Clustering

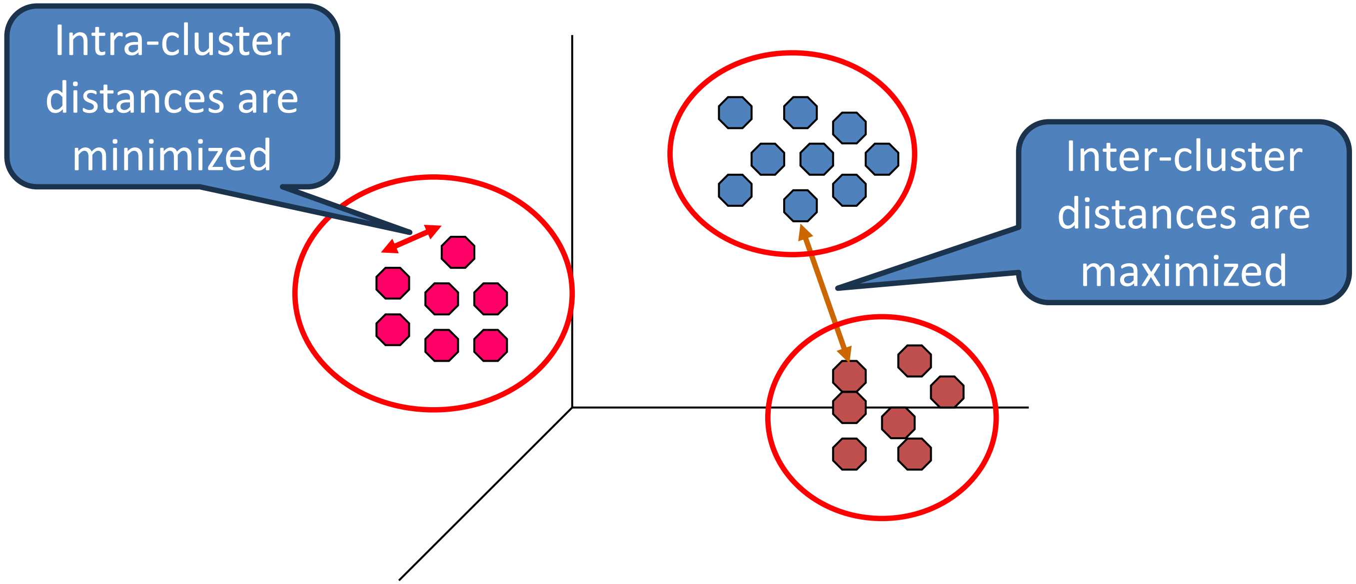
IE500 Data Mining

Outline

- What is Cluster Analysis?
- K-Means Clustering
- Density-based Clustering
  - DBSCAN
- Proximity Measures
- Anomaly Detection
  - Statistical Approaches
  - Distance-based Approaches

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group
  - will be similar to one another
  - and different from the objects in other groups
- Goal: Get a better understanding of the data



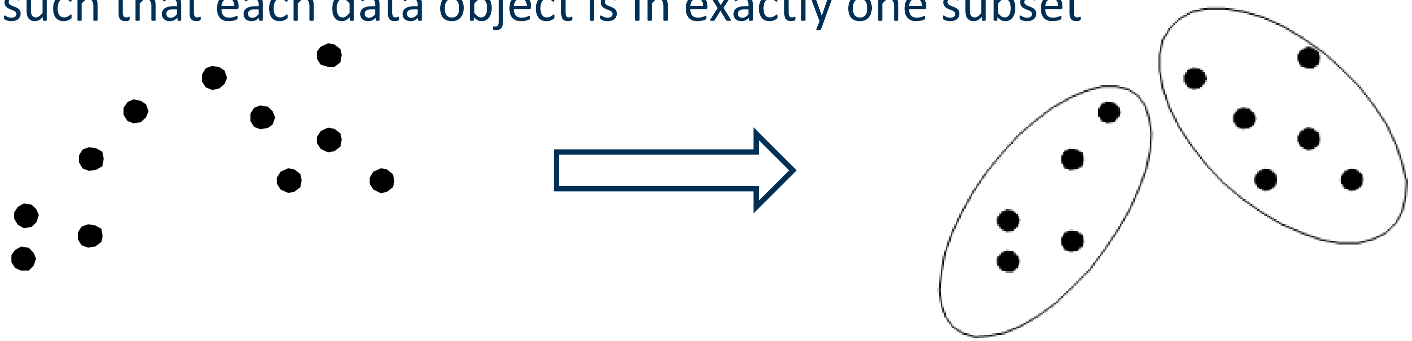
Cluster Analysis as Unsupervised Learning

- **Supervised learning**: Discover patterns in the data that relate data attributes with a target (class) attribute
  - The set of classes is known before
  - Class attributes are usually provided by human annotators
  - Patterns are used for prediction of the target attribute for new data

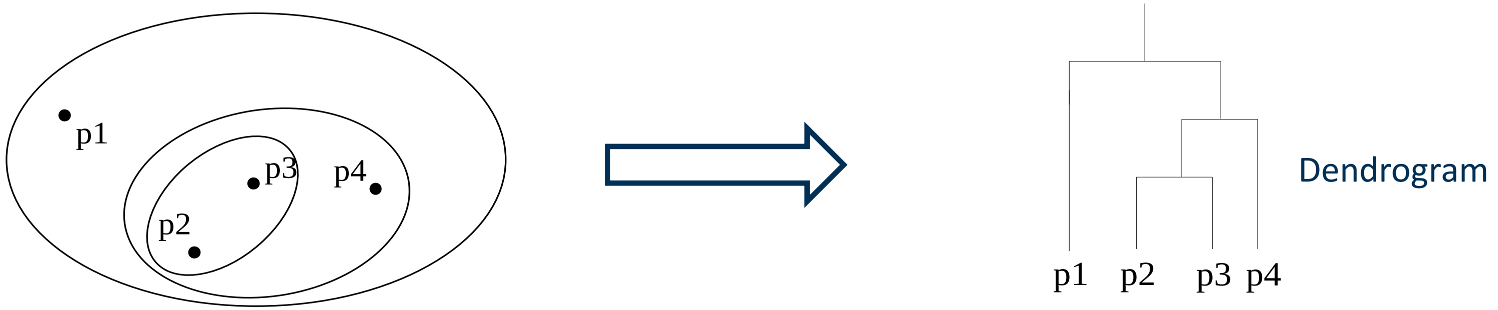
- **Unsupervised learning**: The data has no target attribute
  - We want to explore the data to find some intrinsic structures in it
  - The set of classes/clusters is not known before
  - Cluster Analysis and Association Rule Mining are unsupervised learning tasks

Types of Clusterings

- **Partitional Clustering**
  - A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset



- **Hierarchical Clustering (see additional material)**
  - A set of nested clusters organized as a hierarchical tree



Aspects of Cluster Analysis

UNIVERSITY OF MANNHEIM
Data and Web Science Group

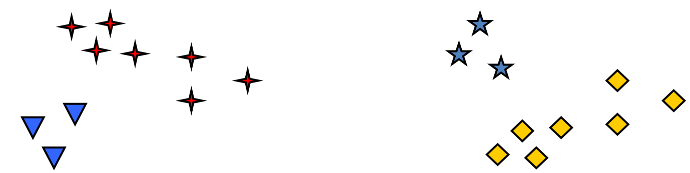
- **A clustering algorithm**
  - Partitional algorithms
  - Density-based algorithms
  - Hierarchical algorithms
  - ...
- **A proximity (similarity, or dissimilarity) measure**
  - Euclidean distance
  - Cosine similarity
  - Data type-specific similarity measures
  - Domain-specific similarity measures
- **Clustering quality**
  - Intra-clusters distance $\Rightarrow$ minimized
  - Inter-clusters distance $\Rightarrow$ maximized
  - The clustering should be useful with regard to the goal of the analysis

The Notion of a Cluster is Ambiguous

How many clusters do you see?

Six Clusters

Two Clusters



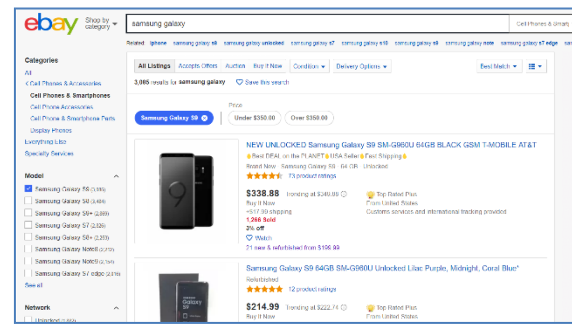
Four Clusters

The usefulness of a clustering depends on
the **goal of the analysis**

Example Applications

- Market Segmentation
  - Goal: Identify groups of similar customers
  - Level of granularity depends on the task at hand

- E-Commerce
  - Identify offers of the same product on electronic markets



- Image Recognition
  - Identify parts of an image that belong to the same object

K-Means Clustering

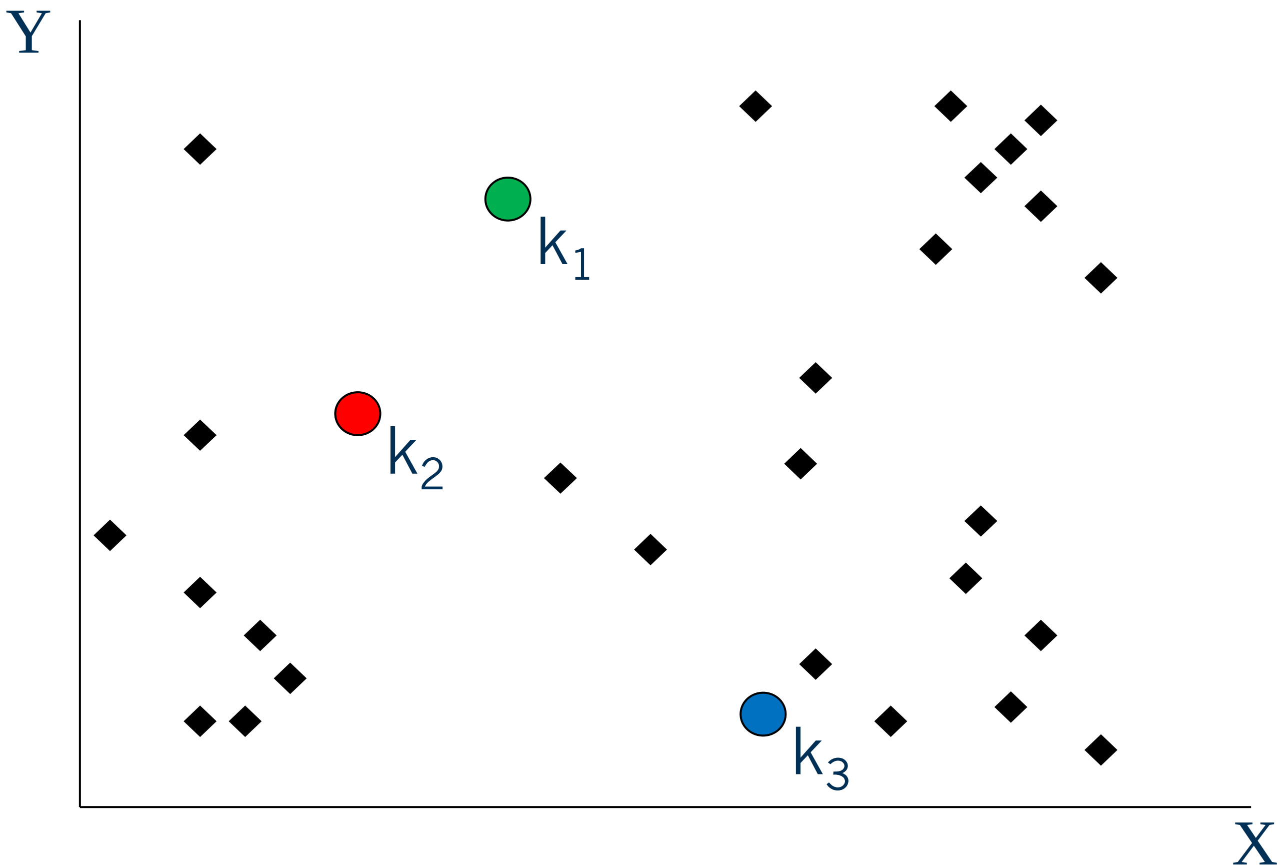
- Partitional clustering algorithm
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified beforehand

- Algorithm:

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

K-Means Example, Step 1

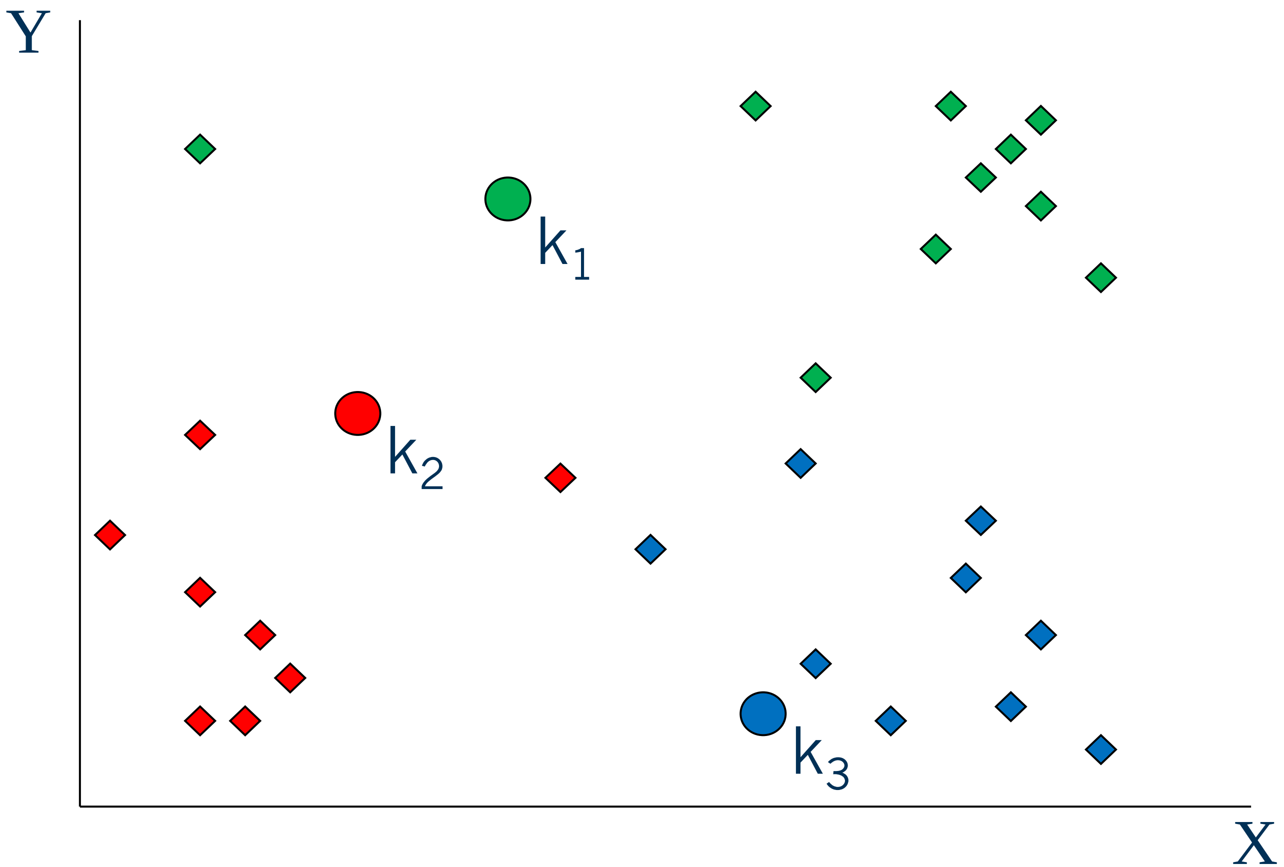
- Randomly pick 3 initial centroids



K-Means Example, Step 2

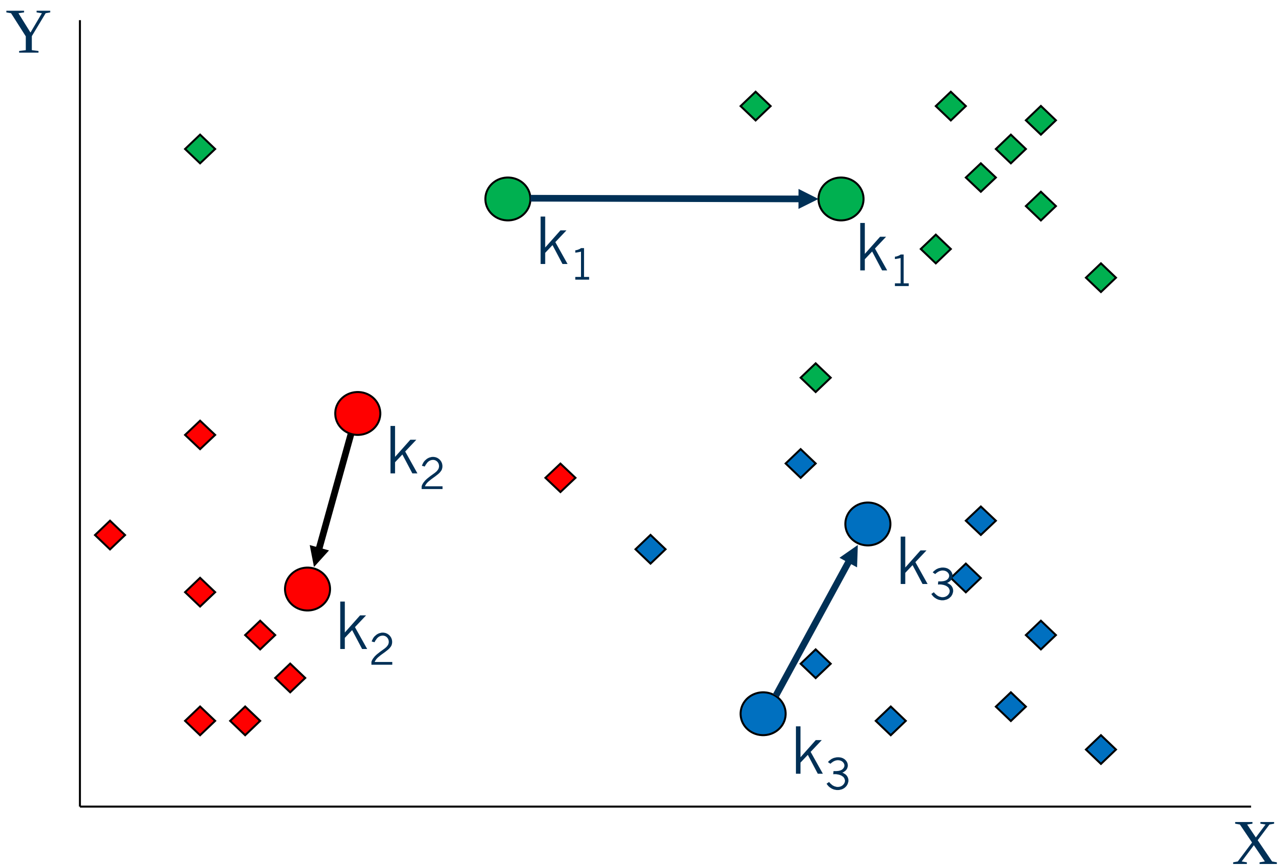
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Assign each point to the closest centroid



K-Means Example, Step 3

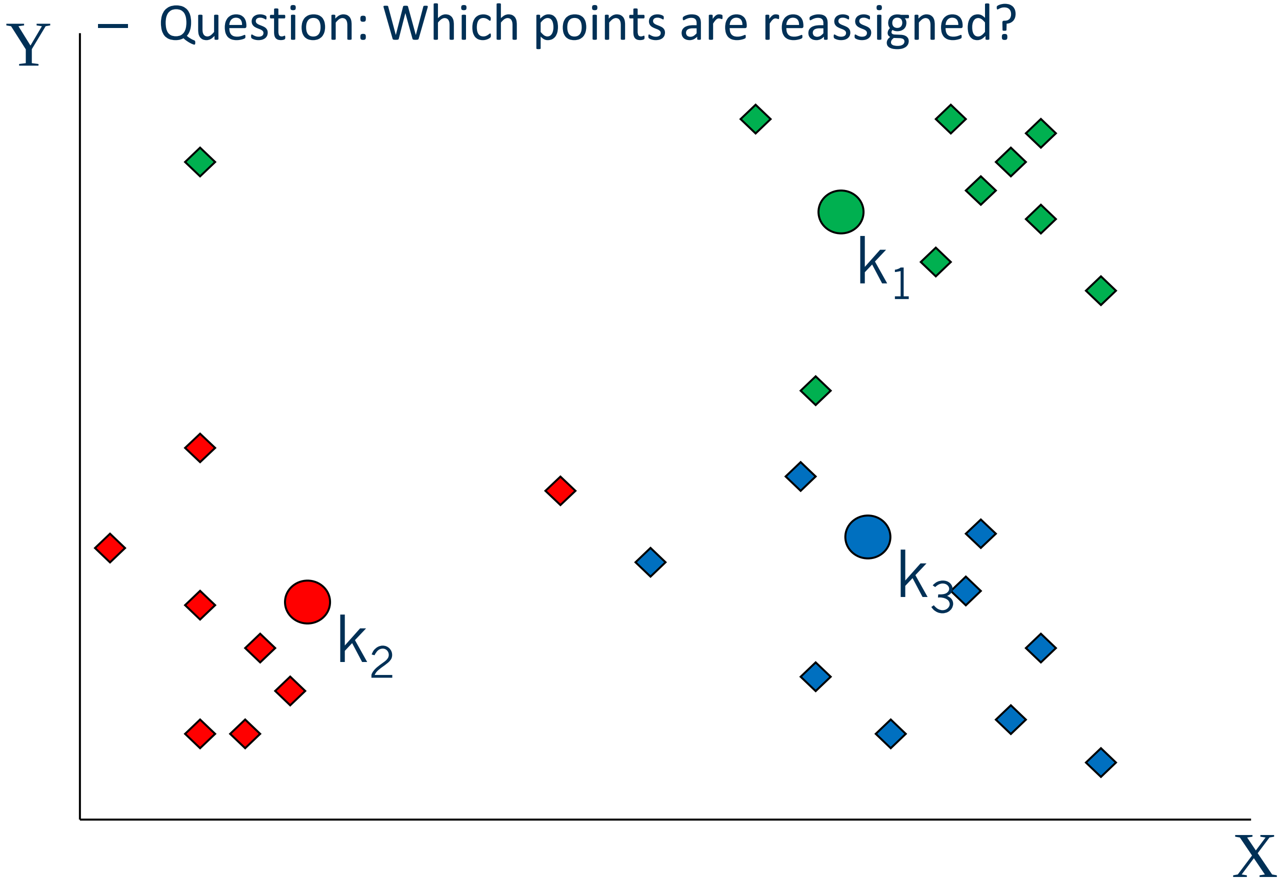
- Move each centroid to the mean of each cluster



K-Means Example, Step 4

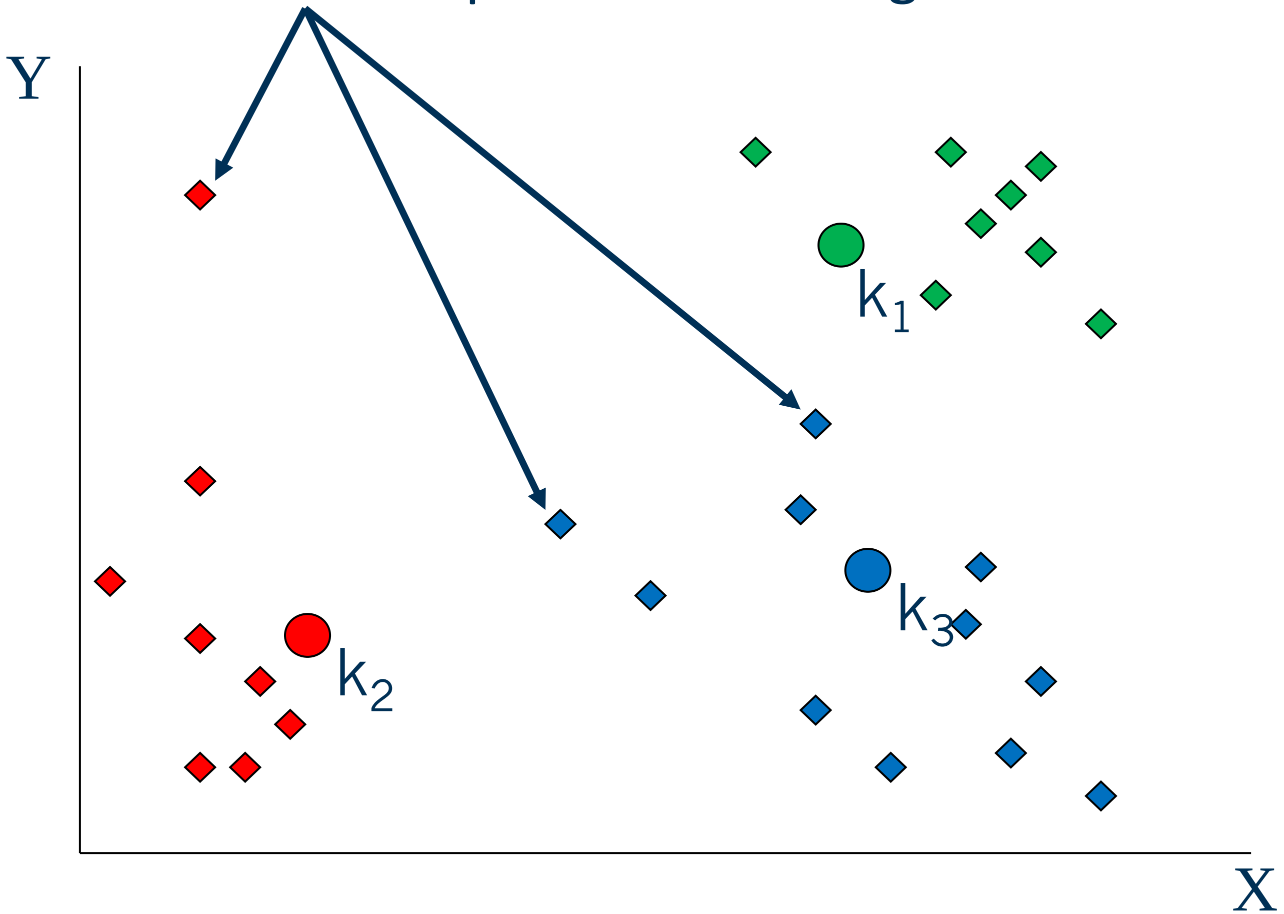
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Reassign points if they are now closer to a different centroid
  - Question: Which points are reassigned?



