Clustering Techniques for Large Data Sets

From the Past to the Future

Alexander Hinneburg, Daniel A. Keim
University of Halle
Introduction

- **Application Example: Marketing**
  - Given:
    - Large data base of customer data containing their properties and past buying records
  - Goal:
    - Find groups of customers with similar behavior
    - Find customers with unusual behavior
Introduction

- **Application Example:**
  Class Finding in CAD-Databases
  - Given:
    - Large data base of CAD data containing abstract feature vectors (Fourier, Wavelet, ...)
  - Goal:
    - Find homogeneous groups of similar CAD parts
    - Determine standard parts for each group
    - Use standard parts instead of special parts (→ reduction of the number of parts to be produced)
Introduction

Problem Description

- Given:
  A data set with $N$ $d$-dimensional data items.

- Task:
  Determine a natural partitioning of the data set into a number of clusters (k) and noise.
Introduction

From the Past ... 

- Clustering is a well-known problem in statistics [Sch 64, Wis 69]
- more recent research in
  - machine learning [Roj 96],
  - databases [CHY 96], and
  - visualization [Kei 96] ...
Introduction

... to the Future

- **Effective** and **efficient** clustering algorithms for *large high-dimensional* data sets with *high noise level*

- Requires **Scalability** with respect to
  - the *number of data points* (**N**)
  - the *number of dimensions* (**d**)
  - the *noise level*
Overview

1. Introduction

2. Basic Methods
   2.1 k-Means
   2.2 Linkage-based Methods
   2.3 Kernel-Density Estimation Methods

3. Methods Improving the Effectiveness and Efficiency
   2.1 Model- and Optimization-based Approaches
   2.2 Density-based Approaches
   2.3 Hybrid Approaches

4. Summary and Conclusions

From the Past ...

... to the Future
K-Means [Fuk 90]

- Determine \( k \) prototypes (\( p \)) of a given data set
- Assign data points to nearest prototype
- Minimize distance criterion:
  \[
  \sum_{i=1}^{k} \sum_{j=1}^{N} d(p_i, x_j^i)
  \]
- Iterative Algorithm
  - Shift the prototypes towards the mean of their point set
  - Re-assign the data points to the nearest prototype
K-Means: Example
Expectation Maximization [Lau 95]

- Generalization of k-Means
  (⇒ probabilistic assignment of points to clusters)

- Basic Idea:
  - Estimate parameters of k Gaussians
  - Optimize the probability, that the mixture of parameterized Gaussians fits the data
  - Iterative algorithm similar to k-Means
Linkage-based Methods (from Statistics) [Boc 74]

- Single Linkage (Connected components for distance $d$)
Linkage-based Methods [Boc 74]

- Method of Wishart [Wis 69] (Min. no. of points: c=4)

  Reduce data set

  Apply Single Linkage
Kernel Density Estimation

Data Set

Influence Function: Influence of a data point in its neighborhood

Density Function: Sum of the influences of all data points

Density Function:

Sum of the influences of all data points

Influence Function:

Influence of a data point in its neighborhood
Influence Function

The influence of a data point \( y \) at a point \( x \) in the data space is modeled by a function \( f_B^y : \mathbb{F}^d \rightarrow \mathbb{R} \),

\[
e.g., \quad f_{\text{Gauss}}^y(x) = e^{-\frac{d(x,y)^2}{2\sigma^2}}.
\]

Density Function

The density at a point \( x \) in the data space is defined as the sum of influences of all data points \( x_i \), i.e.

\[
f_B^D(x) = \sum_{i=1}^{N} f_B^{x_i}(x)
\]
Kernel Density Estimation
Hierarchical Methods

- Single Linkage
- Complete Linkage
- Average Linkage / Centroid Method
  (see also BIRCH)

Diversive: top-down
- Find the most inhomogenius cluster and split

Agglomerative: bottom-up
- Find the nearest pair of clusters and merge
Single Linkage

- Distance between clusters (nodes):
  \[ Dist(C_1, C_2) = \min_{p \in C_1, q \in C_2} \{ dist(p, q) \} \]

