FH
University of Applied Sciences
TECHNIKUM
WIEN

# Machine Learning

*Lecture 7*

David Meyer
Pascal Plank

# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# Lecture 7 – Unsupervised Learning

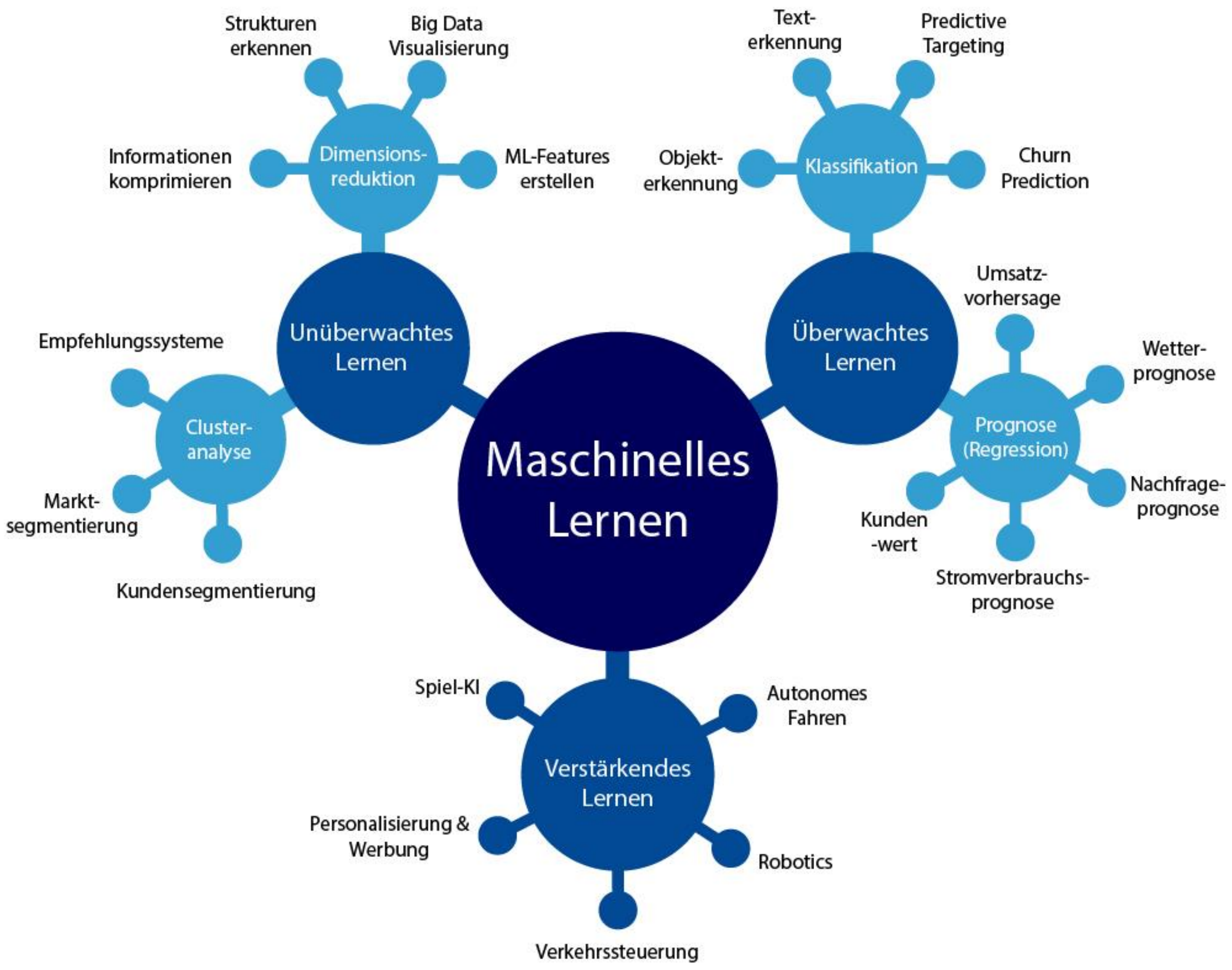
- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# Machine Learning Types

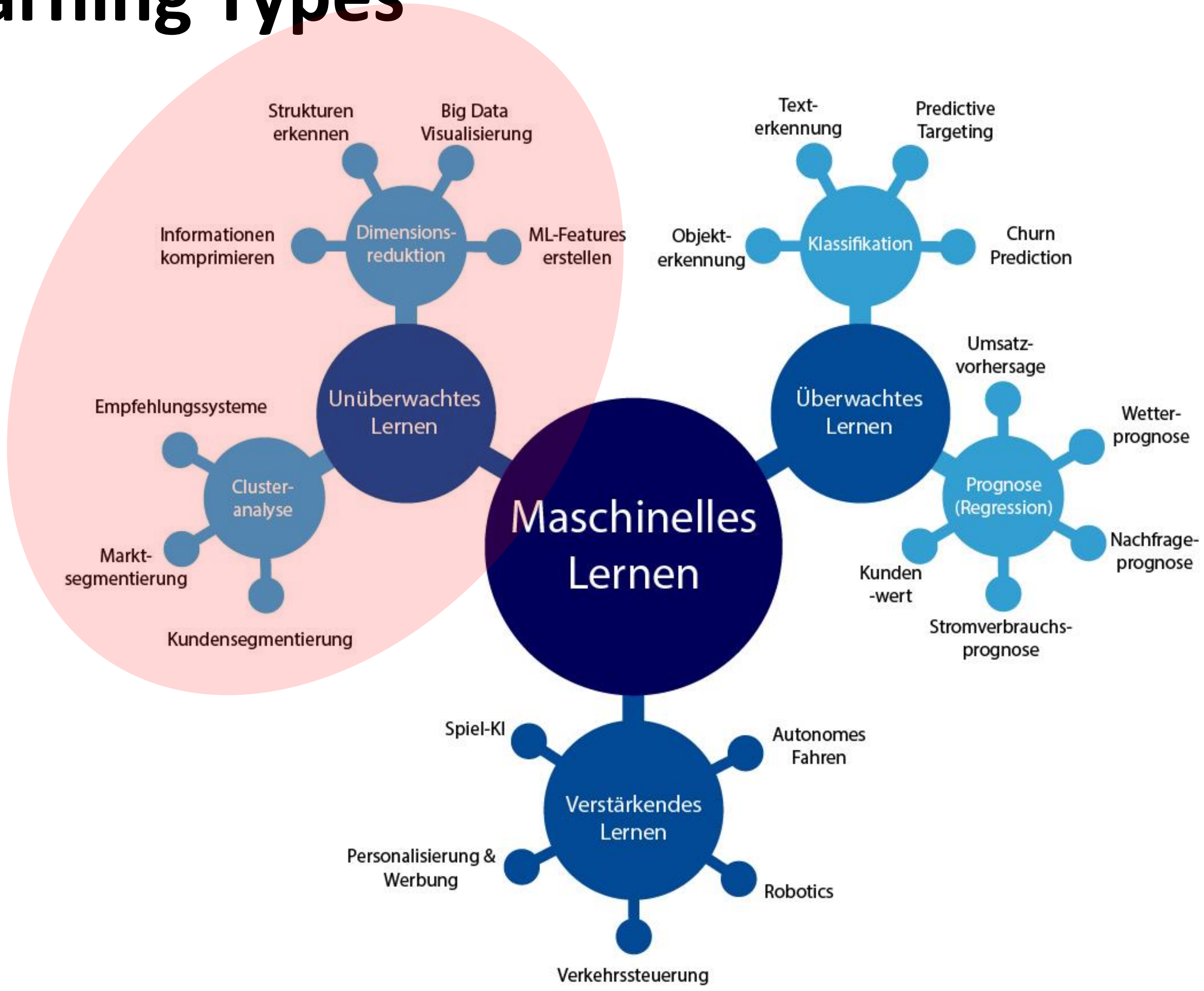
- **Supervised Learning**:
  - labelled data
  - Direct feedback
  - Predict an outcome/future, forecasting
  - E. g. predict customer churn

- **Unsupervised Learning**:
  - No labels
  - No feedback
  - Finding hidden structures
  - E. g. cluster customers

- **Reinforcement Learning**:
  - Decision process
  - Reward system
  - Learn a series of actions
  - E. g. playing Go



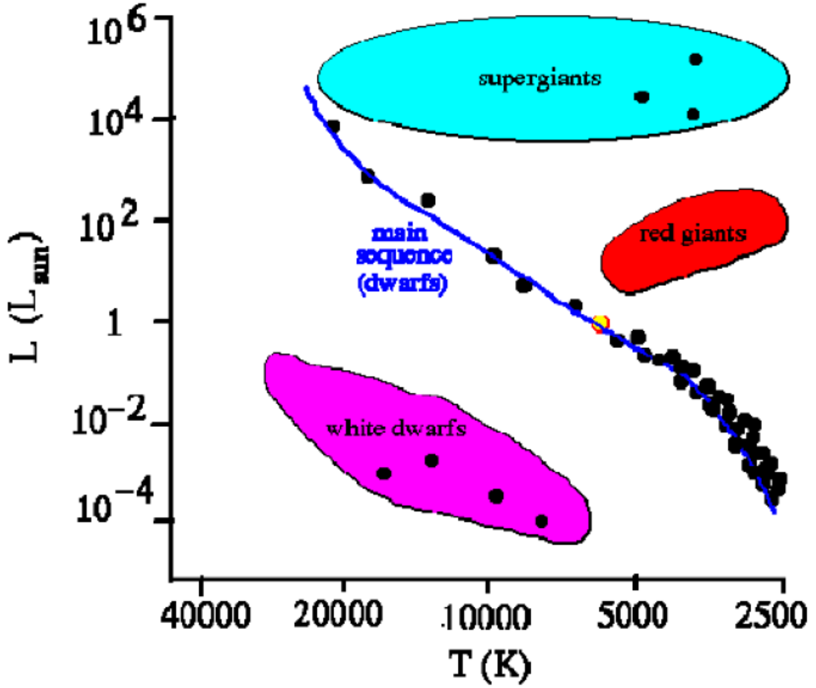
# Machine Learning Types



# Why unsupervised learning?

- Getting enough ground truth data (a.k.a. labelled data) is expensive
- Automatic collection would be nice
- Features can update and the labels should update too

  → e. g. color of fruit over a season; age/growth rate/weight of a person etc.




