}}{\sqrt{\sum_i {x_{1i}}^2}\sqrt{\sum_i {x_{2i}}^2}}$$

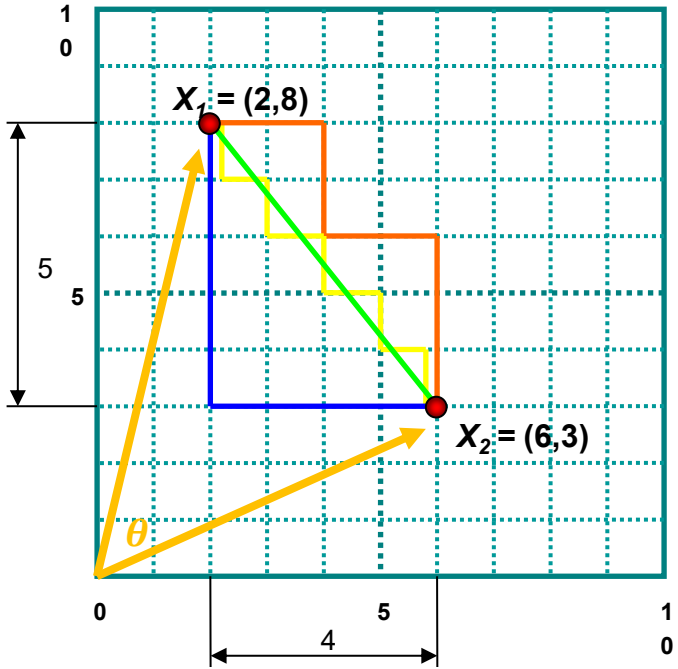
- x_{1i} is the feature from variable y_i in observation $\mathbf{x}_1$
- $\|\mathbf{x}_1\|$ and $\|\mathbf{x}_2\|$ are 2-norm lengths of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$
- $\cos(\mathbf{x}_1, \mathbf{x}_2)$ is the (cosine) similarity of $\mathbf{x}_1$ and $\mathbf{x}_2$



For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2 = \sum_{i=1}^{m} x_{1i}\, x_{2i}$$

Cosine similarity



Euclidean distance

$$l_2(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{|2-6|^2 + |8-3|^2} = \sqrt{41}$$

Manhattan distance

$$l_1(\mathbf{x}_1, \mathbf{x}_2) = |2-6| + |8-3| = 9$$

Cosine similarity

$$\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} = \frac{2\times6+8\times3}{\sqrt{2^2+8^2}\,\sqrt{6^2+3^2}} = 0.65$$

Distances in clustering

- Distances (or similarities) applied between:
  - two observations $d(\mathbf{x}_i, \mathbf{x}_j)$
  - one observation and a cluster $d(\mathbf{x}_i, \mathbf{c}_j)$
  - two clusters $d(\mathbf{c}_i, \mathbf{c}_j)$
- **Cluster**: $\mathbf{c}_j = \{\mathbf{x} \mid d(\mathbf{x}, \mathbf{c}_j) = \min_i d(\mathbf{x}, \mathbf{c}_i)\}$
- **Centroid** of a cluster as its mass center: $\bar{\mathbf{c}}_j$ (e.g., mean/mode value per real/categorical variable)
- Squared **error** of clustering solution: $E = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathbf{c}_k} d(\mathbf{x}, \bar{\mathbf{c}}_k)^2$

Outline

- Distances: advanced notes
- **Clustering approaches**
  - ***k*-means**
  - **model-based (EM)**
  - **deep learning**
- Evaluation recap
- Advanced aspects

Approaches

Partitioning ⇐

- Create partitions and iteratively update them (e.g. *k*-means, *k*-modes, *k*-medoids)

Hierarchical

- Create hierarchical decomposition of data points

Density-based

- Group points based on connectivity and density functions

Model-based ⇐

- Data are seen as a mixture of distributions (e.g. EM)

Deep learning ⇐

- End-to-end neural networks for high-efficacy clustering

Partitioning algorithms

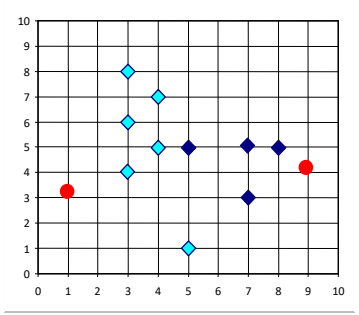
Given *k* clusters:

1. partition observations into *k* non-empty subsets
2. compute the centroid c_j of each subset
   - centroid is the center of mass: e.g. mean or median centers
3. reassign each observation to the cluster with the nearest centroid
4. goto *step 2, stop* when:
   i) assignment does not change, *ii*) $|E^{new} - E^{old}| < \varepsilon$, or *iii*) max iterations is reached

Variants

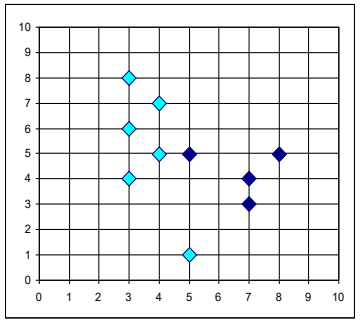
- centroid calculus
- selection of the initial seeds
- adjustments for **batches** of observations (instead of all) for very large datasets

