Solutions for higher Complexities

- if data mining algorithms has a super linear complexity parallel processing and hardware can help, but do not solve the problem
- runtimes can only be reduced by limiting the number of input objects
- solution:
  - reduce the input data to set of objects having a smaller cardinality
  - perform data mining on the reduced input set

⇒ Results may vary from using the complete data set
⇒ parallel processing can be used for this preprocessing step

Sampling

Idea: Select a limited subset from the input data.

- Random sampling: draw k times from the data set and remove the drawn the 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 the distribution w.r.t. to set of attributes (e.g., class labels)
⇒ compute how many instances for attribute value (combination of attribute values) should be contained in the sample
⇒ Partition the input data w.r.t. to the values/ value combinations
⇒ draw the computed amount from each partition
Index-based Sampling [Ester, Kriegel & Xu 1995]

- random sampling is problematic in spatial data
- use spatial index structures to estimate spatial distributions
  - index structures try to group similar objects (similar effect to clustering)
  - index structures are usually built up in an efficient way
  - allow fast access for similarity queries

Method

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

Structure of an R*-tree
Micro-Clustering

*BIRCH* [Zhang, Ramakrishnan & Linvy 1996]

Method

- Build a compact description of micro clusters (cluster features)
- organize cluster features in a tree structure (CF-tree)
- leaves of the tree have a maximal expansion
- data mining algorithms use the leaf nodes as data objects

- Birch is a hierarchical clustering approach
- the topology of the tree is dependent on the insertion order
- building up a Birch tree can be done in linear time

Basic concepts

- *Cluster Features* of a set $C \{X_1, \ldots, X_n\}$: $CF = (N, LS, SS)$
- $N = |C|$ cardinality of $C$
- $LS = \sum_{i=1}^{N} \bar{X}_i$ linear sum of the vectors $X_i$
- $SS = \sum_{i=1}^{N} \bar{X}_i^2 = \sum_{i=1}^{N} \langle X_i, X_i \rangle$ sum of squared vector lengths

CF can be used to compute:

- the centroid of $C$ (representing object)
- standard deviation of the distance from $X_1$ to the centroid
Micro-Clustering

Example:

\[(3,4) \quad (5,1)\]
\[(4,5) \quad (7,3)\]
\[(5,5) \quad (7,4)\]
\[(3,6) \quad (8,5)\]
\[(4,7) \quad \]
\[(3,8) \quad \]

\[\text{CF}_1 = (6, (22,35),299)\]
\[\text{CF}_2 = (4, (27,13),238)\]

Additivity theorem

For two disjunctive clusters \( C_1 \) and \( C_2 \) the following equation holds:

\[\text{CF}(C_1 \cup C_2) = \text{CF}(C_1) + \text{CF}(C_2) = (N_1 + N_2, LS_1 + LS_2, QS_1 + QS_2)\]

i.e., clusters can be computed incrementally and easily merged into one cluster.

**Distance between CFs** = distance between centroids

Definition

A CF-tree is a height balanced tree where all nodes are described by cluster features.
Properties of CF Trees

• Each inner node contains at most $B$ entries $[CF_i, child_i]$ where $CF_i$ is the CF vector of the $i^{th}$ subcluster child$_i$

• A lead node contains at most $L$ entries of the form $[CF_i]$.

• each leaf has two pointers previous and next to a allow sequential access

• for each lead node the diameter of all contained entries is less than the threshold $T$.

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

• transform the data set $p$ into a CF vector $CF_p=(1, p, p^2)$
• insert $CF_p$ into the subtree having the minimal distance
• when reaching the leaf level, $CF_p$ is inserted into the closest leaf
• if after insertion the diameter is still $< T$ then $CF_p$ is absorbed into the leaf
• else $CF_p$ is removed from the leaf and spawns a new leaf.
• if the parent node has more than $B$ entries, split the node:
  – select the pair of CFs having the largest distance seed CFs
  – assign the remaining CFs to the closer one of the seed CFs
Using Birch for Data Clustering