- Use cases are quite diverse, some examples:
  - Data analysis in science (e. g. grouping of genes, proteins, astronomical objects)
  - Marketing (e. g. product analysis, shopping baset analysis, grouping of customers)
  - Databases (e. g. finding similar documents)

# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# Clustering

- Unsupervised learning method to cluster data into groups
- **Goals**:
  - Clusters should be homogeneous (samples in a single cluster are similar)
  - Clusters themselves should be very different
- **Definition:**

  m-clustering of X = {$x_1$, $x_2$, …, $x_N$} is a subdivision of X into m cluster $C_1$,…, $C_m$ such that:
  - $C_i \neq \emptyset, i = 1, \dots m$
  - $\bigcup_{i=1}^{m} C_i = X$
  - $C_i \cap C_j = \emptyset, i \neq j; i, j = 1, \dots m$

  The vectors $x_i$ of cluster $C_i$ are more similar to each other than they are to the vectors of the other clusters. This is called **„hard“ or „crisp“ clustering**.

# Clustering

- **Definition (Fuzzy Clustering):**

  Fuzzy-clustering of X = {$x_1$, $x_2$, …, $x_N$} into m cluster $C_1$,…, $C_m$ is defined by m membership functions $u_j$ with:

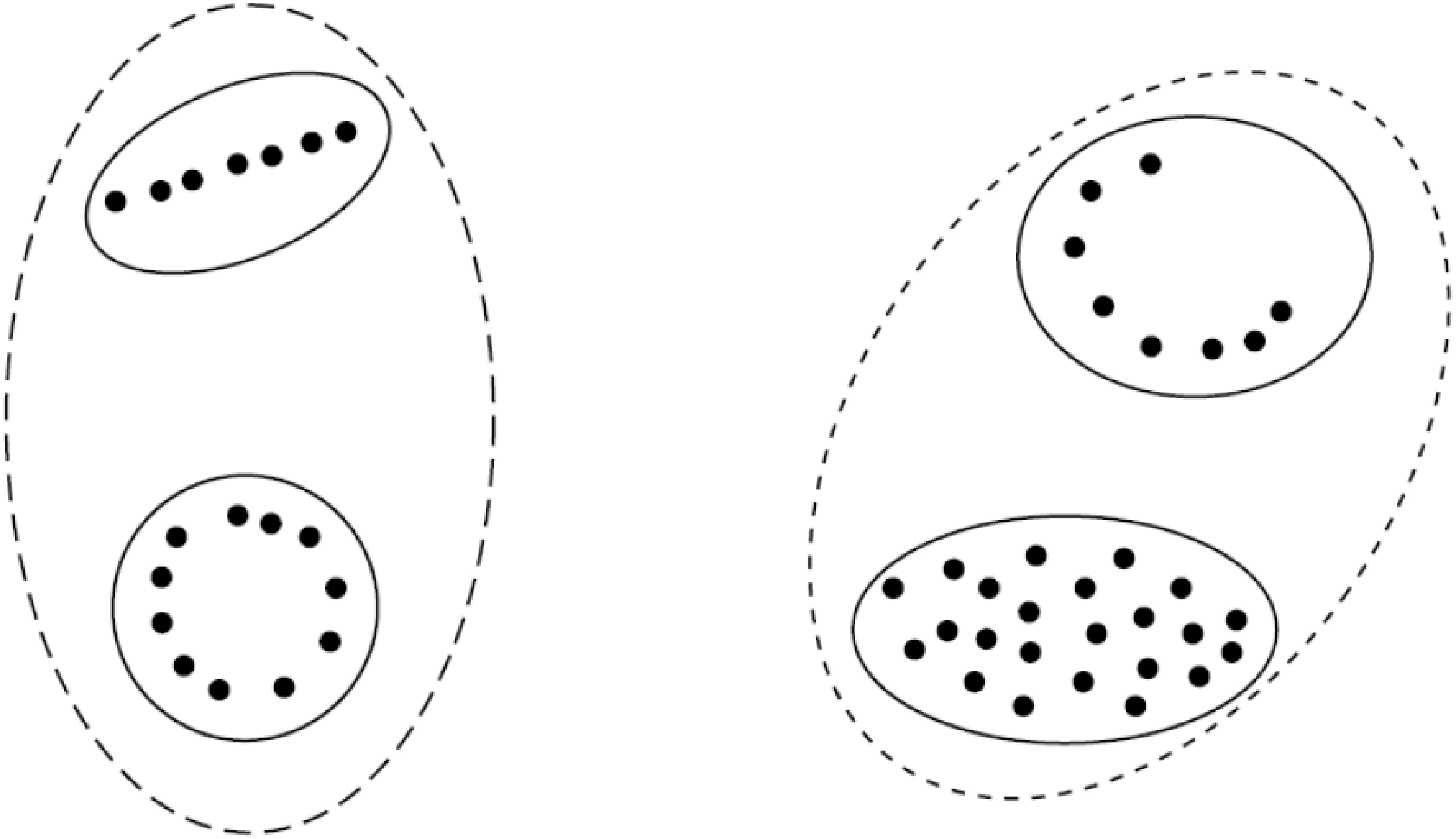
  - $u_i: X \rightarrow [0,1], j = 1, \dots, m \ and$
  - $\sum_{j=1}^{m} u_i(x_i) = 1, i = 1, \dots, N$

  Each vector $x_i$ is part of several clusters at the same time, the value of this membership is quantified by $u_j$. Values near 1 indicate a high degree of membership to a cluster.

# Clustering

- There are many possible solutions depending on the scale

[Quelle: Theodoris et al., 2009]



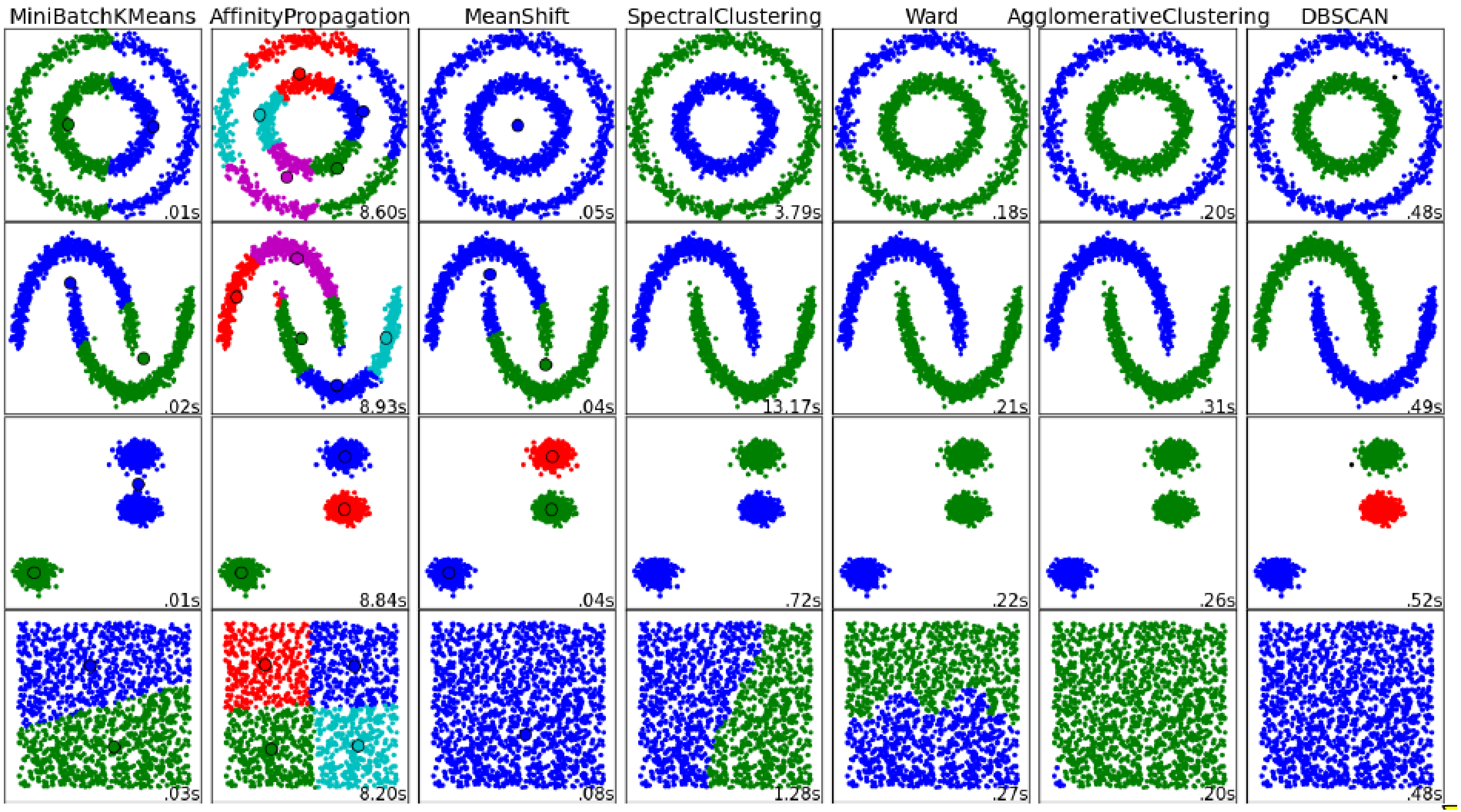
# Clustering

- For clustering to work, you‘ll need:
  - Features and some pre-processing
  - Distance metric
  - Cluster criterion (grouping criterion)
  - Clustering algorithm
  - Validation and interpretation of results

  → Depending on the selection of the above points, you‘ll get very different clusterings

# Clustering

[Quelle: http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html]