k-means

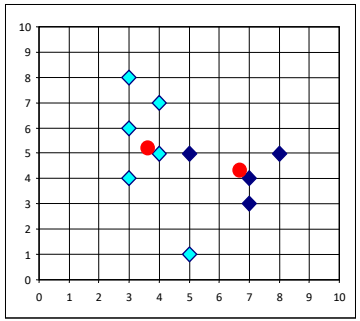


k=2

Arbitrarily choose k object as initial cluster center

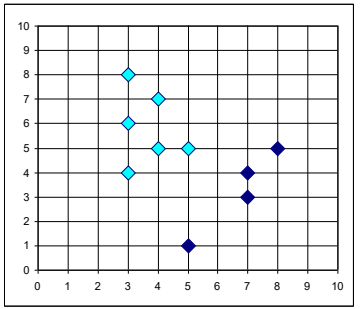


Assign each objects to most similar center

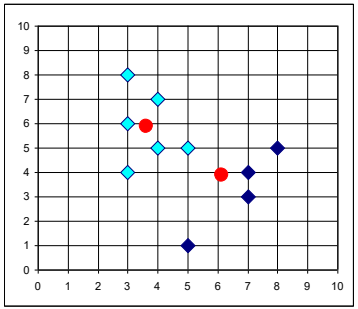


Update the cluster means

reassign

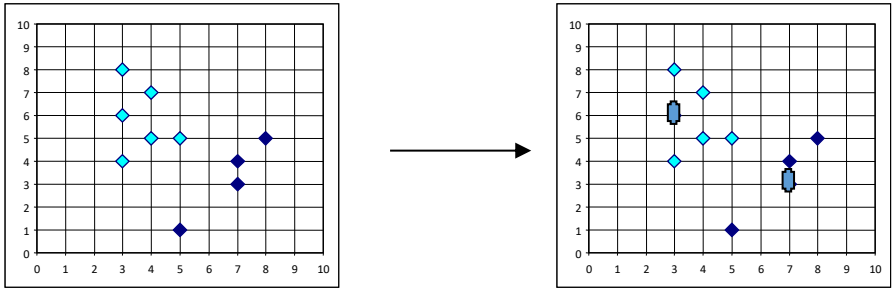


Update the cluster means



k-medoids

- **numeric** data: centroid as the mean or median
- **categoric** data: centroid as the mode – *k*-modes [Huang'98]
  - frequency-based procedure to update modes of clusters
- **mixed** data: centroid combining mean and modes (***k*-prototype**)
- ***k*-medoids**: the **most centrally located observation** in a cluster is the centroid
  - observation with minimum average distance to all observations in the cluster
- What is the algorithm most robust to outliers: *k*-means or *k*-medoids?

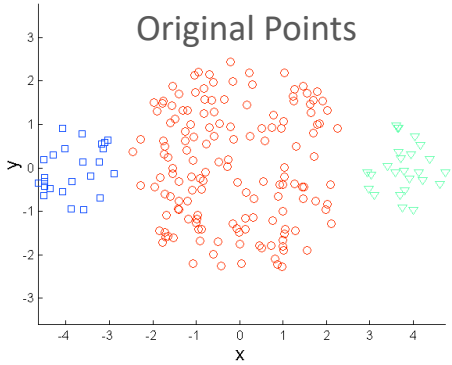


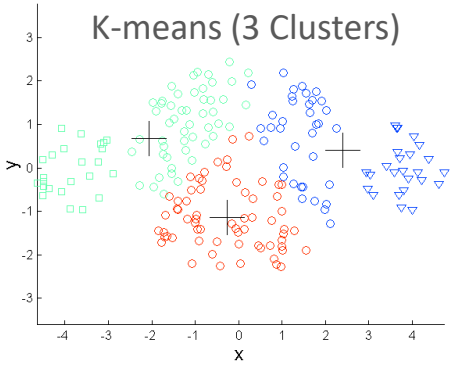
k-means: challenges

- **Efficiency**: $O(tkn)$ n=#observation, k=#clusters, t=#iterations, usually $k, t \ll n$
- **Problems**
  - dependent on initialization
  - sensitive to outliers
  - sensitive noisy data
    - noise can substantially distort centroids
  - not suitable to discover clusters with *non-convex shapes*
    - deal only with clusters with spherical symmetrical point distribution
  - need to specify k, the *number* of clusters, in advance
  - convergence?