- Merge Step: union the two subset of data points

- A single linkage hierarchy can be constructed using the minimal spanning tree
Example: Single Linkage
Complete Linkage

- Distance between clusters (nodes): 
  \[ Dist(C_1, C_2) = \max_{p \in C_1, q \in C_2} \{ \text{dist}(p, q) \} \]
- Merge Step: union the two subset of data points
- Each cluster in a complete linkage hierarchy corresponds a complete subgraph
Example: Complete Linkage
Average Linkage / Centroid Method

- Distance between clusters (nodes):
  \[
  \text{Dist}_{\text{avg}}(C_1, C_2) = \frac{1}{\#(C_1) \cdot \#(C_2)} \sum_{p \in C_1} \sum_{q \in C_2} \text{dist}(p, q)
  \]

- \[
  \text{Dist}_{\text{mean}}(C_1, C_2) = \text{dist}[\text{mean}(C_1), \text{mean}(C_2)]
  \]

- Merge Step:
  - union the two subset of data points
  - construct the mean point of the two clusters
Scalability Problems

- Effectiveness degenerates
  - with dimensionality (d)
  - with noise level

- Efficiency degenerates
  - (at least) linearly with no of data points (N) and
  - exponentially with dimensionality (d)
Scaling Up Cluster Algorithms

- Sampling Techniques [EKX 95]
- Bounded Optimization Techniques [NH 94]
- Indexing Techniques [BK 98]
- Condensation Techniques [ZRL 96]
- Grid-based Techniques [SCZ 98, HK 98]
Indexing [BK 98]

- Cluster algorithms and their index structures
  - BIRCH: CF-Tree [ZRL 96]
  - DBSCAN: R*-Tree [Gut 84], X-Tree [BKK 96]
  - STING: Grid / Quadtree [WYM 97]
  - WaveCluster: Grid / Array [SCZ 98]
  - DENCLUE: B⁺-Tree, Grid / Array [HK 98]
Methods for Improving the Effectiveness and Efficiency

- Model- and Optimization-Based Approaches
- Density-Based Approaches
- Hybrid Approaches
Model- and Optimization-based Methods

- K-Means [Fuk 90]
- Expectation Maximization [Lau 95]
- CLARANS [NH 94]
- Focused CLARANS [EKX 95]
- Self-Organizing Maps [KMS+ 91, Roj 96]
- Growing Networks [Fri 95]
- PROCLUS [APW+ 99]
CLARANS [NH 94]

- Medoid Method:
  - Medoids are special data points
  - All data points are assigned to the nearest medoid

- Optimization Criterion:

\[
average\_distance(c) = \sum_{m_i \in M} \sum_{o \in cluster(m_i)} dist(o, m_i)
\]
Bounded Optimization [NH 94]

- CLARANS uses two bounds to restrict the optimization: \textit{num\_local}, \textit{max\_neighbor}

- Impact of the Parameters:
  - \textit{num\_local} → Number of iterations
  - \textit{max\_neighbors} → Number of tested neighbors per iteration
Graph Interpretation:
- Search process can be symbolized by a graph
- Each node corresponds to a specific set of medoids
- The change of one medoid corresponds to a jump to a neighboring node in the search graph

Complexity Considerations:
- The search graph has \( \binom{N}{k} \) nodes and each node has \( N \times k \) edges
- The search is bound by a fixed number of jumps (\textit{num\_local}) in the search graph
- Each jump is optimized by randomized search and costs \textit{max\_neighbor} scans over the data (to evaluate the cost function)
Sampling [EKX 95]

- R*-Tree Sampling
- Comparison of Effectiveness versus Efficiency (example CLARANS)

Graph:
- Bar chart showing average distance with and without focus for different maximum numbers of neighbors.
- Another bar chart showing runtime in seconds for different maximum numbers of neighbors.
AI Methods

- Self-Organizing Maps [Roj 96, KMS 91]
  - Fixed map topology
    (grid, line)
AI Methods

- Growing Networks [Fri 95]
  - Iterative insertion of nodes
  - Adaptive map topology
Density-based Methods

