Overview

- **What is Knowledge Discovery and Data Mining?**
- **KDD Process**
- Supervised Learning
  - Classification
  - Prediction
- **Unsupervised Learning**
  - **Clustering**
  - **Outlier Detection**
- Frequent Pattern Mining
  - Frequent Item-sets

Definition: Knowledge Discovery in Databases

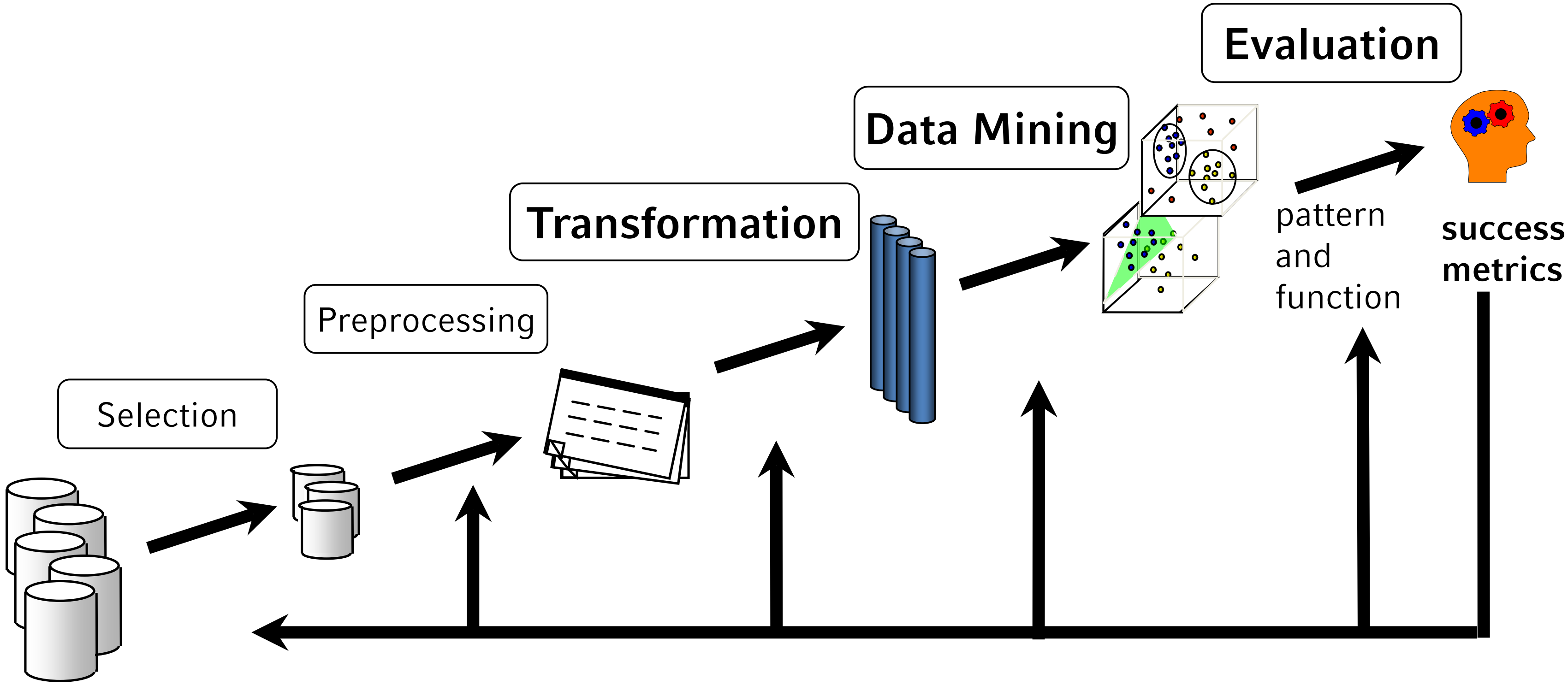
[Fayyad, Piatetsky-Shapiro & Smyth 1996]

Knowledge Discovery in Databases (KDD) is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

Remarks:

- *valid*: in a statistic sense.
- *novel*: not explicitly known yet,
  no common sense knowledge.
- p*otentially useful*: for a given application.
- *ultimately understandable*: the end user should be able to interpret the patterns either immediately or after some postprocessing

Knowledge Discovery Process



- Knowledge Discovery is a process comprising several steps.
- The KDD process is iteratively optimized (back arrow) until the result is acceptable.
- It is important what's the purpose of the analysis.

