Knowledge Discovery in Databases II

Winter Term 2014/2015

Chapter 3: Large Object Cardinalities And High-Performance Data Mining

Lectures : PD Dr Matthias Schubert
Tutorials: Markus Mauder, Sebastian Hollizeck

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

Chapter Overview

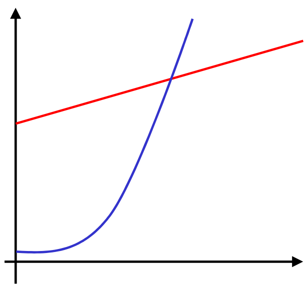
Problem Setting

So far:

- Focus on quality: How can we derive meaningful patterns?
- Feature Selection, Feature Reduction, Metric Learning and Subspace Clustering yield high complexities

In this chapter:

- How can we mine high volume data faster?
- Performance depends on
  - the volume of the data set
  - the scalability of the data mining algorithms



Solution Approaches

Use modern Hardware:

- Cloud Computing => Parallel Data Mining
- Broadband Networks => Distributed Data Mining

Where does it help?

- high volume data repositories (electronic payments, sales data, web pages, emails, ...)

  => every data object must be examined at least once
- preprocessing (select relevant data objects)
- data transformation (data discretization, temporal aggregation etc.)

Solution Approaches

Limitations of high-performance computing architectures:

- best-case speed up of parallel algorithms: linear in the number of machines
- in most cases: less than linear due to communication and result merging overheads
- in problems having a super linear complexity: adding more machines helps but does not make the problem scalable

What can be done in these cases?

Solution Approaches

Reduce the number of objects being processed

$\Rightarrow$ Sampling and Micro-Clustering

Why does this make sense ?

Representative Sample $\neq$ Large Sample

- A too small data set might not be representative
- A very large data set can still be biased and not representative

=> there are redundant samples

=> removing these from the data set does not hurt the representativeness

DATABASE SYSTEMS GROUP

Solution Approaches

LMU

Methods for reducing large data sets:

Sampling:

- Use a subset of the data set by removing redundant instances
- Find redundant features

Micro-Clustering:

- use a clustering algorithm to determine a set of cluster descriptions
- perform data mining on cluster descriptions

DATABASE SYSTEMS GROUP

Parallel and Distributed Data Mining

LMU

Goal:

- use multiple cores /work stations to increase performance
  $\Rightarrow$ parallel data mining

- if data is stored in distributed locations:
  $\Rightarrow$ distributed data mining

- if data is confidential:
  $\Rightarrow$ privacy preserving data mining

Privacy can only be preached if there are at least two parties (data owner and data user).

=> closely related to distributed mining

Examples

- Parallel DBSCAN to speed up processing
- Clustering end customers for distributors:
  - Retailer do not want to share customer information but might share distributions or statistics
  - Retailer needs *„privacy-preserving“* Clustering algorithms to derive general end customer groups
- Pharmaceutical companies collect costumer sales data from pharmacies
  - helps the company to plan the production of pharmaceutics
  - find profitable areas for researching new drugs

  But: Individual drug consume of costumers might be sold to insurance companies or is made available to the public.
  (potential employers, landlords, credit institutions,..)

Parallel and Distributed Data Mining

Parallel Data Mining:

- Data repository is already integrated and available in a common location.
- data has to be analyzed on *k* work stations
- performance gain by following a *„Divide and Conquer“* strategy:

  $\Rightarrow$ Distribute data to worker tasks

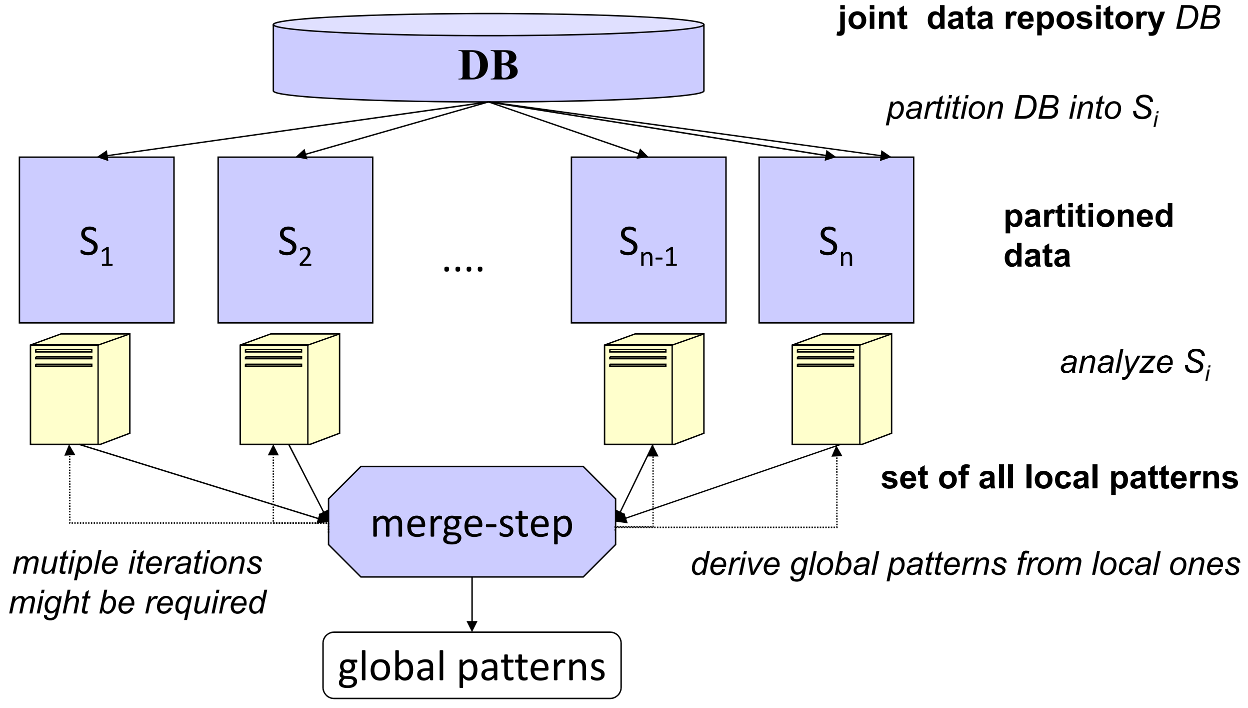
  $\Rightarrow$ each worker analyses the data and returns a local result

  $\Rightarrow$ local results must be combined to global patterns/functions

Important aspects:

- Distribute data in a way that joining local results into global patterns is easy
- Avoid communication between the workers as much as possible

