Advances in Clustering and Applications

Alexander Hinneburg
Institute of Computer Science,
University of Halle, Germany
hinneburg@informatik.uni-halle.de

Daniel A. Keim
Computer & Information Science
University of Constance, Germany
keim@inf.uni-konstanz.de

1 Description

Cluster analysis is one of the basic techniques which is often applied for analyzing large data sets. Originating from the area of statistics, most cluster analysis algorithms have originally been developed for relatively small data sets. In the early years of KDD research clustering algorithms have been improved to efficiently work on large data sets. However, in the last five years appeared a number of advanced topics related to clustering. The advanced topics include clustering with constraints, projected clustering, outlier detection, interactive clustering, database technology for clustering and categorical clustering.

The main goal of the tutorial is to provide an overview of the state-of-the-art in cluster discovery methods for large databases, covering well-known clustering methods from related fields such as statistics, pattern recognition, and machine learning, as well as to discuss the new topics related to clustering. The target audience of the tutorial are newcomers as well as experienced KDD researchers, who are interested in the state-of-the art of cluster discovery methods and applications. The tutorial especially addresses people from academia who are interested in developing new clustering algorithms, and people from industry who want to apply cluster discovery methods in analyzing large databases.

The tutorial is structured as follows: First, we give a brief motivation for clustering from the perspective of modern data mining applications. We discuss important design decisions and explain the interdependencies with the properties of data. In the second section, we introduce a variety of clustering methods developed in the early years of KDD research. The third section covers a large number of advanced topics related to clustering. Last we present some applications where clustering has been successfully used. The tutorial concludes with a discussion of open problems and future research issues.

2 Outline of the Tutorial

In the following outline, many references of clustering techniques and underlying index structures are included.

1. Introduction

   a) Motivation
      - the need for cluster discovery in the KDD process
      - the role of cluster discovery in the KDD process

   b) Properties of the Data

- data characteristics and their impact on the clustering methods
- pre-analysis (e.g., hypothesis generation based on visualization[Kei02])

c) Basic Definitions
- Density Estimation [Sil86, Sco92]
- Classification of the Clustering Approaches

2. Basic Clustering Methods

a) Model- and Optimization-Based Approaches
- K-Means [DH73, Fuk90] / EM [KMN97, Lau95], CLARANS [NH94, EKX95], LBG-U [Fri97], K-Harmonic Means [ZHD99, ZHD00]
- Self-Organizing Maps [KMSK91, Roj96]
- Growing Networks [MS91, Mar93, MS94, Fri95]

b) Linkage-Based Methods
- Linkage-based Methods from Statistics [DH73, Boc74]: Complete-, Single-, Centroid- Linkage; Method of Wishart [Wis69]
- DBSCAN [EKSX96], DBCLASD[XEKS98], OPITCS [ABKS99]
- BIRCH [ZRL96]

c) Density-Based Methods
- STING (+) [WYM97, WYM99]
- Hierarchical Grid Clustering [Sch96]
- Wavelet Cluster [SCZ98]
- DENCLUE [HK98]

3. Advanced Topics in Clustering

a) Clustering with Constraints
- Constraint-based clustering [TNLH01]
- Spatial Clustering with Obstacles [THH01, ECL01]

b) Projected Clustering
- Subspace Clustering CLIQUE [AGGR98]
- OptiGrid [HK99]
- Projected Clustering: ProClus [APW+99] and OrClus [AY00]
- DOC: a Monte Carlo algorithm for fast projective clustering [PJAM02]

c) Outlier Detection
- Distance Based Outliers [KN98, KNT00, RRS00]
- Local Outliers[BKNS99, BKNS00, JTH01, PKGF03]
- Projected Outliers [AY01]

d) Database Technology for Clustering
- Index Structures[BBK01, Gut84, SRF87, BKSS90, KF94, LJF94, WJ96, BKK96, KS97, WSB98]
- SQL Extentsions [WZ00] and SQL-EM [OC00]
- Density Estimation, Aggregation, Data Bubbles [BKKS01, HLH03, ZS03]

e) Categorical Clustering
- ROCK [GRS99]
- STIRR [GKR98]
- CACTUS [GGR99]

f) Interactive Clustering
- Image is Everything [AKN01]

- HD-Eye [HWK99, HKW03]

4. Applications of Clustering
   a) Images [KC03]
   b) Data from BioInformatics [CC00, YWWY03, YWWY02, GE02]
   c) GeoInformatics [SHDZ02]
5. Conclusion
   a) Open Problems
   b) Future Research Issues

Biography

Alexander Hinneburg
is working in the areas of data mining, databases and bioinformatics. He developed and published several algorithms and methods in the context of clustering and visual similarity search. He has given tutorials on clustering at SIGMOD'99, KDD'99 and PKDD'00 and has been tutorial chair of the KDD conference in 2002. He served as (external) referee for a number of conferences including VLDB, SIGMOD and InfoVis as well as referee for the journals IEEE TKDE, IEEE PAMI, IEEE TVCG, Kluwer Journal on Data Mining and Knowledge Discovery and ACM TIS.

He received his diploma (equivalent to an MS degree) in Computer Science from the Martin-Luther-University of Halle in 1997 and his Ph.D. in 2003. Currently he is working in the database group of the Martin-Luther-University of Halle, Germany.

Daniel A. Keim
is working in the area of data mining and information visualization, as well as similarity search and indexing in multimedia databases. He has published extensively in these area and has given tutorials on related issues at several large conferences including Visualization, SIGMOD, VLDB, and KDD; he has been program co-chair of the KDD conference in 2002 and of the IEEE Information Visualization Symposia in 1999 and 2000; and he is editor of IEEE Trans. on Knowledge and Data Engineering, IEEE Trans. on Visualization and Computer Graphics, and Palgrave's Information Visualization Journal.