Steps of a KDD-Process

- **Selection**: Determining a clear objective and approach.
  Example: Use of a recording of TCP-Traffic to train a prediction model, which recognizes if a player is controlled by a bot.

- **Preprocessing**: Selection, integration and ensuring consistency of data to analyze.
  Example: Saving records of normal players' and bots' network traffic. Integration of data from several servers. Elimination of too short or useless records (permanently AFK).

Steps of a KDD-Process

- **Transformation**: Transforming data into an analyzable form.
  Example: Create a vector from average package rates, length and burstiness key-figures.

- **Data Mining**: Use efficient algorithms to derive statistically significant patterns and functions from transformed data.
  Example: Training of a neural network with examples for bots and human players, to predict a new record if it is a bot.

Steps of a KDD-Process

- **Evaluation**: test the quality of the patterns and functions gained from data mining.
  - Compare expected and predicted results.
    (Rate of error)
  - Manual evaluation by experts (Does the result make sense?)
  - Evaluation based on mathematical characteristics of patterns

  Example: Testing an independent set of test-recordings on how likely the neural network predicts a bot with more than 50% confidence.

- **Conclusion**:
  - If test results are unsatisfying, the process is adapted.
  - Adaption is possible in every step: more training data, different algorithms, different parameters, …

Unsupervised Learning

problem setting: only unlabeled objects/no classes or target values

tasks:

- find groups of similar objects. (Clustering)
- find uncommon objects. (Outlier Detection)
- find parts of objects which occur often (Pattern Mining)

pro:

- results are based on less assumptions
- no labeling required

con:

- measuring the results is often a problem (manual evaluation)
- more flexibility often implies more computational complexity
- correlating the result to the actual target is difficult without examples (how to guide the algorithm to achieve the goal of the process)

Example Applications

- **Clustering**: Determine typical tactics for a particular boss encounter.

- **Outlier Detection**: Which player might cheat?

- **Pattern Mining**: Determine standard rotations of ability usage.

Clustering Methods

- identify a finite set of clusters or groups

- *similar objects should be part of the same cluster whereas dissimilar objects should be part of different clusters*

- clustering comprises finding the clusters and assigning new objects to these clusters

Clustering (formal view)

given:

- *dataset* $DB \subseteq F$ *(F is a feature space)*
- $C \subseteq IN_0$ a discrete target variable (cluster id)
- sometimes the number of clusters *|C| is assumed to be known*

goal: find function *f*: $F \rightarrow C$ *assigning objects to clusters.*

find reasonable clusters (e.g. Minimize intra cluster distance and maximize distance between clusters)

quality of a clustering:

- depends on the cluster model:
  - How is an object assigned to a cluster?
  - How is decided whether two objects belong to the same cluster?
- optimize:
  - compactness of clusters
  - cluster separation

Partitioning Clustering(1)

idea:

- there are k clusters and each cluster c *is represented by* o_c
- object o is assigned to c by the distance $dist(o_c, o)$:
  $$cluster(o) = \arg\min_{c \in C} \left(dist(o_c, o) \right)$$
- to achieve compact clusters minimize:
  - distance of objects to the closest cluster representation:
    $$compact(c) = \sum_{o \in \{o \in DB \mid cluster(o) = c\}} dist(o_c, o)$$
  - squared distance to the closest cluster representation:
    $$sqrComp(c) = \sum_{o \in \{o \in DB \mid cluster(o) = c\}} dist(o_c, o)^2$$
- Quality of the clustering :
  $$TD(C) = \sum_{c \in C} compact(c)$$
  $$TD^2(C) = \sum_{c \in C} sqrComp(c)$$