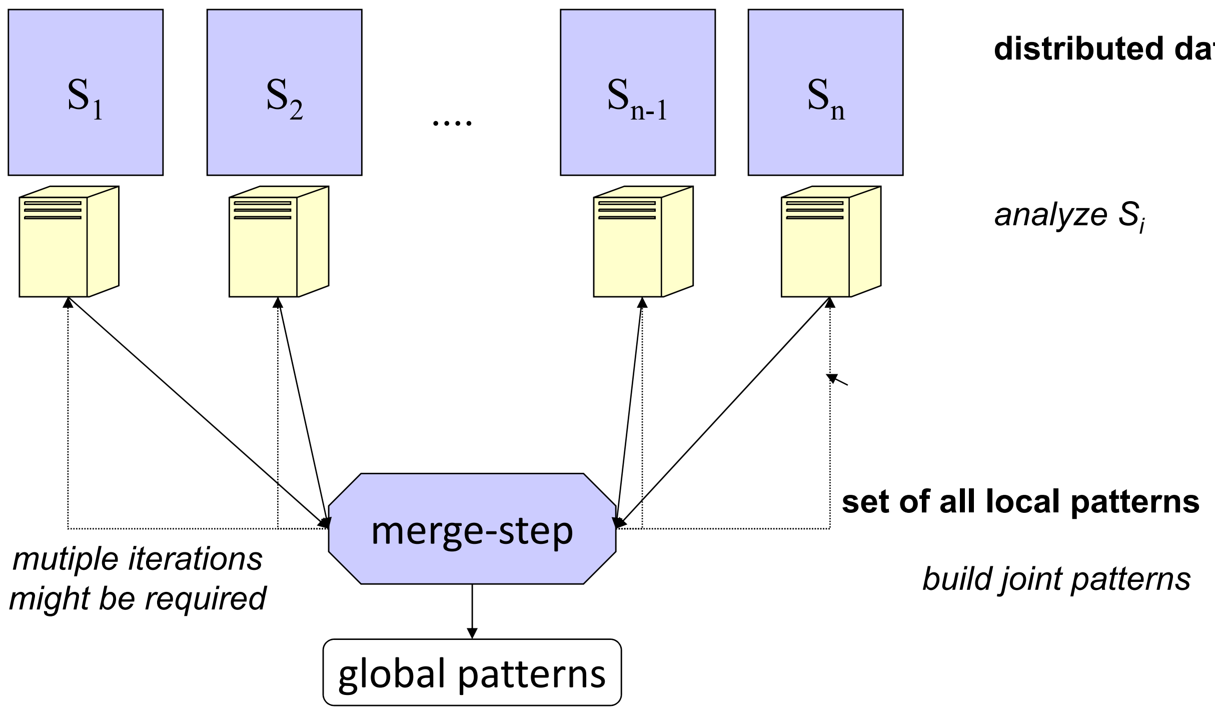
Workflow of parallel Data Mining


joint data repository *DB*
DB
partition DB into S_i
S_1
S_2
....
S_{n-1}
S_n
partitioned data
analyze S_i
set of all local patterns
merge-step
mutiple iterations might be required
derive global patterns from local ones
global patterns

Distributed Data Mining

- The distribution of data to the different Peers is given
  - ⇒ no effort for data partitioning
  - ⇒ local patterns are less controllable
  - ⇒ joining local patterns might be more difficult
- an unfavorable distribution might lead to the following problems:
  - Discrepancies between the result of distributed and stationary mining
  - Large communication effort
- Differences between parallel and distributed:
  - distribution is given
  - network costs are usually assumed to be higher (between companies, mobile clients..)

Workflow Distributed Data Mining

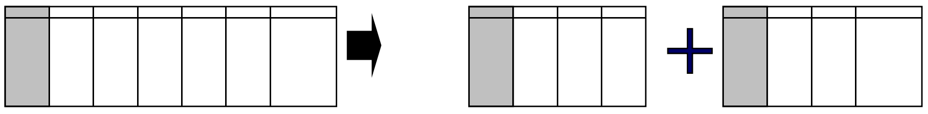


further algorithmic properties

1. Data Partitioning:

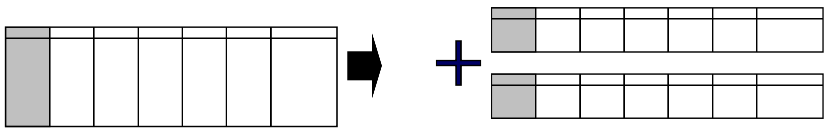
- *Vertical Partitioning*

  Features are distributed. Objects are available everywhere



- *Horizontal Partitioning*

  Objects are distributed over workers and sites.
  Object description is the everywhere the same.



- In practice: Data might be partitioned in both ways.

further algorithmic properties

2. **By Task: C**lassification, Clustering, Association Rules

3. **Partitioning dependency:** Does the result depend on the used/given partitioning of the data.

4. **Type of local patterns:** Approximations, data objects, distributions… examples: Gaussians, hyper rectangles, centroids…

5. **Organization of the distributed Workflow:**

   Master and Slave processes, P2P computation

Parallel Data Mining Algorithms

- Usually the result is expected to be independent from the partitioning (deterministic result)
- Main focus is speeding up the computation
- Partitioning strategy is often a major part of the algorithm:
  - minimize effort for joining local patterns

    $\Rightarrow$ local patterns should be independent from each other

    $\Rightarrow$ in case of dependencies: extra communication is required or inaccurate results have to be accepted

  - Runtime depends on the worst runtime of any worker task

  $\Rightarrow$ all parallel steps should take about the same amount of time

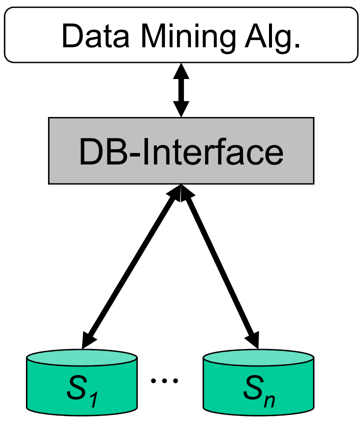
  $\Rightarrow$ all sites should receive the same amount of data

Parallelism via Database Primitives

- In general data mining algorithms can be based on database primitives (e.g. eps-range query)
- parallel computing of the database primitives yield a better hardware support to general data mining algorithms.

Example:

- parallel computation of ε-range queries can accelerate density-based clustering
- parallel kNN queries , allow fast instance-based classification.



distributed query processing

characteristics:

- Joining the results have to be done on one machine
- Partitioning might still play a major role

Parallel Density-Based Clustering

Idea:

- Horizontal and compact Partitioning
- Determine local core points and clusters
- Connect local clusters to global clusters:
  - Clusters from different sites
  - Noise points from other sites

General problem:

What happens to objects where ε-range intersects with of the partition?