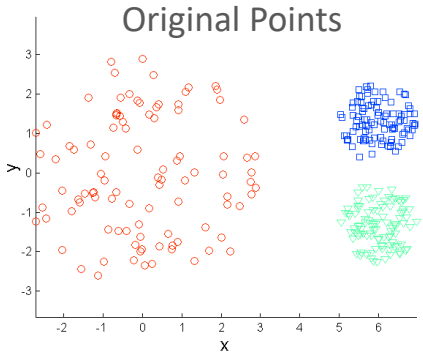
Limitations of k-means

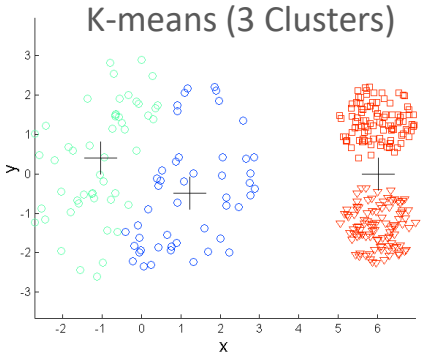
Different sizes





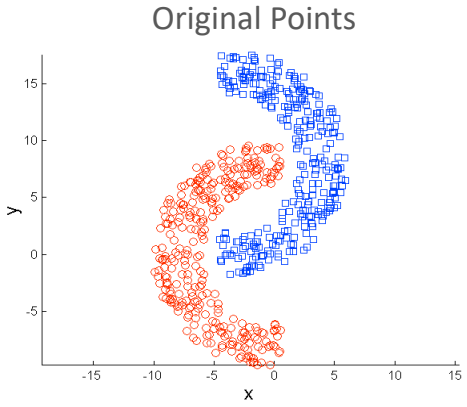
Different densities

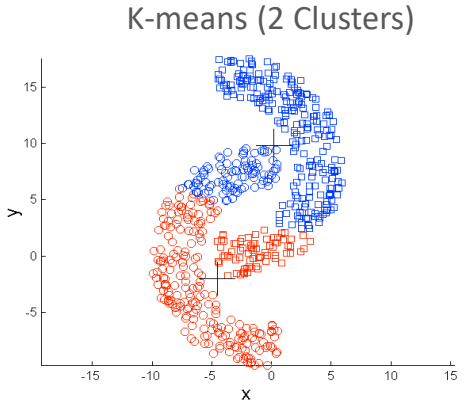




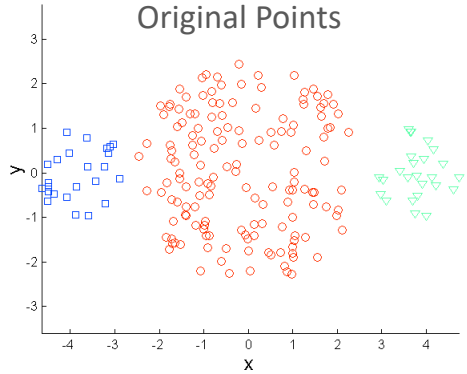
Limitations of k-means

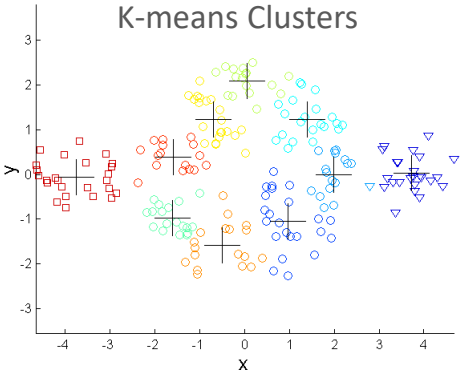
Non-globular shapes



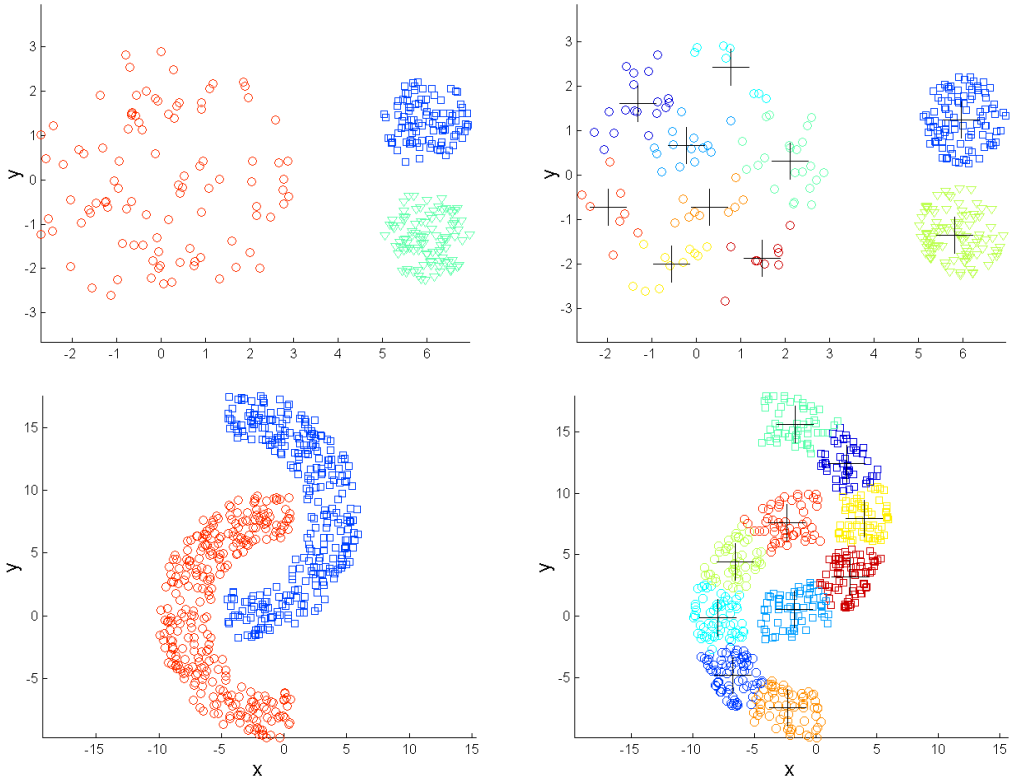


Fixed number of clusters



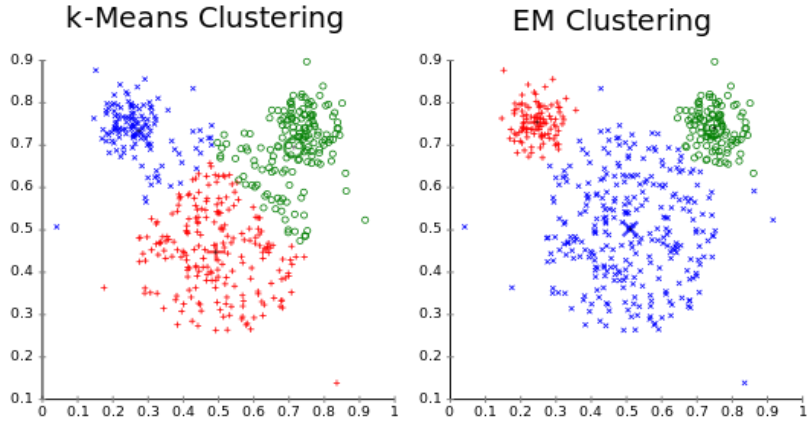


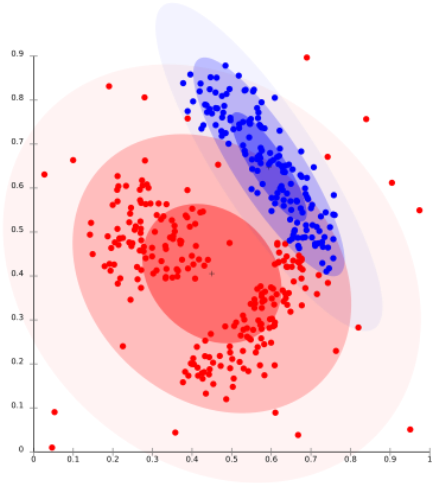
Overcoming k-means limitations



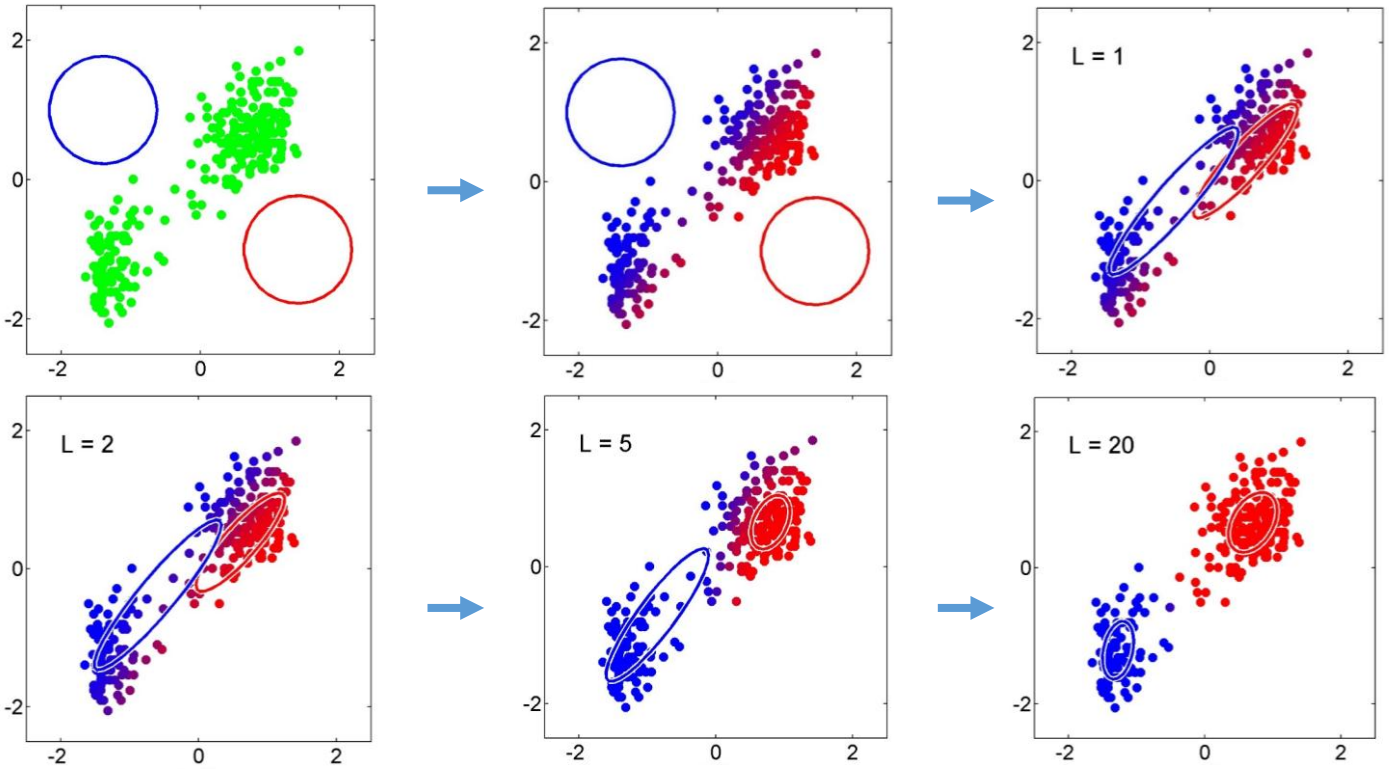
Model-based *vs* partitioning approaches

