Knowledge Discovery in Databases II

Winter Term 2014/2015

Lecture 7:
Volume: Large Object Cardinalities: Privacy Preserving Data Mining, Sampling and Summarization

Lectures : Dr Eirini Ntoutsi, PD Dr Matthias Schubert
Tutorials: PD Dr Matthias Schubert

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

Chapter Overview

DATABASE SYSTEMS GROUP

Why it is important?

Why is privacy preservation important ?

- A lot of data is only provided by the data owners if the privacy is saved
  - Example: Analyze clickstreams from web browsers
- Data mining should not be used as a excuse to collect data.
- 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.

nothing about an individual should be learnable from the database that cannot be learned without access to the database

T. Dalenius, 1977

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 that patterns are preserved but object-specific information is removed

- Solutions depend on the way the 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 achieved through different ways:

- Changing data objects (data perturbation)
- Use a general model for the data (distribution functions)
- Generate new data following the same distribution (sampling from distribution functions)

Basic Techniques: Discretization

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 is too small, possible privacy breach

$\Rightarrow$ Uniform distribution of the number of objects per interval

Basic Techniques: Data Pertubation [AgrSri00]

Data Perturbation:

- Add a noise component to the original data
- Instead of the original value x_i, transmit the sum of the original value x_i and a number *r* drawn from an error distribution *f, i.e,* : $x_i + r$
- Distributions *f* for *r* :
  - Uniform distribution $[-\alpha,..,\alpha]$ $\alpha \in IR^+$
  - Normal distribution with 0 mean and standard deviation σ

- The intent is to allow legitimate users the ability to access important aggregate statistics (such as mean, correlations, etc.) from the entire database while 'protecting' the individual identify of a record.

Privacy-level

- Indicates how closely the sensitive information, that has been hidden, can still be estimated.
- **Definition privacy level**: if it can be estimated with c% confidence that a value x lies in the interval $[x_1, x_2]$, then the width of the interval $(x_2 - x_1)$ defines the amount of privacy at c% confidence level.

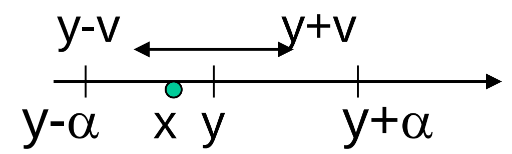
- input: the changed feature values y_i, constructed from the original feature values and the noise component, $x_i + r$. the error distribution
- output: Breadth of the interval [y-v, y+v] in which the original value x is contained with c % probability. (Privacy Level=2v)

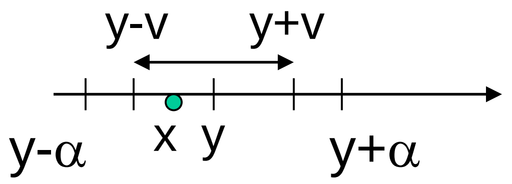
Privacy Level

Example:

- error *r* is uniformly distributed in [-α,..,α].

- For c=100 % confidence (value must been within the interval [y-a, y+a]), the interval is 2a.
  - Privacy level = 2a
- With c=50 % confidence,
  - Privacy level = α
- General formula:
  - v = c% * 2α
  - Privacy increases with a





Reconstruction of the original distribution

Given the noise distribution functions f_Y and the pertubed data set x_1+y_1, x_2+y_2, x_n+y_n, one could reconstruct the original dataset's distribution f_X (but not actual data values)

Input: The set of perturbated data *$W = \{w_1,..,w_n\}$, where $w_i=x_i+y_i$*

and the probability density functions of noise f_Y.

Output: Approximation of the original distribution f_X

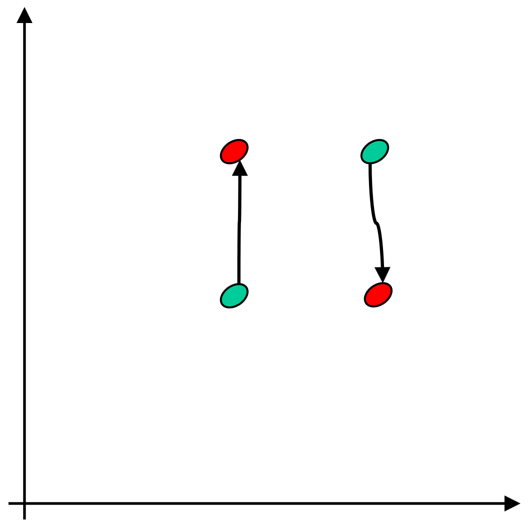
Solution: Iterative approximation algorithms (initialize f_x with uniform)

```
$f^0_X$:= 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 ( $f_X^{j+1}(a) - f_X^j(a)$ < $\varepsilon$)
```

DATABASE SYSTEMS GROUP

Basic techniques: Data swapping

- Data swapping = the values across different records are swapped
  - $\Rightarrow$ original feature vectors cannot be reconstructed
  - $\Rightarrow$ Well-suited for algorithms assuming independent features like Naïve Bayes
  - $\Rightarrow$ Patterns depending on feature correlation are destroyed



Basic techniques: Data approximation

Idea:

- Data owners provide local patterns/distributions instead of instances
- Patterns/Distributions 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 a 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)

References

- V. Verykios, E. Bertino, I.N. Fovino, L. Parasiliti Provenza, Y. Saygin, Y. Theodoridis, *State-of-the-art in Privacy Preserving Data Mining*, SIGMOD Record, 2004.
- Jagannathan G., Wright R.N. : *Privacy Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data*, Proc. 11th ACM SIGKDD, 2005
- Kriegel H.-P., Kröger P., Pryakhin A., Schubert M.: *Effective and Efficient Distributed Model-based Clustering*, Proc. 5th IEEE Int. Conf. on Data Mining (ICDM'05), Houston, TX, 2005
- [AgrSri00] Agrawal R., Srikant R.: *Privacy-preserving data mining*", Proc. of the ACM SIGMOD Conference on Management of Data, Dallas, TX, 2000
- E. Bertino, D. Lin, W. Jiang, *A Survey of Quantification of Privacy Preserving Data Mining Algorithms*,+++

Chapter Overview

Reducing the cardinality of the input set

- If data mining algorithms have a super linear complexity, parallel processing and hardware can help, but do not solve the problem
- In such cases, runtimes can only be reduced by limiting the number of input objects
- Solution:
  - reduce the input data to a smaller set of objects
  - perform data mining on the reduced set

$\Rightarrow$ Results may vary from using the complete data set

$\Rightarrow$ Parallel processing can be used for this preprocessing step

Approaches

Methods for reducing large data sets:

- Sampling
  - Select a subset of the input dataset
  - Use the subset, instead of the original dataset, for mining
- Summarization
  - Summarize the input dataset through appropriate summaries.
  - Use the summaries, instead of the original raw data, for mining.