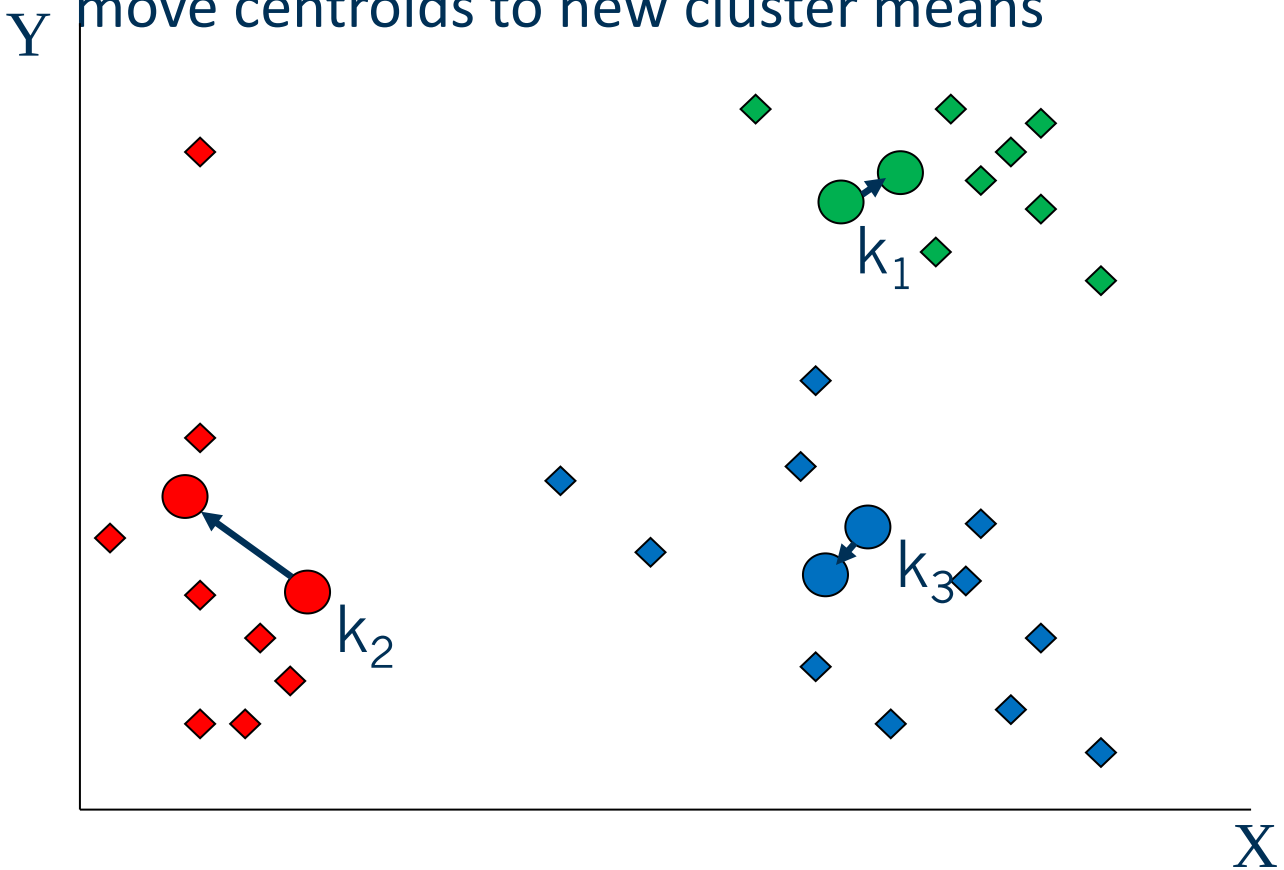
K-Means Example, Step 4

- Answer: Three points are reassigned



K-Means Example, Step 5

- Re-compute cluster means and
  move centroids to new cluster means



Convergence Criteria

- Default convergence criterion
  - No (or minimum) change of centroids

- Alternative convergence criteria
  - No (or minimum) re-assignments of data points to different clusters
  - Stop after x iterations
  - Minimum decrease in the sum of squared error (SSE)
    - See next slide

Evaluating K-Means Clusterings

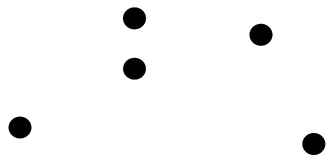
- Widely used cohesion measure: **Sum of Squared Error (SSE)**
  - For each point, the error is the distance to the nearest centroid
  - To get SSE, we square these errors and sum them

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} dist(x, m_j)^2$$

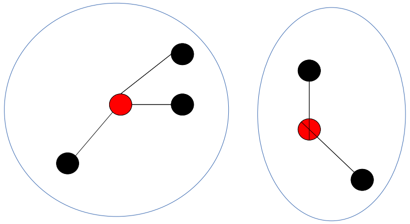
  - C_j is the j-th cluster
  - m_j is the centroid of cluster C_j (the mean vector of all the data points in C_j)
  - $dist(x, m_j)$ is the distance between data point x and centroid m_j
- Given several clusterings (= groupings), we should prefer the one with the smallest SSE

Illustration: Sum of Squared Error

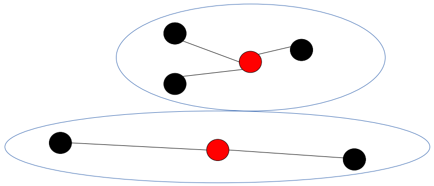
- Clustering problem given:



- Good clustering
  - small distances to centroids

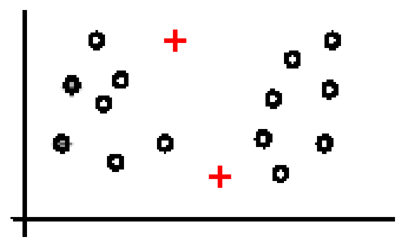


- Not so good clustering
  - larger distances to centroids

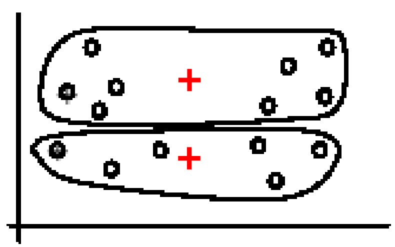


Weaknesses of K-Means: Initial Seeds

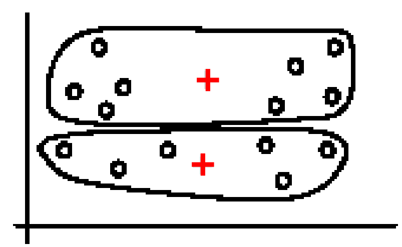
- Clustering results may vary significantly depending on initial choice of seeds (**number** and **position** of seeds)



(A). Random selection of seeds (centroids)



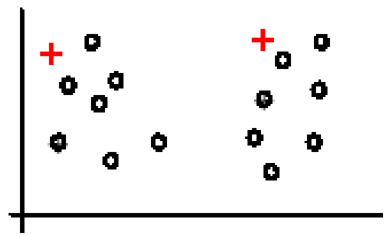
(B). Iteration 1



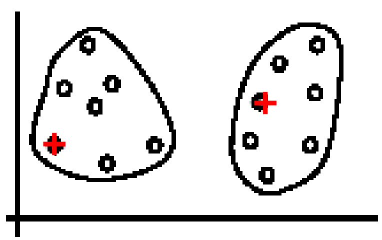
(C). Iteration 2

Weaknesses of K-Means: Initial Seeds

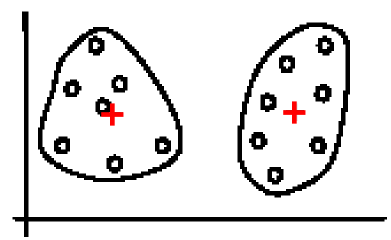
- If we use **different seeds**, we get good results



(A). Random selection of *k* seeds (centroids)



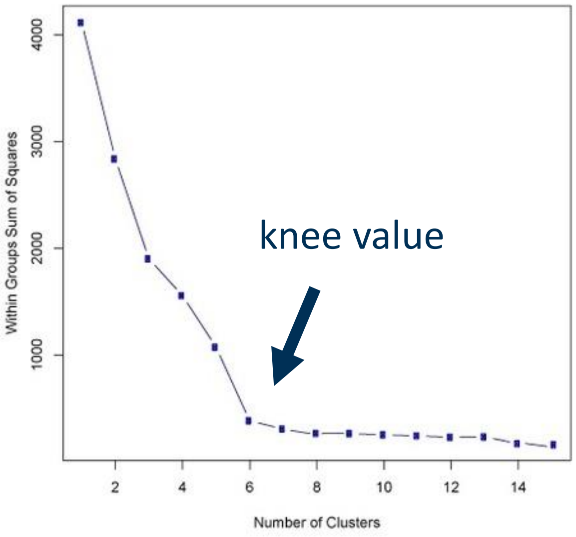
(B). Iteration 1



(C). Iteration 2

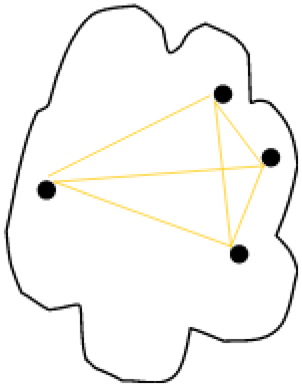
Improving the Clustering Results

- Restart a number of times with different random seeds
  - chose the resulting clustering with the smallest sum of squared error (SSE)
- Run k-means with different values of k
  - The SSE for different values of k cannot directly be compared
    - Think: what happens for k → number of examples?
  - Workarounds
    - Choose k where SSE improvement decreases (knee value of k)
    - Employ X-Means
      - Variation of K-Means algorithm that automatically determines k
      - Starts with small k, then splits large clusters until improvement decreases

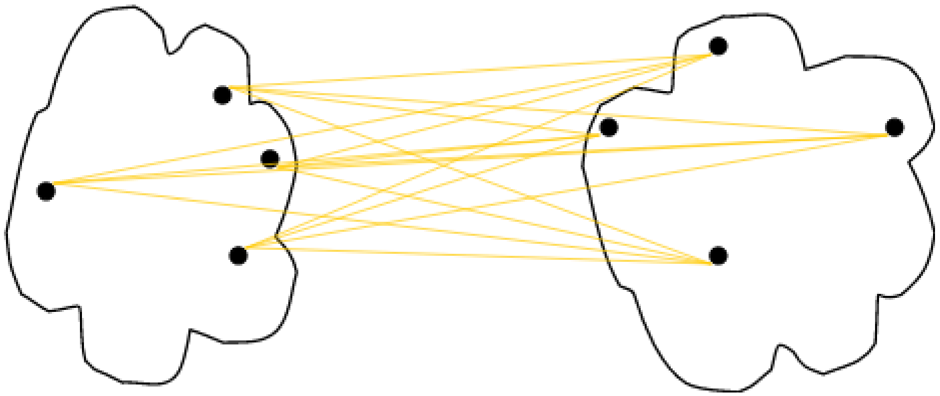


Choosing k – Cluster Evaluation

- Recap: we want to maximize
  - Cohesion: measures how closely related are objects in a cluster
  - Separation: measure how distinct or well-separated a cluster is from other clusters



Cohesion



Separation

Silhouette Coefficient

- Cohesion $a(x)$: Average distance of x to all other vectors in the same cluster
- Separation $b(x)$: Average distance of x to the vectors in other clusters. Find the minimum among the clusters.
- Silhouette $s(x)$:

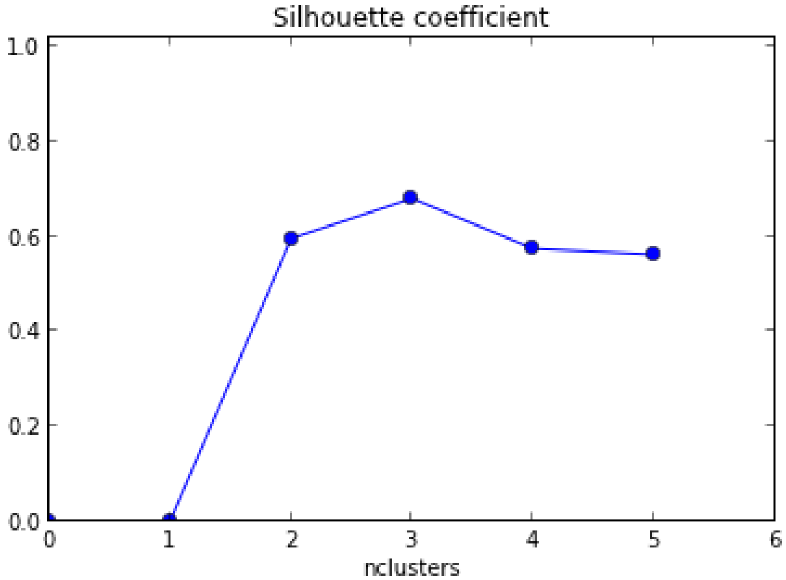
$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

- $s(x) = [-1,1]$ -1=bad, 0=indifferent, 1=good
- Silhouette coefficient (SC):

$$SC = \frac{1}{N} \sum_{i=1}^{N} s(x_i)$$

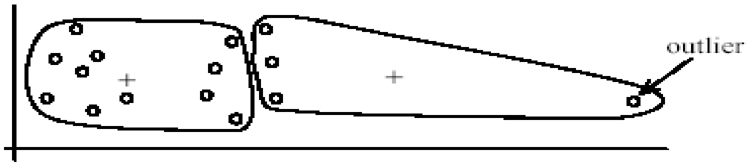
Selecting k
Using the Silhouette Coefficient