- mirror marginal objects
- requires communication between the partitions

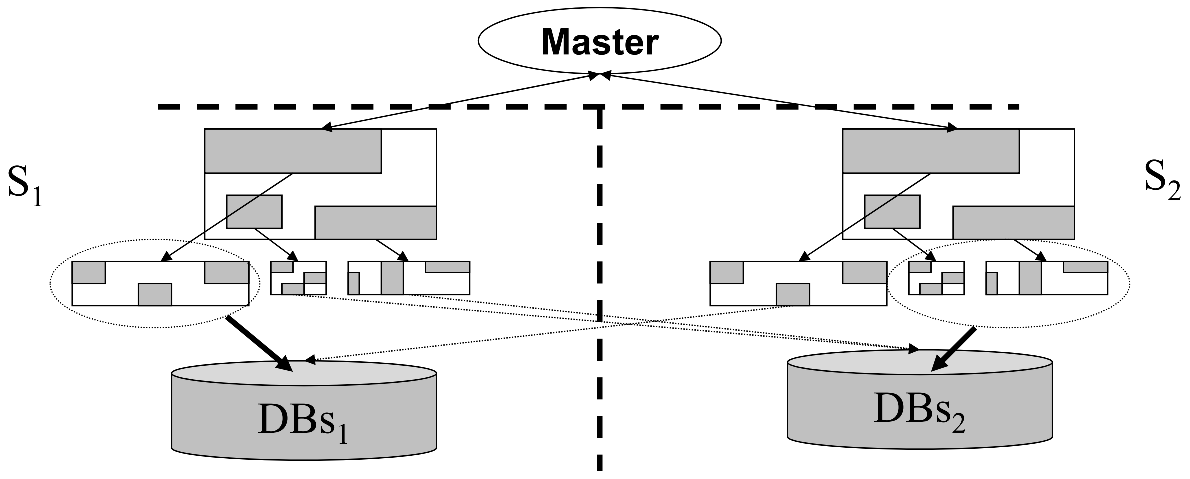
Solution

PDBSCAN [Xu, Jäger,Kriegel99]

Idea:

- Store vectors in a distributed index structure (dR*-Tree)
  - Directory of the R-tree in each Site
  - Data pages partitioned w.r.t. spatial distance
- compute local clusters for each site
- Marginal points cause a cross-site queries
- Afterwards merge clusters having commong points

Parallel DBSCAN



dR*-tree:

- Perform range queries on the complete data set
- Queries on S_i, being completely processible on DBs_i can be answered completely simultaneously
- Access to pages on other sites reduce concurrence and raise communication costs

=> Algorithms should employ as much local queries as possible

PDBSCAN

Data partitioning in the dR*-tree

(the same amount of pages on each site)

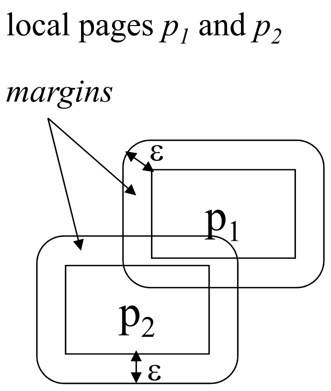
Idea:

- DBSCAN runs for each point in p_1 and p_2
- If ε-range intersects with the margin:
  1. Margins might have to be loaded from other sides to determine core points
  2. Expanding clusters beyond the margin is to expensive would lead to large communication overheads.

     => store points in merge list
- Join local cluster having common merge points:

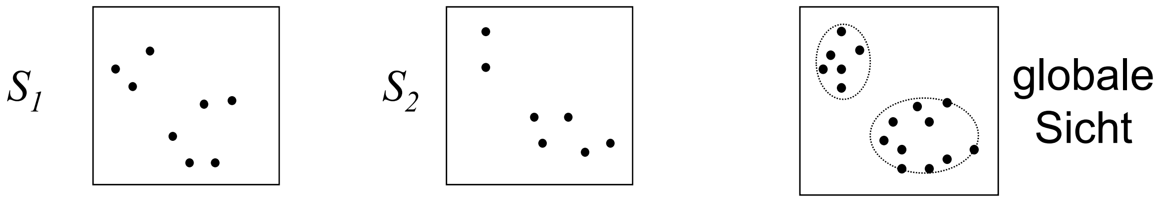
  merge point needs to be a core point in at least one partition

  => merge clusters



Distributed Density-based Clustering

- Arbitrary distribution of points over the sites
- Partitions might spatially overlap
  - Each site S_i might store elements of the ε–range of point p
  - *p* might be a core point, even if p isn‘t a local core point.