# Clustering – Similarity metrics

- **Euclidean norm**
- **dot-product** (scalar product) of two vectors x, y:

$$s_{scalar}(x, y) = x^T y = \sum_{i=1}^{l} x_i y_i$$

- **Pearson's correlation coefficient:**

$$s_{pearson}(x, y) = \frac{x_d^T y_d}{\|x_d\| \|y_d\|},$$

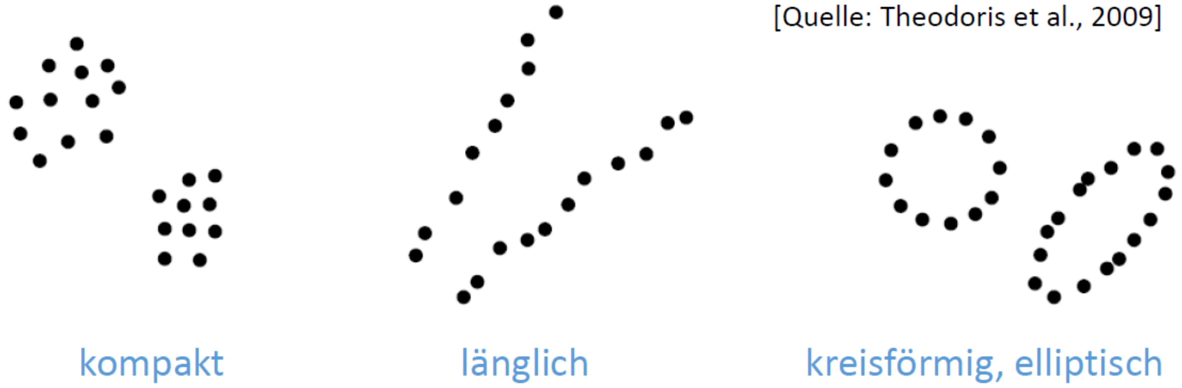
with $x_d [x_1 - \bar{x}, \dots, x_l - \bar{x}]^T \; and \; y_d [y_1 - \bar{y}, \dots, y_l - \bar{y}]^T$

# Clustering – Similarity metrics

- **Distance between a vector and a subset of X**

  → needed for assessing whether a vector should be assigned to a cluster C
- Several options exist:
  - Minimal dissimilarity: d(x,C) = min(d(x,y)) for all y in C
  - Maximal similarity: s(x,C) = max(s(x,y)) for all y in C
  - Average (dis)similarity or distance to a **centroid** average cluster coordinates)
- Depending on the cluster type, a different distance calculation might be reasonable

[Quelle: Theodoris et al., 2009]



# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# K-means clustering

- K-means is a **centroid-based clustering method. It is partitional**
- **Goal**: find the k cluster centers and assign the objects to the nearest cluster center (centroids), such that the squared distances from the cluster are minimized
- **NP-hard** problem → we search for an approximate solution
- Number of clusters k needs to be specified (hyperparameter)
- Conceptually close the to k-nearest neighbor classification algorithm
- Some variations:
  - k-median (more robust against outliers)
  - k-medoids (centroids need to be a data sample)
  - k-means++ (choosing initial centers less randomly)
  - fuzzy c-means (allowing fuzzy cluster assignment)

# K-means clustering

- **The algorithm consists of the following steps:**
  1. Choose number of clusters k and initial centroids (usually randomly)
  2. Assignment step: assign samples to the cluster with the closest (Euclidean distance) centroid
  3. Update step: Recalculate centroids (means) with the assignments of step 2
  4. Repeat steps 2 and 3 until there is no significant improvement anymore

- The algorithm is not guaranteed to find the global optimum
- It is based on spherical clusters that are separable and approximately the same size so that the mean converges towards the cluster center

# K-means clustering

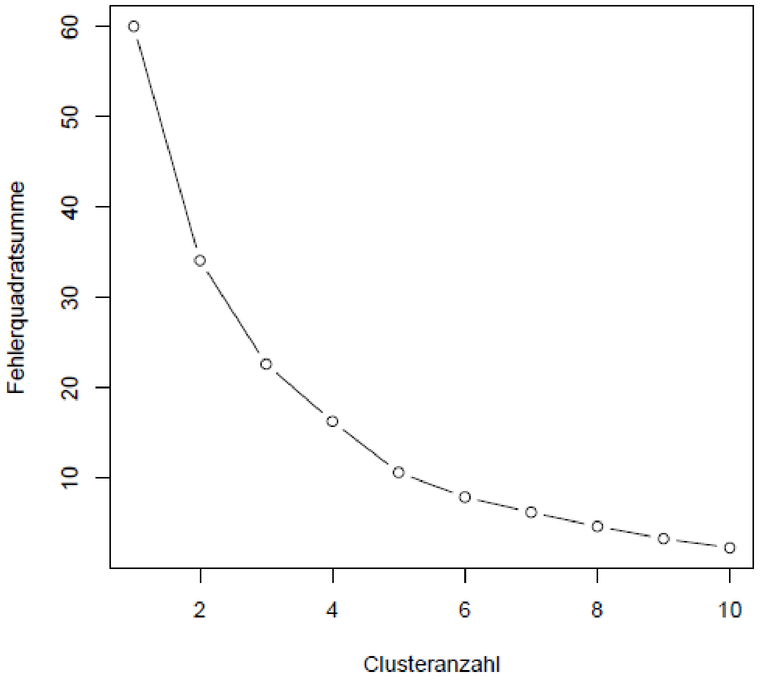
- To assess „significant improvement“ we need a criterion, e. g., sum of squared distances of samples to their centroids
- Since we use distance metrics **standardized variables** are strongly recommened!
- Initial centroids are critical to the overall result

  → repeat algorithm several times with random initial centroids

  → choose the best run
- The resulting clusters can be used as an additional categorical variables for further statistical methods, e. g., clustering customers and running analysis on customer clusters.

# K-means clustering – Choosing k

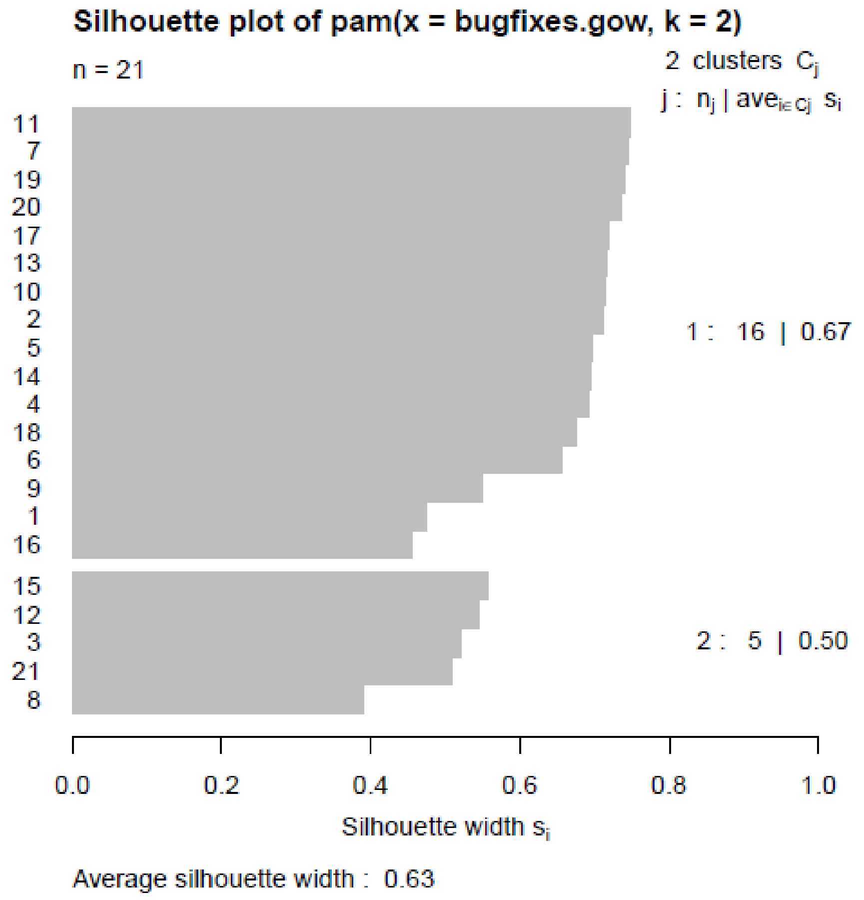
- K is very important to the final result
- To find a good k, run the algorithm several times with different k and plot the errors




- Look out for a plateauing point or a sharp change in error trend that indicates that improvement with higher k is limited after this point.

# K-means clustering – Evaluating results with silhouettes

- How good the samples are clustered can be assessed with a silhouette plots
- Silhouettes can be calculated for the overall solution or individual clusters
- Silhouette values and their approx. meaning:
  - 1.0 – 0.71: well clustered samples, clustering found strong structures
  - 0.7 – 0.51: medium and somewhat acceptable structures were found
  - 0.5 – 0.26: weak structures
  - <0.25: no substantial structures were found, and samples lie between clusters

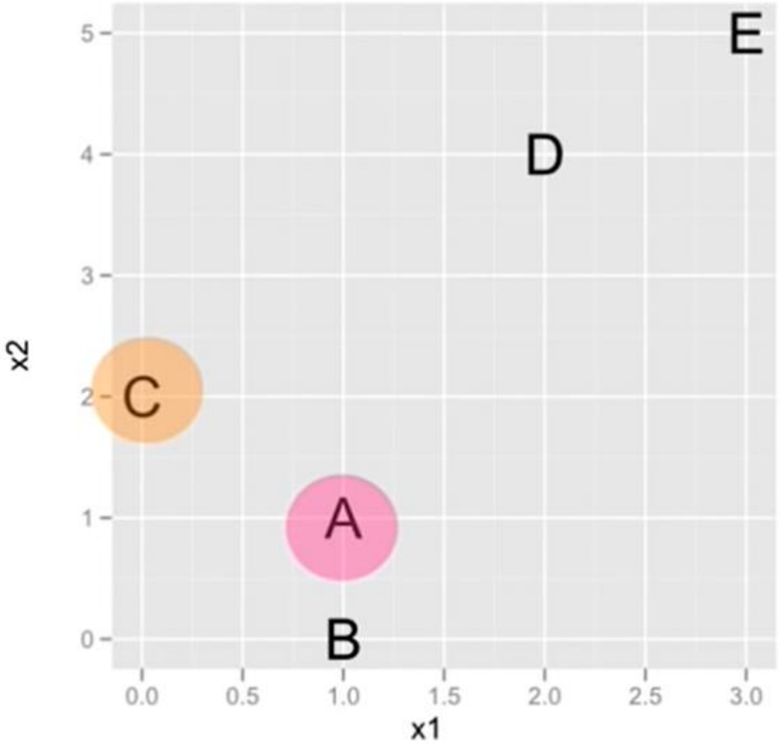



- Example for n=21 samples in 2 clusters:
  - Clusters are medium (0.5) or strong clusters (0.67)
  - Average silhouette width is acceptable (0.63)
  - Individual silhouettes are >0, i. e., no samples are in wrong cluster or close to a border