Daniel Keim received his diploma (equivalent to an MS degree) in Computer Science from the University of Dortmund in 1990 and his Ph.D. in Computer Science from the University of Munich in 1994. He has been assistant professor in the CS department of the University of Munich, associate professor in the CS department of the Martin-Luther-University Halle, and he is currently full professor and head of the database and visualization group in the CS department of the University of Constance, Germany.

References

[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999.

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, 1998, Seattle, Washington, USA*, pages 94–105. ACM Press, 1998.

[AKN01] Amihood Amir, Reuven Kashi, and Nathan S. Netanyahu. Analyzing quantitative databases: Image is everything. In *The VLDB Journal*, pages 89–98, 2001.

[APW+99] Charu C. Aggarwal, Cecilia Magdalena Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 61–72. ACM Press, 1999.

[AY00] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 70–81. ACM, 2000.

[AY01] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *SIGMOD'2001, Proc. of the ACM Intl Conf. on Management of Data*, pages 427–438. ACM Press, 2001.

[BBK01] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3):322–373, 2001.

[BKK96] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The x-tree : An index structure for high-dimensional data. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 28–39. Morgan Kaufmann, 1996.

[BKKS01] Markus M. Breunig, Hans-Peter Kriegel, Peer Kröger, and Jörg Sander. Data bubbles: quality preserving performance boosting for hierarchical clustering. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 79–90. ACM Press, 2001.

[BKNS99] M. M. Breunig, H.P. Kriegel, R. Ng, and J. Sander. Optics of: Identifying local outliers. In *In Lecture Notes in Computer Science*, volume 1704, pages 262–270. Springer, 1999.

[BKNS00] S. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD'2000, Proc. of the ACM Int. Conf. on Management of Data*, pages 93–104. ACM Press, 2000.

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pages 322–331. ACM Press, 1990.

[Boc74] H. H. Bock. *Autmatic Classification*. Vandenhoeck and Ruprecht, Göttingen, 1974.

[CC00] Yizong Cheng and George M. Church. Biclustering of expresssion data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, ISMB 2000*. AAAI, 2000.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classifaication and Scene Analysis*. Wiley and Sons, New York, 1973.

[ECL01] Vladimir Estivill-Castro and Ickjai Lee. Fast spatial clustering with different metrics and in the presence of obstacles. In *ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems, Atlanta, GA, USA, November 9-10, 2001*, pages 142–14. ACM, 2001.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96, Proc. of the 2nd Intl Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[EKX95] M. Ester, H.P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In *SSD'95, 4th Int. Symp. on Large Spatial Databases, August 6-9, 1995, Portland, Maine, Lecture Notes in Computer Science Vol. 951*, pages 67–82. Springer, 1995.

[Fri95] B. Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7:625–632, 1995.

[Fri97] B. Fritzke. The lbg-u method for vector quantization - an improvement over lbg inspired from neural networks. *Neural Processing Letters*, 5(1), 1997.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, 1990.

[GE02] A.P. Gasch and M.B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11):1–22, 2002.

[GGR99] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Cactus - clustering categorical data using summaries. In *KDD'99, Proc. of the Fifth Int'l Conf. on Knowledge Discovery and Data Mining*, pages 73–83. ACM Press, 1999.

[GKR98] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 311–322. Morgan Kaufmann, 1998.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 512–521. IEEE Computer Society, 1999.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57. ACM Press, 1984.

[HK98] A. Hinneburg and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD'98, Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 58–65. AAAI Press, 1998.

[HK99] Alexander Hinneburg and Daniel A. Keim. Clustering methods for large databases: From the past to the future. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, page 509. ACM Press, 1999.

[HKW03] A. Hinneburg, D. A. Keim, and M. Wawryniuk. Using projections to visually cluster high–dimensional data. *IEEE Computing in Science & Engineering*, 5(2):14–25, 2003.

[HLH03] Alexander Hinneburg, Wolfgang Lehner, and Dirk Habich. Combi-operator: Database support for data mining applications. In *VLDB'03, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany, DE*, 2003. to appear.

[HWK99] A. Hinneburg, M. Wawryniuk, and D. A. Keim. Hdeye: Visual mining of high-dimensional data. *Computer Graphics & Applications Journal*, 19(5):22–31, September/October 1999.

[JTH01] Wen Jin, Anthony Tung, and Jiawei Han. Mining top-n local outliers in large databases. In *Proceedings of the seventh conference on Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*, pages 293 – 298. ACM Press, 2001.

[KC03] Deok-Hwan Kim and Chin-Wan Chung. Qcluster: relevance feedback using adaptive clustering for content-based image retrieval. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 599–610. ACM Press, 2003.

[Kei02] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 8(1):1–8, January–March 2002.

[KF94] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 500–509. Morgan Kaufmann, 1994.

[KMN97] M. Kearns, Y. Mansour, and A. Ng. An informationtheoretic analysis of hard and soft assignment methods for clustering. In *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence*, page 282293. Morgan Kaufmann, 1997.

[KMSK91] T. Kohonen, K. Mkisara, O. Simula, and J. Kangas. *Artificial Networks*. Amsterdam, 1991.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 392–403. Morgan Kaufmann, 1998.

[KNT00] Edwin M. Knorr, Raymond T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.

[KS97] Norio Katayama and Shin'ichi Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 369–380. ACM Press, 1997.