- Linkage-based Methods [Boc 74]
- Kernel-Density Estimation [Sil 86]
- BIRCH [ZRL 96]
- DBSCAN [EKS+ 96]
- DBCLASD [XEK+ 98]
- STING [WYM 97]
- Hierarchical Grid Clustering [Sch 96]
- WaveCluster [SCZ 98]
- DENCLUE [HK 98]
- OPTICS [ABKS 99]
BIRCH [ZRL 96]

Clustering in BIRCH

Data

Phase 1: Load into memory by building a CF tree

Initial CF tree

Phase 2 (optional): Condense into desirable range by building a smaller CF tree

smaller CF tree

Phase 3: Global Clustering

Good Clusters

Phase 4: (optional and off line) : Cluster Refining

Better Clusters
Basic Idea of the CF-Tree

- Condensation of the data \( \{ \vec{X}_i \} \) using CF-Vectors
  \[
  \text{CF} = (N, \bar{L}_S, SS)
  \]
  \[
  \bar{L}_S = \sum_{i=1}^{N} \vec{X}_i, \quad SS = \sum_{i=1}^{N} \vec{X}_i^2
  \]

- CF-tree uses sum of CF-vectors to build higher levels of the CF-tree
BIRCH

Insertion algorithm for a point $x$:

1. Find the closest leaf $b$
2. If $x$ fits in $b$, insert $x$ in $b$; otherwise split $b$
3. Modify the path for $b$
4. If tree is too large, condense the tree by merging the closest leaves
BIRCH

CF-Tree Construction

1. Increase T.
2. Rebuild CF tree t2 of new T from CF tree t1:
   - if a leaf entry of t1 is potential outlier and disk space available,
     write to disk; otherwise use it to rebuild t2.
3. t1 ← t2.

Out of memory

Result?

Finish scanning data

Out of disk space

Re-absorb potential outliers into t1

Re-absorb potential outliers into t1

Continue scanning data and insert to t1

Start CF tree t1 of Initial T
Condensing Data

- **BIRCH [ZRL 96]:**
  - Phase 1-2 produces a condensed representation of the data (CF-tree)
  - Phase 3-4 applies a separate cluster algorithm to the leafs of the CF-tree

- Condensing data is crucial for efficiency
Problems of BIRCH

- Centroid Method with fixed order of the points
Clusters are defined as Density-Connected Sets (wrt. MinPts, $\varepsilon$)

- **(a)** $p$: border point
  - $q$: core point

- **(b)** $p$: directly density-reachable from $q$

- **(c)** $p$: density-reachable from $q$
  - $q$: not density-reachable from $p$

- **(d)** $p$ and $q$: density-connected to each other by $o$
DBSCAN

- For each point, DBSCAN determines the $\epsilon$-environment and checks, whether it contains more than MinPts data points.
- DBSCAN uses index structures for determining the $\epsilon$-environment.
- Arbitrary shape clusters found by DBSCAN.
**DBCLASD** [XEK+ 98]

- Distribution-based method
- Assumes arbitrary-shape clusters of uniform distribution
- Requires no parameters
- Provides grid-based approximation of clusters

Before the insertion of point p

![Before insertion](image1)

After the insertion of point p

![After insertion](image2)
Definition of a cluster $C$ based on the distribution of the NN-distance ($NNDistSet$):

1. 
2. 
3. 

The expected and the observed distance distributions for cluster 1
DBCLASD

- Step (1) uses the concept of the $\chi^2$-test

- Incremental augmentation of clusters by neighboring points (order-depended)
  - unsuccessful candidates are tried again later
  - points already assigned to some cluster may switch to another cluster
DBSCAN / DBCLASD

- DBSCAN and DBCLASD use index structures to speed-up the $\varepsilon$-environment or nearest-neighbor search
- the index structures used are mainly the R-tree and variants
R-Tree: [Gut 84]
The Concept of Overlapping Regions
Variants of the R-Tree

**Low-dimensional**
- R⁺-Tree [SRF 87]
- R*-Tree [BKSS 90]
- Hilbert R-Tree [KF94]

**High-dimensional**
- TV-Tree [LJF 94]
- X-Tree [BKK 96]
- SS-Tree [WJ 96]
- SR-Tree [KS 97]
Effects of High Dimensionality