# Class Assignment

- <u>**Assignment:**</u>
  Use **K=2** with points **A and C as initial cluster means** and perform the **k-means clustering algorithm** for **two iterations** *(Euclidean norm* $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ *)*

| | i | $X_1$ | $X_2$ |
|---|---|---|---|
| $\bar{X}_1^0$ | A | 1 | 1 |
| | B | 1 | 0 |
| $\bar{X}_2^0$ | C | 0 | 2 |
| | D | 2 | 4 |
| | E | 3 | 5 |



# Class Assignment – Solution 1/7

- Step 1 (Iteration 1): Calculate the distances between each of the cluster means and all other points

| | i | $X_1$ | $X_2$ |
|---|---|---|---|
| $\bar{X}_1^0$ | A | 1 | 1 |
| | B | 1 | 0 |
| $\bar{X}_2^0$ | C | 0 | 2 |
| | D | 2 | 4 |
| | E | 3 | 5 |

| i | 1 | 2 |
|---|---|---|
| A | 0 | 1.4 |
| B | 1 | 2.2 |
| C | 1.4 | 0 |
| D | 3.2 | 2.8 |
| E | 4.5 | 4.2 |

# Class Assignment – Solution 2/7

- Step 2 (Iteration 1): Assign each point to the cluster with the lowest distance. Use this new cluster assignment to calculate the new cluster means

| i | 1 | 2 | Cluster |
|---|---|---|---|
| A | 0 | 1.4 | 1 |
| B | 1 | 2.2 | 1 |
| C | 1.4 | 0 | 2 |
| D | 3.2 | 2.8 | 2 |
| E | 4.5 | 4.2 | 2 |

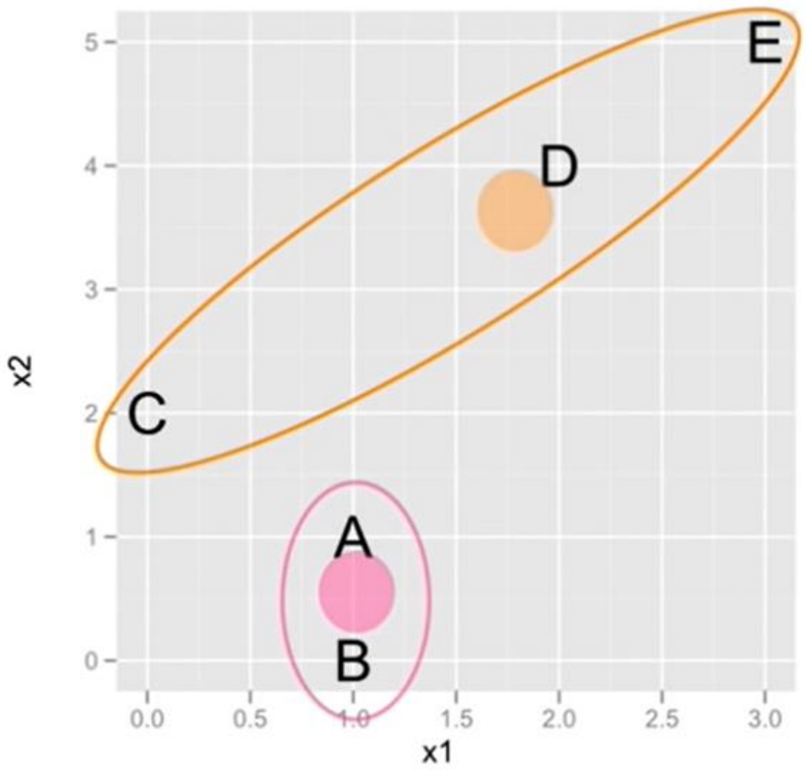
| i | $X_1$ | $X_2$ |
|---|---|---|
| A | 1 | 1 |
| B | 1 | 0 |
| C | 0 | 2 |
| D | 2 | 4 |
| E | 3 | 5 |

$\bar{X}_1^1$

$\bar{X}_2^1$

# Class Assignment – Solution 3/7

- Iteration 1 final: Updated clusters in the graph:




$\bar{X}_1^1 = (1,0.5)$

$\bar{X}_2^1 = (1.7,3.7)$

# Class Assignment – Solution 4/7

- Step 3 (Iteration 2): Recalculate the distances of points to the new cluster means

| i | $X_1$ | $X_2$ |
| --- | --- | --- |
| A | 1 | 1 |
| B | 1 | 0 |
| C | 0 | 2 |
| D | 2 | 4 |
| E | 3 | 5 |

$\bar{X}_1^1 = (1,0.5)$

$\bar{X}_2^1 = (1.7,3.7)$

| i | 1 | 2 |
| --- | --- | --- |
| A | 0.5 | 2.7 |
| B | 0.5 | 3.7 |
| C | 1.8 | 2.4 |
| D | 3.6 | 0.5 |
| E | 4.9 | 1.9 |

# Class Assignment – Solution 5/7

- Step 4 (Iteration 2): Assign each point to the cluster with the lowest distance.

| i | 1 | 2 | Cluster |
|---|---|---|---|
| A | 0.5 | 2.7 | 1 |
| B | 0.5 | 3.7 | 1 |
| C | 1.8 | 2.4 | 1 |
| D | 3.6 | 0.5 | 2 |
| E | 4.9 | 1.9 | 2 |

| i | $X_1$ | $X_2$ |
|---|---|---|
| A | 1 | 1 |
| B | 1 | 0 |
| C | 0 | 2 |
| D | 2 | 4 |
| E | 3 | 5 |

$\bar{X}_1^2$

$\bar{X}_2^2$

# Class Assignment – Solution 6/7

- Step 5 (Iteration 2): Assign each point to the cluster with the lowest distance.

| i | 1 | 2 | Cluster |
|---|---|---|---|
| A | 0.5 | 2.7 | 1 |
| B | 0.5 | 3.7 | 1 |
| C | 1.8 | 2.4 | 1 |
| D | 3.6 | 0.5 | 2 |
| E | 4.9 | 1.9 | 2 |

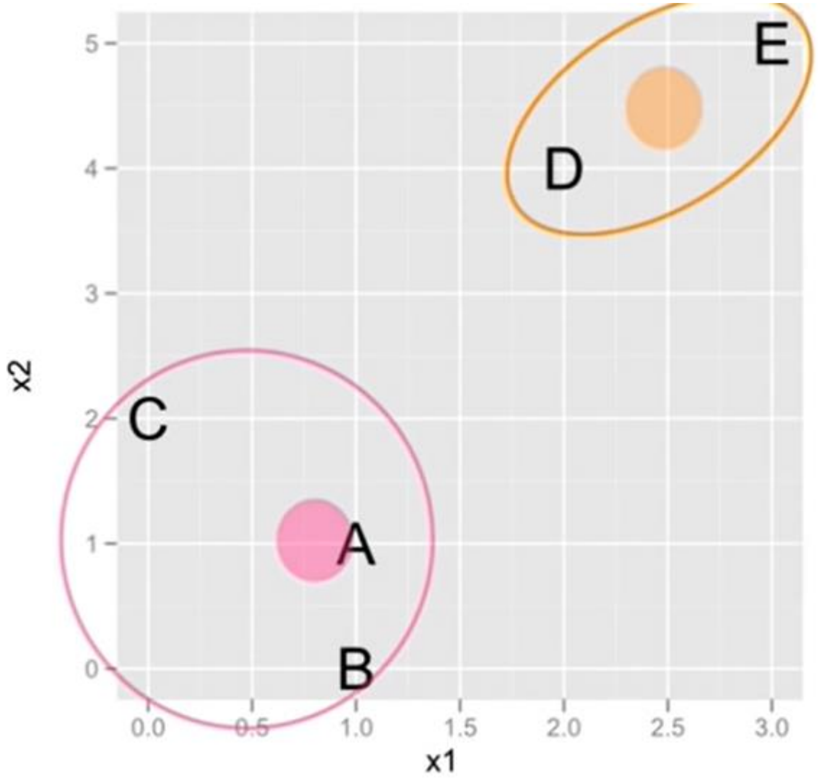
| i | $X_1$ | $X_2$ |
|---|---|---|
| A | 1 | 1 |
| B | 1 | 0 |
| C | 0 | 2 |
| D | 2 | 4 |
| E | 3 | 5 |

$\bar{X}_1^2$

$\bar{X}_2^2$

# Class Assignment – Solution 7/7

- Step 6 (Iteration 2): Recalculate cluster means
  → Algorithm has already converged ☺