[Lau95] S. L. Lauritzen. The em algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[LJF94] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos. The tv-tree: An index structure for high-dimensional data. *VLDB Journal*, 3(4):517–542, 1994.

[Mar93] T. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks*, pages 427–434, Amsterdam, 1993. Springer.

[MS91] T. Martinetz and K. J. Schulten. *A Neural-Gas Network Learns Topologies*, pages 397–402. Elsevier Science Publishers, North-Holland, 1991.

[MS94] T. Martinetz and K. J. Schulten. Topology representing networks. *Neural Networks*, 7(3):507–522, 1994.

[NH94] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 144–155. Morgan Kaufmann, 1994.

[OC00] Carlos Ordonez and Paul Cereghini. SQLEM: fast clustering in SQL using the EM algorithm. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 559–570. ACM Press, 2000.

[PJAM02] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 418–427. ACM Press, 2002.

[PKGF03] Sprios Papadimitriou, Hirozuki Kitagawa, Phillip Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *ICDE'03, Proc. of the 19th Int. Conf. on Data Engeneering*, pages 315–326. IEEE Computer Society, 2003.

[Roj96] R. Rojas. *Neural Networks A Systematic Introduction*. Springer, Berlin, 1996.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD'2000, Proc. of the ACM Int. Conf. on Management of Data*, pages 427–438. ACM Press, 2000.

[Sch96] Erich Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *In Proc. 13th Int. Conf. on Pattern Recognition, volume 2*, pages 101–105, Vienna, Austria, October 1996. IEEE Computer Society Press.

[Sco92] D.W. Scott. *Multivariate Density Estimation*. Wiley and Sons, 1992.

[SCZ98] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 428–439. Morgan Kaufmann, 1998.

[SHDZ02] Shashi Shekhar, Yan Huang, Judy Djugash, and Changqing Zhou. Vector map compression: a clustering approach. In *Proceedings of the tenth ACM international symposium on Advances in geographic information systems*, pages 74–80. ACM Press, 2002.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[SRF87] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In Peter M. Stocker, William Kent, and Peter Hammersley, editors, *VLDB'87, Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England*, pages 507–518. Morgan Kaufmann, 1987.

[THH01] A. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. In *17th International Conference on Data Engineering (ICDE' 01)*, pages 359–367. IEEE, 2001.

[TNLH01] Anthony K. H. Tung, Raymond T. Ng, Laks V. S. Lakshmanan, and Jiawei Han. Constraint-based clustering in large databases. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 405–419. Springer, 2001.

[Wis69] D. Wishart. Mode analysis: A generalisation of nearest neighbor, which reducing chaining effects. pages 282–312. A. J. Cole, 1969.

[WJ96] David A. White and Ramesh Jain. Similarity indexing: Algorithms and performance. In *Storage and Retrieval for Image and Video Databases, SPIE IV, SPIE Proceedings Vol. 2670*, pages 62–73, 1996.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 194–205, 1998.

[WYM97] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 186–195. Morgan Kaufmann, 1997.

[WYM99] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting+: An approach to active spatial data mining. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 116–125. IEEE Computer Society, 1999.

[WZ00] Haixun Wang and Carlo Zaniolo. Using SQL to build new aggregates and extenders for object-relational systems. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 166–175. Morgan Kaufmann, 2000.

[XEKS98] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 324–331. IEEE Computer Society, 1998.

[YWWY02] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. d-clusters: Capturing subspace correlation in a large data set. In *18th International Conference on Data Engineering (ICDE'02)*, pages 517–528. IEEE, 2002.

[YWWY03] Jiong Yang, Haixun Wang, Wei Wang, and Philip S. Yu. Enhanced biclustering on expression data. In *Third IEEE Symposium on BioInformatics and BioEngineering (BIBE'03)*, pages 321–327. IEEE, 2003.

[ZHD99] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-harmonic means - a data clustering algorithm. Technical Report HPL-1999-124, HP Research Labs, 1999.

[ZHD00] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-harmonic means - a spatial clustering algorithm with boosting. In *Temporal, Spatial, and Spatio-Temporal Data Mining, First International Workshop TSDM 2000*, pages 31–45. Springer, 2000.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

[ZS03] Jianjun Zhou and Jörg Sander. Data bubbles for non-vector data: Speeding-up hierarchical clustering in arbitrary metric spaces. In *VLDB'03, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany, DE*, 2003. to appear.

Advances in Clustering and Applications

Alexander Hinneburg,
University of Halle

Daniel A. Keim,
Unversity of Constance

Overview

- **Introduction**
  - Motivation
  - Properties of the Data
  - Basic Definitions
- **Basic Clustering Methods**
  - Model- and Optimization-based Approaches
  - Linkage-based Methods
  - Density-based Methods

Overview (2)

- **Advanced Topics in Clustering**
  - Clustering with Constraints
  - Projected Clustering
  - Outlier Detection
  - Database Technology for Clustering
  - Categorical Clustering
  - Interactive Clustering
- **Applications of Clustering**
  - Images, Bio Data, Geo Data
- **Conclusions**
  - Open Problems
  - Future Research

Introduction

- Motivation
  - Why is clustering needed in the KDD process?
  - What is the role of clustering in the KDD process?
- Properties of the Data
  - Data characteristics and impact on Clustering
  - Pre-analysis necessary (hypothesis generation)
- Basic Definitions
  - Density Estimation
  - Classification of Clustering Approaches

Motivation - Preliminary Remarks

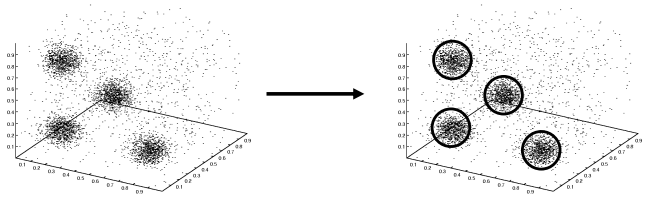
Problem: Analyze a (large) set of objects and form a small number of groups using the similarity and factual closeness between the objects.