Phase 1
- construct a CF-tree by successively adding new vectors

Phase 2 (loop until number of leafes is small enough)
- if the CF-tree $B_1$ still contains to many leafes, adjust the threshold to $T_2 > T_1$
- Construct CF-tree $B_2$ w.r.t. $T_2$ by successively inserting the leaf CF’s of $B_1$

Phase 3
- Apply clustering algorithms
- Clustering algorithm can use special distance measures on CFs
Micro-Clustering

Discussion

advantages:
• compression rate is adjustable
• efficient method to build a representative sample
  – construction on secondary storages: $O(n \log n)$
  – construction in main memory CF-Baums: $O(n)$

disadvantages:
• only suitable for numerical vector spaces
• results depend on insertion order

Data Bubbles [Breunig, Kriegel, Kröger, Sander 2001]

Original DB and OPTICS-plot
1 mio. data points

OPTICS-plots for compressed data

Sampling

BIRCH

$\begin{array}{ccc}
 k=10,000 & k=1,000 & k=200 \\
\end{array}$
There are three problems making the result unstable:

1. Lost Objects
   - many objects are not in the plot (plot is too small)
2. Size Distortions
   - Cluster are too small or too large relative to other clusters
3. Structural Distortions
   - hierarchical cluster structures are lost

Solutions:

- Post processing Lost Objects and Size Distortions
  - use nn-classifier and replace all representative objects by the set of represented objects
- Data Bubbles can solve structural distortions

reasons for structural distortions:

- Distance between the original objects is only badly described by the distance between the centroids.

- the reachability distance of a representative objects does really approximate the reachability distance of the represented objects
Data Bubble: adds more information to representations

**Definition: Data Bubble**

- Data Bubble $B=(n, M, r)$ of the set $X=\{X_i\}$ containing $n$ objects

$$M = \frac{\sum_{i=1}^{n} X_i}{n}$$  

is the centroid of $X$.

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

is the radius of $X$.

- Expected $k$NN Distance of the objects $X_i$ in a data bubble (assuming a uniform distribution)

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

- Data Bubbles can either be generated from samples or CFs

**Micro-Clustering**

- **Definition: Distance between Data Bubbles**

$$\text{dist}(B.M, C.M)$$

- $(B.r + C.r)$

$=>$ $\text{dist}(B, C)$

- **Definition: Core- and reachability distance for data bubbles**

are defined in the same way as for points

- **Definition: virtual reachability distance of a data bubble**

  - expected $k$NN-distance within the the data bubble
  - better approximation of the reachability distance of the represented objects
Micro-Clustering

- Clustering using Data Bubbles:
  - Generate \( m \) Data Bubbles from \( m \) sample objects or CF-features
  - Cluster the set of data bubbles using OPTICS
  - Generate reachability plot:
    for each data bubble \( B \):
    - Plot the reachability distance \( B_{\text{reach}} \) (generated by the running OPTICS on the data bubbles)
    - For points being represented by \( B \), plot the virtual reachability distance of \( B \)

\[ B = (n_B, M_B, r_B) \]

reachability distance of \( B \)  virtual reachability distance of \( B \)

Results of using data bubbles based on sampling and CF

\( k = 10,000 \) \( k = 1,000 \) \( k = 200 \)
**Micro-Clustering**

**Speed-Up**

w.r.t to number of data objects

for 1 million data objects

**Discussion Micro-Clustering and Sampling**

- often the same results can be achieved on much smaller samples
- it is important that the data distributed is sufficiently represented by the data set
- Sampling and Micro-Clustering try to approximate the spatial data distribution by a smaller subset of the data
- there are similar approaches for classification instance selection:
  - 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
Literature


• 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


• Zhao W., Ma H., He Q.: Parallel K-Means Clustering Based on MapReduce, CloudCom, (pp 674-679) 2009


• Ene A., Im S., Moseley B.: Fast Clustering using MapReduce, 17th int. Conf. on Knowledge Discovery and Data Mining (KDD‘11), 2011


• Tian Zhang, Raghu Ramakrishnan, and Miron Livny: BIRCH: an efficient data clustering method for very large databases, SIGMOD Rec. 25, 2 (June 1996), 103-114. DOI=10.1145/235968.233324 http://doi.acm.org/10.1145/235968.233324