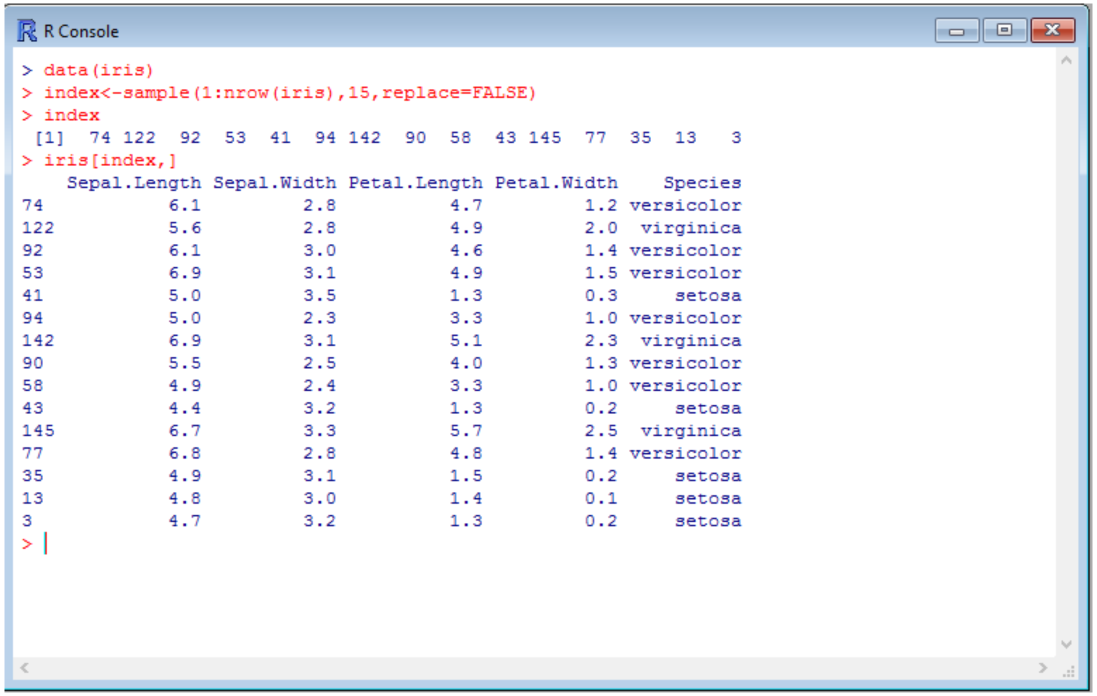
Sampling

- Idea: Select a limited subset from the original dataset.
- It is important that the group selected be *representative* of the population, and not biased in a systematic manner
- Sampling approaches:
  - Random sampling: draw k times from the data set and remove the drawn elements
  - Bootstrapping: draw k times from the input data set but do not exclude the drawn elements from the set
  - Stratified sample: Draw a sample which maintains a distribution w.r.t. to some attributes (e.g., class labels)

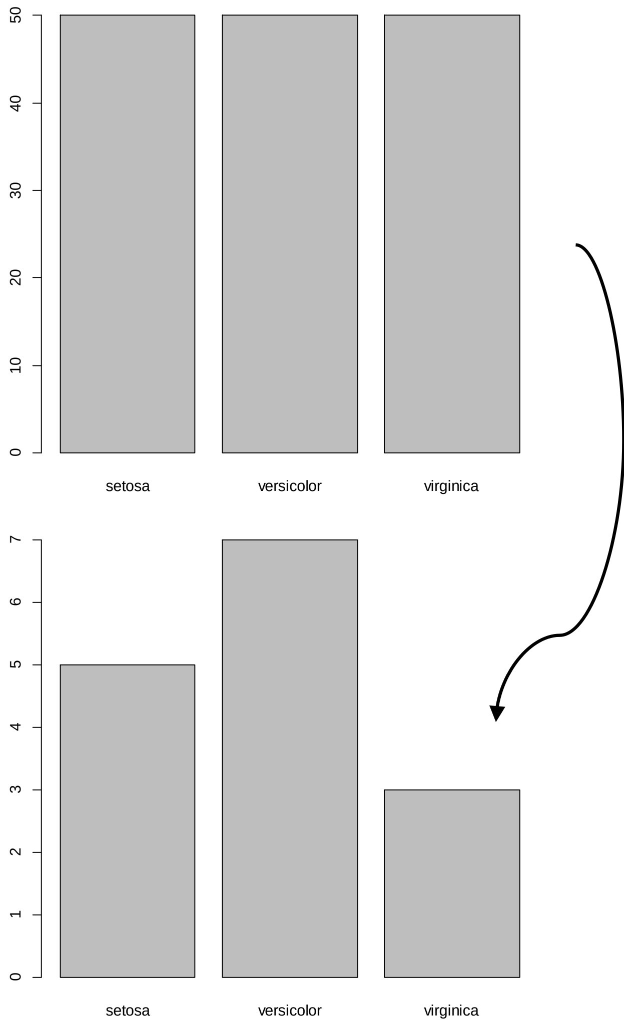
Random sampling

- Each instance of the input dataset is chosen entirely by chance.
- Each instance has an equal chance of being included in the sample.
- Procedure: if you want to select *k* out of *n* instances (*n*: the cardinality of the input dataset):
  - Generate an independent random integer i, $1 \leq i \leq n$
  - Draw instance *i*, if not already selected
  - Repeat *k* times

Random sampling (R)



```
> data(iris)
> index<-sample(1:nrow(iris),15,replace=FALSE)
> index
 [1]  74 122  92  53  41  94 142  90  58  43 145  77  35  13   3
> iris[index,]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
74           6.1         2.8          4.7         1.2 versicolor
122          5.6         2.8          4.9         2.0  virginica
92           6.1         3.0          4.6         1.4 versicolor
53           6.9         3.1          4.9         1.5 versicolor
41           5.0         3.5          1.3         0.3     setosa
94           5.0         2.3          3.3         1.0 versicolor
142          6.9         3.1          5.1         2.3  virginica
90           5.5         2.5          4.0         1.3 versicolor
58           4.9         2.4          3.3         1.0 versicolor
43           4.4         3.2          1.3         0.2     setosa
145          6.7         3.3          5.7         2.5  virginica
77           6.8         2.8          4.8         1.4 versicolor
35           4.9         3.1          1.5         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
3            4.7         3.2          1.3         0.2     setosa
> |
```