Goals:

- Find representatives for homogenous groups
  -> ***Data Reduction***
- Find "natural" clusters and describe their properties
  -> ***"natural" Data Types***
- Find useful and suitable groupings
  -> ***"useful" Data Classes***
- Find unusual data objects -> ***Outlier Detection***

Motivation - Preliminary Remarks

- Examples:
  - Plant / Animal classification
  - Book ordering
  - Sizes for clothing
  - Fraud detection



Motivation - Preliminary Remarks

- Goal:
  objective instead of ***subjective*** clustering
- Preparations:
  - Data Representation
    - Feature Vectors, real / categorical values
    - Strings, Key Words
  - Similarity Function, Distance Matrix

Motivation

- **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

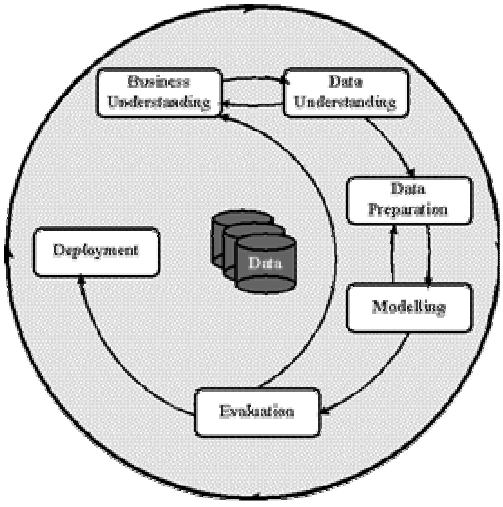
Motivation

- **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
      ($\rightarrow$ reduction of the number of parts to be produced)

Clustering and the KDD-Process

- Why is clustering needed in the KDD process?
- Where in the KDD process does clustering fit in?

The KDD-Process (CRISP)

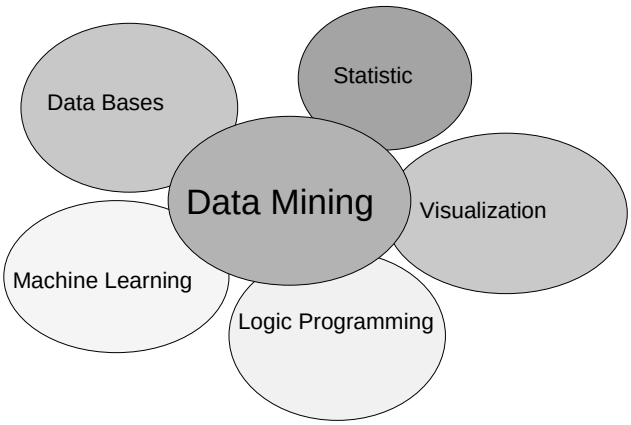


Data Mining vs. Statistic

- Algorithms scale to large data sets
- Data is available independent of data mining (secondary use)
- DM-Tools are for End-User with Background
- Strategy:
  - explorative
  - cyclic

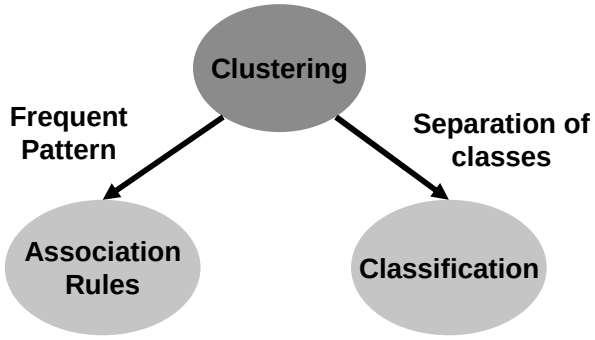
- Many Algorithms with quadratic run-time
- Data is collected for statistics (primary use)
- Statistical Background is often required
- Strategy:
  - conformational
  - verifying
  - few loops

Data Mining, an inter-disciplinary Research Area



Role of Clustering in the KDD Process

- Clustering is a basic technique for Knowledge Discovery.

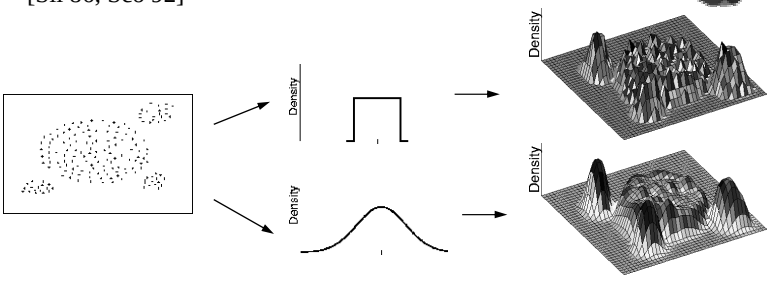


Basic Definitions

- Density Estimation
- Classification of Clustering Approaches

Kernel-Density Estimation

[Sil 86, Sco 92]



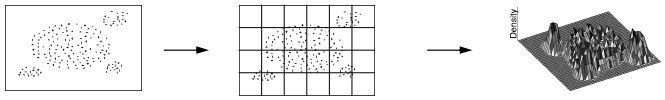
Data Set ***Kernel Function*** ***Density Function***

