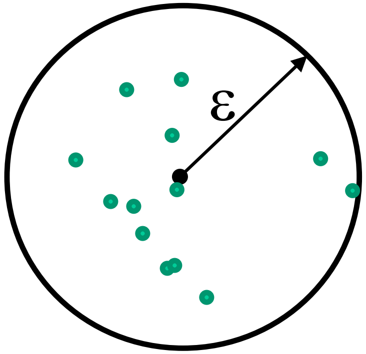
# Density-Based Clustering

**idea**: Clusters are dense regions in feature space F.

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




**here**:

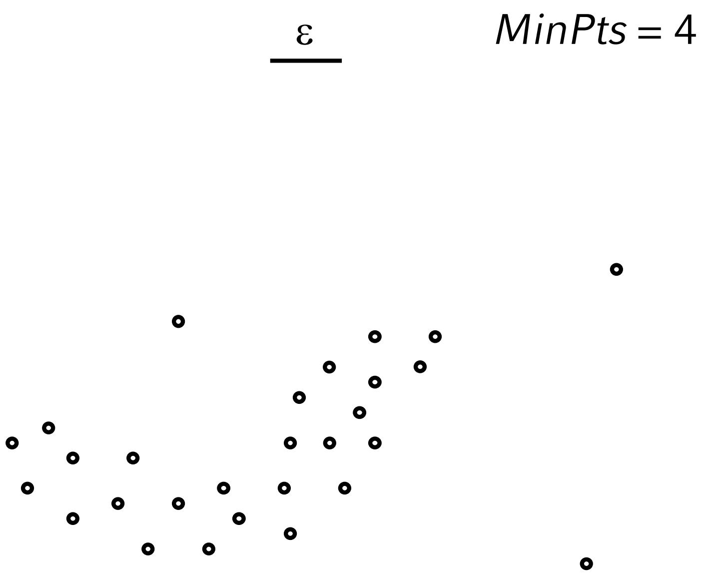
- ***volume***: ε-neighborhood for object o w.r.t. distance measure *dist(x,y)*
- ***dense region:*** ε-neighborhood contains MinPts objects

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

# Density-Based Clustering

## intuition

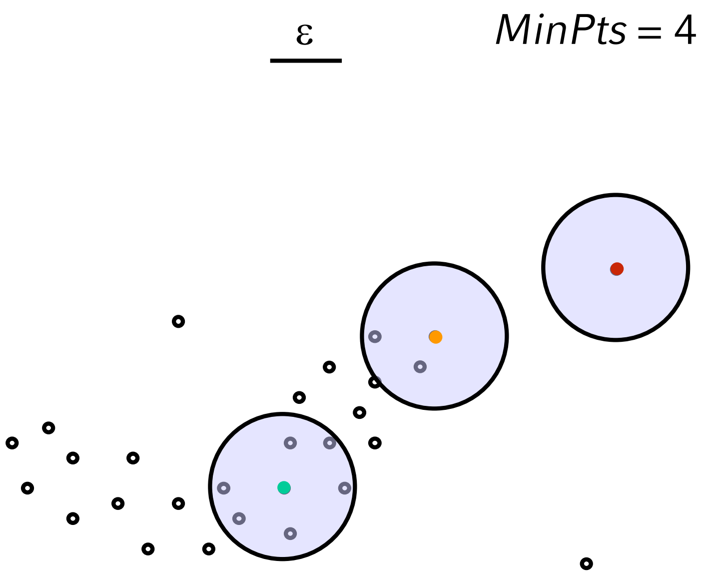
parameters $\varepsilon \in IR$ and $MinPts \in IN$ *specify the density threshold*