- *k*-means algorithm performs a **hard assignment** of data points to clusters, in which each data point is associated uniquely with one cluster
  - in *k*-means the shape of the cluster is described by Euclidean distance function
- Model-based clustering makes a **soft assignment** based on the posterior probabilities
  - Expectation Maximization (EM) is the paradigmatic algorithm



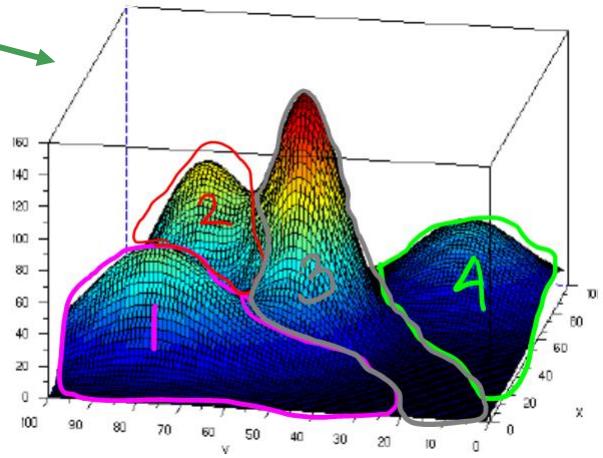


EM: example



EM: principles

- Fix the number of clusters, assume each cluster follows a specific distribution
  - multivariate Gaussian mixture for numeric variables
  - frequentist view for discrete variables
  - joint probabilities under independence assumption for mixed data