Kernel Function: Influence of a data point in the neighborhood

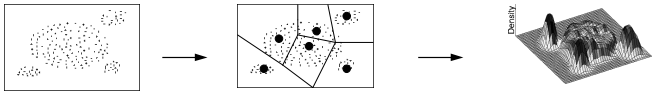
Density Function: Sum of the influences of all data points

Fast Density Estimation

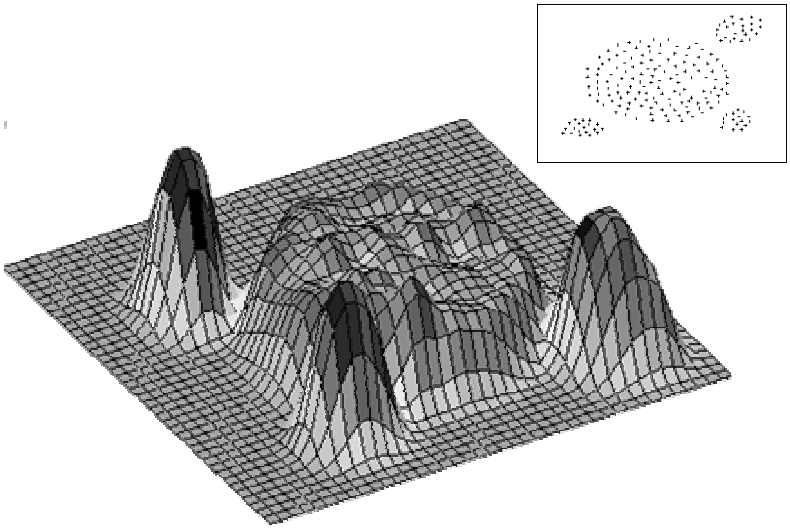
- Histograms (Reduction of the Influence)



- Centroid-based (Reduction of the data points)

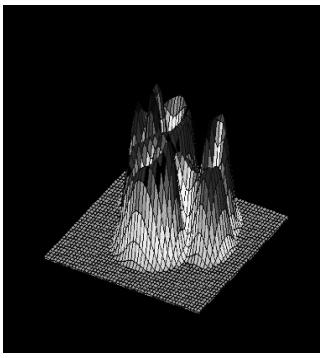


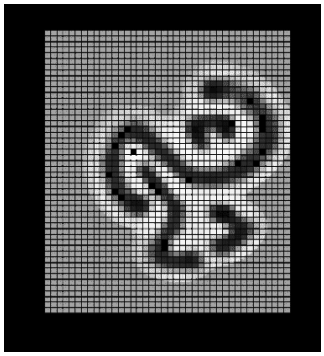
Kernel Density Estimation



Density Estimation

Impact of different Significance Levels (ξ)

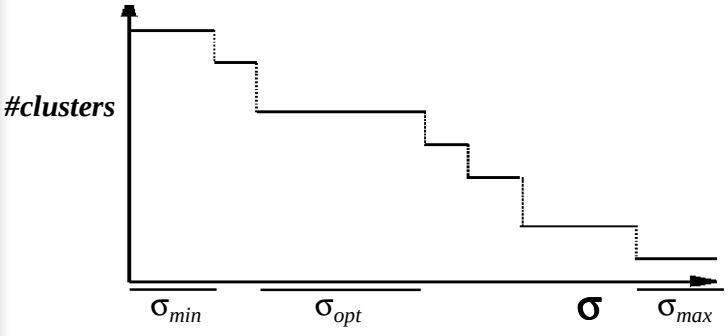




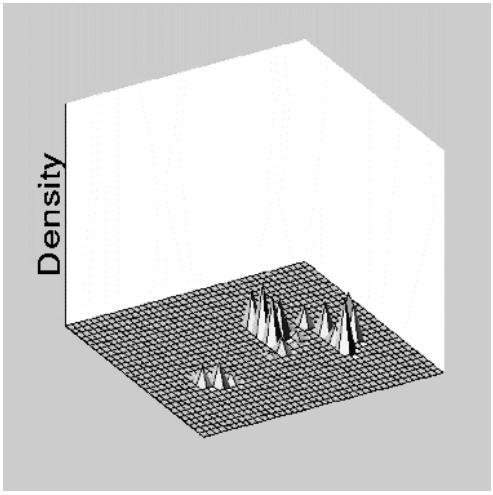
Kernel Density Estimation

Choice of the Smoothness Level (σ)

Choose σ such that *number of density attractors* is constant for a long interval of σ!



Kernel Density Estimation
Impact of the smoothness (σ)

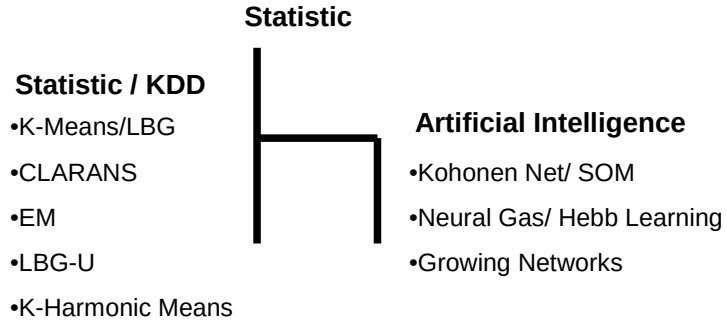


Basic Clustering Methods

- Model- and Optimization based Approaches
  - k Means, CLARANS
  - Expectation Maximization, EM
  - Self-Organizing Maps
  - Growing Networks
- Linkage Based Methods
  - Single, Complete, Avg and Centroid Linkage
  - Method of Wishart and DBSCAN
  - OPTICS
  - BIRCH
- Density Based Methods
  - STING, STING Plus
  - Wave Cluster
  - DENCLUE