- Approach
  - Run k-means with different k values
  - Plot the Silhouette Coefficient
  - Pick the best (i.e., highest) silhouette coefficient
  - Note: silhouette coefficient does not depend on no. of clusters

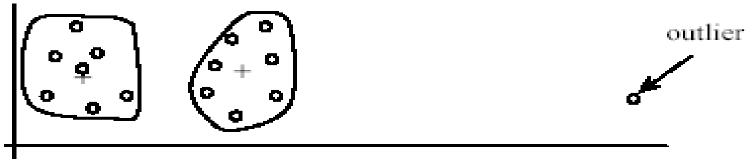


Weaknesses of K-Means: Problems with Outliers

- Possible remedy:
  - Remove data points far away from centroids
  - To be safe: monitor these possible outliers over a few iterations and then decide to remove them
- Other remedy: random sampling
  - After determining the centroids based on random samples, assign the rest of the data points (also improves runtime performance)



(A): Undesirable clusters



(B): Ideal clusters

K-Medoids

- K-Medoids is a K-Means variation that uses the **medians** of each cluster instead of the mean
- Medoids are the **most central existing data points** in each cluster
- K-Medoids is more robust against outliers as the median is not affected by extreme values:
  - Mean and Median of 1, 3, 5, 7, 9 is 5
  - Mean of 1, 3, 5, 7, 1009 is 205
  - Median of 1, 3, 5, 7, 1009 is 5

K-Means Clustering Summary

- **Advantages**
  - Simple, understandable
  - Efficient time complexity:
    O(n * K * I * d)
    where
    - n = number of points
    - K = number of clusters
    - I = number of iterations
    - d = number of attributes
- **Disadvantages**
  - Need to determine number of clusters
  - All items are forced into a cluster
  - Sensitive to
    - Outliers
    - Initial seeds

Python

```python
# import KMeans
from sklearn.cluster import KMeans

# create clusterer
estimator = KMeans(n_clusters = 3)

# create clustering
cluster_ids = estimator.fit_predict(dataset[['Att1', 'Att2']])
```

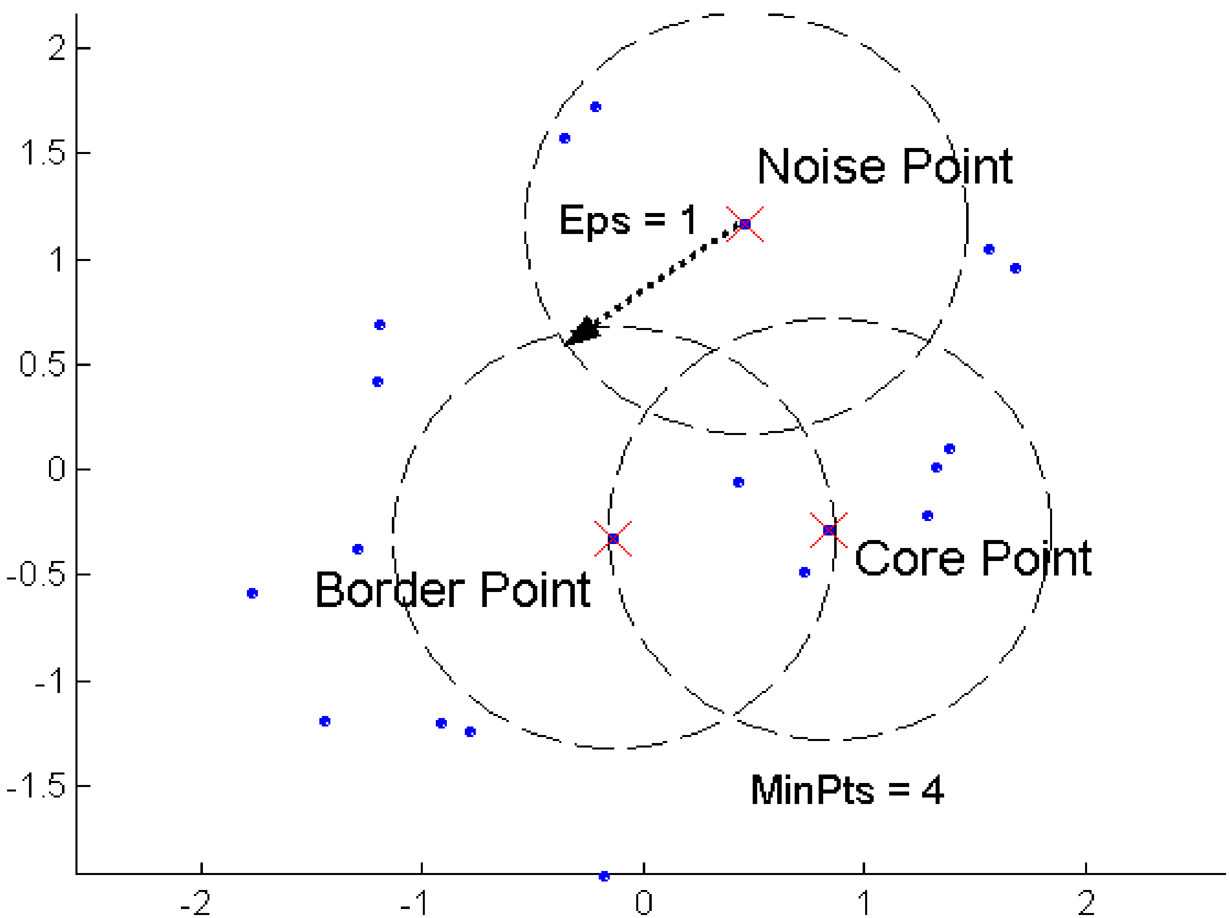
Density-based Clustering

- Challenging use case for K-Means because
  - Problem 1: Non-globular shapes
  - Problem 2: Outliers / noise points

DBSCAN

- DBSCAN is a density-based algorithm
  - **Density** = number of points within a specified radius Epsilon (Eps)
- Divides data points into three classes:
  - A point is a **core point** if it has at least a specified number of neighboring points (*MinPts*) within the specified radius *Eps*
    - the point itself is counted as well
    - these points form the interior of a dense region (cluster)
  - A **border point** has fewer points than *MinPts* within *Eps*, but is in the neighborhood of a core point
  - A **noise point** is any point that is not a core point or a border point

Examples of Core, Border, and Noise Points



DBSCAN Algorithm

UNIVERSITY OF MANNHEIM
Data and Web Science Group

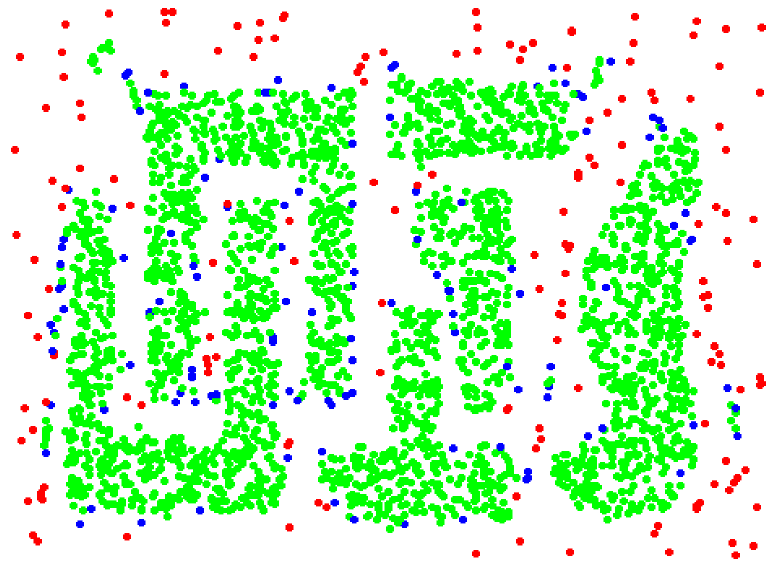
- Eliminate noise points
- Perform clustering on the remaining points

$current\_cluster\_label \leftarrow 1$
for all core points **do**
  if the core point has no cluster label **then**
    $current\_cluster\_label \leftarrow current\_cluster\_label + 1$
    Label the current core point with cluster label $current\_cluster\_label$
  end if
  for all points in the Eps-neighborhood, except i^{th} the point itself **do**
    if the point does not have a cluster label **then**
      Label the point with cluster label $current\_cluster\_label$
    end if
  end for
end for

perform recursion for all points in the Eps-neighborhood of the point

Examples of Core, Border, and Noise Points

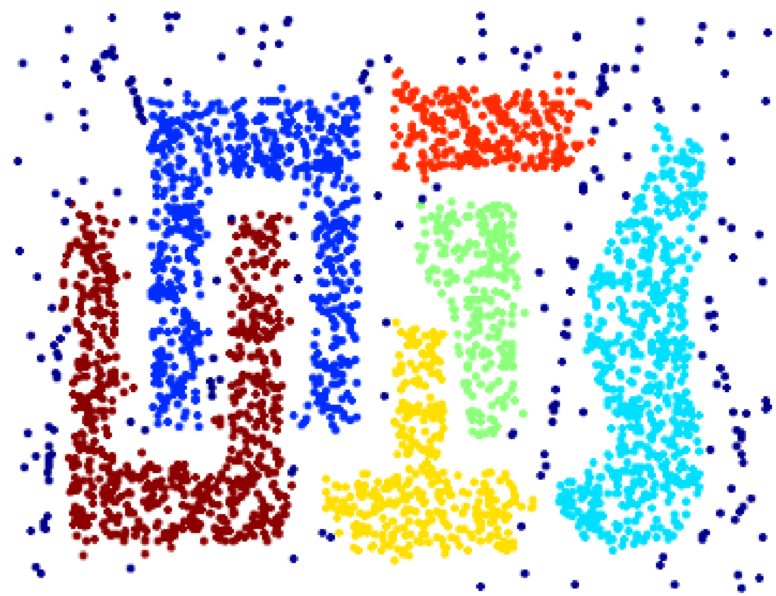
Original Points



Point types:
core, border and noise

Examples of Core, Border, and Noise Points

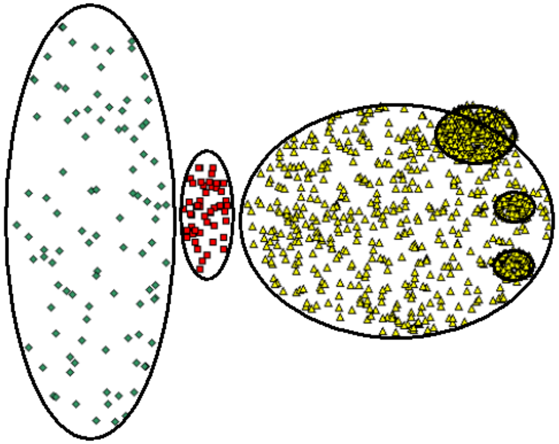
Original Points



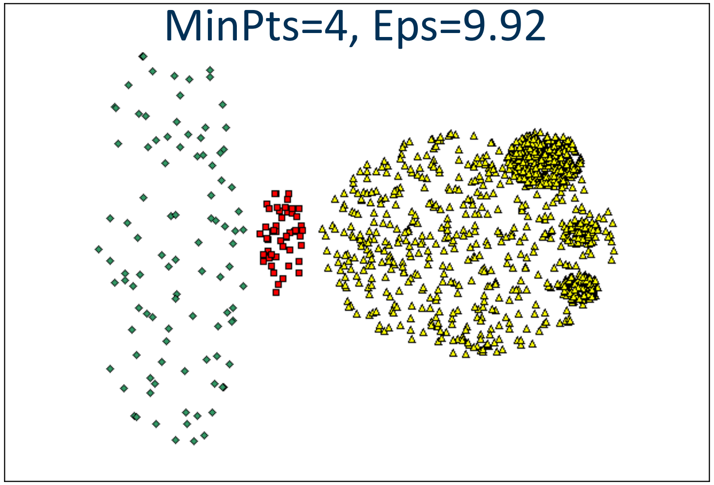
Clusters

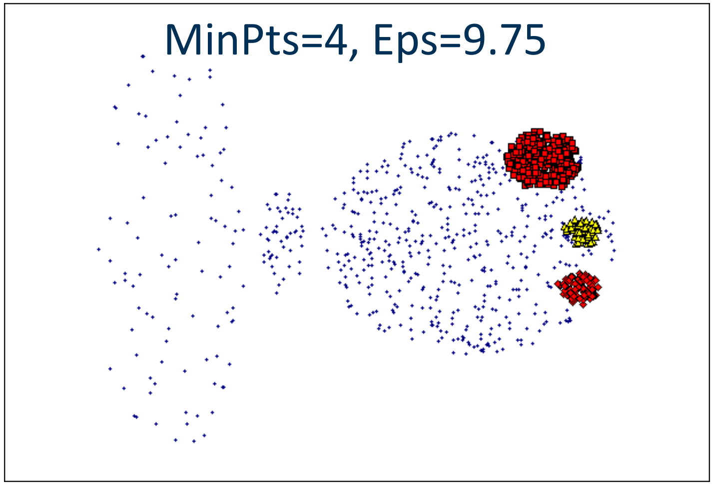
When DBSCAN Does NOT Work Well

- Varing densities
- High-dimensional data



Original Points

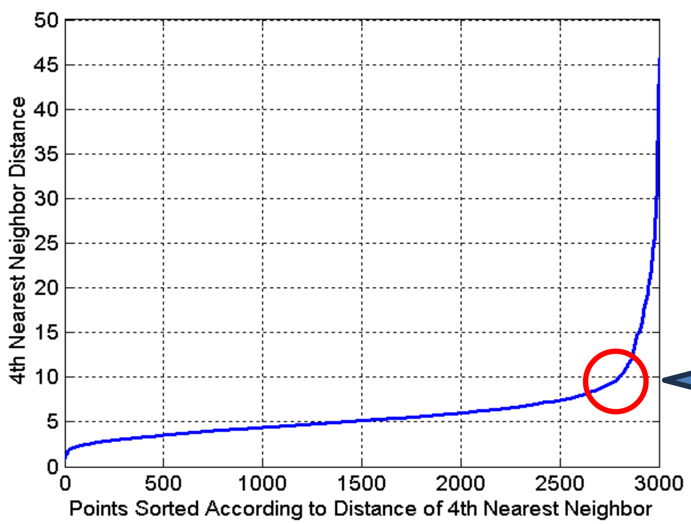




DBSCAN: Determining EPS and MinPts

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Idea: for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- Plot sorted distance of every point to its k^{th} nearest neighbor