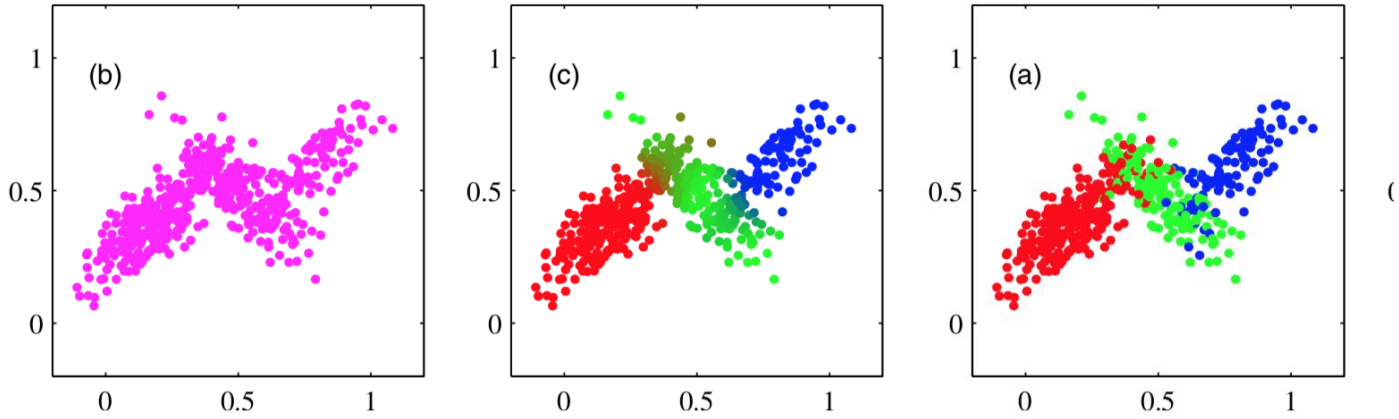
- EM algorithm
  - Iterate between two steps until convergence

    Expectation step: assign observations to clusters

    Maximation step: update the model parameters (adjust distributions)

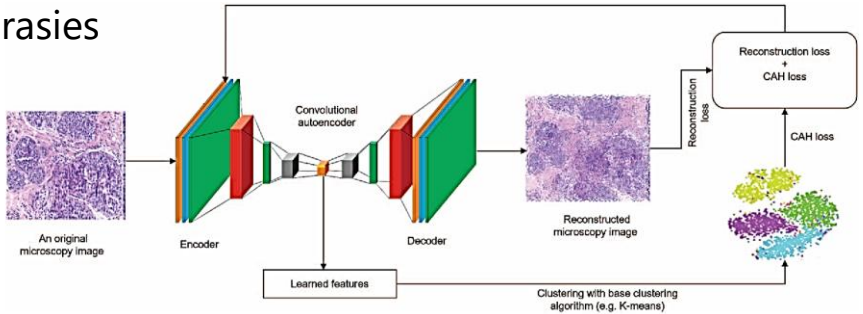
EM and soft clustering

- each observation has a likelihood of being generate by every cluster, $p(\mathbf{x}|c)$
  - can be assigned to the cluster with higher likelihood, $\text{argmax}_c\ p(c|\mathbf{x})$
  - can be assigned to more than one cluster (if both satisfy minimum likelihood) or none
- tackles major problems of density-based approaches (e.g. DBSCAN cannot partition data below)



Deep learning approaches

- Check our presentation on *representation learning* to recover the foundations!
- Two major paradigms
  1. **learn good representations** from data (whether simple or complex data structures) using deep learning *followed by* **classic clustering stances** (e.g. *k*-means)
     - addresses data idiosyncrasies (e.g., non-iid variables)



  2. apply emerging **end-to-end deep learning** pipelines for clustering

Outline

- Distances: advanced notes
- Clustering approaches
  - *k*-means
  - model-based (EM)
  - deep learning
- **Evaluation recap**
- Advanced aspects

Recall: clustering evaluation

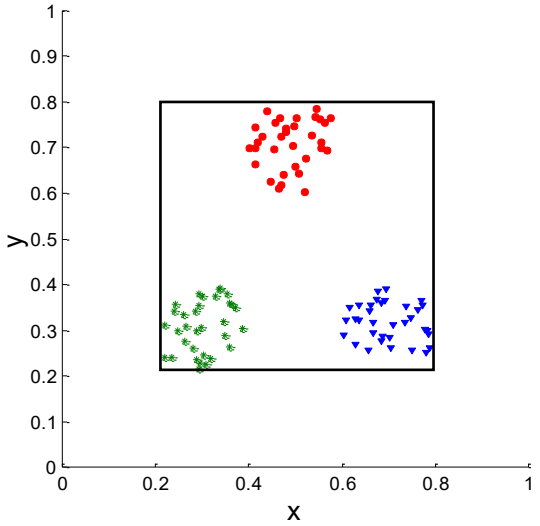
- 3 kinds of measures: external, internal and relative indexes
- **External** (supervised): extent to which cluster labels match true labels
  - requires prior or expert knowledge
- **Internal** (unsupervised): goodness without external information
  - how well they are separated (e.g. silhouette)
  - should be independent from algorithm-specific functions (unbiased)
- **Relative**: compare different cluster analyses (different parameters/algorithms)