Model-based Approaches

- Optimize the parameters for a given model



Model-based Methods: Statistic/KDD

- K-Means [DH73,Fuk 90]
- Expectation Maximization [Lau 95, KMN97]
- CLARANS [NH 94]
- LBG-U [Fri 97]
- K-Harmonic Means [ZHD 99, ZHD 00]

K-Means / LBG [DH73, Fuk90]

- Determine *k* prototypes (*p)* of a given data set
- Assign data points to nearest prototype

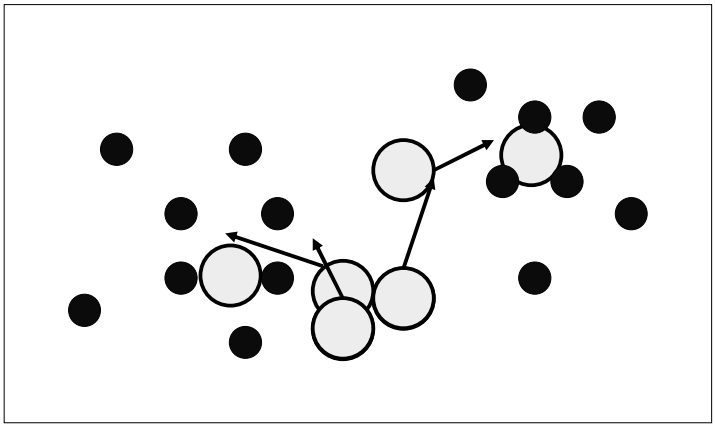
$$p \rightarrow R_p \text{ Voronoi - Set}$$

- Minimize distance criterion:

$$E(D,P) = 1/|D| \sum_{p \in P} \sum_{x \in R_p} dist(p,x)$$

- Iterative Algorithm
  - Shift the prototypes towards the mean of their point set
  - Re-assign the data points to the nearest prototype

K-Means: Example

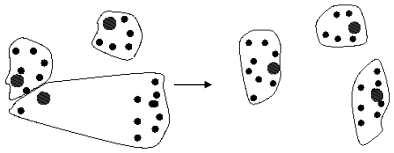


Expectation Maximization [Lau 95]

- 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

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: *num_local, max_neighbor*
- Impact of the Parameters:
  - *num_local* → Number of iterations
  - *max_neighbors* → Number of tested neighbors per iteration

CLARANS

- 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*k* edges
  - The search is bound by a fixed number of jumps *(num_local)* in the search graph
  - Each jump is optimized by randomized search and costs *max_neighbor* scans over the data (to evaluate the cost function)

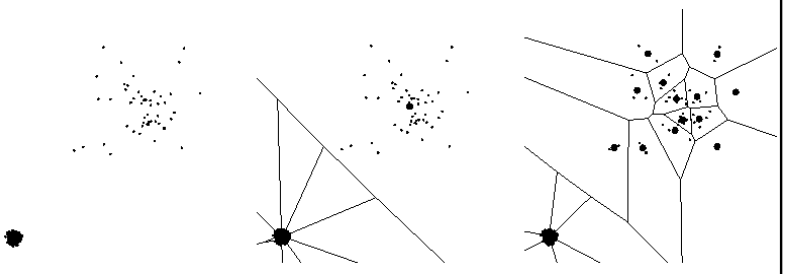
LBG-U [Fri 97]

$$\text{Utility } U(p) = E(D, P \setminus \{p\}) - E(D, P)$$
$$= \sum_{x \in R_p} dist(p_2, x) - dist(p, x)$$

$$\text{Quantization Error } E(p) = 1/R_P \sum_{x \in R_p} dist(x, p)$$

- Pick the prototype p with min. Utility and set it near the prototype p' with max. Quantization Error .
- Run LBG again until convergence

LBG-U: Example



Data Set with 1000 elements

LBG-Clustering Result 1

LBG-Clustering Result 2

K-Harmonic Means [ZHD 99,00]

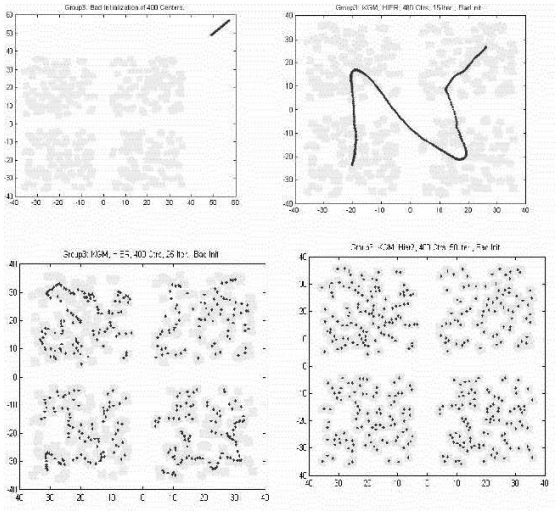
- Different Optimization Function:

$$Perf_{KHM}(\{x_i\}_{i=1}^{N},\{m_l\}_{l=1}^{K}) = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{\| x_i - m_l \|^2}}$$

- Update Formula for Prototypes:

$$m_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{3} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{2}})^2} x_i}{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{3} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{2}})^2}} \qquad d_{i,j} = dist(x_i, m_j)$$

K-Harmonic Means [ZHD 99,00]