$\overline{X}_1^2 = (0.7,1)$

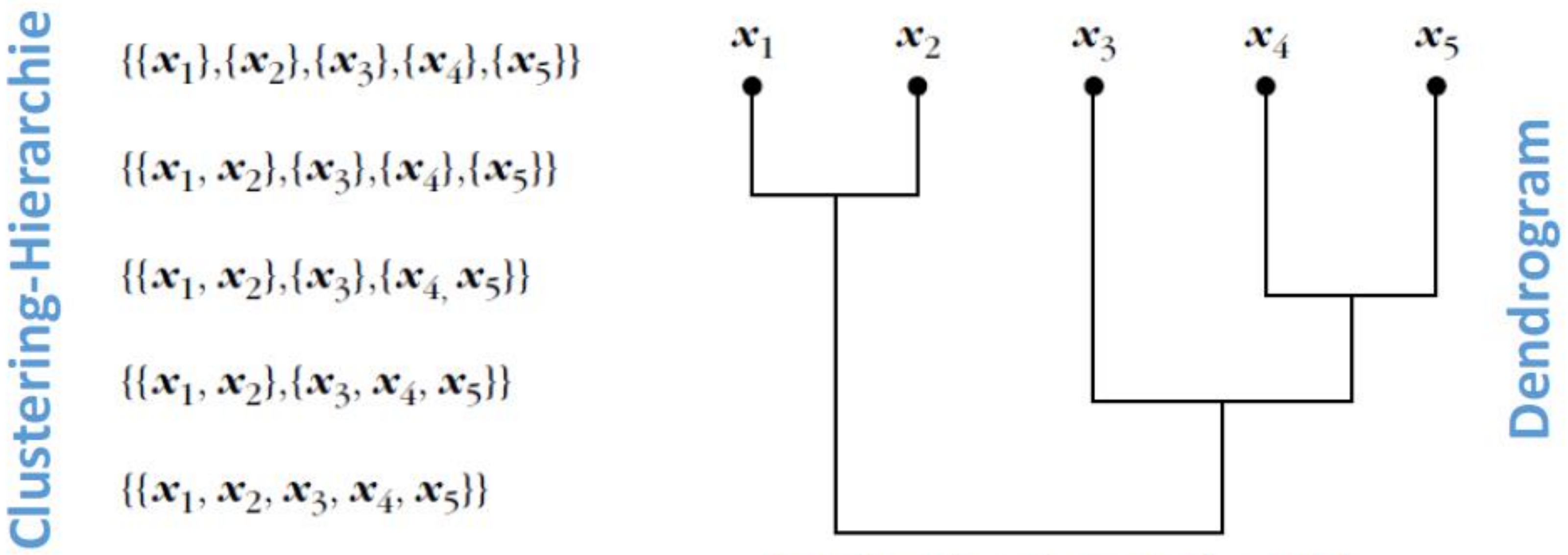
$\overline{X}_2^2 = (2.5,4.5)$

# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# Hierarchical clustering

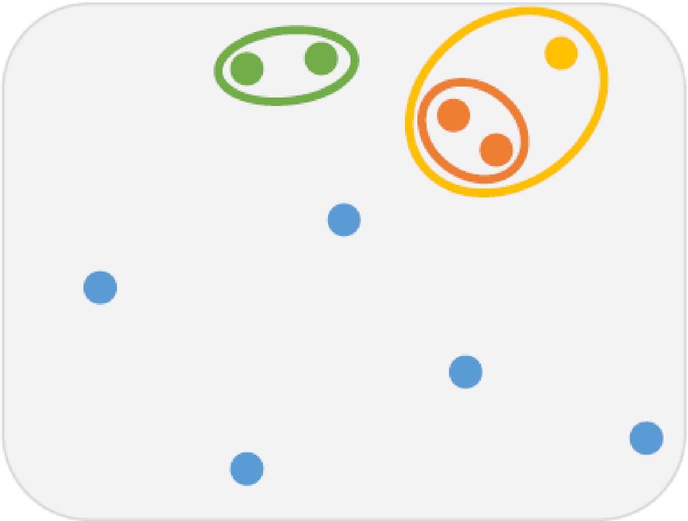
- **Two types:**
  - Agglomerative
  - Divisive
- In addition to homogeneous clusters, hierarchical clustering results in a hierarchical structure of the data (a **dendrogram**)

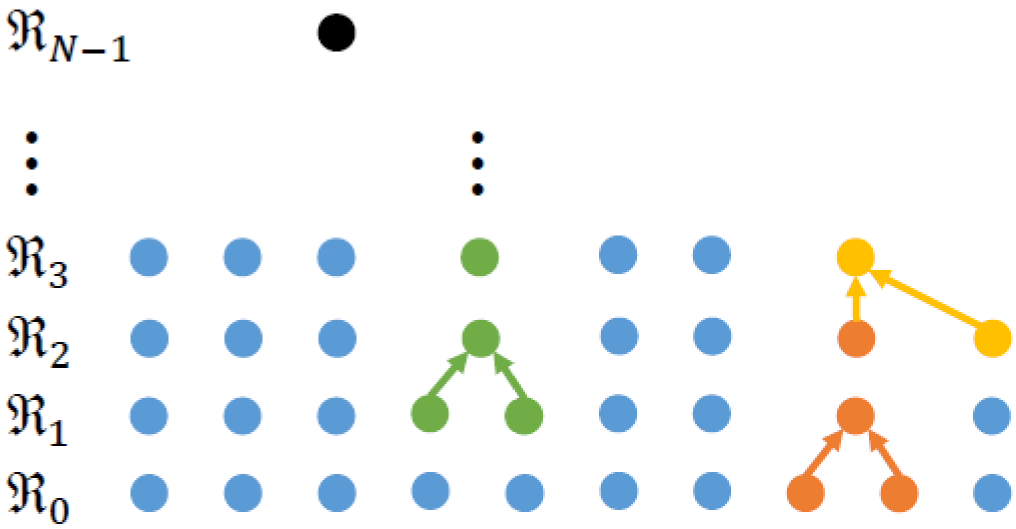



[Quelle: Theodoris et al., 2009]

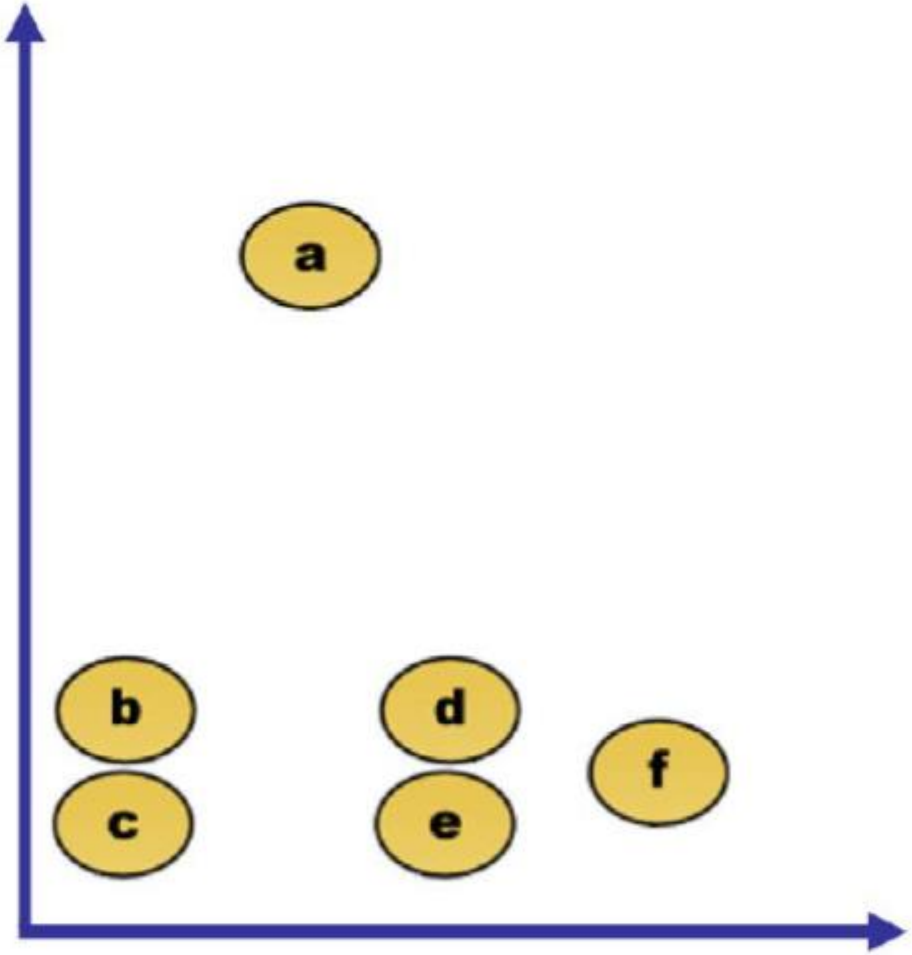
# Hierarchical clustering – agglomerative approach

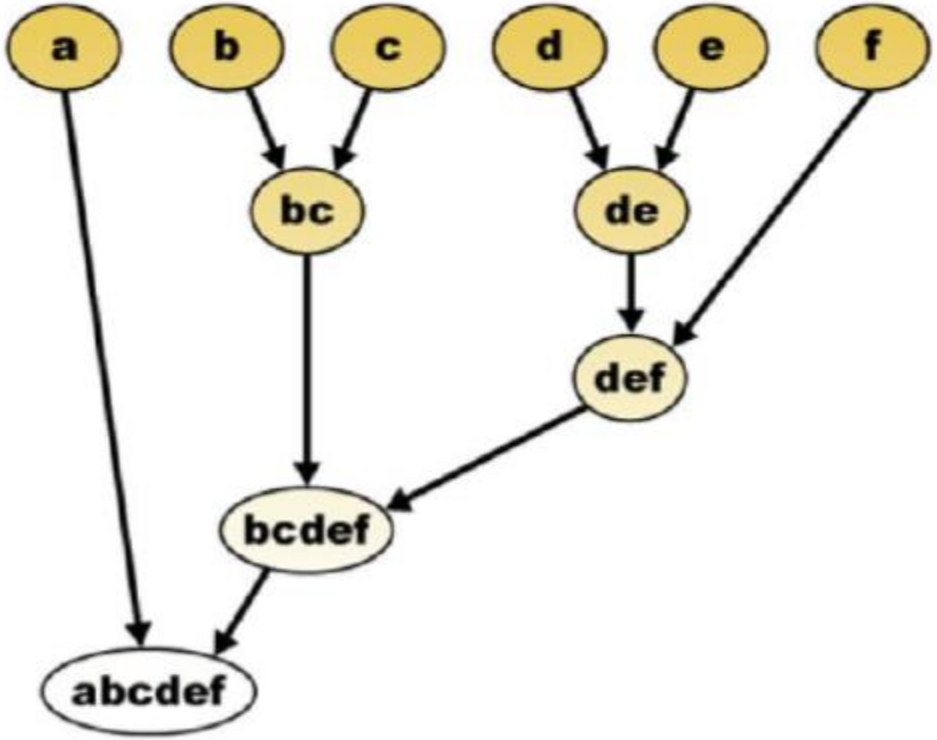
- Start with each sample in a separate cluster and then group them into larger and larger clusters until there is only one cluster left
- Grouping criterion determines how this agglomeration works:
  - Single Linkage Clustering: smallest distance between samples in clusters
  - Complete Linkage Clustering: largest distance between samples in clusters
  - Average Linkage Clustering: mean distance between samples in clusters