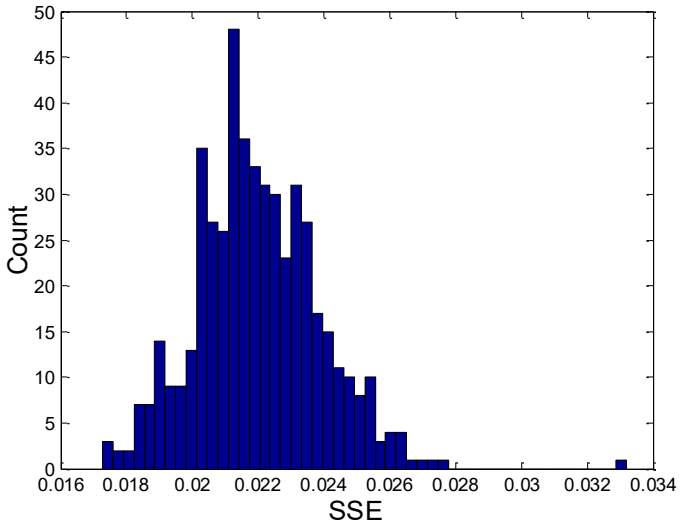
Statistical significance

- **Why?** need to robustly interpret measure
  - if our measure of evaluation has the value, 10, is that good, fair, or poor? ⇒ *p*-value
- **How?** Compare the values of an index that result from a randomized dataset against the index values from the target cluster analysis
  - If the value of the index is unlikely, then results are significant
- To test superiority between two clustering solutions, simpler statistical tests can be considered
- Interesting fact: the more “atypical” a clustering result is, the more likely it represents valid structure in the data

Statistical significance

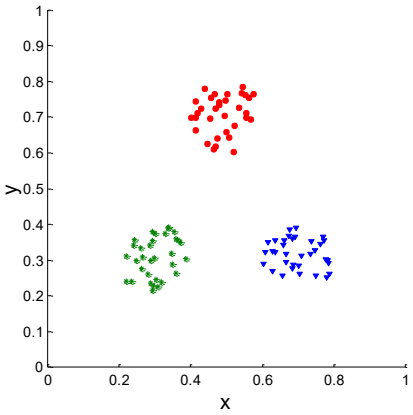
- Compare SSE of 0.005 against SSE of clusters in 500 sets of random data points of size 100 (histogram)



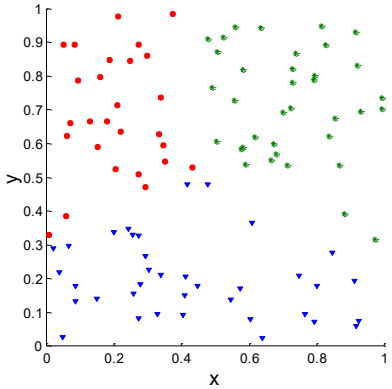


Statistical significance

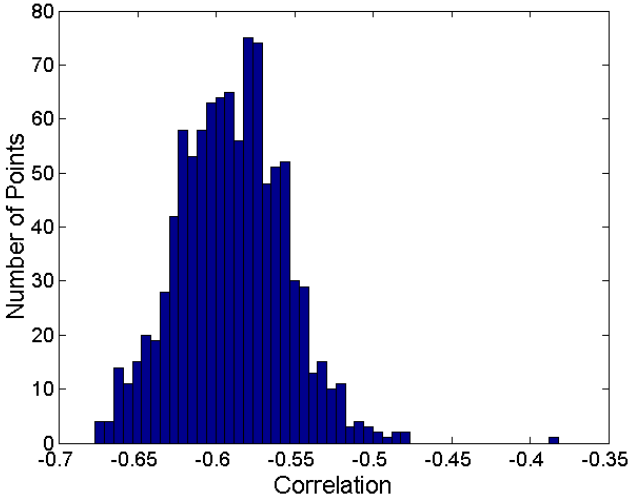
- Are correlations of -0.92 and -0.58 statistically significant?
  Histogram shows correlation on randomized data with K-means



Corr = -0.9235



Corr = -0.5810

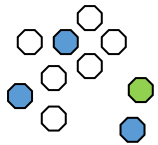


Outline

- Distances: advanced notes
- Clustering approaches
  - *k*-means
  - model-based (EM)
  - deep learning
- Evaluation recap
- **Advanced aspects**

Recall: clustering modes

- memberships: **hard** (observation belongs to a single cluster) *versus* **soft** (each observation has a probability of belonging to each cluster – e.g. fuzzy, model-based clustering)
- supervision
  - **unsupervised** (*default*): cluster observations without knowning their outcome variable
  - **semi-supervised**: cluster observations when labels of some observations may be known *or* pairs of observations are known to belong to the same cluster
  - **"supervised":** cluster observations in the presence of variables of interest (e.g. targets as inputs, class-conditional clustering)
- separation of clusters: **exclusive** versus **non-exclusive** (overlapping clusters)
- **complete** versus **partial** (cluster some objects)
- **uniform** versus **weighted attributes** or **observations**



Visualizing clustering solutions

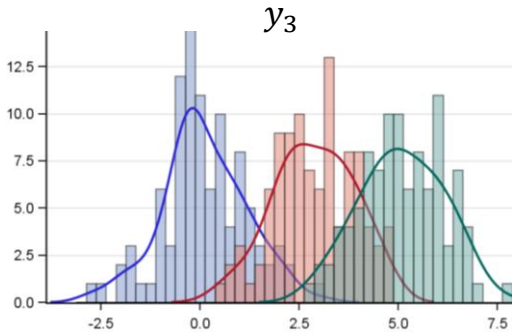
Key principles for knowledge discovery (to be mastered during our course!)

- Compute the variables that are better separated by the clusters
  - add a clusters column to the dataset and run ANOVA (*f-classif* in *sklearn*) to assess the **discriminative power** of each **input variable** (use p-values to rank them by importance)
- Retrieve the **centroids**
- Compute the **cluster-conditional distributions** for the most important features
- Retrieve **observation memberships/distances** to each cluster/centroid
- **Visualize** clustering solutions in a **2D or 3D space**
  - select specific features of interest (importance or domain knowledge)
  - project the original m-dimensional space into a 2D or 3D space using uMAP, PCA, tSNE

Visualizing clustering solutions

Key principles for knowledge discovery (to be mastered during our course!)