- Density-based clustering does not use a compact cluster model

  ⇒ Transfer local points to determine global clusters

Distritbuted Partitioning Clustering

Idea:

- If the cardinality of the transferred points is small, multiple iterations between the sites are no problem (low traffic)
- *Centroids and cluster quality in k*-Means or related methods is suitable for distributed computing:

$$TD^2 = \sum_{o \in DB} \left(\min_{C_i \in C} \{d(o, C_i)\} \right)^2 = \sum_{S_j \in DB} \left(\sum_{o \in S_j} \left(\min_{C_i \in C} \{d(o, C_i)\} \right)^2 \right)$$

- global centroid C_j is computed from local centroids $C_{i,j}$:

$$c_j = \frac{1}{\sum_{C_{i,j} \in DB} |C_{i,j}|} \cdot \sum_{C_{i,j} \in DB} \sum_{o \in C_{i,j}} o$$

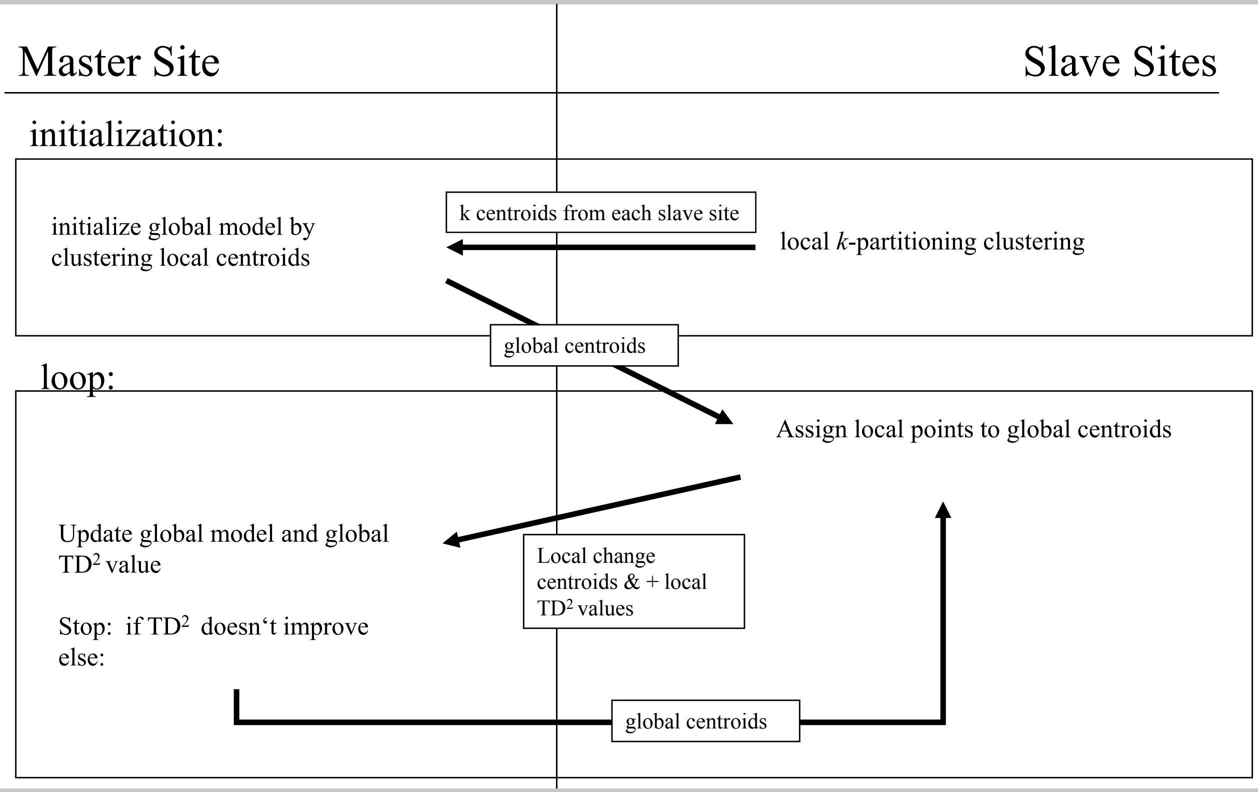
- **Summary**: In each iteration it is possible to optimize the global clustering by adding up local components.