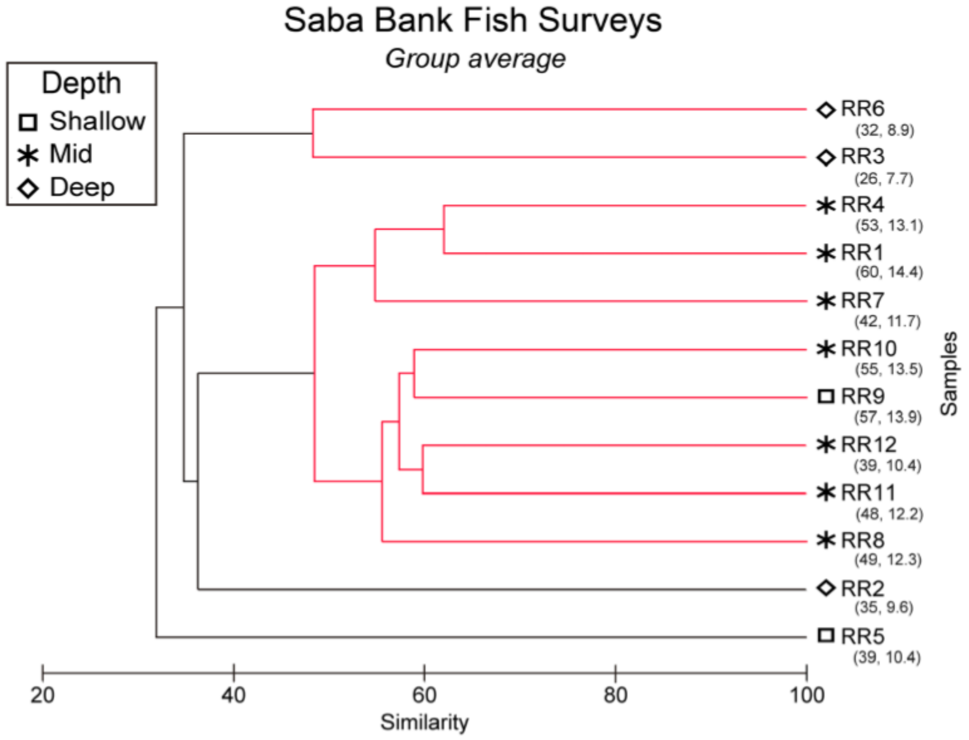
# Hierarchical clustering – agglomerative approach



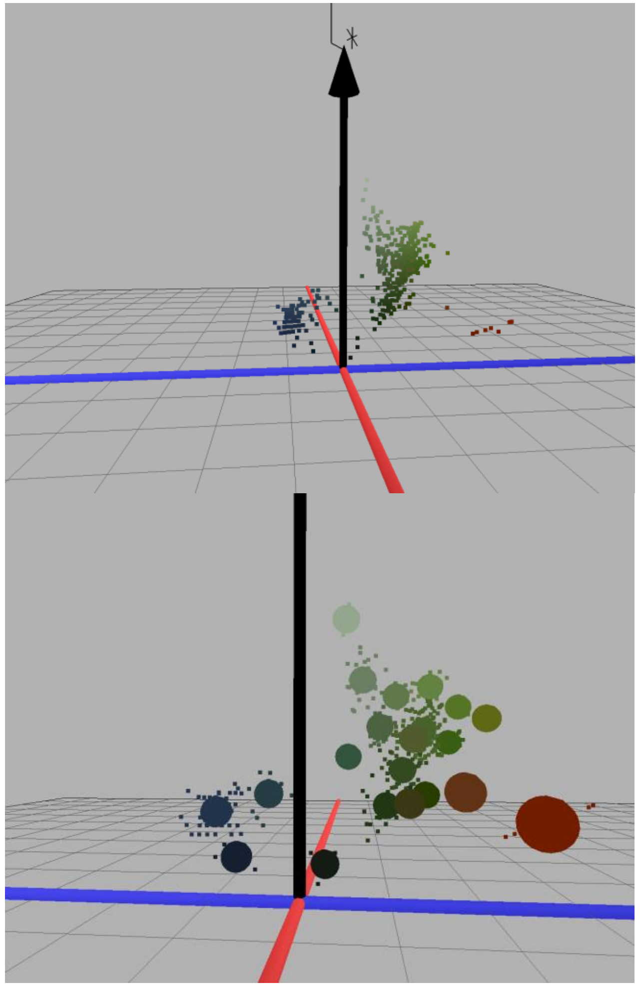
# Hierarchical clustering – divisive approach

- Opposite of agglomerative approach: start with all samples in one cluster and recursively split until each sample is in its own cluster
- Naïve method: Search for best subdivision of a set into two clusters out of the $2^{N-1}-1$ possibilities based on some criterion.



# Hierarchical clustering – Remarks

- There is no way to recover from a “bad grouping” (or splitting)
- To get k clusters just stop early or cut off the appropriate branches in the dendrogram
- Example: reduce color palette to 500 colors (or even fewer)



# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

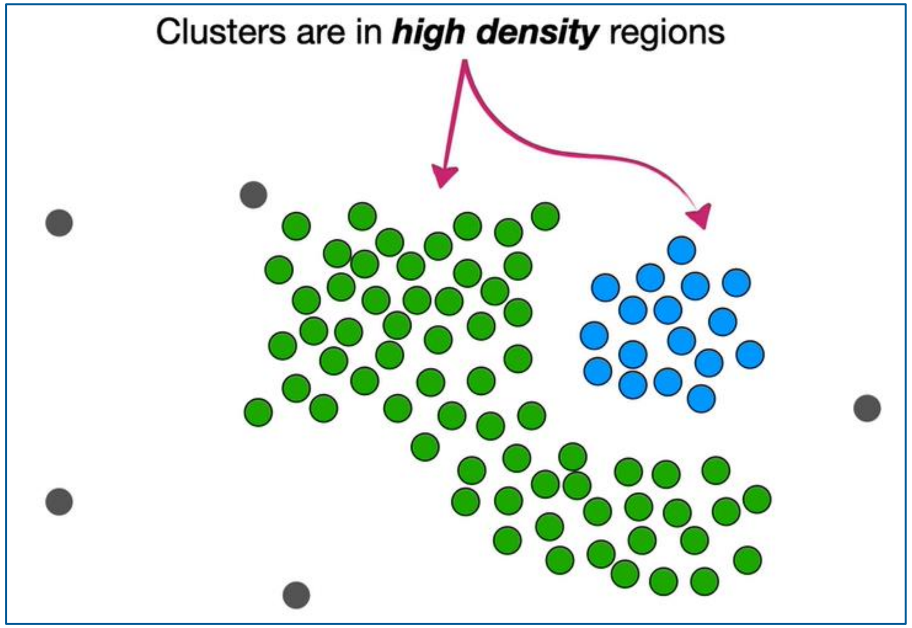
# DBSCAN

- DBSCAN … *"Density-Based Spatial Clustering of Applications with Noise"*

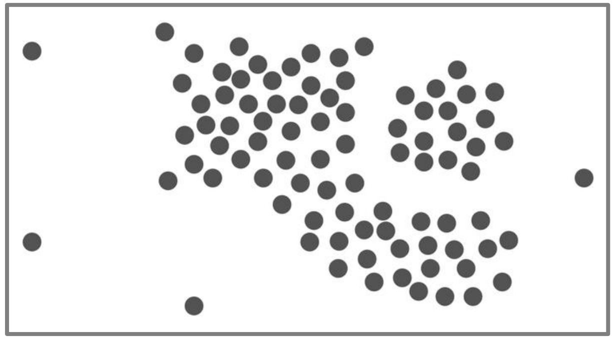
- Works well even with "nested clusters" (also in higher dimensions with the right distance metric)
  + reduced risk of *single-link clusters*

- Sequential algorithm that also marks outliers

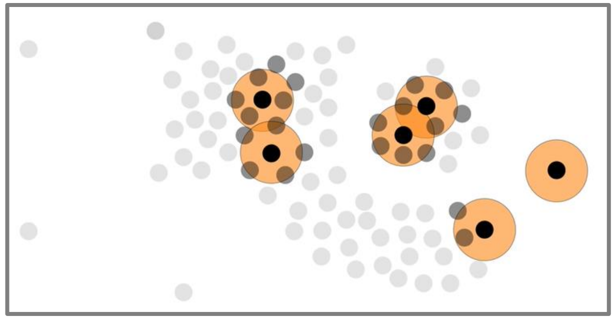
- **Hyperparameters**: distance d, min. neighbors n



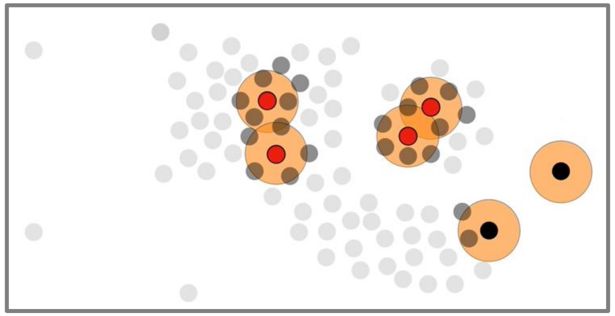
# DBSCAN – Algorithm



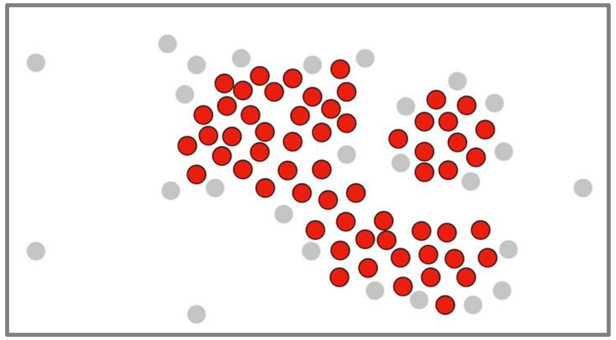
*1.) Starting point: raw unlabelled points*



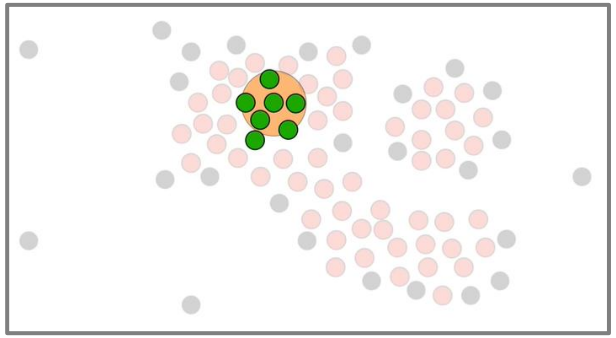
*2.) Count the number of points within **distance d** (orange, hyperparameter) for each point*