- Retrieve the **centroids** and/or **medoids**
- Compute the **cluster-conditional distributions** for the most relevant variables



y_7 y_{12}

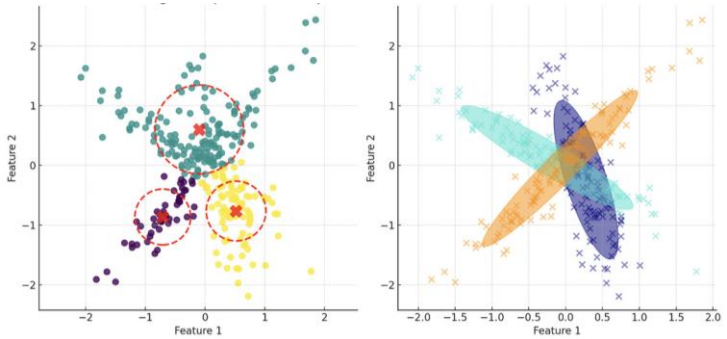
... ...

- Retrieve **observation memberships/distances** to each cluster/centroid

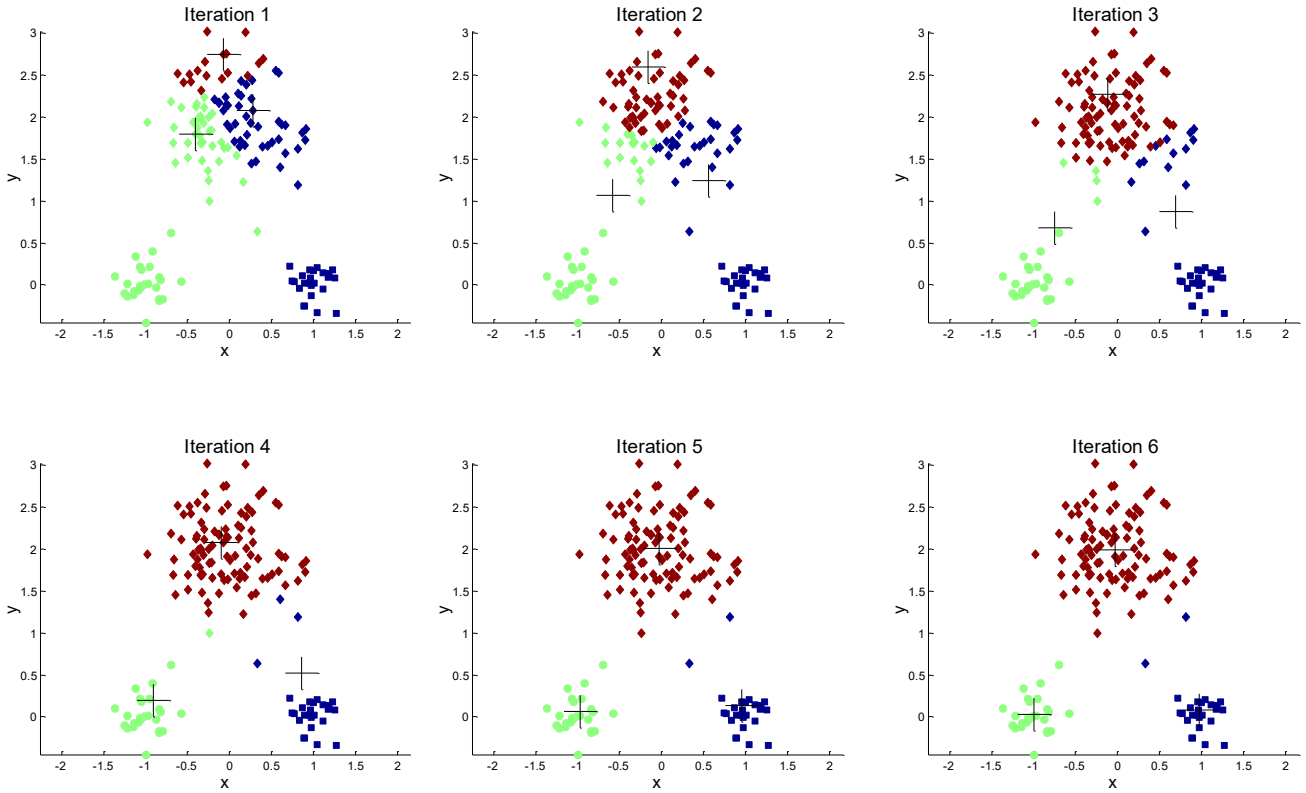
Visualizing clustering solutions

Key principles for knowledge discovery (to be mastered during our course!)

- Most informative variables? The ones that were better separated by the clusters
  - add a column with clusters to the dataset
  - run ANOVA (*f-classif* in *sklearn*) to assess the **discriminative power** of each **input variable**
  - use p-values to rank variables by importance
- **Visualize** clustering solutions in a **2D or 3D space**
  - select specific features of interest (importance or domain knowledge)
  - project the original m-dimensional space into a 2D or 3D space using uMAP, PCA, tSNE



Importance of seeding initial centroids



Importance of seeding initial centroids

- If there are *k* real clusters…

  … chance of selecting one centroid from each cluster is small when *k* is large

  - e.g. if *k* = 10, then probability = 10!/10^10 = 0.00036

- **Difficulty**: sometimes centroids readjust in ‘right’ way, sometimes don’t
- **Solution**
  - multiple runs (helps, but probability is not on your side)
  - hierarchical clustering to determine initial centroids
  - select more than *k* initial centroids and then select among these initial centroids (select most widely separated)
  - postprocessing

Number of clusters?

How many clusters?