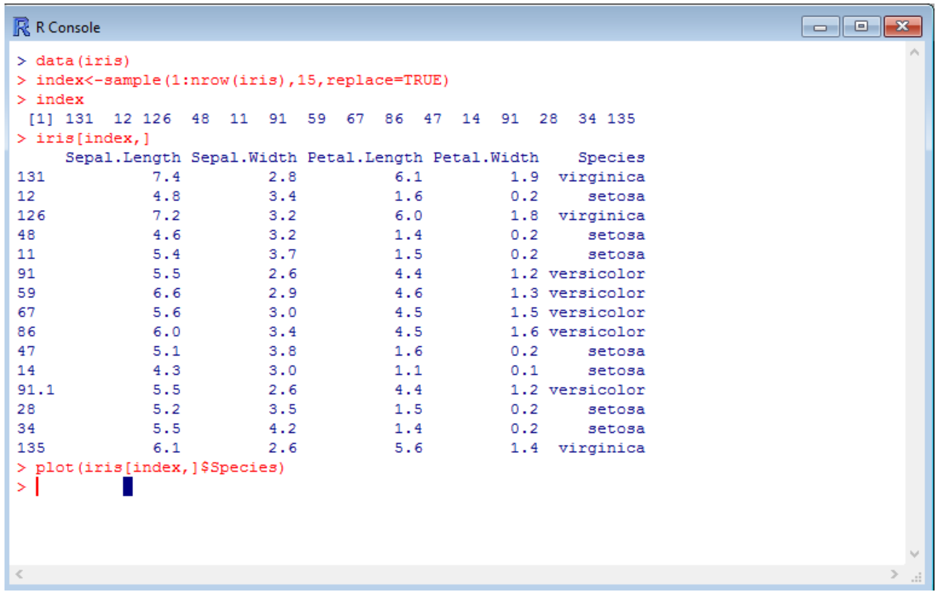


Bootstrapping

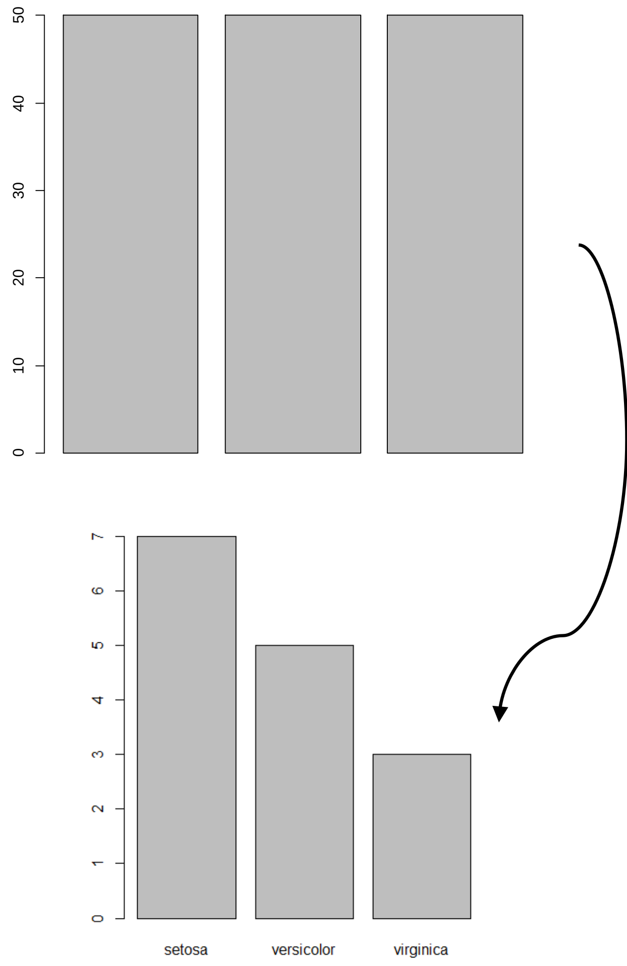
- Random sampling with replacement
- Procedure: if you want to select *k* out of *n* instances (*n*: the cardinality of the input dataset):
  - Generate an independent random integer i, $1 \leq i \leq n$
  - Draw instance *i*
  - Repeat until *k* instances have been drawn.

- Difference to random sampling: The already drawn elements are not excluded from future selection.

Bootstrap sampling (R)



```
> data(iris)
> index<-sample(1:nrow(iris),15,replace=TRUE)
> index
 [1] 131  12 126  48  11  91  59  67  86  47  14  91  28  34 135
> iris[index,]
     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
131           7.4         2.8          6.1         1.9  virginica
12            4.8         3.4          1.6         0.2     setosa
126           7.2         3.2          6.0         1.8  virginica
48            4.6         3.2          1.4         0.2     setosa
11            5.4         3.7          1.5         0.2     setosa
91            5.5         2.6          4.4         1.2 versicolor
59            6.6         2.9          4.6         1.3 versicolor
67            5.6         3.0          4.5         1.5 versicolor
86            6.0         3.4          4.5         1.6 versicolor
47            5.1         3.8          1.6         0.2     setosa
14            4.3         3.0          1.1         0.1     setosa
91.1          5.5         2.6          4.4         1.2 versicolor
28            5.2         3.5          1.5         0.2     setosa
34            5.5         4.2          1.4         0.2     setosa
135           6.1         2.6          5.6         1.4  virginica
> plot(iris[index,]$Species)
>
```