# Density-Based Clustering

## intuition

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




- *core points*

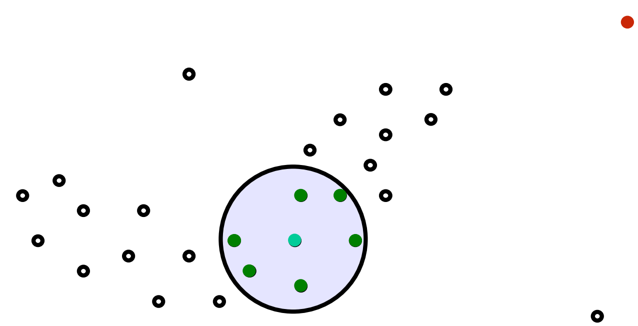
# Density-Base Clustering

## intuition

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

$\varepsilon$ $MinPts = 4$

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



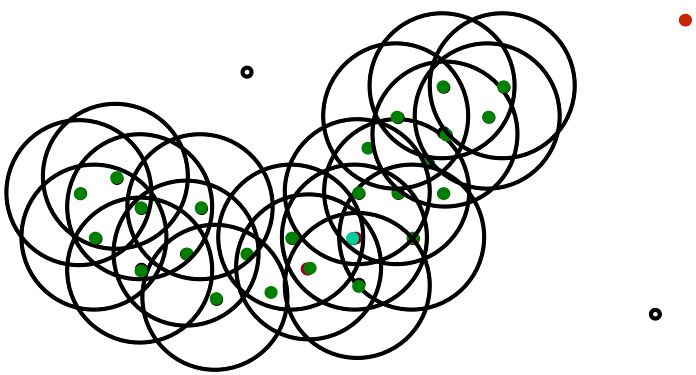
# Density-Based Clustering

## intuition

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

$\varepsilon$ $MinPts = 4$

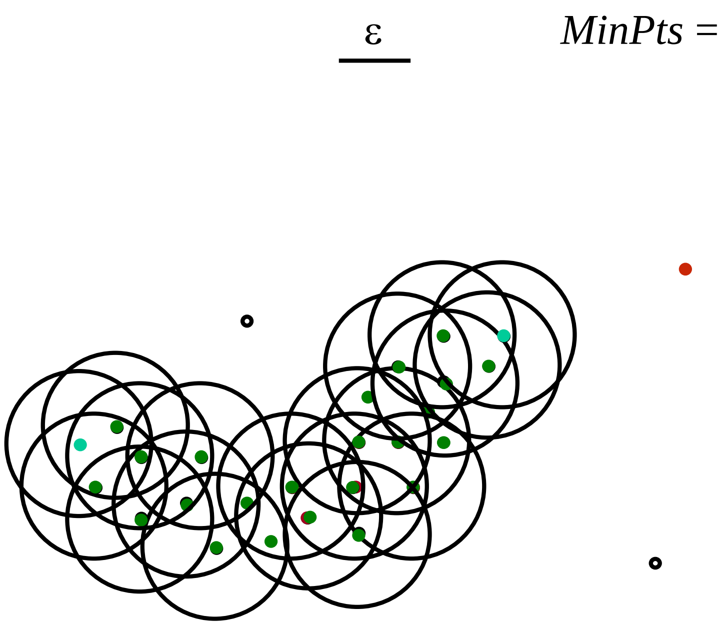
- *core points*
- *direct density-reachability*
- *density reachability*



# Density-Based Clustering

## intuition

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

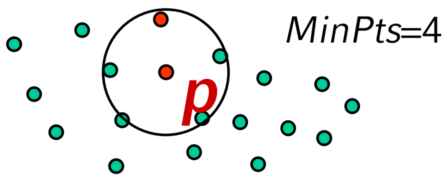



- *core points*
- *direct density-reachability*
- *density reachability*
- *density connectivity*

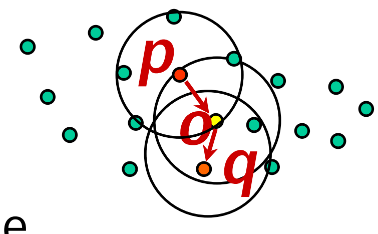
# Density-Based Clustering

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

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

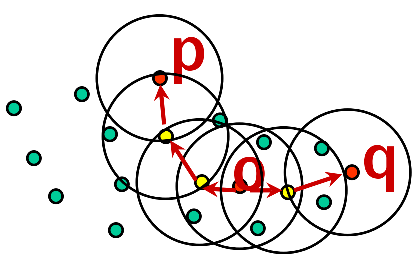



- Object $p \in DB$ is direct density reachable from $q \in DB$ wr.t. $\varepsilon$ and $MinPts$, if:
  $p \in RQ(q,\varepsilon)$ and $q$ is a core object in $DB$.