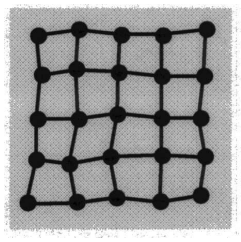
Model-based Methods: AI

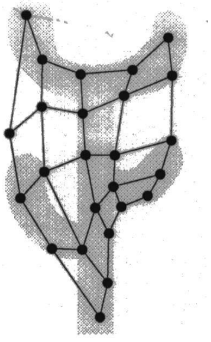
- Online Learning vs. Batch Learning
- Self-Organizing Maps [KMSK 91, Roj 96]
- Neural Gas & Hebb. Learning [MS 91,Mar93,MS94]
- Growing Networks [Fri 95]

Self Organizing Maps

[Roj 96, KMS+ 91]

– Fixed map topology (grid, line)

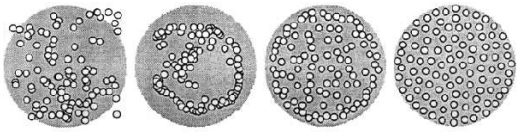




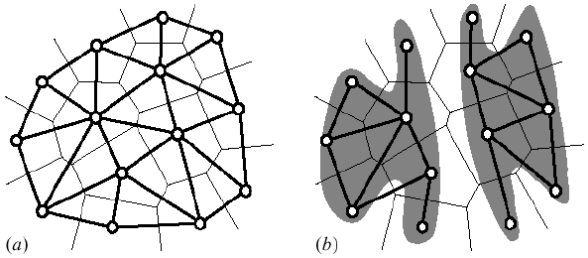
Neural Gas / Hebb Learning

[MS 91,Mar93,MS94]

- Neural Gas:

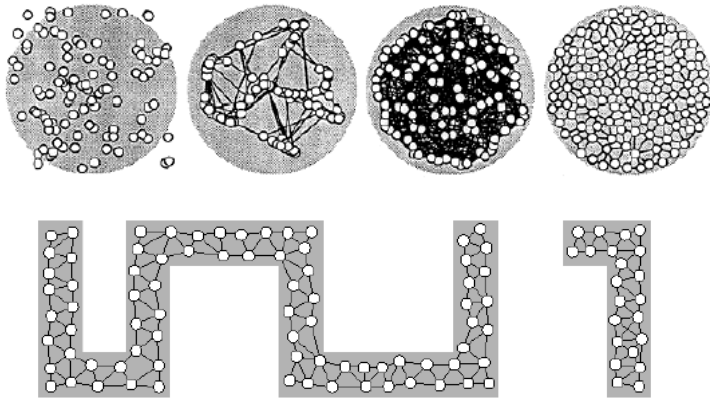


- Hebbian Learning:



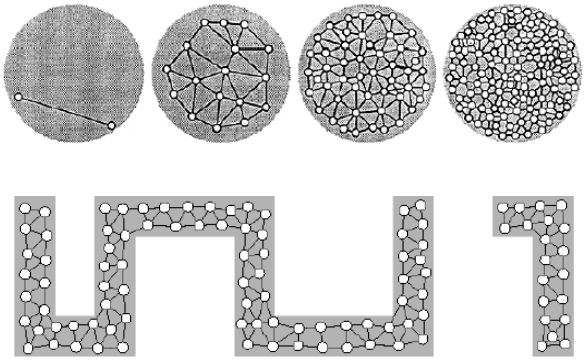
Neural Gas / Hebb Learning

[MS 91,Mar93,MS94]



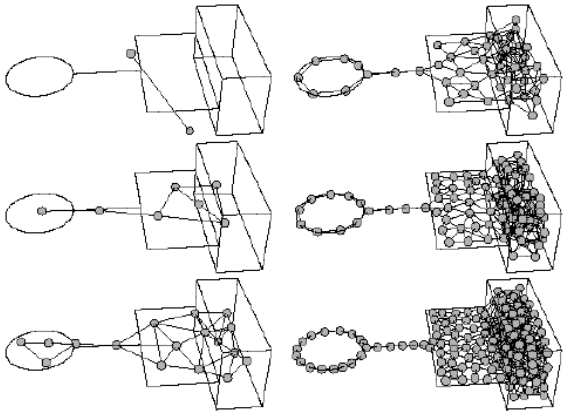
Growing Networks [Fri 95]

– Iterative insertion of nodes



Growing Networks [Fri 95]

- Adaptive map topology



Linkage-based Methods

- Hierarchical Methods:
  - Single / Complete / Avg / Centroid Linkage
  - BIRCH
- Graph Partitioning based Methods:
  - Single Linkage
  - Method of Wishart
  - DBSCAN
  - DBCLASD
  - OPTICS

Linkage Hierarchies [DH73, Boc 74]

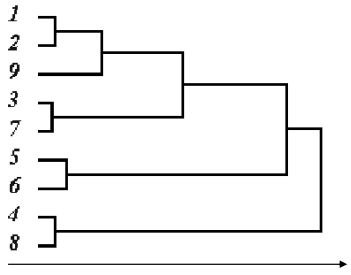
- Single Linkage (Minimum Spanning Tree)
- Complete Linkage
- Average Linkage
- Centroid Linkage (see also BIRCH)

Top-down (Dividing):

- Find the most inhomogeneous cluster and split

Bottom-up (Agglomerating):

- 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 of two subset of data points
- A single linkage hierarchy can be constructed using the Minimal Spanning Tree