Area where a good Epsilon value is assumed to be found

DBSCAN in Python

Python

```
# import DBSCAN
from sklearn.cluster import DBSCAN

# create the clusterer
clusterer = DBSCAN(min_samples=3, eps=1.5, metric='euclidean')

# create the clusters
clusters = clusterer.fit_predict(dataset[['Att1', 'Att2']])
```

Proximity Measures

- So far, we have seen different clustering algorithms
  - all of which rely on proximity (distance, similarity, ...) measures
- Similarity
  - Numerical measure of how alike two data objects are (higher: more alike)
  - Often falls in the range [0,1]
- Dissimilarity / Distance
  - Numerical measure of how different are two data objects (higher: less alike)
  - Minimum dissimilarity is often 0
  - Upper limit varies
- A wide range of different measures is used depending on the requirements of the application

Proximity of Single Attributes

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Similarity and dissimilarity for simple attributes

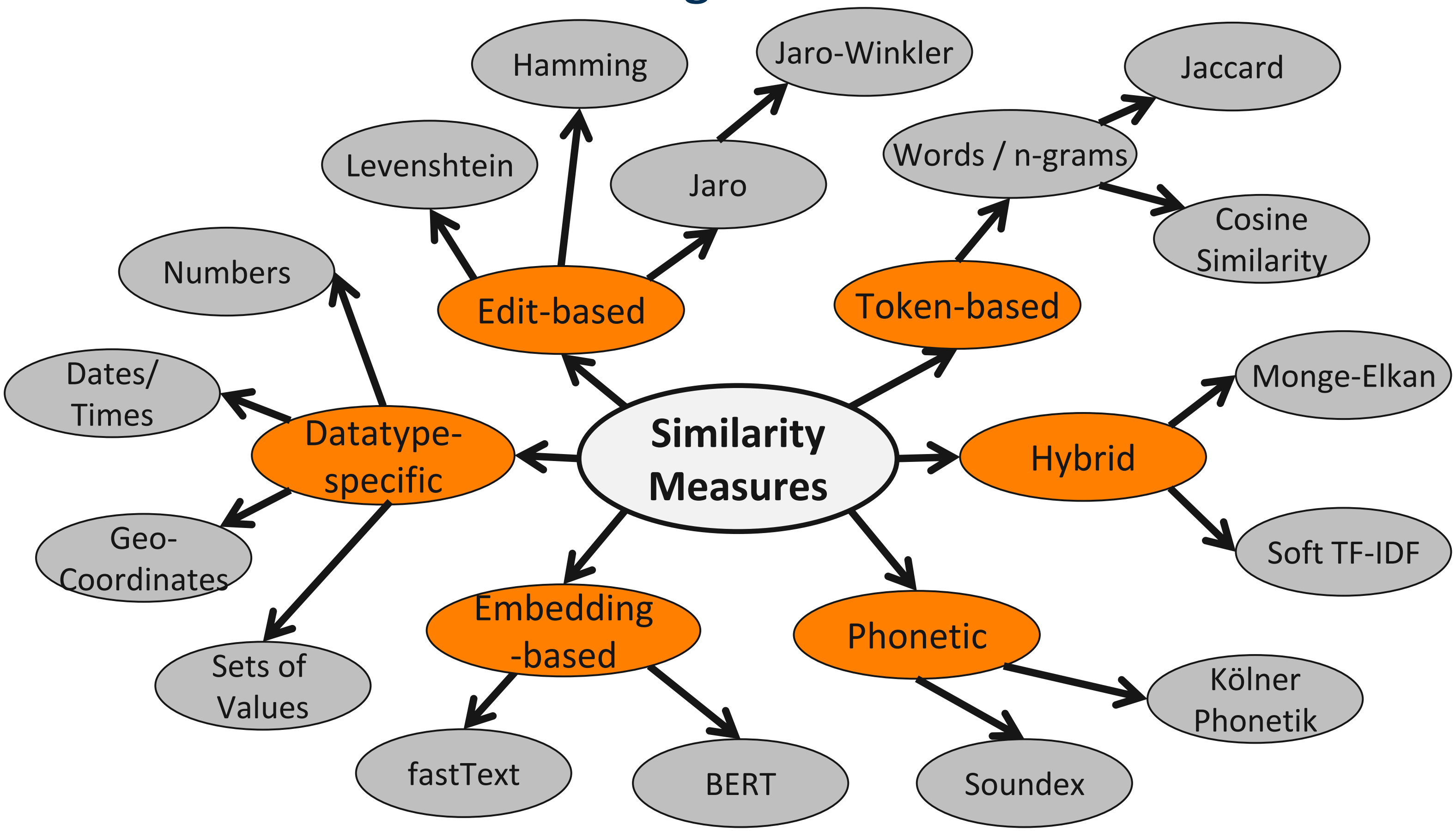
p and q are attribute values for two data objects

Levenshtein Distance

- Measures the dissimilarity of **two strings**
- Measures the **minimum number of edits** needed to transform one string into the other
- Allowed edit operations:
  - **Insert** a character into the string
  - **Delete** a character from the string
  - **Replace** one character with a different character
- Examples:
  - levensthein('table', 'cable') = 1 (1 substitution)
  - levensthein('Doe, Jane', 'Jane Doe') = 8 (7 substitution, 1 deletion)

Further String Similarity Measures

- See course: Web Data Integration



Proximity of Multidimensional Data Points

- All measures discussed so far cover the proximity of single attribute values
- But we usually have data points with many attributes
  - e.g., age, height, weight, sex...
- Thus, we need proximity measures for data points
  - See next slide

- Clustering approaches heavily depend on a similarity/distance between records
  - Attributes should be **normalized** so that all attributes can have **equal impact** on the computation of distances

Norms

- Euclidean Distance (L_2 - norm)
  - $dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$

Where n is the number of dimensions (attributes) and p_k and q_k are the k^th attributes of data points p and q

- More general (L_p - norm)
  - $dist = \sqrt[p]{\sum_{k=1}^{n} |p_k - q_k|^p}$

$$= \left(\sum_{k=1}^{n} |p_k - q_k|^p\right)^{\frac{1}{p}}$$

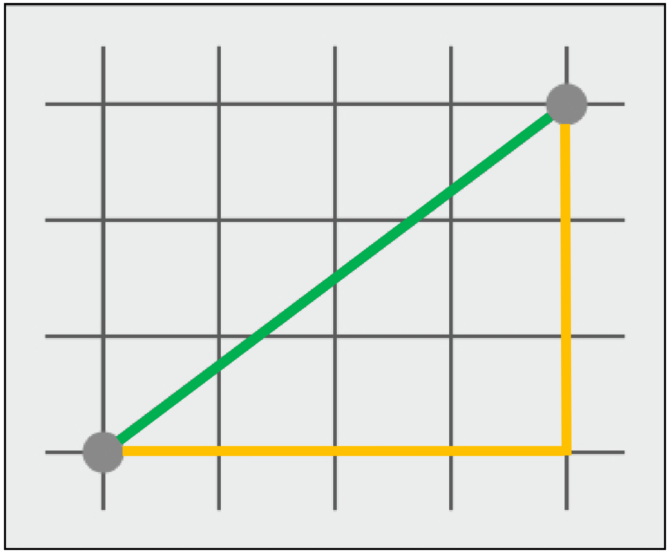
- Manhattan distance (L_1 - norm)
  - $dist = \sum_{k=1}^{n} |p_k - q_k|$
  - Minimum distance to go from one crossing to another
    - In a squared city (like Manhattan)

Or Mannheim;)

Example:

$L_2 = \sqrt{4^2 + 3^2} = 5$

$L_1 = 4+3=7$



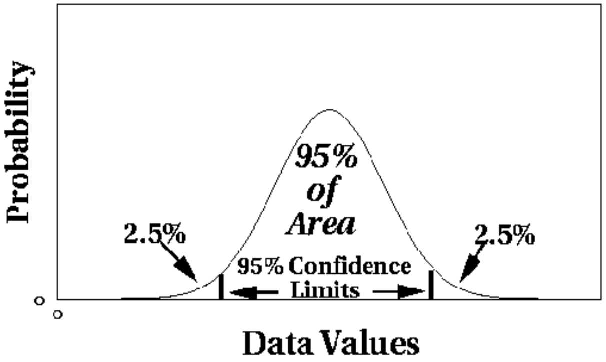
Anomaly Detection

- Also known as “Outlier Detection”
- Automatically identify data points that are somehow different from the rest
- Working assumption:
  – There are considerably more “normal” observations than “abnormal” observations (outliers/anomalies) in the data
- Methods:
  – Statistical Approaches (IQR, MAD)
  – Distance-based Approaches
  – Density based Approaches
  – Clustering based

Statistical Approaches

- Assume a parametric model describing the distribution of the data (e.g., normal distribution)

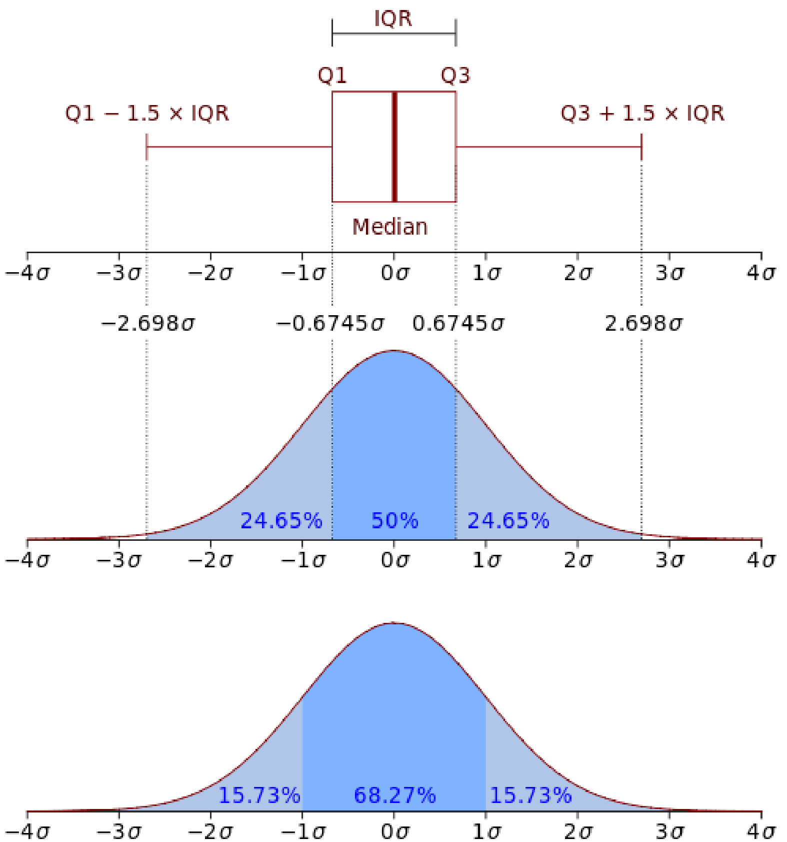
- Apply a statistical test that depends on
  - Data distribution
  - Parameter of distribution (e.g., mean, variance)
  - Number of expected outliers (confidence limit)



Interquartile Range (IQR)

- Divides data in quartiles
  - Assumes a normal distribution
- Definitions:
  - Q1: x ≥ Q1 holds for 75% of all x
  - Q3: x ≥ Q3 holds for 25% of all x
  - IQR = Q3-Q1
- Outlier detection:
  - All values outside [Q1-1.5*IQR ; Q3+1.5*IQR]
- Example:
  - 0,1,1,3,3,5,7,42 → median=3, Q1=1, Q3=7 → IQR = 6
  - Allowed interval: [1-1.5*6 ; 7+1.5*6] = [-8 ; 16]

Thus, 42 is an outlier



https://en.wikipedia.org/wiki/File:Boxplot_vs_PDF.svg

Median Absolute Deviation (MAD)

- MAD is the median deviation from the median of a sample, i.e.

$$\tilde{X} = median(X)$$
$$MAD = median(|X_i - \tilde{X}|)$$

- MAD can be used for outlier detection
  - All values that are k*MAD away from the median are considered to be outliers
    - E.g., k=3
- Example:
  - X =0,1,1,3,5,7,42 → $\tilde{X} = 3$
  - $|X_i - \tilde{X}|$ (Deviations): 3,2,2,0,2,4,39
  - Deviations sorted: 0,2,2,**2**,3,4,39→ MAD = 2
  - Allowed interval: [3-3*2 ; 3+3*2] = [-3;9]