- 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. $\varepsilon$ 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. $\varepsilon$ and $MinPts$ is the complete set of all density-based clusters w.r.t. $\varepsilon$ 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. $\varepsilon$ and $MinPts\}$.

# Density-Based Clustering

## 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 $\varepsilon$ 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

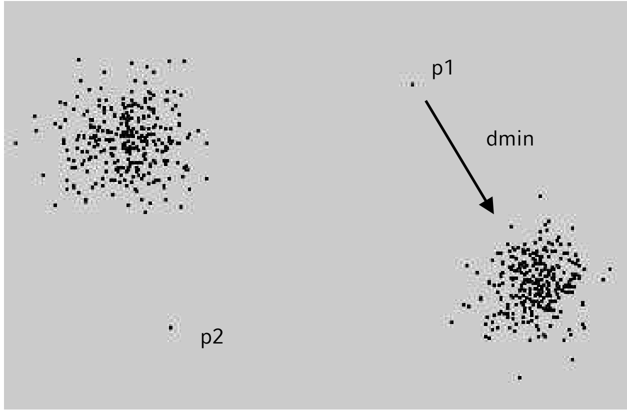
  $\Rightarrow$ max. distance to all cluster centers (part. clustering)

  $\Rightarrow$ 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 large 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 $p_1$

# Frequent Pattern Mining

**until now**: patterns describe subsets of DB

**idea**: patterns might be parts of object description

**here**: patterns are identical parts in object descriptions from large subsets of DB

**example**: group composition in MMORPG:

| | |
|---|---|
| Grp1 | priest, mage, druid, rogue, warrior |
| Grp3 | mage, mage, hunter, priest, warrior |
| Grp3 | Priest, priest, paladin, druid, warrior |
| Grp4 | warlock, paladin, shaman, warrior, priest |
| Grp5 | priest, warrior, hunter, rogue, warrior |

warrior, priest,?,?,?

# Directions in Frequent Pattern Mining

- **Frequent Itemset Mining** (=> KDD1 and in the following)
  objects are subsets of items
  => find frequent subsets
- **Frequent Substring Mining** (=> next chapter)
  data objects are string/sequences over a discrete alphabet
  => find frequent subsequences

- **Frequent Subgraph Mining** (=> KDD2)
  data objects are graphs with discrete labels for nodes and/or edges
  => find frequent groups of isomorphic subgraphs

- **Rule Mining**: find rules based on frequent patterns:
  *If pattern A is present in a data object, then its extension A+B is present with likelihood p.*

# Basic on Frequent Itemsets (1)

- **items** $I = \{i_1, \ldots, i_m\}$ is a set of literals

  e.g., all articles in a shop

- **itemset** $X$: subset of items $X \subseteq I$

  e.g., a set of items purchased together

- **transaction** *$T = (tid, X_T)$ where tid is the transaction ID and $X_T \subseteq I$ is the complete itemset in the transaction*

  e.g., all items being purchased together and the corresponding invoice number

- **database** $DB$: set of transactions $T$

  e.g., the set of all purchases in a certain time interval

- Items in itemsets can be sorted **lexicographical**:

  itemset $X = (x_1, x_2, \ldots, x_k)$, where $x_1 \leq x_2 \leq \ldots \leq x_k$

- **length of an itemset**: $|X|$: number of items in X

- ***k*-Itemset**: an itemset of length $k$

  {butter, bread, milk, sugar} is a 4-itemset
  {flour, sausage} is a 2-itemset

# Basics on Frequent Itemsets(2)

- **cover** of itemset $X$: set of transactions $T$ *containing* $X$ :