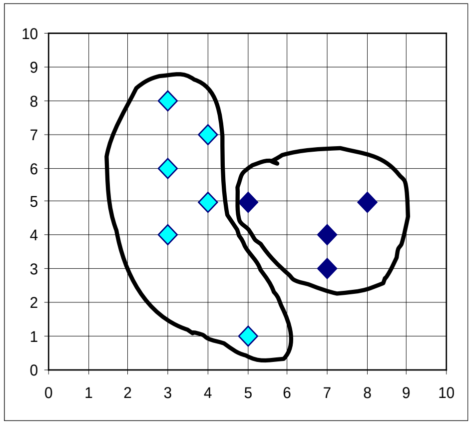
Partitionierendes Clustering (2)

- typical cluster representations:
  - centroid: $centroid(c) = \frac{1}{\left|\{o \in DB \mid cluster(o) = c\}\right|} \sum_{o \in \{o \in DB \mid cluster(o) = c\}} o$
  - medoid: $medoid(c) = \underset{o \in \{o \in DB \mid cluster(o) = c\}}{\arg\min} \left(\sum_{p \in \{p \in DB \mid cluster(o) = c\}} dist(o, p) \right)$

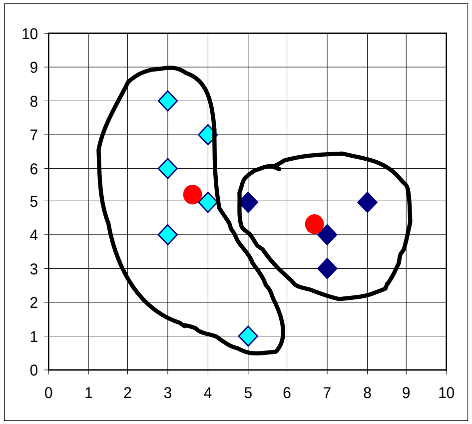
minimize TD or TD²:

- *TD* and TD^2 are not convex and might have multiple local minima
- *TD* and TD^2 are discontinuous (e.g. when switching clusters)
- apply greedy search to minimize TD/ TD^2
  1. Step: for all o $\in$ DB cluster(o) is known
     => compute cluster representations $\{o_{c1}, \ldots, o_{cn}\}$
  2. Step: given the cluster representation $\{o_{c1}, \ldots, o_{cn}\}$
     => assign all objects to their closest clusters and go to step 1
  3. terminate if TD/ TD^2 do not change (no cluster switch => local minimum)