Thus, 42 is an outlier

Carl Friedrich Gauss, 1777-1855

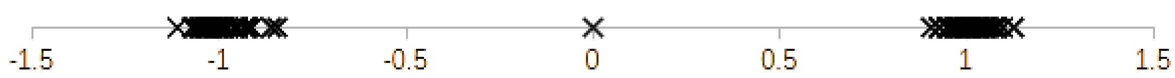
Outliers vs. Extreme Values

- So far, we have looked at extreme values only
  - But outliers can occur as non-extremes
  - In that case, methods like IQR fail
- IQR on the example below:
  - Q2 (Median) is 0
  - Q1 is -1, Q3 is 1

    → IQR = 2

    → everything outside [-4,+4] is an outlier

    → there are no outliers in this example



Distance-based Approaches

- Nearest-neighbor based
  - Compute the distance between every pair of data points
  - There are various ways to define outliers:
    - Data points for which there are fewer than p neighboring points within a distance D
    - The top n data points whose distance to the k^{th} nearest neighbor is greatest
    - The top n data points whose average distance to the k nearest neighbors is greatest

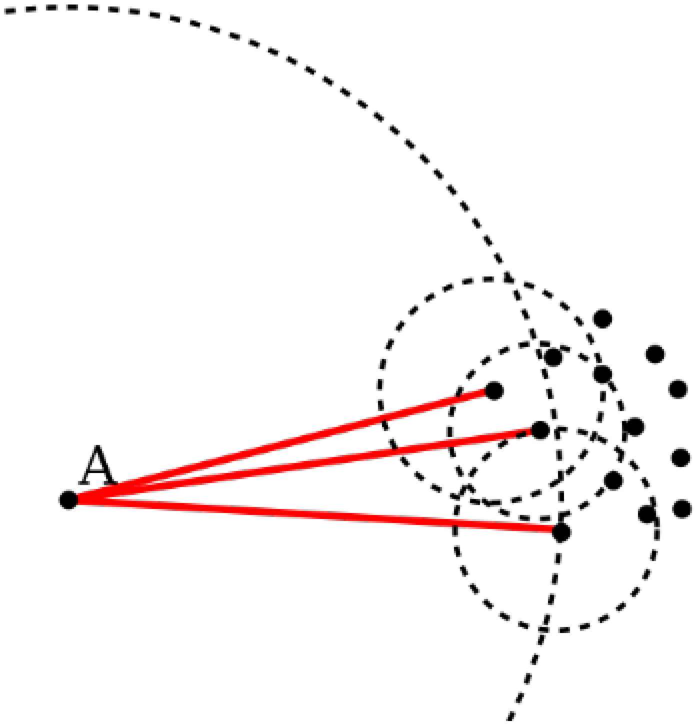
Package PyOD

Density-based: LOF approach

- For each point, compute the density of its local neighborhood
  - If that density is higher than the average density, the point is in a cluster
  - If that density is lower than the average density, the point is an outlier
- Compute local outlier factor (LOF) of a point A
  - Ratio of average density of A's neighbors to density of point A
- Outliers are points with large LOF value
  - Typical: larger than 1

LOF: Illustration

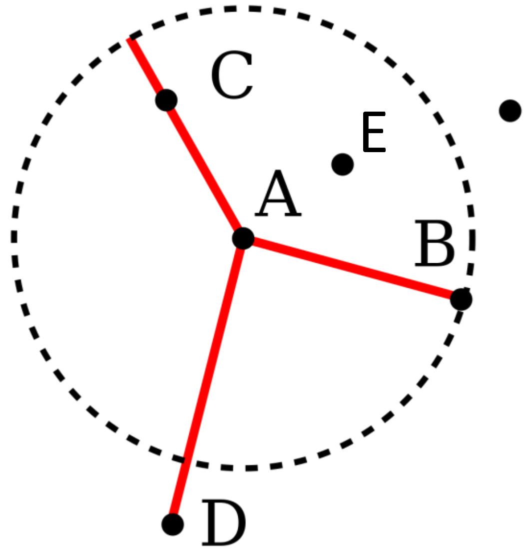
- Using 3 nearest neighbors
- We compute
  - the average density of A
  - the average density of A's neighbors
- If the density of A is lower than the neighbors' density
  - A might be an outlier



http://commons.wikimedia.org/wiki/File:LOF-idea.svg

LOF: Defining Density

- LOF uses a concept called “reachability distance”
- All points within the k-neighborhood have the same k-distance
  - In the example: $d_3(A,B) = d_3(A,C) = d_3(A,E)$
- Reachability distance $rd_k(A,B)$:
  - Distance of A,B, lower bound by $d_k(B)$
  - $rd_k(A,B) = \max(d_k(B), distance(A,B))$
- In the example:
  - $rd_k\ (D,A) = d(D,A)$, but
  - $rd_k\ (C,A) =$ k-distance(A)
- Rationale: all sufficiently close points are regarded as equally close
  - lessens the impact of small variations

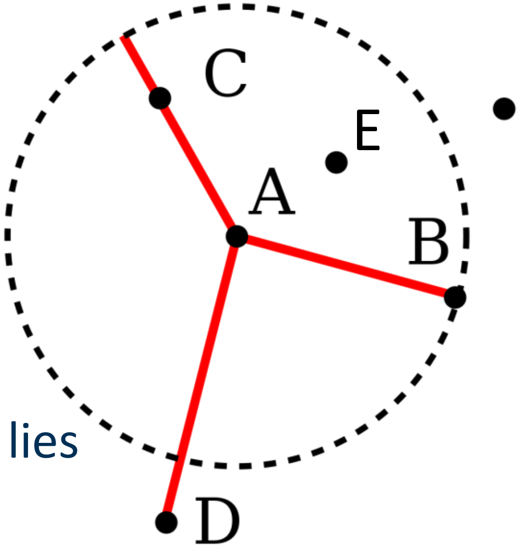


https://commons.wikimedia.org/wiki/File:Reachability-distance.svg

LOF: Defining Density

- Average reachability distance
  - $avgrd_k(A) = \frac{\sum_{P:k\ nearest\ neighbors\ of\ A} rd_k(A,P)}{|N_k(A)|}$

no. of k nearest neighbors of A, usually =k



- Density is defined as the inverse
  - Idea: the larger the avg. reachability distance, the sparser the region in which the data point lies
  - *Local reachability density* $lrb_k(A) = 1/avgrd_k(A)$

- Local outlier factor: relation of density of A's neighbors to A's density:

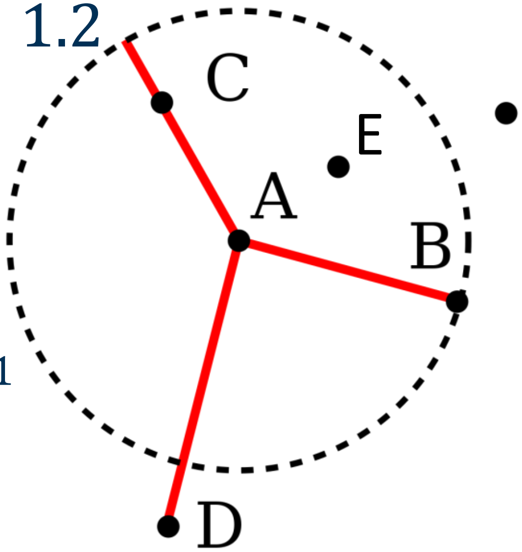
$$LOF_k(A) = \frac{\sum_{P:k\ nearest\ neighbors\ of\ A} \frac{lrb_k(P)}{lrb_k(A)}}{|N_k(A)|} = \frac{\sum_{P:k\ nearest\ neighbors\ of\ A} lrb_k(P)}{|N_k(A)| * lrb_k(A)}$$

https://commons.wikimedia.org/wiki/File:Reachability-distance.svg

LOF: Example

- $d(A,B) = 1,\ d(A,C) = 0.75,\ d(A,D) = 1.2$
  - $rd_k(A,B) = rd_k(A,C) = rd_k(A,E) = 1$
  - $rd_k(A,D) = 1.2$
- Average reachability:
  - $avgrd_k(A) = \frac{\sum_{P:k\ nearest\ neighbors\ of\ A} rd_k(A,P)}{|N_k(A)|} = \frac{1+1+1}{3} = 1$
- Density $lrb_k(A) = \frac{1}{avgrd_k(A)} = 1$
- Let's assume: $avgrd_k(B) = 0.8, avgrd_k(C) = 1.2, avgrd_k\,(E) = 0.6$
  - $lrb_k(B) = 1.25\,, lrb_k(C) = 0.83\,, lrb_k(E) = 1.67$
- Local outlier factor of A:

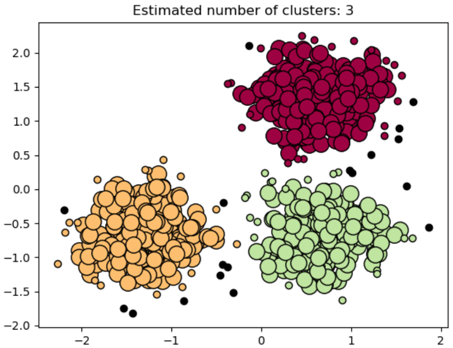
$$LOF_k(A) = \frac{\sum_{P:k\ nearest\ neighbors\ of\ A} lrb_k(P)}{|N_k(A)| * lrb_k(A)} = \frac{1.25 + 0.86 + 1.67}{3 * 1} = 1.26$$



>1 ⟶ outlier

DBSCAN for Outlier Detection

- DBSCAN directly identifies noise points
  - These are outliers not belonging to any cluster
    - In scikit-learn: label -1
  - Allows for performing outlier detection directly



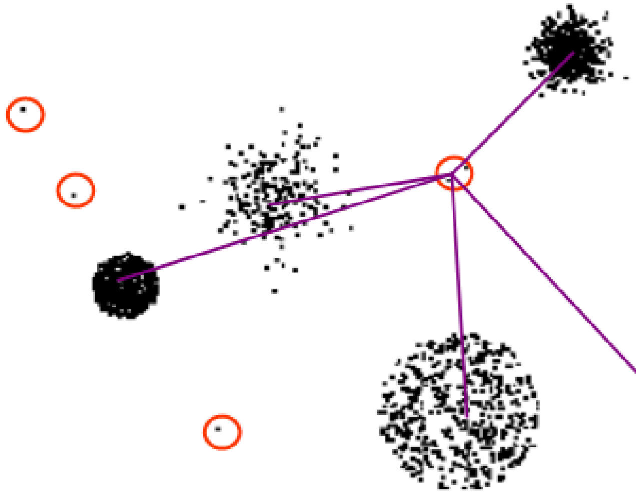

```
# Apply DBSCAN with eps=0.5 and min_samples=5
dbscan = DBSCAN(eps=0.1, min_samples=5)
dbscan.fit(X)

# Identify the noise points
noise_mask = dbscan.labels_ == -1
print(noise_mask)

# Remove the noise points from the dataframe
X = X[~noise_mask]
```

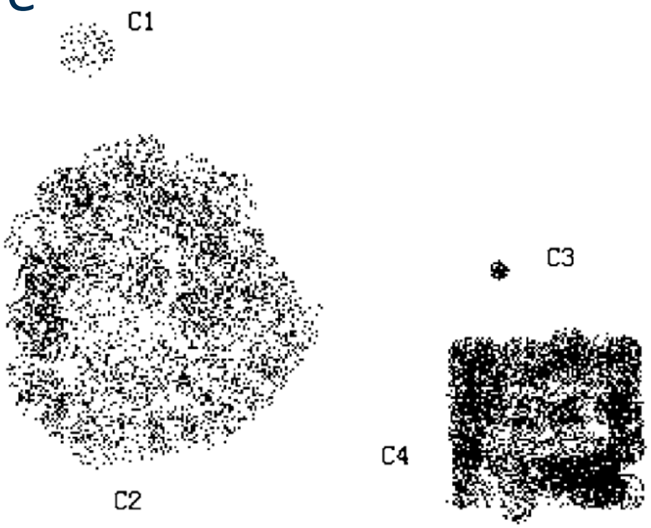
Clustering-based Outlier Detection

- Basic idea:
  - Cluster the data into groups of different density
  - Choose points in small cluster as candidate outliers
  - Compute the distance between candidate points and non-candidate clusters
  - If candidate points are far from all other non-candidate points, they are outliers



Clustering-based Local Outlier Factor

- Idea: anomalies are data points that are
  - In a very small cluster or
  - Far away from other clusters



- CBLOF is run on clustered data
- Assigns a score based on
  - The size of the cluster a data point is in
  - The distance of the data point to the next large cluster

Clustering-based Local Outlier Factor

- General process:
  - First, run a clustering algorithm (of your choice)
  - Then, apply CBLOF

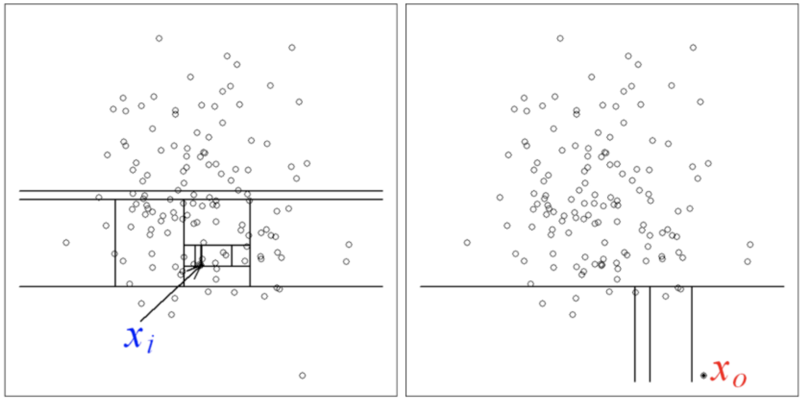
Package PyOD

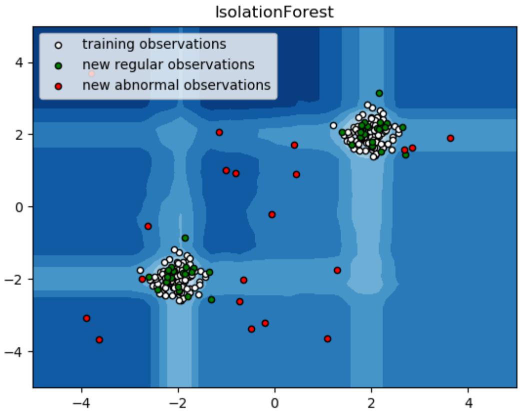
- Result: data points with outlier score

```
from sklearn.cluster import KMeans
from pyod.models.cblof import CBLOF
# clustering
clust = KMeans()
# outlier detection
detector = CBLOF(n_clusters=8,clustering_estimator=clust)
detector.fit(X)
# removal
noise_mask =detector.predict(X) == 1
X = X[~noise_mask]
```