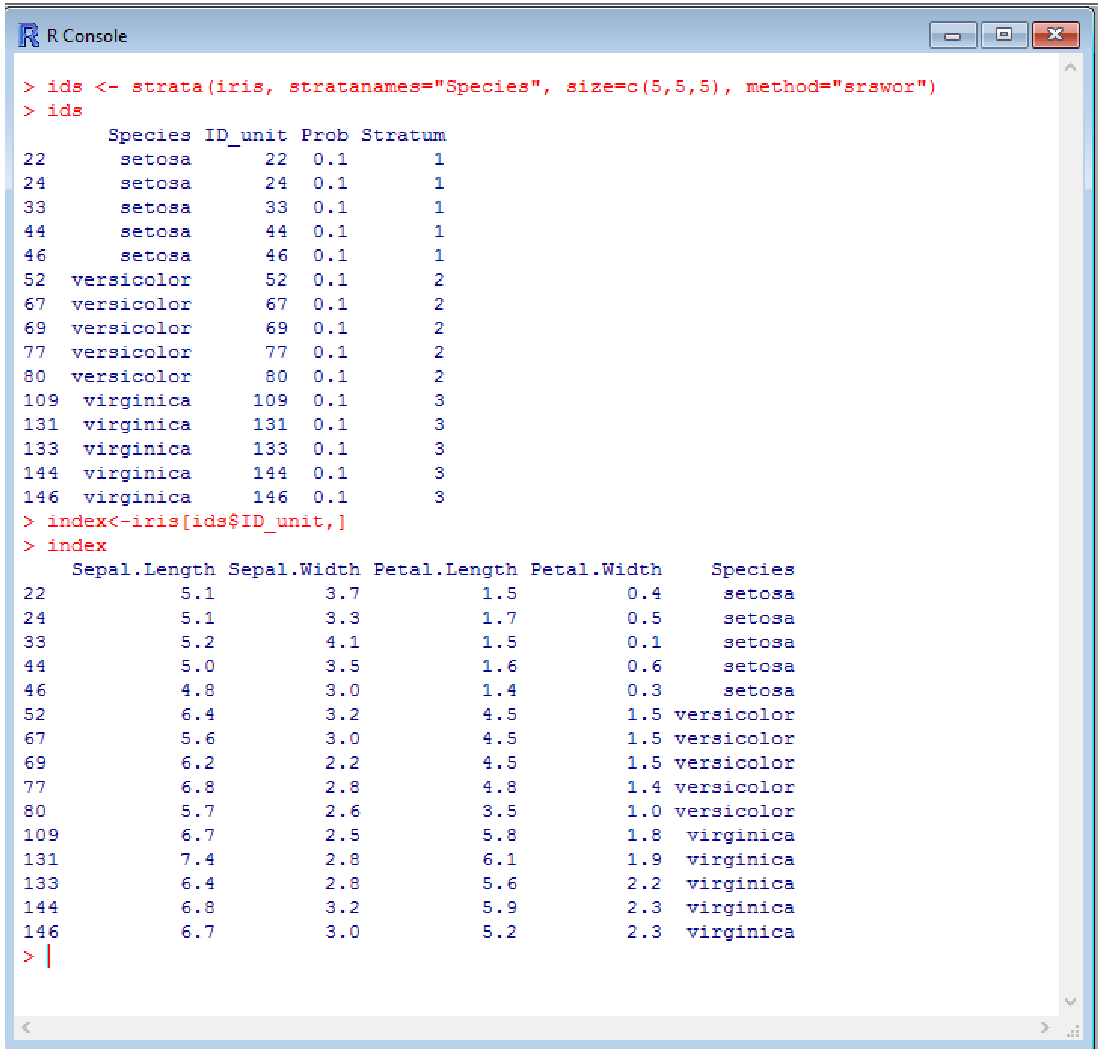
Stratified sampling

- Stratification is the process of dividing the initial dataset into homogeneous subgroups (*strata*) before sampling.
  - Every instance is assigned to only one stratum
  - No instance is excluded
- Apply random sampling, within each stratum.

- It ensures the presence of key subgroups in the sample
  - Even small groups are represented

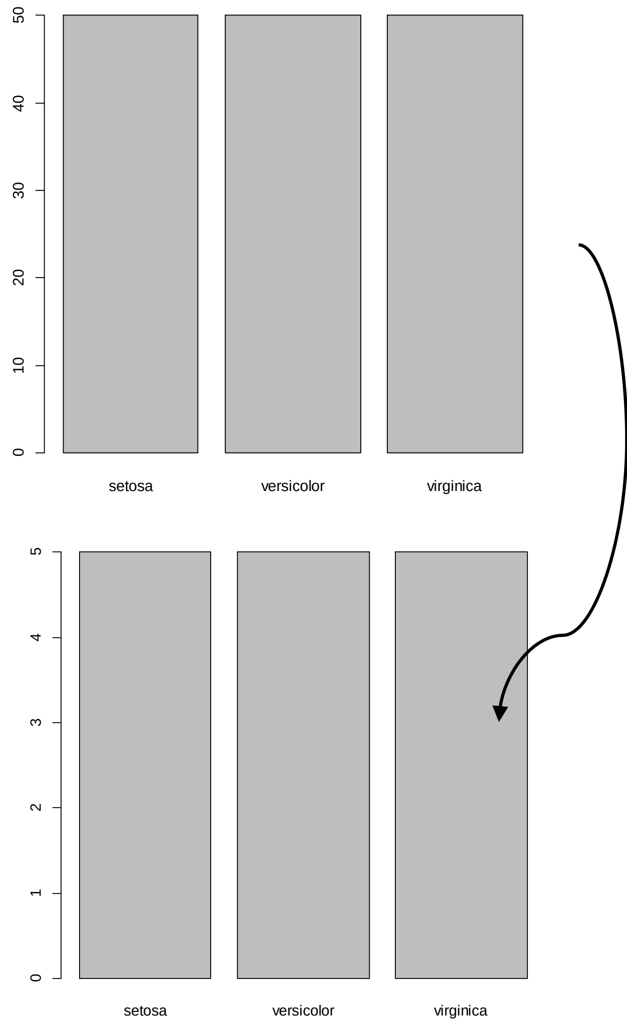
- Class labels is a typical way to define the strata.
  - Stratified sampling to deal with unbalanced data classification
- Other (sets of) dimensions can be also used, e.g., demographics

Stratified sampling example (R)



R Console

```
> ids <- strata(iris, stratanames="Species", size=c(5,5,5), method="srswor")
> ids
        Species ID_unit Prob Stratum
22       setosa      22  0.1       1
24       setosa      24  0.1       1
33       setosa      33  0.1       1
44       setosa      44  0.1       1
46       setosa      46  0.1       1
52   versicolor      52  0.1       2
67   versicolor      67  0.1       2
69   versicolor      69  0.1       2
77   versicolor      77  0.1       2
80   versicolor      80  0.1       2
109   virginica     109  0.1       3
131   virginica     131  0.1       3
133   virginica     133  0.1       3
144   virginica     144  0.1       3
146   virginica     146  0.1       3
> index<-iris[ids$ID_unit,]
> index
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
22           5.1         3.7          1.5         0.4     setosa
24           5.1         3.3          1.7         0.5     setosa
33           5.2         4.1          1.5         0.1     setosa
44           5.0         3.5          1.6         0.6     setosa
46           4.8         3.0          1.4         0.3     setosa
52           6.4         3.2          4.5         1.5 versicolor
67           5.6         3.0          4.5         1.5 versicolor
69           6.2         2.2          4.5         1.5 versicolor
77           6.8         2.8          4.8         1.4 versicolor
80           5.7         2.6          3.5         1.0 versicolor
109          6.7         2.5          5.8         1.8  virginica
131          7.4         2.8          6.1         1.9  virginica
133          6.4         2.8          5.6         2.2  virginica
144          6.8         3.2          5.9         2.3  virginica
146          6.7         3.0          5.2         2.3  virginica
> |
```