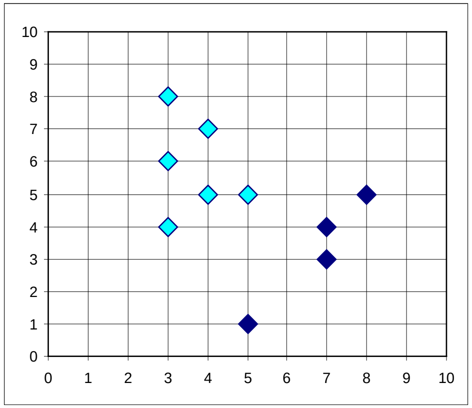
Example: Partitioning Clustering



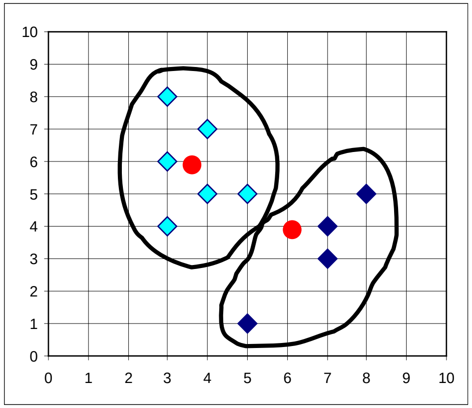
compute centroids



assign data objects



compute centroids



Algorithm

```
ClusteringVarianceMinimization(Objectset DB, Integer k)
    build initial clustering by splitting DB into k Cluster;
    compute representatives C'={C_1, ..., C_k}
    C = {};
    TD2 = sqrTD(C',DB);
    repeat
        TD2old = TD2;
        C = C';
        build k clusters by assigning each object to the next
          centroid in C;
        compute the new representatives C'={C'_1, ..., C'_k};
        TD2 = sqrTD(C',DB);
    until TD2 == TD2old;
    return C;
```

Partitioning Clustering

variants:

- *k*-Means: update a single object and then re-compute affected centroids.
- Expectation Maximization Clustering (EM)

  cluster=density distribution, Bayesian model, soft-clustering
- k-Medoid Clusterings:
  - cluster representations are mediods
  - cluster adaption is done by switching objects and medoids

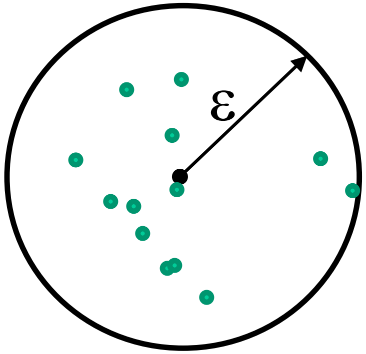
properties:

- all algorithms depend on the initialization
- centroid-based are very efficient $O(i \cdot n \cdot k)$. (#Iterations i)
- medoid-based are generic but slow $O(i \cdot n^2 \cdot k)$ (#Iterations i)

Density-Based Clustering

idea: Clusters are dense regions in feature space F.

density: $$\frac{|\text{ objects }|}{\text{volume}}$$



here:

- ***volume***: ε-neighborhood for object o w.r.t. distance measure *dist(x,y)*
- ***dense region:*** ε-neighborhood contains MinPts objects

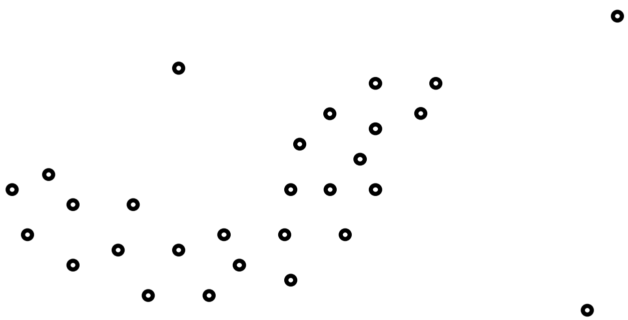
  => o is called core point
- „connected" core points form ***clusters***
- Objects outside cluster is considered ***noise***

Density-Based Clustering

intuition

parameters $\varepsilon \in IR$ and $MinPts \in IN$ *specify the density threshold*