Location and Shape of Data Pages

- Data pages have large extensions
- Most data pages touch the surface of the data space on most sides
The X-Tree [BKK 96] (eXtended-Node Tree)

- **Motivation:** Performance of the R-Tree degenerates in high dimensions
- **Reason:** overlap in the directory
The X-Tree

- X-tree avoids overlap in the directory by using
  - an overlap-free split
  - the concept of supernodes

![Diagram of X-Tree]

- Supernodes
- Normal Directory Nodes
- Data Nodes
Speed-Up of X-Tree over the R*-Tree

Point Query

10 NN Query
Effects of High Dimensionality

Selectivity of Range Queries

- The selectivity depends on the volume of the query

\[ e = \sqrt[d]{Vol_{cube}} \]

\( \Rightarrow \) no fixed \( \varepsilon \)-environment (as in DBSCAN)

selectivity = 0.1 %
Effects of High Dimensionality

Selectivity of Range Queries

- In high-dimensional data spaces, there exists a region in the data space which is affected by ANY range query (assuming uniformly distributed data).

⇒ difficult to build an efficient index structure
⇒ no efficient support of range queries (as in DBSCAN)
**STING** [WYM 97]

- Uses a quadtree-like structure for condensing the data into grid cells
- The nodes of the quadtree contain statistical information about the data in the corresponding cells
- STING determines clusters as the density-connected components of the grid
- STING approximates the clusters found by DBSCAN
Hierarchical Grid Clustering [Sch 96]

- Organize the data space as a grid-file
- Sort the blocks by their density
  \[ DB = \frac{p_B}{V_B} \quad \rightarrow \quad \langle B_1', B_2', \ldots, B_b' \rangle \]
- Scan the blocks iteratively and merge blocks, which are adjacent over a (d-1)-dim. hyperplane.
- The order of the merges forms a hierarchy
WaveCluster [SCZ 98]

- Clustering from a signal processing perspective using wavelets

*Input:* Multidimensional data objects’ feature vectors
*Output:* clustered objects

1. Quantize feature space, then assign objects to the units.
2. Apply wavelet transform on the feature space.
3. Find the connected components (clusters) in the subbands of transformed feature space, at different levels.
4. Assign label to the units.
5. Make the lookup table.
6. Map the objects to the clusters.
WaveCluster

- Grid Approach
  - Partition the data space by a grid → reduce the number of data objects by making a small error
  - Apply the wavelet-transformation to the reduced feature space
  - Find the connected components as clusters

- Compression of the grid is crucial for the efficiency

- Does not work in high dimensional space!
WaveCluster

- Signal transformation using wavelets

- Arbitrary shape clusters found by WaveCluster
Hierarchical Variant of WaveCluster [SCZ 98]

- WaveCluster can be used to perform multiresolution clustering
- Using coarser grids, clusters start to merge
DENCLUE

Data Set

Influence Function: Influence of a data point in its neighborhood

Density Function: Sum of the influences of all data points
Density Attractor/Density-Attracted Points

- local maximum of the density function
- density-attracted points are determined by a gradient-based hill-climbing method
Center-Defined Cluster

A center-defined cluster with density-attractor \( x^* \) (\( f_B^D(x^*) > \xi \)) is the subset of the database which is density-attracted by \( x^* \).

Multi-Center-Defined Cluster

A multi-center-defined cluster consists of a set of center-defined clusters which are linked by a path with significance \( \xi \).
DENCLUE

Impact of different Significance Levels (\(\xi\))
DENCLUE

*Choice of the Smoothness Level (\(\sigma\))*

Choose \(\sigma\) such that *number of density attractors* is constant for a long interval of \(\sigma\)!
Building Hierarchies ($\sigma$)
DENCLUE

Noise Invariance

Assumption: Noise is uniformly distributed in the data space.

Lemma: The density-attractors do not change when increasing the noise level.