  $cover(X) = \{tid \mid (tid, X_T) \in DB,\ X \subseteq X_T\}$

- **support of itemsets** $X$ *in* $DB$: number of transactions containing X:

  $support(X) = |cover(X)|$

  remark: $support(\emptyset) = |DB|$

- **frequency of itemsets** $X$ *in* $DB$:

  relative frequency of $X$ being contained in any $T \in DB$ :

  $frequency(X) = P(X) = support(X) \,/\, |DB|$

- **frequent itemset** $X$ *in* $DB$:

  $support(X) \geq s$ $\qquad (0 \leq s \leq |DB|)$

  $s$ is an absolute threshold

  alternative: $frequency(X) \geq s_{rel}$ $\qquad$ where $s = \lceil s_{rel} \cdot |DB| \rceil$

# problem setting : Frequent Itemset Mining

**given**:

- a set of items $I$
- a transaction database $DB$ over $I$
- an absolute frequency threshold $s$

**task**: find all frequent itemsets in $DB$, i.e. $\{X \subseteq I \mid support(X) \geq s\}$

| TransaktionsID | Items |
|---|---|
| 2000 | A,B,C |
| 1000 | A,C |
| 4000 | A,D |
| 5000 | B,E,F |

support of 1-itemsets:

(A): 75%, (B), (C): 50%, (D), (E), (F): 25%,

support of 2-itemsets:

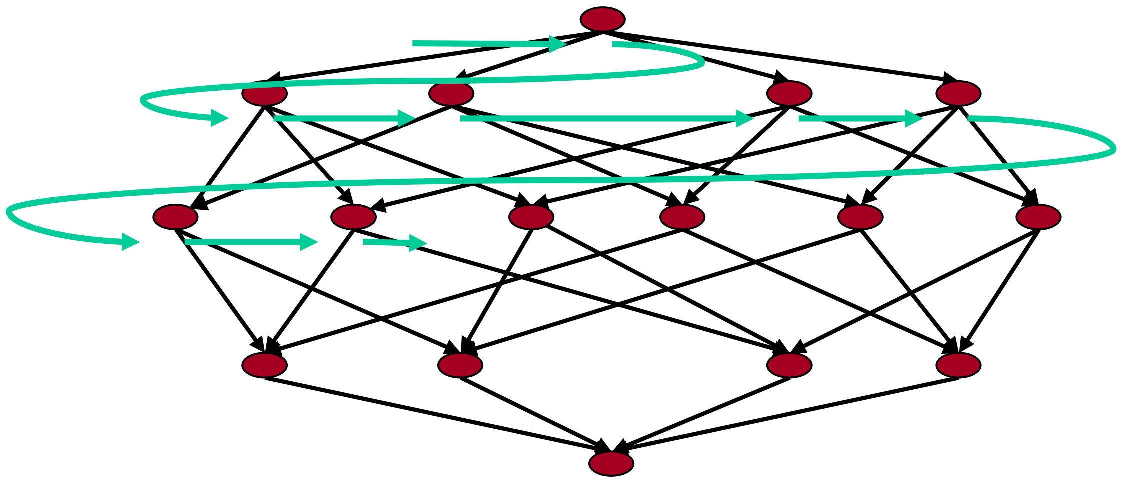
(A, C): 50%,

(A, B), (A, D), (B, C), (B, E), (B, F), (E, F): 25%

# Itemset Mining

## Apriori Algorithm [Agrawal & Srikant 1994]

- start by computing 1-itemsets, 2-itemsets and so on.
  (breadth-first search)



- find all frequent *k*+1-itemsets:
  - Consider only those *k*+1 itemsets for which all k-itemsets are frequent
- Determine support by scanning all transactions in DB (only one Scan)