Isolation Forests

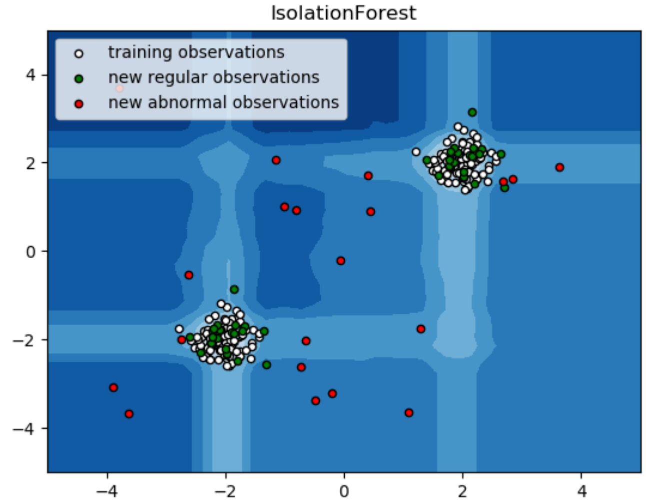
- Isolation tree:
  – A decision tree that has only leaves with one example each
- Isolation forests:
  – Train a set of random isolation trees
- Idea:
  – Path to outliers in a tree is shorter than path to normal points
  – Across a set of random trees, average path length is an outlier score





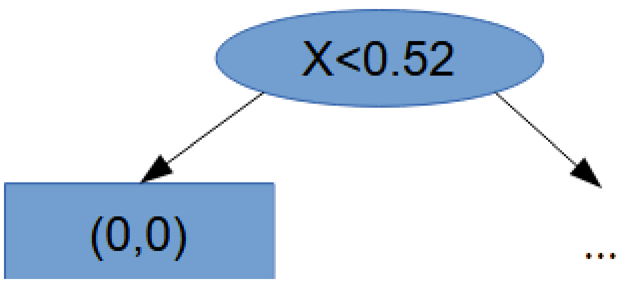
Isolation Forest

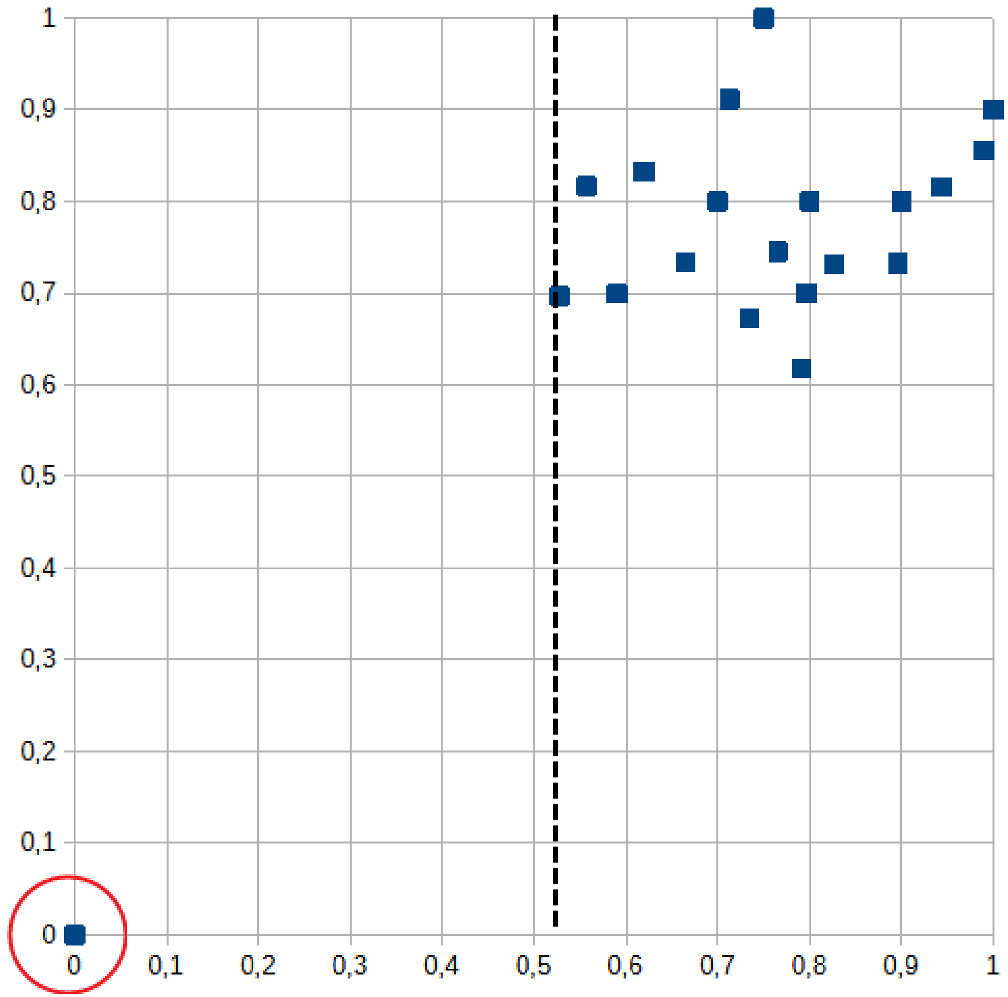
- Training a single isolation tree
  - For each leaf node w/ more than one data point
    - Pick an attribute Att and a value V at random
    - Create inner node with test Att<V
      - Train isolation tree for each subtree
- Output
  - A tree with just one instance per node
  - Usually, an upper limit on height is used



Isolation Forest

- Probability of (0,0) ending in a leaf at height 1
  - Pick Att X, pick V<0.52





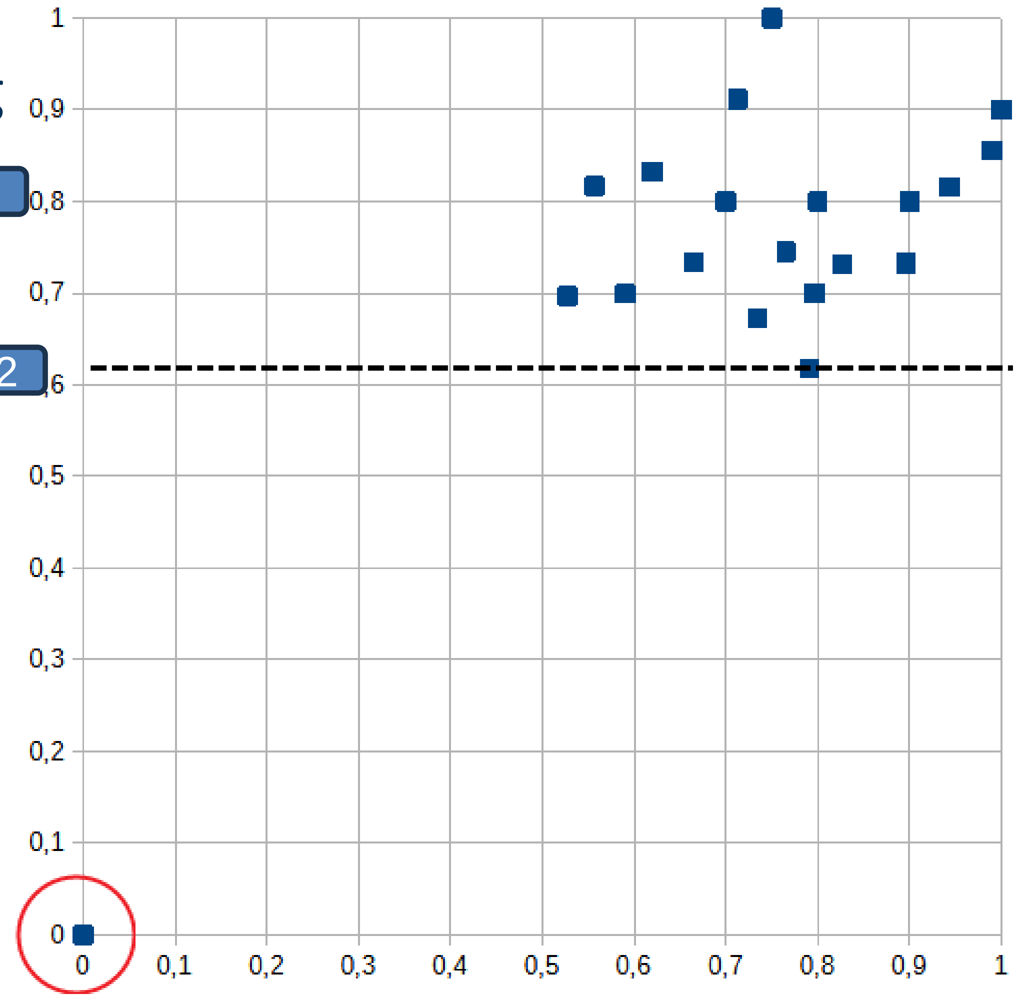
Isolation Forest

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Probability of (0,0) ending in a leaf at height 1
  - pick Att X, pick V<0.52, or
  - pick Att Y, pick V<0.62