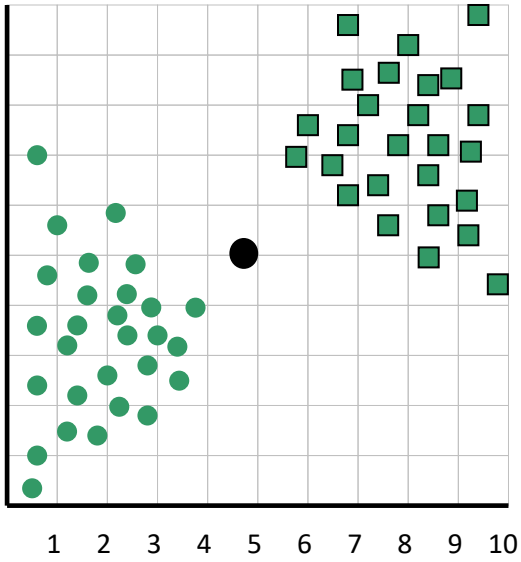
Six Clusters

Two Clusters

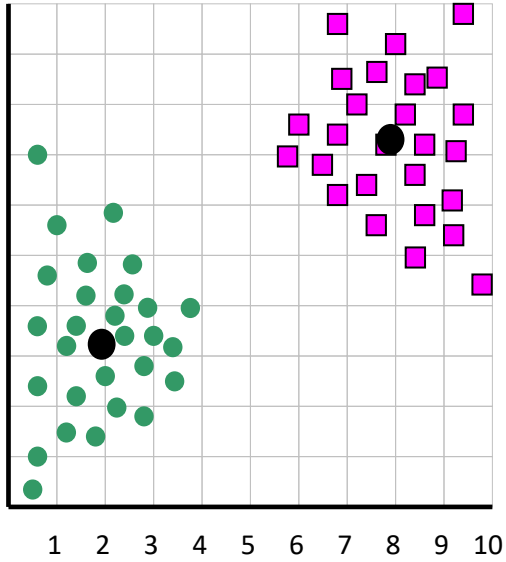
Four Clusters

Number of clusters?

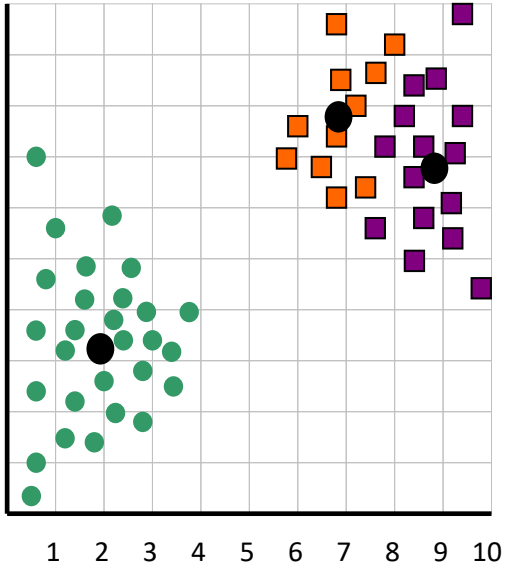
k = 1: cohesion (SSE) is 873.0



k = 2: cohesion (SSE) is 173.1



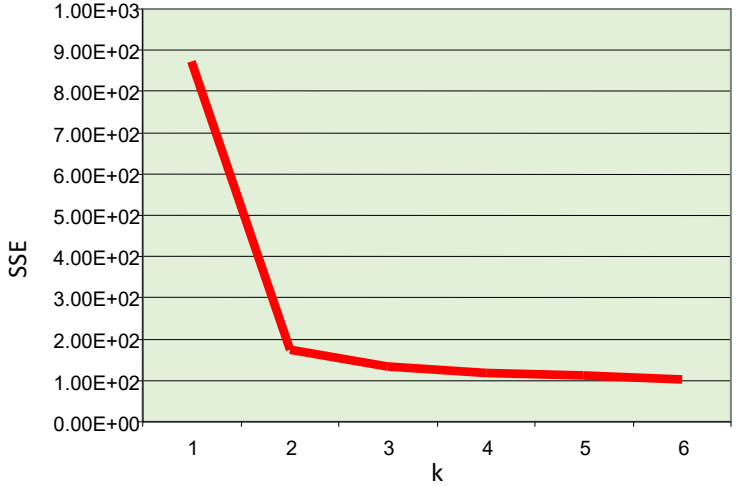
k = 3: cohesion (SSE) is 133.6



Number of clusters?

Knee/elbow method:

- plot the SSE for different *k*
- find the elbow. Example: abrupt changes (*k*=2) are highly suggestive of two clusters in the data



Other methods

- plot metrics less sensitive to the #clusters (e.g., silhouette)
- find the k with maximum value or higher neighborhood average centered on k

Pre- and post-processing

- **Pre-processing**
  - normalize data
  - data reduction and transformation
  - remove outliers
- **Post-processing**
  - eliminate small clusters that may represent outliers
  - split '*loose*' clusters (clusters with relatively high SSE)
  - merge clusters that are '*close*' (clusters with relatively low SSE)
  - these steps can be integrated within the clustering process

Requirements

- clustering **quality**
- ability to deal with different **data types**
  - attribute: numeric, nominal, ordinal and mixtures
  - structure: vectors, events, time series, etc.
  - dimensionality
- ability to deal with **noise**, **missings** and **outliers**
- discovery of clusters with arbitrary **shape**
- **actionability**
- **efficiency**/scalability
- **interpretability** and usability
- **parameter-free** (unfixed number of clusters)
- clustering in the presence of domain knowledge and user-specified constraints

How data characteristics affect clustering?

- sparseness
- variable domain
- data structure
- data distribution
- size and dimensionality
- noise and outliers

Thank you!

Rui Henriques

rmch@tecnico.ulisboa.pt