Spatial Sampling 1/2

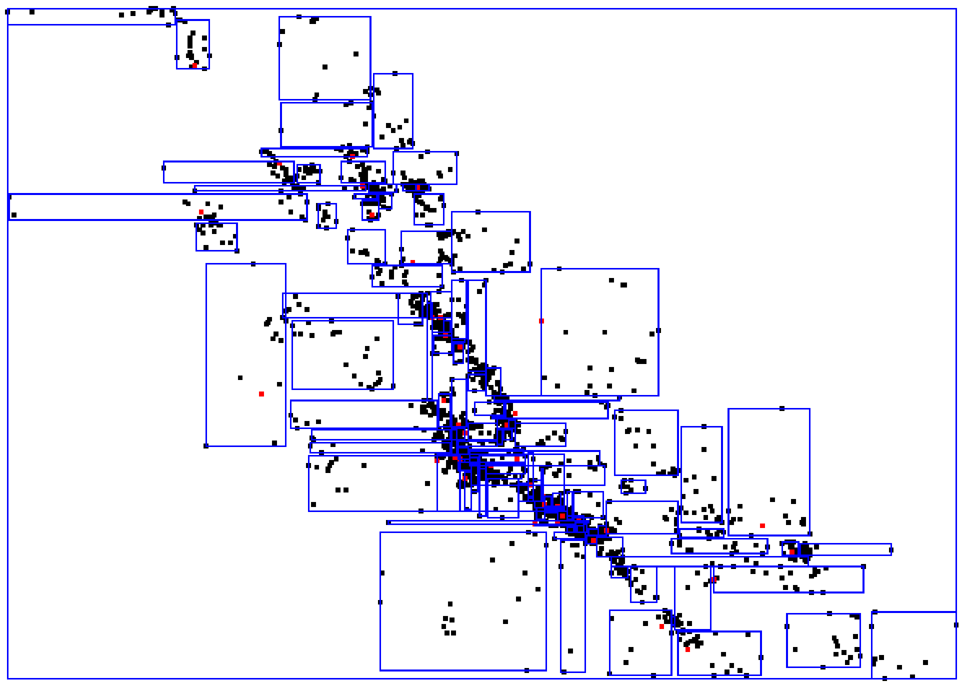
Index-based sampling [EstKriXu95]

- Random sampling is problematic in spatial data
- Use spatial index structures to estimate spatial distributions
    - index structures obtain a coarse pre-clustering; neighboring objects are stored on the same / a neighboring disk block
    - index structures are efficient to construct (sometimes are already there)
    - allow fast access methods for similarity queries

Spatial Sampling 2/2

Method

- build an R*-tree
- sample a set of objects from all leaf nodes



Structure of an R*-tree

Summarization

- Instead of selecting a subset of the input dataset, summarize the input data into a set of summaries.
- “Forget” the raw data, apply data mining algorithms upon the summaries afterwards

- Cluster feature vectors/ BIRCH algorithm
- Data bubbles

Micro-Clustering / BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [ZhaRamLiv96]

BIRCH overview

- Build a compact description of the dataset using cluster feature (CF) vector summaries
- Organize custer features in a tree structure (CF-tree)
- The leaves of the tree have a maximal expansion which control how tight these summaries are.
- Use leaves as data objects for data mining algorithms

- BIRCH is the first approach for clustering large scale data
- BIRCH introduced the idea of cluster feature vectors
  - also known as microclusters (more on this in the stream clustering lecture)

The cluster feature vector (CF) summary

Given N d-dimensional points in a cluster C, the cluster feature (CF) vector of C is defined as a triple:

$$CF = (N, \overrightarrow{LS}, SS)$$

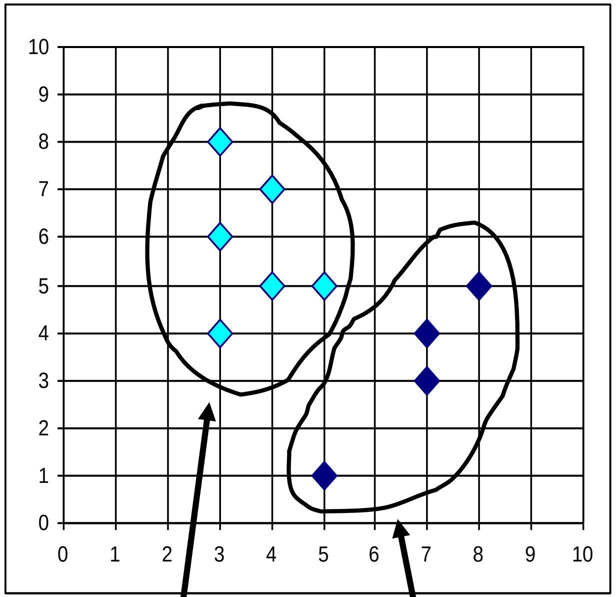
where

- N = |C|, the number of points in C
- $\overrightarrow{LS} = \sum_{i=1}^{N} \vec{X}_i$, the linear sum of the N data points
- $SS = \sum_{i=1}^{N} \vec{X}_i^{\,2} = \sum_{i=1}^{N} \langle X_i, X_i \rangle$, the square sum of the N data points

CF vector example

Example:

(3,4)
(4,5)
(5,5)
(3,6)
(4,7)
(3,8)



(5,1)
(7,3)
(7,4)
(8,5)

$CF_1 = (6, (22,35), 299)$

$CF_2 = (4, (27,13), 238)$

CF vector properties 1/2

The CF vector is not only *efficient,* as it compresses the input dataset, but also *accurate*, as it is sufficient to compute several measures we need for clustering.

- the centroid of C: $\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X_i}}{N} \Longrightarrow \frac{\overrightarrow{LS}}{N}$

- the radius of C (avg distance to the centroid):

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X_i} - \vec{X0})^2}{N}\right)^{\frac{1}{2}} \Longrightarrow \sqrt{\frac{SS}{n} - \left(\frac{LS}{n}\right)^2}$$

- the diameter of C (avg pairwise distance within a cluster)

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X_i} - \vec{X_j})^2}{N(N-1)}\right)^{\frac{1}{2}} \Longrightarrow$$

Homework!

CF vector properties 2/2

- CF additivity property: Let two disjoint clusters C_1 und C_2. The CF vector of the cluster that is formed by merging the two disjoint clusters, is:

  $$\mathrm{CF}(C_1 \cup C_2) = CF\,(C_1) + CF\,(C_2) = (N_1 + N_2,\ LS_1 + LS_2,\ QS_1 + QS_2)$$

- CF incremental property: The updated CF of a cluster C_1 after the addition of a new point p, is:

  $$\mathrm{CFT}(C_1 \cup p) = \mathrm{CFT}(C_1) + p$$

- Based on CFs several distance functions can be supported:

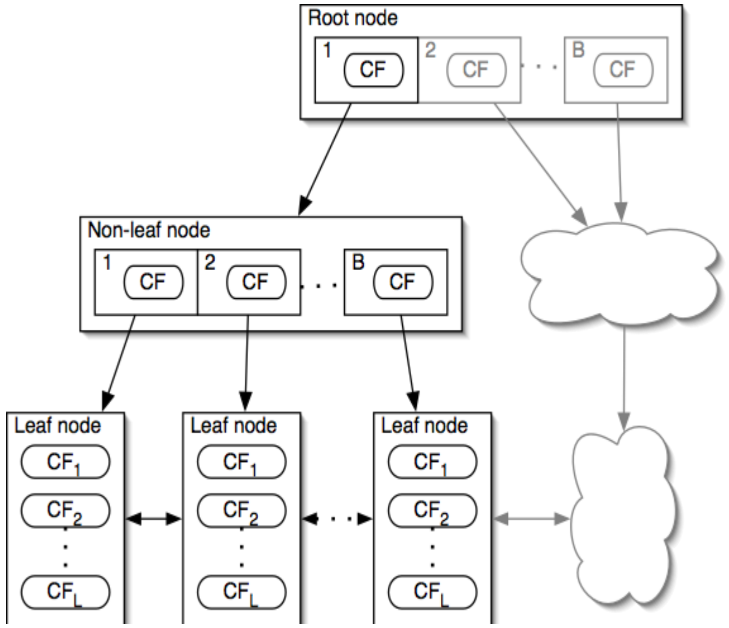
  $$D0 = ((\vec{X0}_1 - \vec{X0}_2)^2)^{\frac{1}{2}}$$ *Centroid Euclidean distance*

  $$D1 = |\vec{X0}_1 - \vec{X0}_2| = \sum_{i=1}^{d} |\vec{X0}_1^{(i)} - \vec{X0}_2^{(i)}|$$ *Centroid Manhattan distance*

- Why CFs?
  - Summarize info for a single cluster
  - Easy merge of clusters based on the additivity property
  - Easy incorporation of new points based on the incremental property

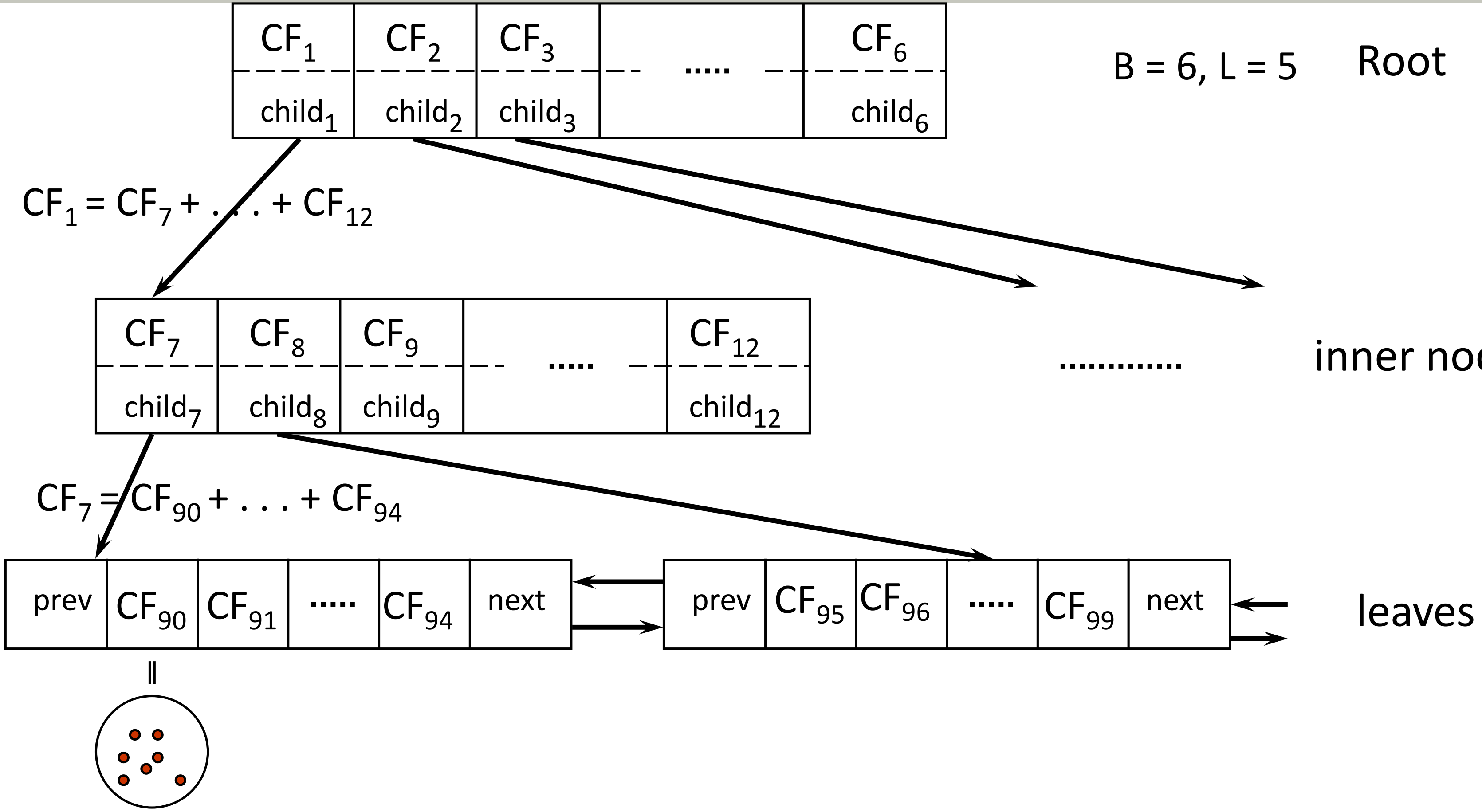
CF-tree

- A CF tree is a height balanced tree with 2 parameters
  - a branching factor B & a threshold T
  - A *non-leaf node*, contains at most B entries of the form [CF_i,$child_i$], i=1:B, where CF_i is the CF vector of the subcluster represented by the ith child.
  - A *leaf node* contains at most L entries of the form [CF_i].
    - The diameter (or radius) of all contained entries is less than the threshold T
    - Each leaf has two pointers previous and next to a allow sequential access



Source:[Miroslav11]

CF-tree example



CF_1 CF_2 CF_3 ..... CF_6
$child_1$ $child_2$ $child_3$ $child_6$
B = 6, L = 5
Root
$CF_1 = CF_7 + \ldots + CF_{12}$
CF_7 CF_8 CF_9 ..... CF_{12}
$child_7$ $child_8$ $child_9$ $child_{12}$
....
inner nodes
$CF_7 = CF_{90} + \ldots + CF_{94}$
prev CF_{90} CF_{91} ..... CF_{94} next
prev CF_{95} CF_{96} ..... CF_{99} next
leaves

A very compact representation of the dataset:
Each leaf entry is not a single point but a subcluster which absorbs many data points with diameter (or radius) under the threshold T.