Distributed Partitioning Clustering

Distributed clustering using variance minimization (Master-Slave):

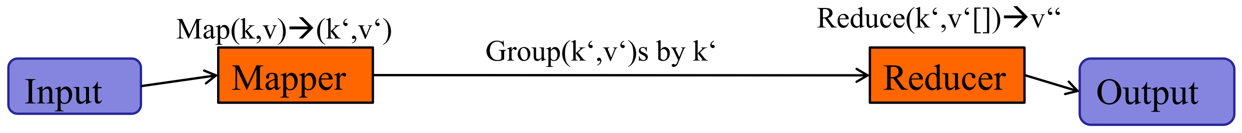
```
Determine initial distribution and start-Centroids
loop:
  transfer centroids to all sites
  assign local points to the current centroids
      => compute local centroids and local $TD^2$ values
  After retransferring local centroids, cluster
  cardinalities and $TD^2$ values
   $\Rightarrow$ Add local sum-vectors, cluster cardinalities and $TD^2$
     values (implies new global centroids)
   $\Rightarrow$ Determine global $TD^2$ value
   if $TD^2$ value does not improve => terminate
```

Workflow Distributed K-Means



Master Site
Slave Sites
initialization:
initialize global model by clustering local centroids
k centroids from each slave site
local *k*-partitioning clustering
global centroids
loop:
Assign local points to global centroids
Update global model and global TD² value
Local change centroids & + local TD² values
Stop: if TD² doesn‘t improve
else:
global centroids

Parallel Data Analysis with MapReduce



- Based on statements from functional programming: Map and Reduce
- Originally developed by Google to allow batch processing for large amounts of data with little implementation effort (e.g. calculating PageRank)
- Open source version: Hadoop (used by Facebook)
- Data is represented as <key, value> pairs
  example: <DokID3, „Today is a beautiful day..“>
- Both mapper and reducer can be distributed over multiple worker tasks
- Optimization is done automatically by adding workers if necessary
- Trade-off: Parallelism vs. Bandwidth

Workflow of a MapReduce Program

Processing in 3 steps:

1. Map-Step: Input: <key1, value1> Output:<key2,value2>

each input pair is computed into 0 to n output pairs

Example: <ID, Text> => <Today, 1>,<is,1>, <a,1>….

2. Shuffle-Step: Output of the mapper is grouped w.r.t. keys.

Example: <Today, <1,1,1,1,1,1,1,1,1,1,1,1>>

3. Reduce-Step: Each pair from step 2 is processed into an output.

Example: <Today,12>

For complex problems multiple MapReduce steps might be necessary to implement an algorithm.

Only the complex steps are computed in a parallel way.

Optional Steps

- ***Partitioner***: Controls the distribution of data over the mapper tasks.
  Default: HashPartitioner
- ***Combiner***: Local aggregation step which summarizes data from a mapper
  Step is performed between Map step and shuffle step.
  => Transfer volume from the Mappers to shuffle step can be reduced
  Example: <Today, <1,1,1,1,>> -> <Today,4>

K-Means using MapReduce

input: A data set *D* having n feature vectors, *k* number of clusters

output: *k* centroids minimizing *TD²*

$$TD^2(D,C) = \sum_{x \in D} \left(\min_{c \in C} \left(dist(c,x) \right) \right)^2$$

Steps:

- Assign vectors to next clusters
- Compute centroids for a set of objects
- Compute TD^2

$\Rightarrow$ All steps can be done by a linear scan of D

$\Rightarrow$ Results are additive. Cluster and TD^2 are sums and therefore, computable in a distributed way. (**associative law**)

k-Means using MapReduce

```
Master:
    Sample k initial centroids C.
    WHILE TD2 < oldValue
        oldValue = TD2
        assign vectors in D to centroids in C (Mapper)
        compute centroids C and quality TD2 (Reducer )
    RETURN C
```

Remark:

- Only the expensive steps are processed in a distributed way
- one MapReduce task for each iteration
- C has to be transferred to mappers and reducers

K-Means: Map Step

```
Input: D: set of vectors,C: set of centroids ,k=|C|: number of centroids
Output: <centroid_id, vector>
FOR EACH v  in D DO
    bestCluster = null; minDist = ∞
    FOR EACH C_i in C DO
        IF minDist > dist(C_i, v) THEN
            minDist = dist(C_i, v)
            bestCluster = C_i
        ENDIF
    END FOR
    OUTPUT<bestCluster, v>
END FOR
```