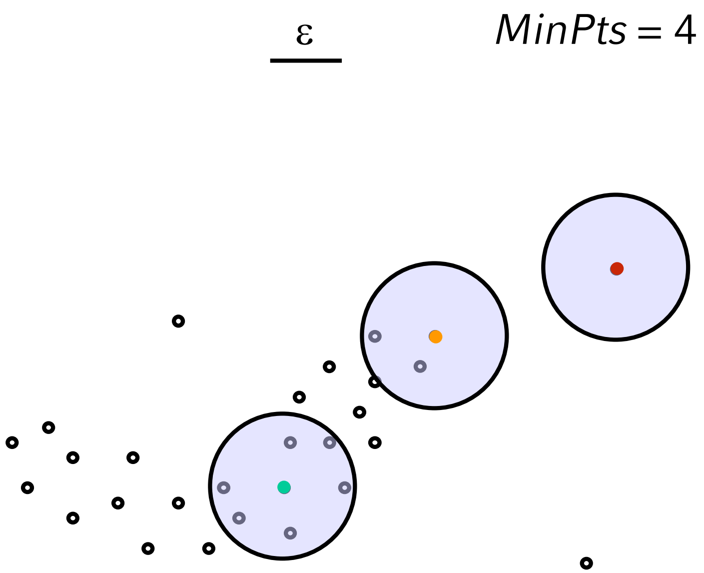
ε $MinPts = 4$



Density-Based Clustering

intuition

parameters $\varepsilon \in IR$ and $MinPts \in IN$ specify density threshold



- *core points*

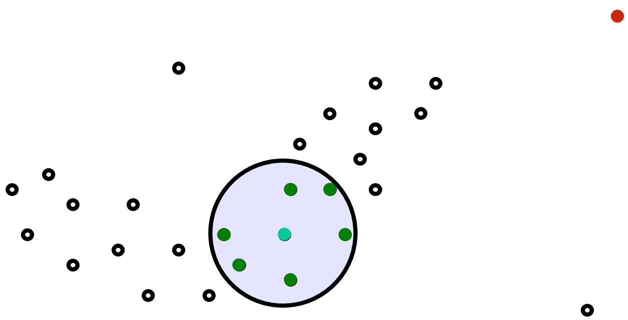
Density-Base Clustering

intuition

parameters $\varepsilon \in IR$ and $MinPts \in IN$ specify density threshold

ε $MinPts = 4$

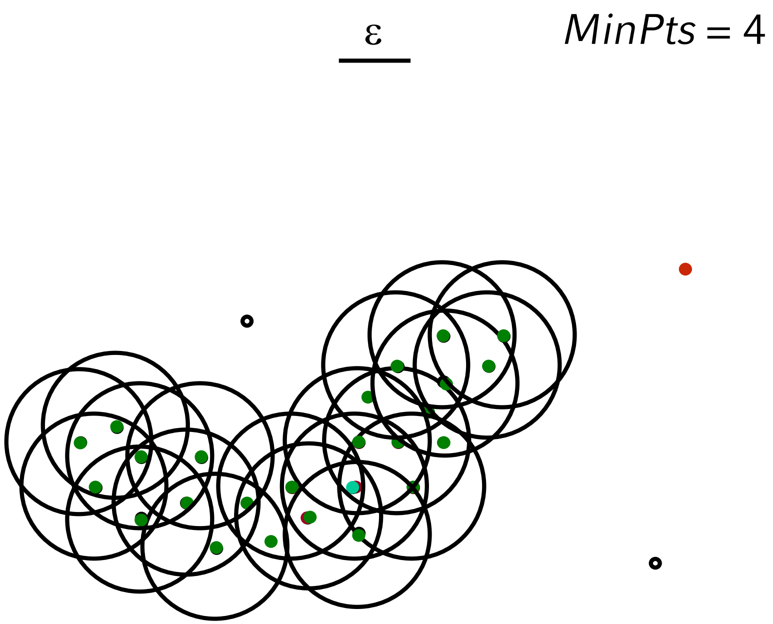
- *core points*
- *direct density-reachability*