# Apriori Algorithm

$C_k$: candidate set of potentially frequent items of length $k$

$L_k$: set of all frequent itemsets of length $k$

**Apriori**($I$, $DB$, $minsup$)

$L_1$ := {frequent 1-Itemsets aus $I$};

$k$ := 2;

**while** $L_{k-1} \neq \emptyset$ **do**

  $C_k$ := AprioriCandidateGeneration ($L_{k-1}$);

  **for each** Transaktion $T \in DB$ **do**

    $CT$ := Subset($C_k$, $T$);// all candidates $C_k$ contained in T

    **for each** candidate $c \in CT$ **do**

      $c.count$++;

    $L_k := \{c \in C_k \mid c.count \geq minsup\}$;

  $k$++;

**return** $\bigcup_k L_k$;

# candidate generation(1)

requirements to candidate itemsets:

- $C_k \supseteq L_k$
- $|C_k|$ should be much smaller then the set of all possible k-itemsets

**step 1**: Join

- p and q are frequent (k-1)-itemsets
- join p and q if they share a common (k-2)-prefix

$p \in L_{k-1}$ (beer, chips, pizza)

(beer, chips, pizza, wine) $\in C_k$

$q \in L_{k-1}$ (beer, chips, wine)

# Candidate generation(2)

**step2**: pruning

remove all candidate k-itemsets containing
non-frequent k-1-itemsets

**example**:

$L_3$ = {(1 2 3), (1 2 4), (1 3 4), (1 3 5), (2 3 4)}

after join step: candidates = {(1 2 3 4), (1 3 4 5)}

pruning step:

delete (1 3 4 5) since (1 4 5) is not frequent

=> $C_4$ = {(1 2 3 4)}

# Example Apriori Algorithms

minsup = 2

| TID | Items |
|---|---|
| 100 | 1 3 4 |
| 200 | 2 3 5 |
| 300 | 1 2 3 5 |
| 400 | 2 5 |

Scan DB

$C_1$

| itemset | sup. |
|---|---|
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {4} | 1 |
| {5} | 3 |

$L_1$

| itemset | sup. |
|---|---|
| {1} | 2 |
| {2} | 3 |
| {3} | 3 |
| {5} | 3 |

$C_2$

| itemset |
|---|
| {1 2} |
| {1 3} |
| {1 5} |
| {2 3} |
| {2 5} |
| {3 5} |

Scan DB

$C_2$

| itemset | sup |
|---|---|
| {1 2} | 1 |
| {1 3} | 2 |
| {1 5} | 1 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

$L_2$

| itemset | sup |
|---|---|
| {1 3} | 2 |
| {2 3} | 2 |
| {2 5} | 3 |
| {3 5} | 2 |

$C_3$

| itemset |
|---|
| {2 3 5} |

Scan DB

$L_3$

| itemset | sup |
|---|---|
| {2 3 5} | 2 |

# What you should know by now

- What is data mining and KDD?
- Steps in the KDD process
- What is supervised learning?
  - linear regression
  - kNN Classification
  - Bayesian  learning
  - metrics for supervised learning
- What is unsupervised learning?
  - clustering (k-means, DBSCAN)
  - outlier detection (distance-based outliers)
  - frequent pattern mining
    (frequent itemset mining ,apriori algorithm)

# Literatur


- **script KDD I**:
  http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_I_(KDD_I)
- Ester M., Sander J.:
  **Knowledge Discovery in Databases: Techniken und Anwendungen**
  Springer, September 2000.
- Han J., Kamber M., Pei J.:
  **Data Mining: Concepts and Techniques**
  3rd edition, Morgan Kaufmann Publishers, March 2011.
- H.-P. Kriegel, M. Schubert, A. Zimek
  **Angle-Based Outlier Detection in High-dimensional Data**
  In Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Las Vegas, NV: 444–452, 2008.
- M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander
  **OPTICS: Ordering Points To Identify the Clustering Structure**
  In Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA: 49–60, 1999.