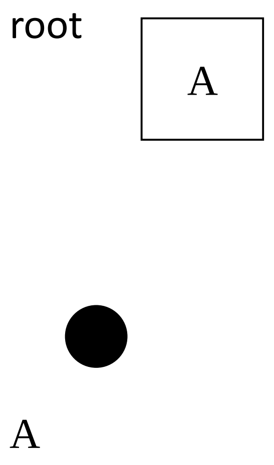
CF-tree construction

Construction of a CF-tree (analogue to constructing a B^+-tree)

- For each point p in the initial dataset ($CF_p=(1, p, p^2)$)
- *Identify the proper leaf node*: Starting from root and traversing the CF tree by choosing the closest child based on distance function (e.g., D0, D1)
- *Modifying the lead node:* Find its closest leaf entry L and check the threshold T in L after addition of p
- If T is not violated,
  - CF_p is absorbed into M
- else
  - A new entry for p is added to the leaf
  - If there is space for the new entry OK, otherwise, we must split the leaf node
    - select the pair of CFs having the largest distance as seeds
    - assign the remaining CFs to their closest seeds
- *Modifying the path to the leaf:* The addition of p should be reflected in the path

CF-tree example (from [Miroslav11]) 1/4

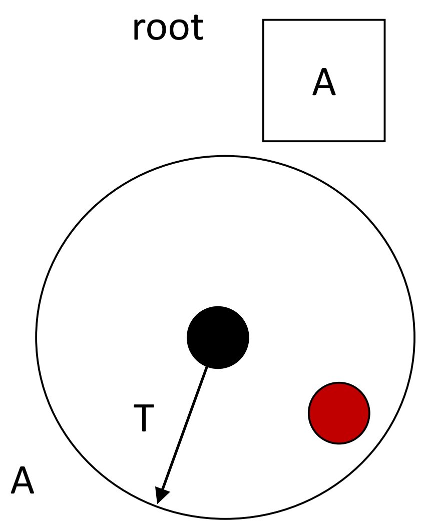
Put the first point in its own cluster A



CF-tree example 2/4

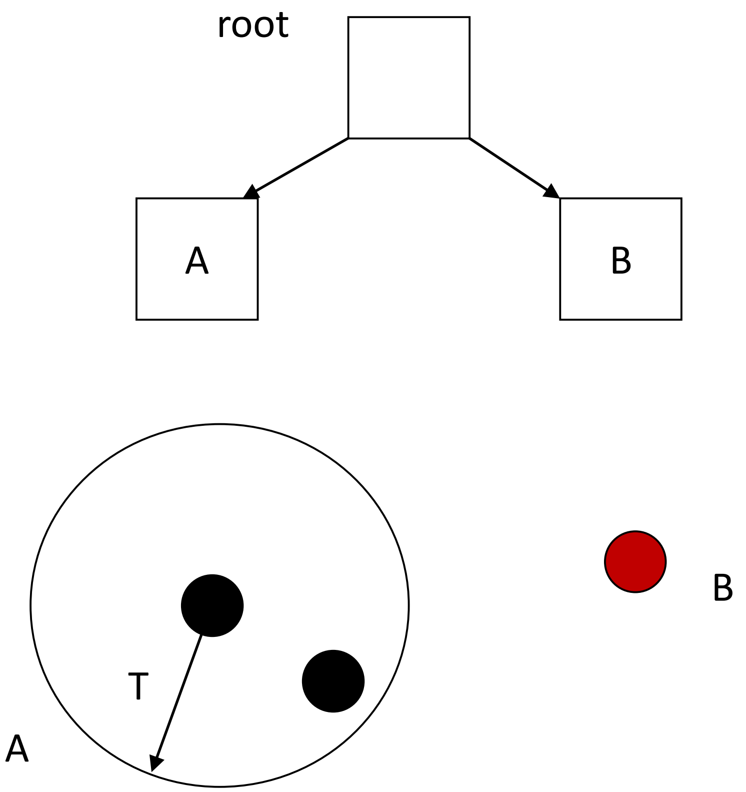
When a new point arrives, find the closest cluster (A) and check whether its radius exceeds the threshold *T*.

- If T is not violated, the new point is absorbed by the cluster (A)



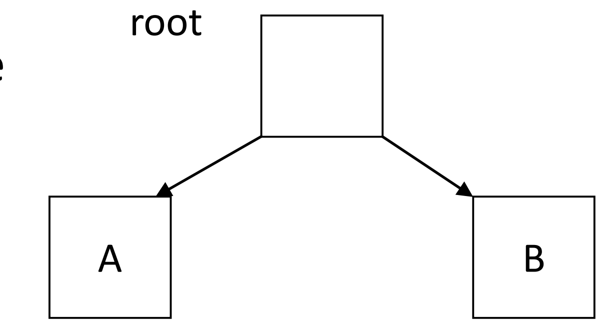
CF-tree example 3/4

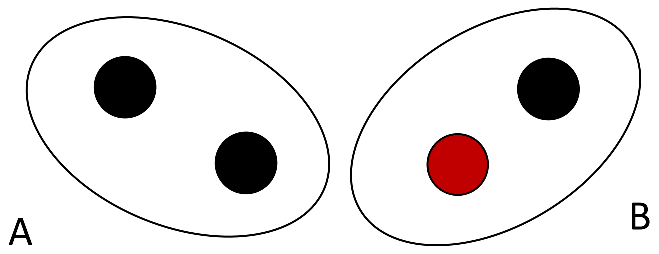
- If T is violated, the cluster is split
  into two clusters
  and the points
  are redistributed.



CF-tree example 4/4

At each node of the tree,
CF vectors are maintained which are adequate for all the node-related operations.





BIRCH: Clustering using clustering feature vectors

Phase 1

- construct a CF-tree by successively adding new points → B_1

Phase 2 [Optional]

- if B_1 is to big, scan its leaf entries and build smaller CF-tree (B_2) by merging subclusters and removing outliers

Phase 3

- Cluster all leaf entries through some clustering algorithm
- Each subcluster can be treated as a single point (the centroid)
    - Or, the cardinality of the clusters can be also considered
- The clustering algorithm can use special distance measures on CFs (e.g., D0, D1)

Cluster feature vectors discussion

Advantages:

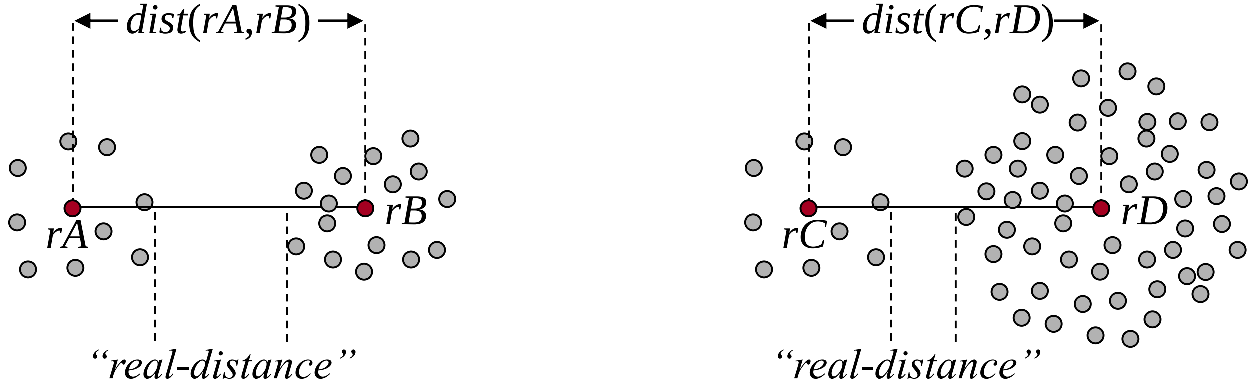
- Compression rate is adjustable
- The tree is a function of T: larger T→smaller tree
- Scans the whole dataset only once (to build the CF tree)
  - Additional passes are optional to refine the result.
- Handles outliers