Density-Based Clustering

intuition

parameters $\varepsilon \in IR$ and $MinPts \in IN$ specify density threshold

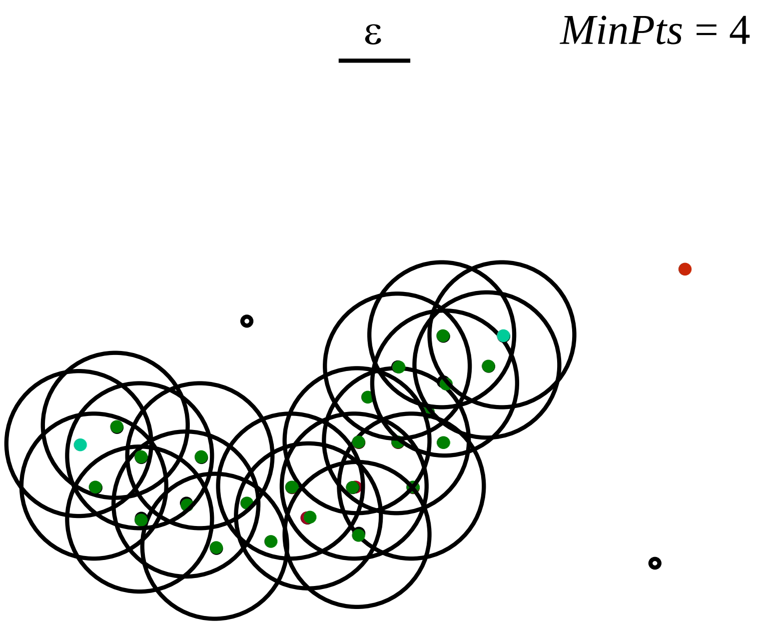


- *core points*
- *direct density-reachability*
- *density reachability*

Density-Based Clustering

intuition

parameters $\varepsilon \in IR$ and $MinPts \in IN$ specify density threshold

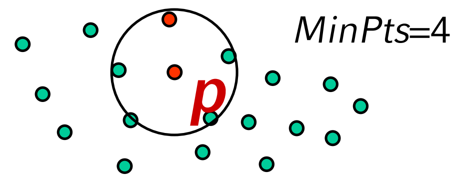


- *core points*
- *direct density-reachability*
- *density reachability*
- *density connectivity*

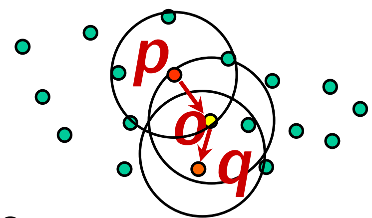
Density-Based Clustering

formal: [Ester, Kriegel, Sander & Xu 1996]

- Object $p \in DB$ is a core object, if:
  - $|RQ(p,\varepsilon)| \geq MinPts$
  - $RQ(p,\varepsilon) = \{o \in DB \mid dist(p,o) <= \varepsilon\}$

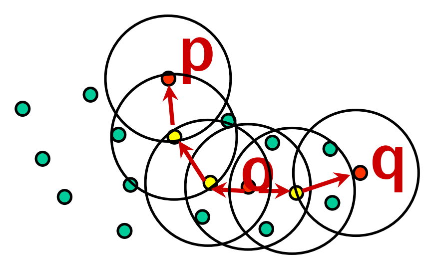


- Object $p \in DB$ is direct density reachable from $q \in DB$ wr.t. ε and *MinPts*, if:
  $p \in RQ(q,\varepsilon)$ and q is a core object in *DB*.



- Object *p* is *density-reachable* from object *q*, if there is a sequence of direct density reachable objects from q to p.

- Two objects *p* and *q* are density-connected, if both p and q are density reachable from a third object o.



Density-Based Clustering

formal:

A density-based cluster C w.r.t. ε and *MinPts* is a none-empty subset of *DB with the following properties:*

Maximality: $p,q \in DB$: $p \in C$ and q *is* density-reachable from $p \Rightarrow q \in C$.

Connectivity: $p,q \in C \Rightarrow p$ *and* q are density-connected.

Density-Based Clustering

formal

- Clustering
  A density-based *clustering* *CL* of *DB* w.r.t. ε and *MinPts* is the complete set of all density-based clusters w.r.t. ε and *MinPts.*

- Noise
  The set $Noise_{CL}$ is defined as the subset of objects in DB which are not contained in any cluster.

- idea behind the DBSCAN algorithm
  Let C be a density-based cluster and let $p \in C$ be a core object, then
  $C = \{o \in DB \mid o$ density reachable from p w.r.t. ε and $MinPts\}$.