0.5 0.52 0.5 0.62

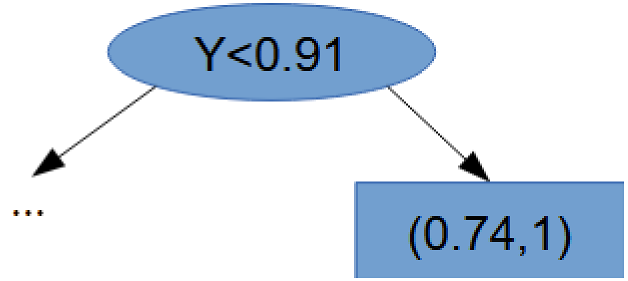
- 0.5*0.52 + 0.5*0.62
  → 0.57

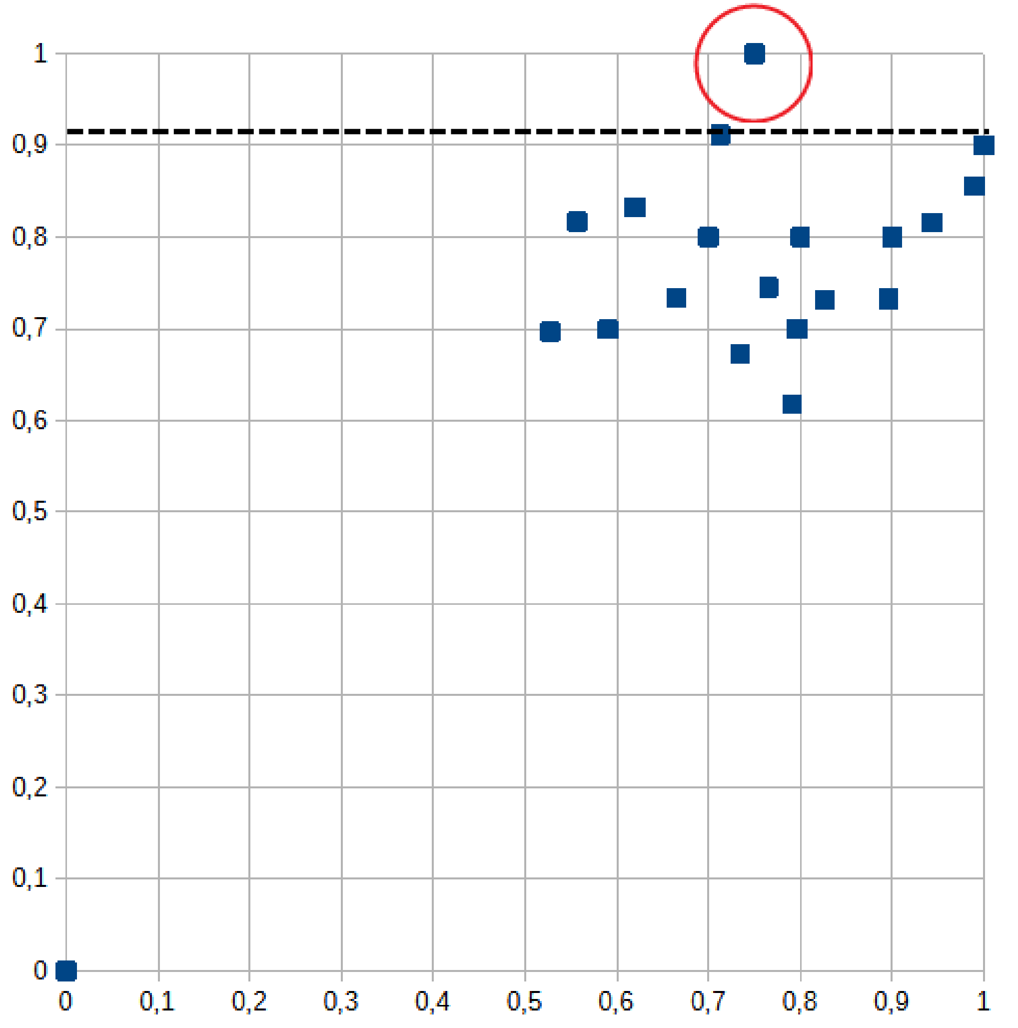


Isolation Forest

- Probability of (0.74,1) ending in a leaf at height 1
- Pick Att Y, pick V>0.91

- 0.5 * 0.09
  → 0.045

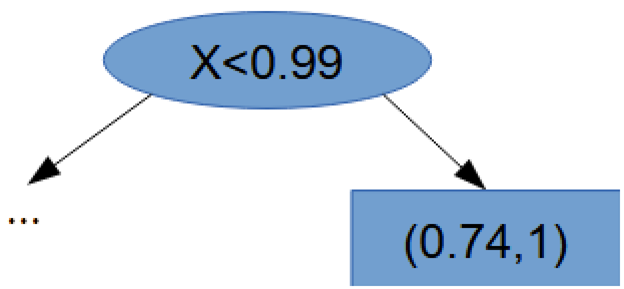


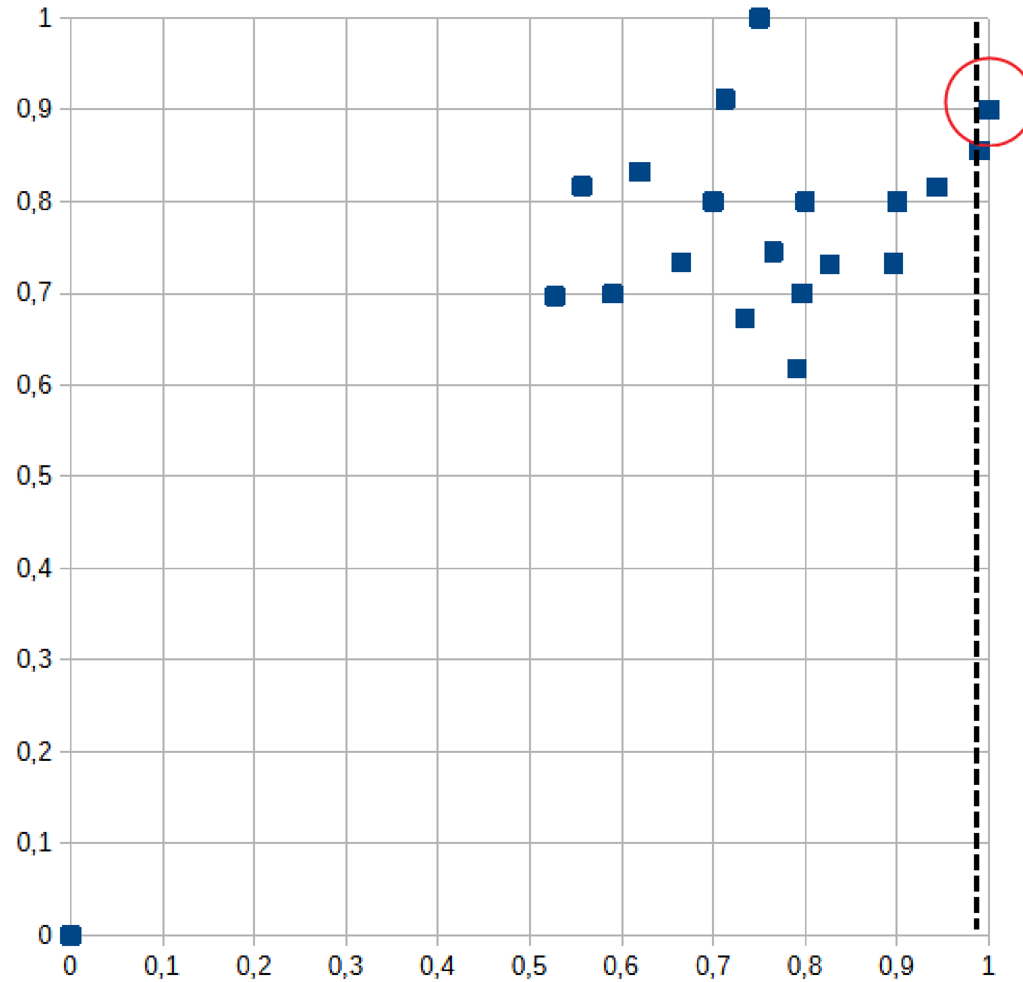


Isolation Forest

- Probability of (1,0.9) ending in a leaf at height 1
- Pick Att X, pick V>0.98

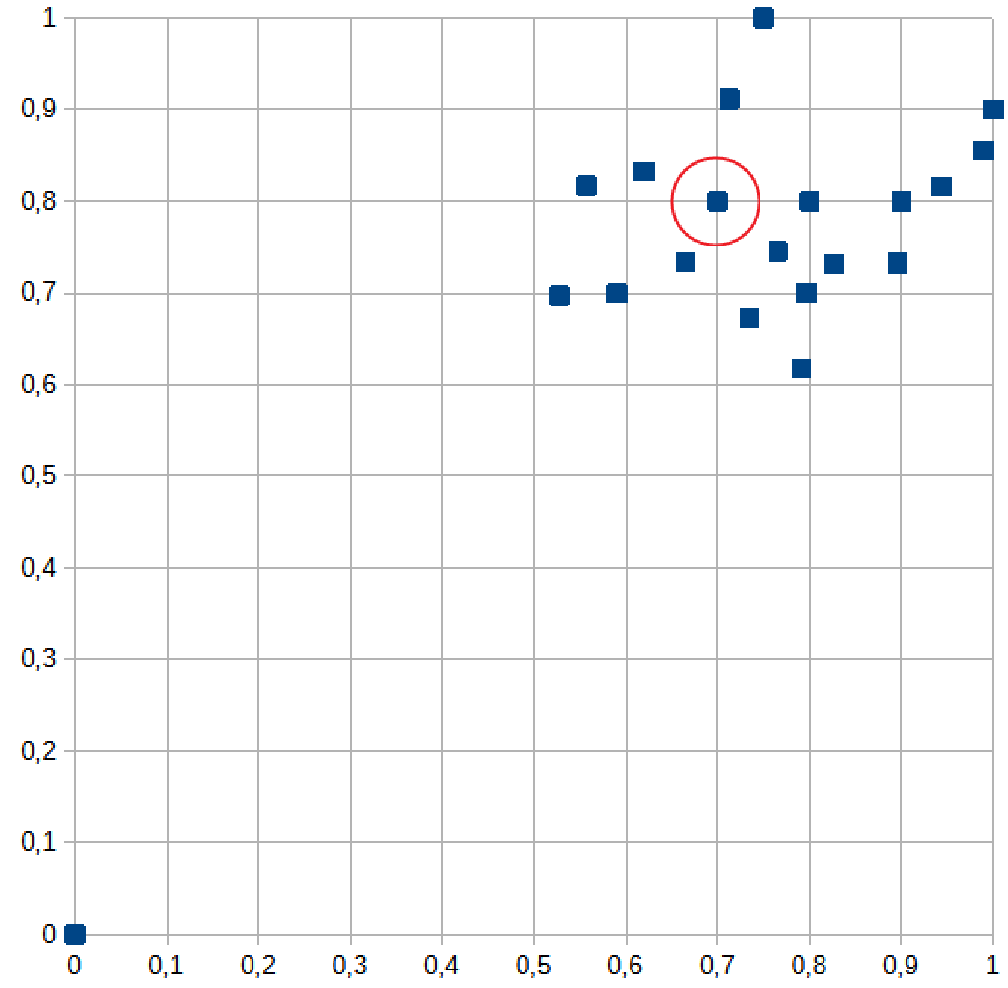
- 0.5 * 0.02
  → 0.01





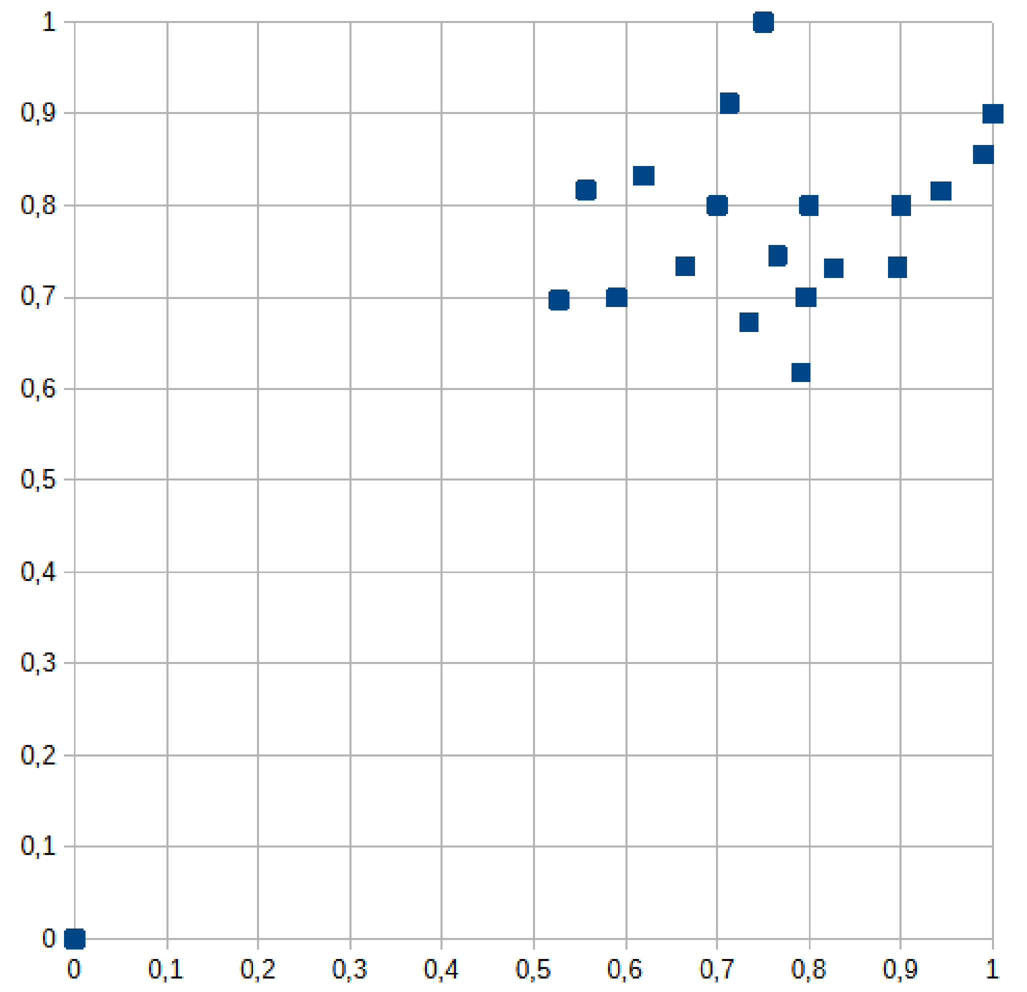
Isolation Forest

- Probability of any other data point ending in a leaf at height 1
  - This is not possible!
  - At least two tests are necessary



Isolation Forest

- Observations
  - Data points in dense areas need more tests
    - i.e., they end up deeper in the trees
  - Data points far away from the rest have a higher probability to be isolated earlier
    - i.e., they end up *higher* in the trees



Additional material

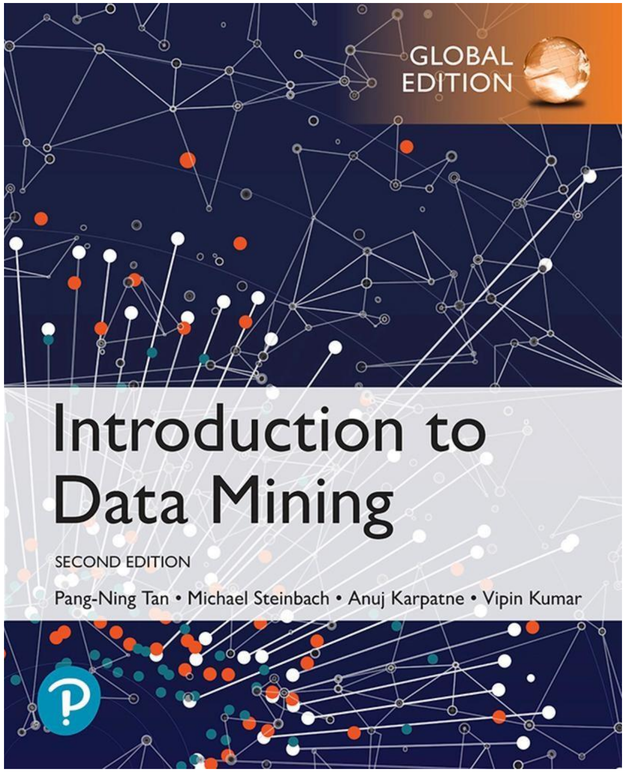
- This week additional material is about
  - Hierarchical Clustering

- Additional material is exercise and exam relevant

Questions?

Literature for this Slideset

- Pang-Ning Tan, Michael Steinbach, Karpatne,Vipin Kumar: Introduction to Data Mining. 2nd Edition. Pearson.



- Chapter 5: Cluster Analysis
  - Chapter 5.2: K-Means
  - Chapter 5.3: Agglomerative Hierarchical Clustering
  - Chapter 5.4: DBSCAN
- Chapter 2.4: Measures of Similarity and Dissimilarity