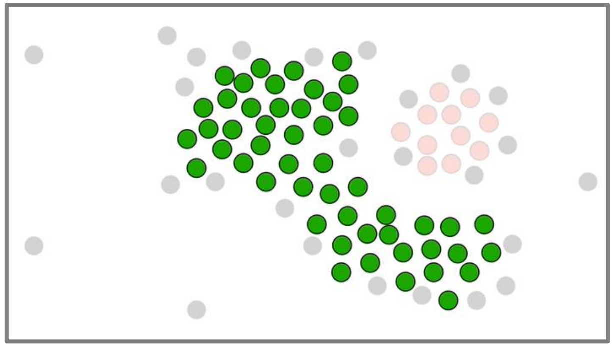
*3.) Label points as **„core“ points (red)** if they have more than **n neighbors** (hyperparameter, e. g. 4) within **distance d**, otherwise as **„non-core“ points (black)***



*4.) Each point is either **core** or **non-core***

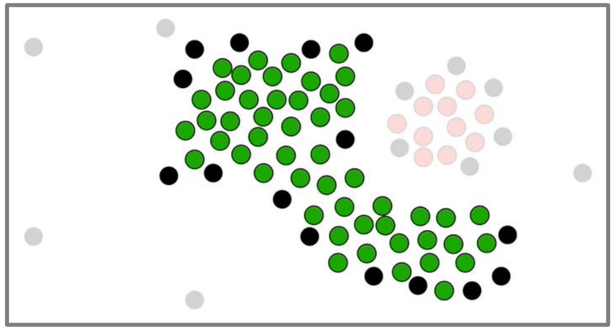


*5.) Randomly pick a **core point** and assign it to **cluster 1**. Add all core points within **distance d** to that point to the cluster.*

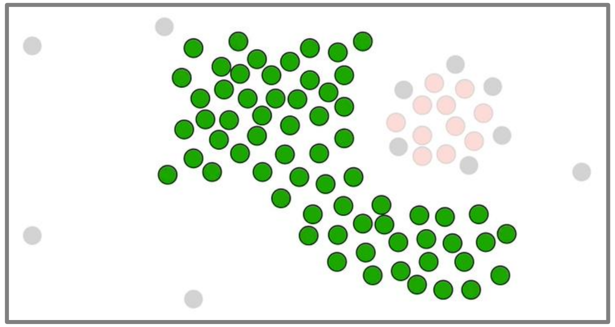


*6.) Repeat step 5 for all added core points recursively until there are no more core points to add*

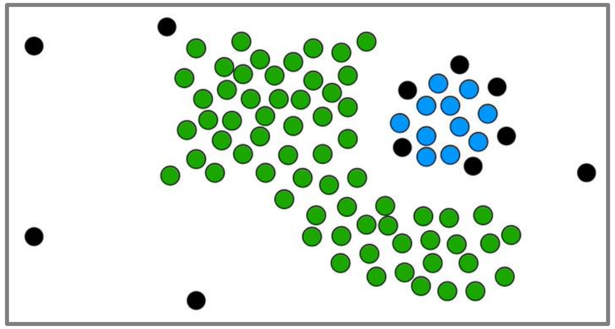
# DBSCAN – Algorithm



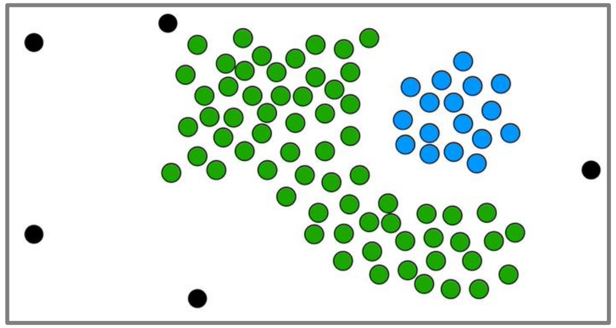
*7.) Add all **non-core points** within distance d to **cluster 1** (only once, not recursively!)*



*8.) That creates our **first cluster (green)***



*9.) Like in step 5, pick a new **non-clustered core point as a seed for cluster 2 (blue)** and recursively add all core points closer than distance d*



*10.) Like in steps 7 & 8, add the **non-core points** and we now have our **full cluster 2***

*11.) If there are more non-clustered core points left, repeat the steps until there are no more unclustered core points left.*

*All **remaining non-core points** are marked as **outliers.***

# DBSCAN – Remarks

1. No initial number of clusters necessary

2. Mostly deterministic except for some non-core boundary points and cluster order

3. Run time complexity $O(n^2)$ but some tradeoffs can be made by using efficient memory structures (e. g. R-tree variations) and pre-calculations

4. Robust to outliers (we even get a list of them)

5. Clusters should have similar densities for the algorithm to work properly (otherwise the min neighbor parameter only fits selected clusters well)

# Lecture 7 – Unsupervised Learning

- Intro Unsupervised Learning
- Clustering
- K-means clustering
- Hierarchical clustering
- DBSCAN
- Association rules

# Association rules

- Association Rule Mining is a rule-based machine learning method that helps to uncover meaningful correlations between different products according to their co-occurrence in a data set
- in sales/marketing also called “Market Basket Analysis”
- Association rules represent relationships between set of elements in transactions (not an individual's preference)
- *“Customers also bought xxx”* → Recommendation
- *“Frequently bought together items”* → Association
- Market Basket Analysis is very simple:
  → Basic idea: check likelihood of different elements occurring together
- We use the Apriori-Algorithm
- Association rules use:
  - **Support**
  - **Confidence**
  - **Lift**


**Example Use Cases:**
- A and B can be placed together so that when a customer buys one of the product he doesn't have to go far away to buy the other product.
- People who buy one of the products can be targeted through an advertisement campaign to buy the other.
- Collective discounts can be offered on these products if the customer buys both of them.
- Both A and B can be packaged together.

# Association rules

Rules need to meet a **user-specified minimum support and minimum confidence**

$$Rule: X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

- Support … (general) probability that X occurs over all transactions
- Confidence … conditional probability of Y when X occurred/is present
- Lift … how likely an item is purchased when another item is purchased, while controlling for how popular both items are
  *(higher value == higher association; >1 … "association is present")*

# Association rules – Apriori algorithm

**Apriori principle**: if an itemset is infrequent, then all its supersets must also be infrequent
→ use this for effective pruning of possible combinations to reduce the required computational effort