Afterwards shuffle step generates aggregated pairs : <Cluster,<v1,..,vl>>

K-Means: Reduce Step

```
Input: <Cluster,<v1,..,vl>>
Output: <newC, TD2> new centroids and parts of TD2
FOR EACH <C, <v1,..,vl>>  DO
    linearSum = 0;
    count = 0;
    TD2 = 0;
    FOR EACH v_j in <v1,..,vl> DO
        linearSum += v_j
        count = count+1
        TD2 +=dist(vj,C)
    END FOR
    newC = linearSum/count
    OUTPUT< newC, TD2>
END FOR
```

K-Means using MapReduce

- Number of calls corresponds to the number of iterations
- Optimization by local combiners which precompute parts of the linear sums.
- Algorithms does not solve the problem of a suitable initialization
- newer methods use sampling techniques to cluster data in sublinear time.

Privacy Preservation

Connection to distributed Data Mining:

⇒ only any issue if multiple parties are involved:

Data-Owner: knows exact data, but does not want to reveal it completely

Data-User: is interested in deriving general patterns

Privacy Preserving Data Mining:

Allows the d*ata user* global patterns from the *data being provided by the data owners* without revealed specific information about data objects:

1. *Data user* must not be able to draw conclusions over the specific data objects during the mining process.

2. The generated patterns must not reveal information about specific objects.

Privacy Preservation

Why is privacy preservation important ?

1. A lot of data is only provided by the data owners if the privacy is saved
   Example: Analyze clickstreams from web browsers
2. Data mining should not be used as a excuse to collect data.
3. Protection from misuse of the provided information by third parties.
   Example: Publication of results about the surfing behavior is used to personalize spam mails and fishing attempts.

Conclusion:

Data Mining does not necessarily violate privacy constraints.

Results are general patterns which should not contain object specific information.

The same patterns can be derived from different samples drawn from the data distribution.

Privacy Preservation

- **idea**: Change data in a way preserving the patterns but removes object-specific information

- **Solutions depends on the way data is generalized in the given data mining method**:
  overfitting patterns have the tendency to contain too specific information
  $\Rightarrow$ generality of the patterns is complementary to privacy preservation

Privacy protection is done by:

- change data objects (data perturbation)
- General model for the data (distribution functions)
- Generate new data following the same distribution

(sampling from distribution functions)

Basic Techniques to Privacy Preservation

discretization:
- Disjunctive distribution of the value set into several discrete subsets
- Actual values are replaced by an interval of values

example:
- *original information*: Person A earns 42.213 € p.a.
- *discrete information*:
  Person A earns between 35.000 € and 55.000 € p.a.

problem:
- information is weaker but still detectable
- If the number of objects in any interval too small possible privacy breach

$\Rightarrow$ Uniform distribution of the number of objects per interaval
(no equidistant distribution over the value set)

Basic Techniques of Privacy Preservation

Data Perturbation:
- Transmit the sum of the original value *w* and a random number *r*: w_i+r_i
- Distributions for *r* :
  - Uniform distribution $[-\alpha,..,\alpha]$ $\alpha \in IR^+$
  - Normal distribution with mean 0 and standard deviation σ
- The perturbation distribution has a mean value of 0
  => the mean value of the summed up distribution is E(W)
  (mean values of the sum of two random variables add up)

Basic Techniques to Privacy Preservation

„Privacy-Level“ (Quality of Privacy Preservation)

Idea:

If it is possible to predict that x is in the interval $[x_1, x_2]$ with c% then the size of the interval $[x_1, x_2]$ corresponds to the privacy level having the confidence level c.

Formal:

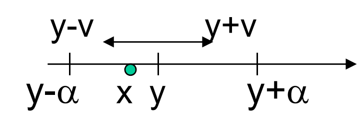
input: changed feature value y being constructed from *r* and the original feature value *x*.

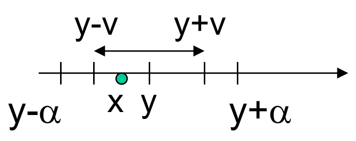
output: Breadth *2v* of the interval *[y-v..y+v] in which* the original value *x* is contained with *c %* probability. (*2v* ≡ Privacy Level)

Privacy Level

Example:

- error R is uniform distributed in $[-\alpha,..,\alpha]$.
- for 100 % ⇒ $\alpha = v$ ⇒ privacy level = 2α (value must be within the interval)
- for 50 % ⇒ $2v/2\alpha = 0.5$ ⇒ $v = 0.5\alpha$
- generally: $v = c\% * \alpha$