Algorithmus DBSCAN

```
DBSCAN(dataset DB, Real ε, Integer MinPts)
  // in beginning all objects are unlabeled,
  // o.ClId = UNLABELED for all o ∈ DB

   ClusterId := nextId(NOISE);
   for i from 1 to |DB| do
      Objekt := DB.get(i);
      if Objekt.ClId = UNLABELED then
         if ExpandCluster(DB, Objekt, ClusterId, ε, MinPts)
         then ClusterId:=nextId(ClusterId);
```

Density-Based Clustering

```
ExpandCluster(DB, startObject, clusterId, ε, MinPts): Boolean
seeds:= RQ(startObject, ε);
if |seeds| < MinPts then // startObject is not a core object
  startObject.ClId := NOISE;
  return false;
// else: startObject is a core object
forall o ∈ seeds do o.ClId := clusterId;
remove startObject from seeds;
while seeds ≠ Empty do
  select object o from seeds;
  neighborhood := RQ(o, ε);
  if | neighborhood | ≥ MinPts then // o is a core object
    for i from 1 to | neighborhood | do
         p := neighborhood.get(i);
         if p.ClId in {UNLABELED, NOISE} then
                if p.ClId = UNLABELED then
                       add p to seeds;
                p.ClId := ClusterId;
  remove o from seeds;
return true;
```

Discussion Density-Based Clustering

- number of clusters is determined by the algorithm
- Parameters ε and MinPts generally less problematics
- Time complexity is $O(n^2)$ for general data objects
- Density-based methods only require a distance measure
- Border points make DBSCAN dependent on processing order
- No cluster model or parameter optimization
- Assigning new points is done with nearest neighbor classification

Outlier Detection

Hawkins' Definition [Hawkins 1980]:

"An outlier is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism."

What does „mechanism" mean?

- intuition from Bayesian statistics:

  "Outliers have a small likelihood to be generated by the assumed generative model."
- connection to clustering:
  - a clustering describes the distribution of data
  - outliers describe errors/noise

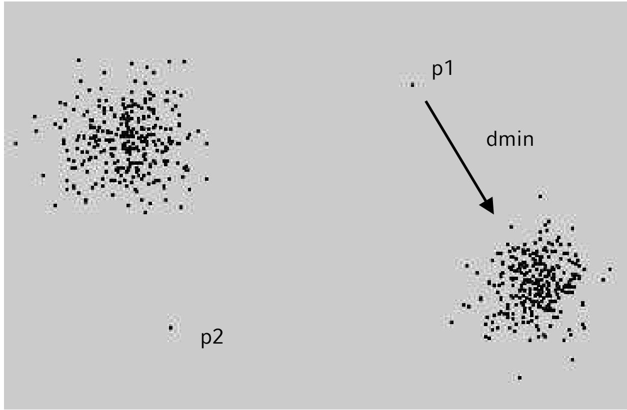
  $\Rightarrow$ max. distance to all cluster centers (part. clustering)

  $\Rightarrow$ noise in density-based clustering

Example: distance-based Outliers

- Definition „*(pct,dmin)*-Outlier" [Knorr, Ng 97]
  - An object *p* in data set *DB* is called *(pct,dmin)*-outlier, if at least *pct* -percent of the objects from *DB* have a larger distance to p then *dmin*.

- Selection of *pct* and *dmin* is left to the user



- <u>example:</u> $p_1 \in DB$, $pct=0.95$, $dmin=8$

- *p_1 is a (0.95,8)-outlier*
  => *95%* of objects in *DB* display a distance > *8* to *p_1*

Tutorial Exercise

- Implement **ClusteringVarianceMinimization** in Java

  - Use the code from the lecture web-page

  - Implement the *ClusteringVarianceMinimization.varianceMinimization* method

  - Test your Implementation with „gradlew test“ (or start the JUnit test case)