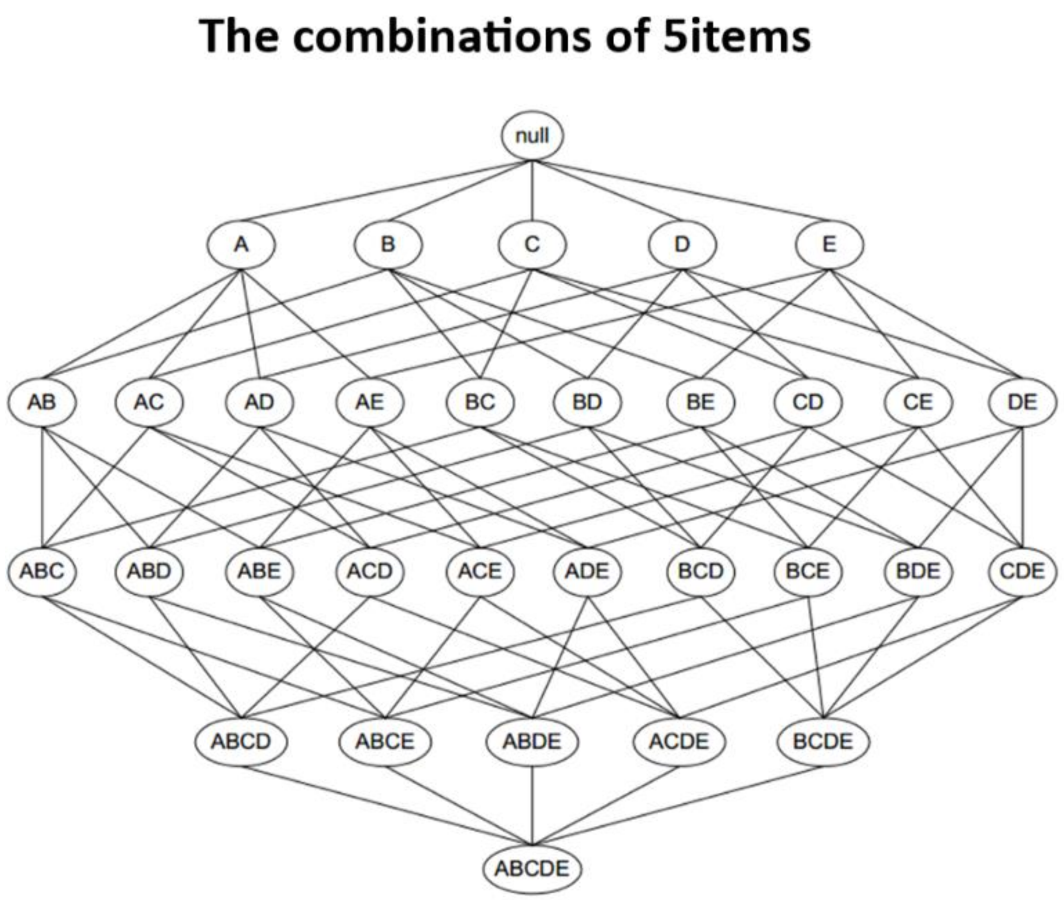
**Steps:**

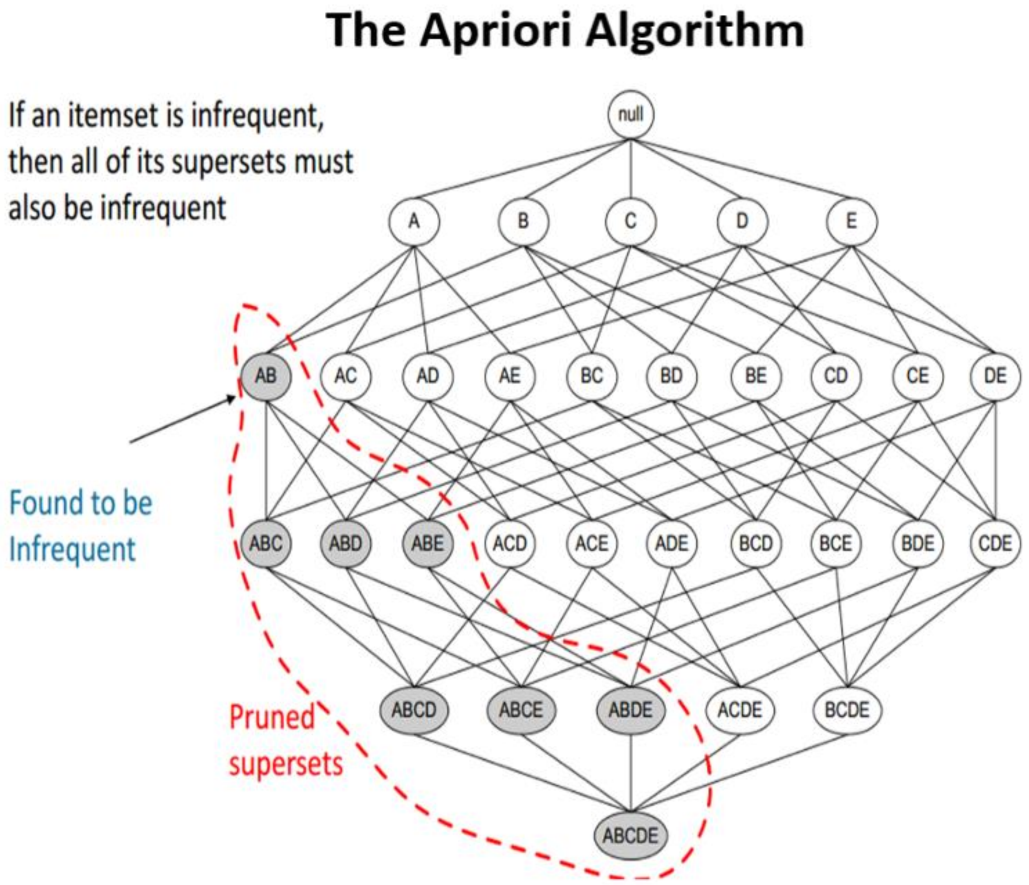
1. **Hyperparameter Setting:** set a minimum value for **support** and **confidence**. This means that we are only interested in finding rules for the items that have certain default existence (i. e. support) and have a minimum value for co-occurrence with other items (i. e. confidence)
2. **Frequent Itemset Generation**: extract all the itemsets having higher value of support than minimum threshold
   a) Start with itemset containing only 1 element, e. g., {bread}, {butter}
   b) Calculate the support for itemsets and only keep those itemsets that reach a specified minimum support threshold *(on-the-fly-pruning – see next slide for animation)*
   c) Generate all possible (non-empty) itemset combinations
   d) Repeat step b and c until there are no more new valid itemset combinations
3. **Strong Rule Generation:** generate all possible rules from the itemsets and only select those with confidence values higher than the minimum threshold
4. **Order list of association rules:** order the rules by descending order of Lift

# Association rules – Apriori algorithm animation (pruning via Support)

https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html/2

# Association rules – Apriori algorithm animation (pruning via Support)




https://www.kdnuggets.com/2019/12/market-basket-analysis.html

# Apriori algorithm example

For this example we will use a minium support of 2 and a minimum confidence of 50%

| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

# Apriori algorithm example

| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

*original dataset*

**Frequent Itemset Generation:**

1. Calculate support values and only keep those with values larger than 2 (itemset size 1)
2. Combine sets and check again for the support value (itemset size 2)
3. Combine those sets and check again for the support value (itemset size 3)

*(→ Remember: we use sets, so we do not care about the order of items in an itemset)*

| Itemset | sup_count |
|---|---|
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

*Step 1 (Level 1)*

| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I4 | 1 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I3,I4 | 0 |
| I3,I5 | 1 |
| I4,I5 | 0 |

| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I2,I5 | 2 |

*Step 2 (Level 2) with filtering for min support of 2*

| Itemset | sup_count |
|---|---|
| I1,I2,I3 | 2 |
| I1,I2,I5 | 2 |

*Step 3 (Level 3) with filtering for min support of 2*

# Apriori algorithm example

| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

*original dataset*

**Strong Rule Generation:**

For each frequent itemset:

1. Generate all combinations of possible rules with the items in your itemset
2. Calculate the Confidence values
3. Only keep those with Confidence > min Confidence

**Example for {I1, I2, I3}:**

- *[I1^I2]=>[I3] //confidence = sup(I1^I2^I3)/sup(I1^I2) = 2/4*100=50%*
- *[I1^I3]=>[I2] //confidence = sup(I1^I2^I3)/sup(I1^I3) = 2/4*100=50%*
- *[I2^I3]=>[I1] //confidence = sup(I1^I2^I3)/sup(I2^I3) = 2/4*100=50%*
- *[I1]=>[I2^I3] //confidence = sup(I1^I2^I3)/sup(I1) = 2/6*100=33%*
- *[I2]=>[I1^I3] //confidence = sup(I1^I2^I3)/sup(I2) = 2/7*100=28%*
- *[I3]=>[I1^I2] //confidence = sup(I1^I2^I3)/sup(I3) = 2/6*100=33%*

*Example: A confidence of 50% means that 50% of the customers, who purchased I1 and I2 also bought I3.*

# Apriori algorithm example

**Lift:**

- We now have rules of frequent itemsets where we are confident that there is a correlation
- BUT: some items are just bought very often and might create fluke rules

→use a final ordering of rule via the Lift value to sort that out

**Example:**

| Transaction | Support | Confidence | Lift |
|---|---|---|---|
| Canned Beer → Soda | 1% | 20% | 1.0 |
| Canned Beer → Berries | 0.1% | 1% | 0.3 |
| Canned Beer → Male Cosmetics | 0.1% | 1% | 2.6 |

- Lift = 1: no actual association (because beer and soda are just bought quite often)
- Lift < 1: negative association (when somebody buys beer they are usually not buying berries)
- Lift > 1: positive association (when somebody buys beer they buy male cosmetics – although the effect is small with a confidence of 1%)

# Association rules – Apriori algorithm

- Drawbacks:
  - Computationally expensive
    → time and space complexity is exponential $O(2^{\{|D|\}})$ with $|D|$ being the horizontal width (= total number of items)
  - Spurious associations: some rules may seem logical

- Common improvements:
  - Hash-based itemset counting
  - Transaction reduction: A transaction that does not contain any frequent k-itemsets useless in subsequent scans
  - Partitioning: An itemset that is potentially frequent must be frequent in at least one of the partitions
  - Sampling: Mining on a subset of given data, lower support threshold + a method to determine the completeness
  - Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

*Articles about association rules and Python examples:*
- https://towardsdatascience.com/data-mining-market-basket-analysis-with-apriori-algorithm-970ff256a92c
- https://ml2021.medium.com/market-basket-analysis-apriori-algorithm-4fb052ffd2f2
- https://www.geeksforgeeks.org/apriori-algorithm/
- https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/

# Final remarks on Machine Learning

- All fields of ML are evolving quite fast at the moment as more companies want to utilize their data and big-tech is investing heavily

- Selection of some current trends in ML:
  1. Improved security: homomorphic encryption, federated learning, private set intersection, anonymizing data etc.
  2. Ethics and avoiding biases in AI
  3. Regulation of AI (e. g. on EU-level)
  4. Responsible/Sustainability in AI: use less computation/simpler models to reduce $CO_2$ emissions
  5. Few Shot, One Shot, & Zero Shot Machine Learning
  6. Understandable/Transparent AI
  7. Auto ML and democratizing AI
  8. Process adaptations like MLOps, DataOps, Data Governance
  9. AI Strategy on EU-, national- and company-level
  10. Tiny-ML, on-edge-inference/computing, specialized hardware (e. g. TPU)
  11. Multi-Modal Learning and Multi-Objective Models
  12. Quantum ML (hardware is not yet ready but algorithms are being adapted and software/middleware are being developed right now)

# Final remarks on Machine Learning

- Some recommendations for continuous learning online:
  - https://towardsdatascience.com/
  - https://www.kdnuggets.com/
  - https://www.kaggle.com/
  - https://www.geeksforgeeks.org/
  - https://www.reddit.com/r/datascience/
  - https://machinelearningmastery.com/

*and many more (also on youtube & Co)*

# Control Questions

**You should be able to answer the following questions:**

- How is unsupervised learning different from supervised and reinforcement learning?
- What is soft and hard clustering? Discuss a few use cases for clustering.
- How does k-means work and which variations do you know. Explain them and why they might make sense for a use case. What is the elbow method?
- How can you evaluate the quality of clustering results?
- What is hierarchical clustering? What two types exist? What is a dendrogram?
- How does DBSCAN work? What are it‘s advantages?
- What are association rules? Where would you use association rules? Benefits and drawbacks?
- How does the apriori algorithm work and which principle does it use?

# Links and Sources


- C. Bishop, Pattern Recognition and Machine Learning. 2006.

- Images and content adapted from slides of Nicole Artner, TU Wien, LV “Einführung Mustererkennung”
- https://en.wikipedia.org/wiki/K-means_clustering
- https://en.wikipedia.org/wiki/K-means%2B%2B
- https://en.wikipedia.org/wiki/Silhouette_(clustering)
- https://en.wikipedia.org/wiki/Self-organizing_map
- Williams, J. T.; Carpenter, K. E.; Van Tassell, J. L.; Hoetjes, P.; Toller, W.; Etnoyer, P.; Smith, M. (2010). "Biodiversity Assessment of the Fishes of Saba Bank Atoll, Netherlands Antilles".
- https://www.kdnuggets.com/2019/12/market-basket-analysis.html
- https://en.wikipedia.org/wiki/Association_rule_learning
- https://medium.com/analytics-vidhya/association-rule-mining-concept-and-implementation-28443d16f611
- https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html/2
- https://www.youtube.com/watch?v=mtkWR8sx0NA
- **Distances in higher dimensions and why they loose their “meaning”: https://towardsdatascience.com/the-surprising-behaviour-of-distance-metrics-in-high-dimensions-c2cb72779ea6**