Idea of the Proof:
- partition density function into signal and noise

\[ f^D(x) = f^{Dc}(x) + f^N(x) \]
- density function of noise approximates a constant \((f^N(x) \approx \text{const.})\)
DENCLUE

Noise Invariance
DENCLUE

*Noise Invariance*
DENCLUE Algorithm [HK 98]

**Basic Idea**

- Use *Local Density Function* which approximates the Global Density Function
- Use *CubeMap Data Structure* for efficiently locating the relevant points
DENCLUE

Local Density Function

Definition
The local density $\hat{f}_B^D(x)$ is defined as

$$\hat{f}_B^D(x) = \sum_{x_i \in \text{near}(x)} f_B^{x_i}(x).$$

Lemma (Error Bound)
If $\text{near}(x) = \{x_i \in D \mid d(x, x_i) \leq k\sigma\}$, the error is bound by:

$$\text{Error} = \sum_{x_i \in D, d(x_i, x) > k\sigma} e^{-\frac{d(x, x_i)^2}{2\sigma^2}} \leq \| \{x_i \in D \mid d(x, x_i) > k\sigma\}\| \cdot e^{-\frac{k^2}{2}}$$
CubeMap

Data Structure based on regular cubes for storing the data and efficiently determining the density function
DENCLUE Algorithm

DENCLUE \((D, \sigma, \xi)\)

(a) \(MBR \leftarrow \text{Determine}MBR(D)\)
(b) \(C_p \leftarrow \text{DetPopCubes}(D, MBR, \sigma)\)
(c) \(C_{sp} \leftarrow \text{DetHighlyPopCubes}(C_p, \xi_c)\)
(d) \(\text{map}, C_r \leftarrow \text{ConnectMap}(C_p, C_{sp}, \sigma)\)
(e) \(\text{clusters} \leftarrow \text{DetDensAttractors}(\text{map}, C_r, \sigma, \xi)\)
Effects of High Dimensionality

Number of Surfaces and Grid Cells

- Number of $k$-dimensional surfaces in a $d$-dimensional hypercube?
  \[ \binom{d}{k} \cdot 2^{(d-k)} \]

- Number of grid cells resulting from a binary partitioning?
  \[ 2^d \]

\[ \Rightarrow \] grid cells cannot be stored explicitly
\[ \Rightarrow \] most grid cells do not contain any data points
The Surface is Everything

- Probability that a point is closer than 0.1 to a \((d-1)\)-dimensional surface

\(\Rightarrow\) no of directions (from center) increases exponentially
Effects of High Dimensionality

Number of Neighboring cells

- Probability that Cutting Planes partition clusters increases

⇒ cluster can not be identified using the grid
Hybrid Methods

- CLIQUE [AGG+ 98]
- OptiGrid [HK 99]
- ...
CLIQUE [AGG+ 98]

- Subspace Clustering
- Monotonicity Lemma: If a collection of points $S$ is a cluster in a $k$-dimensional space, then $S$ is also part of a cluster in any $(k-1)$-dimensional projection of this space.
- Bottom-up Algorithm for determining the projections
OptiGrid [HK 99]

- Optimal Grid Partitioning:

Generalized Grid

Recursive Partitioning
Summary and Conclusions

- A number of effective and efficient Clustering Algorithms is available for small to medium size data sets and small dimensionality.

- **Efficiency** suffers severely for large dimensionality \((d)\).

- **Effectiveness** suffers severely for large dimensionality \((d)\), especially in combination with a high *noise level*. 
Open Research Issues

- Efficient Data Structures for large $N$ and large $d$
- Clustering Algorithms which work effectively for large $N$, large $d$ and large Noise Levels
- Integrated Tools for an Effective Clustering of High-Dimensional Data (combination of automatic, visual and interactive clustering techniques)
References

[CHY 96] Ming-Syan Chen, Jiawei Han, Philip S. Yu: *Data Mining: An Overview from a Database Perspective*. TKDE 8(6), pp. 866-883, 1996.


• [MG 93] R. Mehrotra, J. Gary, Feature-Based Retrieval of Similar Shapes, Proc. 9th Int. Conf. on Data Engeneering, April 1993.

• [NH 94] R.T. Ng, J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, Proc. 20th Int. Conf. on Very Large Data Bases, pp. 144-155, 1994.