Disadvantages:

- results depend on insertion order
- only suitable for numerical vector spaces
- More suitable for spherical clusters (uses radius/ diameter to control the boundary of a cluster)

Data bubbles motivation [BreKriKroSan01]

- CF vectors of BIRCH are effective for k-means type of clustering
- For hierarchical clustering though, their success is limited
- Hierarchical clustering is based on distances between points which are not represented well by the distances between representative objects, especially if the compression rate is high
  - The structural distortion problem
- Below, based only on centroids: dist(rA,rB)=dist(rC,rD).



Data bubbles definition

Data Bubbles: add more information to the representation

A generic definition for data bubbles:

Let X={X_i} be a set of *n* objects. The data bubble B w.r.t. X is defined as a tuple:

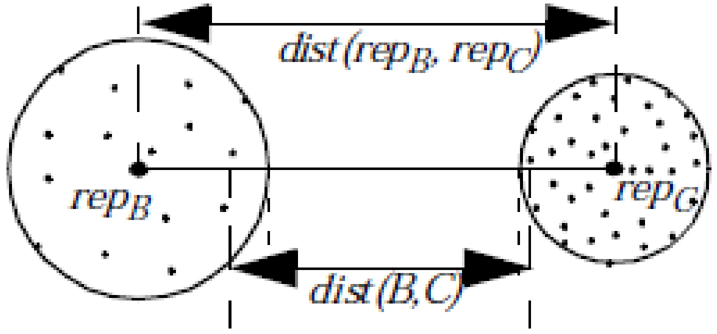
$$B=(\text{rep}, n, extent, nnDist)$$

- rep is a representative point for X (not necessarily a X_i)
- n is the number of points
- extent is a real number such that “most“ objects of X are located within a radius “extent“ around rep
- nnDist(k,B) is a function denoting the estimated avg kNN distance within X for some value k.

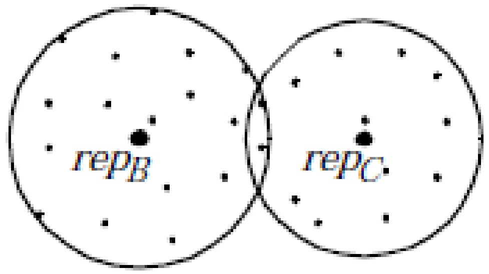
Distance between data bubbles

- Let B, C be two data bubbles, their distance is given as follows

$$dist(B,C) = \begin{cases} 0 & \text{if } B = C \\ dist(rep_B, rep_C) - (e_B + e_C) + nnDist(1, B) + nnDist(1, C) & \\ & \text{if } dist(rep_B, rep_C) - (e_1 + e_2) \geq 0 \\ max(nnDist(1, B), nnDist(1, C)) & \text{otherwise} \end{cases}$$



(a) non-overlaping Data Bubbles



(b) overlapping Data Bubbles

Data bubbles for Euclidean Vector Spaces

Definition: Let X={X_i} be a set of n objects. The data bubble B w.r.t. X is defined as a tuple:

$$B=(\mathrm{rep}, n, extent, nnDist)$$

- rep is the center of X

$$M = \left(\sum_{i=1}^{n} X_i \right) / n$$

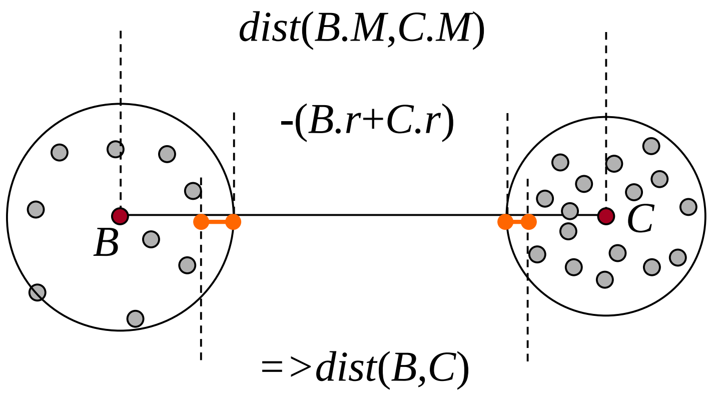
- extent is the radius of X

$$r = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \left(X_i - X_j \right)^2}{n \cdot (n-1)}}$$

- nnDist

$$nnDist(k, B) = \left(\frac{k}{n} \right)^d \bullet r$$

  – Expected *k*NN distance in X (assuming a uniform distribution of the points inside the sphere with center M and radius r)



Clustering using data bubbles

- Data bubbles can be generated in different ways
  - from m sampled objects and by assigning the rest instances to their closest objects
  - from the leaf nodes of a CF-tree
- Apply clustering over the data bubbles
  - e.g., a classical hierarchical clustering algorithm like single link
  - e.g., OPTICS

Summaries and sampling: discussion

- Often, the same results can be achieved on much smaller samples
- Both sampling and summarization try to approximate the data distribution by a smaller subset of the data
- It is important that the sample is representative of the input dataset
- There are similar approaches for instance selection in classification
  - Select samples from each class which allow to approximate the class margins
  - Samples being very “typical” for a class might be useful to learn a discrimination function of a good classifier.
  - Similar to the concept of support vectors in SVMs

DATABASE
SYSTEMS
GROUP

Literature

- [EstKriXu95] M. Ester, H.-P. Kriegel, X. Xu: *A Database Interface for Clustering in Large Spatial Databases,* In Proceedings of the 1st ACM International Conference on Knowledge Discovery and Data Mining (KDD), Montreal, QC, Canada: 94–99, 1995.
- [ZhaRamLiv96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny: *BIRCH: an efficient data clustering method for very large databases, SIGMOD Rec.* 25, 2.
- [Miroslav11] Faculty of Electrical Engineering, University of Belgrade, *The BIRCH Algorithm*, Davitkov Miroslav, 2011/3116.
- [BreKriKroSan01] Markus M. Breunig, Hans-Peter Kriegel, Peer Kröger, and Jörg Sander: *Data bubbles: quality preserving performance boosting for hierarchical clustering, SIGMOD Rec.* 30, 2 (May 2001), 79-90.