Reconstruction of the original Distribution

Input: data of perturbated data *W = {w_1,..,w_n}*

- Probability density function of *R*: $f_R: IR \rightarrow [0..1]$

Output: Approximation of the original distribution X $f_X: IR \rightarrow [0..1]$

- ***Approximation of the approach***

$$F'_{X_1}(a) = \int_{-\infty}^{a} f_{X_1}(z \mid X_1 + Y_2 = w_1)\,dz = \int_{-\infty}^{a} \frac{f_{X_1+Y_1}(w_1 \mid X_1 = z) f_{X_1}(z)}{f_{X_1+Y_1}(w_1)}\,dz$$

$$= \frac{\int_{-\infty}^{a} f_{X_1+Y_1}(w_1 \mid X_1 = z) f_{X_1}(z)\,dz}{\int_{-\infty}^{\infty} f_{X_1+Y_1}(w_1 \mid X_1 = z) f_{X_1}(z)\,dz} = \frac{\int_{-\infty}^{a} f_{Y_1}(w_1 - z) f_{X_1}(z)\,dz}{\int_{-\infty}^{\infty} f_{Y_1}(w_1 - z) f_{X_1}(z)\,dz} = \frac{\int_{-\infty}^{a} f_Y(w_1 - z) f_X(z)\,dz}{\int_{-\infty}^{\infty} f_Y(w_1 - z) f_X(z)\,dz}$$

Reconstruction of the Original Distribution

- *The distribution over all values can be derived from $F_{X1}(a)$* :

$$F'_X(a) = \frac{1}{n} \sum_{i=1}^{n} F_{X_i}(a)$$

- The following probability density function can be determined by differentiation

$$f'_X(a) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(w_i - a) f_X(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X(z)\,dz}$$

- Since $f_X(a)$ *is* unknown, $f'_X(a)$ iteratively approximated from $f_X(a)$.

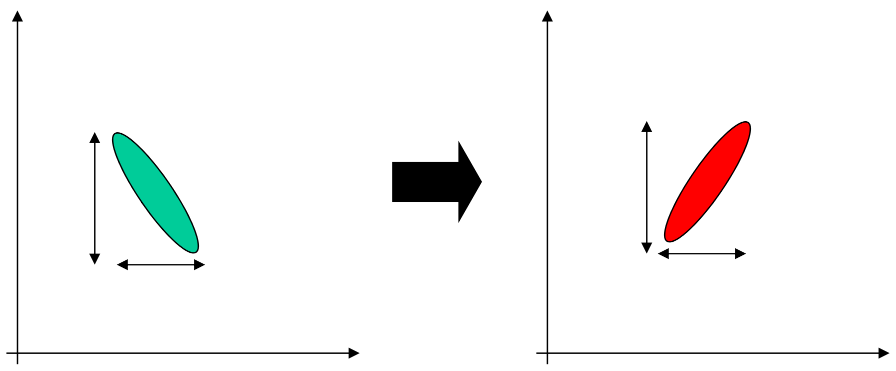
Reconstruction of the original distribution

Iterative approximation algorithms for reconstructing the original distribution:

```
$f^0_X$:= uniform distribution
j:= 0 //iteration counter
repeat
        $f_X^{j+1}(a) := \frac{1}{n}\sum_{i=1}^{n}\frac{f_Y(w_i - a)f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z)f_X^j(z)dz}$
        j:=j+1
until ( $f_X^{j+1}(a) - f_X^j(a) < \varepsilon$ )
```

Further Techniques Privacy Preservation

- Swapping = exchange values o1.attr1 and o2.attr1
  - $\Rightarrow$ original feature vectors cannot be reconstructed
  - $\Rightarrow$ Well-suited for algorithms assuming independent features like Naïve Bayes, Decision trees
  - $\Rightarrow$ Patterns depending on feature correlation are destroyed



Privacy Preservation by data approximation

Idea:

- Data owners provide local patterns/distributions instead of instances
- Patterns/Distribution must be general enough to preserve the privacy

Possible solutions:

- distribution function
- explicit cluster models (centroids covariance matrix)
- Locally frequent patterns

Summary Privacy Preservation

- Distributed mining is based on sharing data for a special purpose
- Privacy breach when the shared data is used for other purposes
- Generally patterns, functions, models are not problem for privacy if generalization is performed well-enough
- Data mining algorithms can be made aware of this problems and can be tuned to allow a certain level of privacy

Caution:

- Data Mining can be used to learn private information
  (e.g. link prediction, predict unknown values from available information)