Complete Linkage

- Distance between clusters (nodes):

$$Dist(C_1, C_2) = \max_{p \in C_1, q \in C_2} \{dist(p, q)\}$$

- Merge Step:
  Union of two subset of data points
- Each cluster in a complete linkage hierarchy corresponds to a complete subgraph

Average Linkage / Centroid Method

- Distance between clusters (nodes):

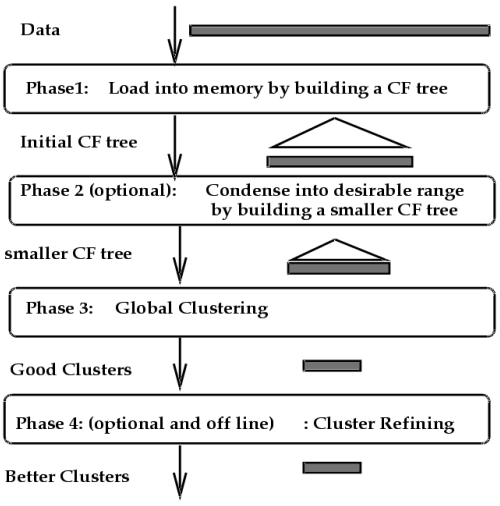
$$Dist_{avg}(C_1, C_2) = \frac{1}{\#(C_1) \cdot \#(C_2)} \sum_{p \in C_1} \sum_{p \in C_2} dist(p, q)$$

$$Dist_{mean}(C_1, C_2) = dist[mean(C_1), mean(C_2)]$$

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

BIRCH [ZRL 96]

Clustering in BIRCH



BIRCH

Basic Idea of the CF-Tree

- Condensation of the data $\{\vec{X}_i\}$ using CF-Vect $\mathbf{CF} = (N, \vec{LS}, SS)$

$$\vec{LS} = \sum_{i=1}^{N} \vec{X}_i, \; SS = \sum_{i=1}^{N} \vec{X}_i^{\,2}$$

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

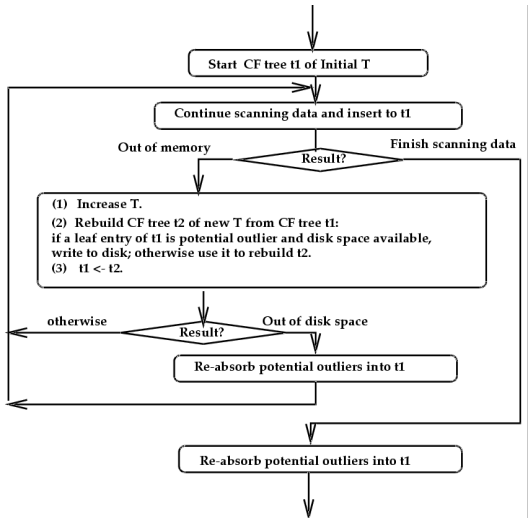
BIRCH

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 to large, condense the tree
by merging the closest leaves

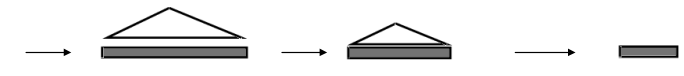
BIRCH

CF-Tree Construction



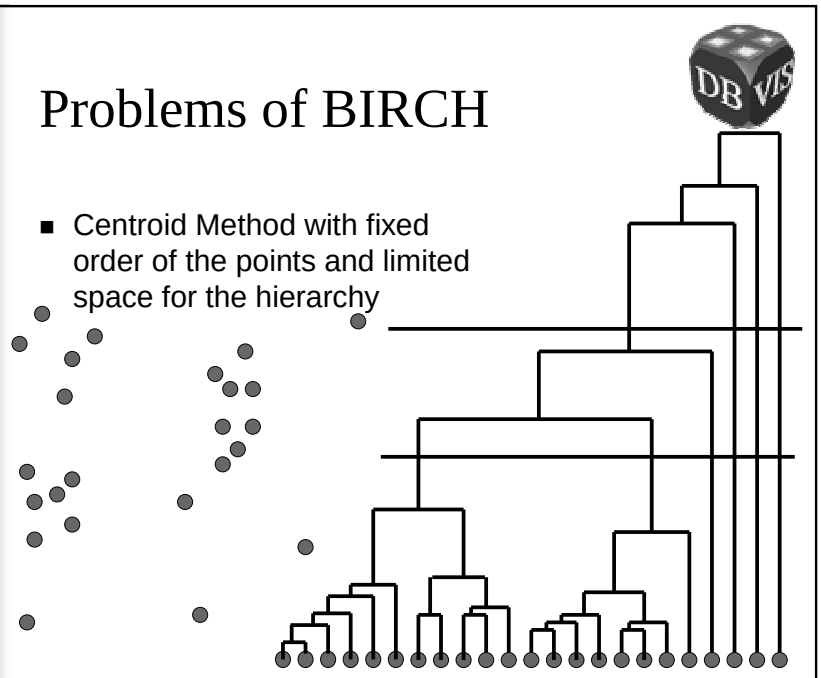
Condensing Data

- 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 and limited space for the hierarchy



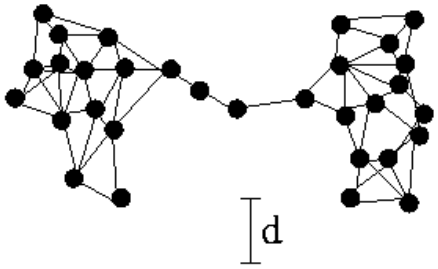
Linkage-based Methods

- Graph-Partitioning based Methods
  - Single Linkage [Boc 74]
  - Method of Wishart [Wis 69]
  - DBSCAN [EKSX 96]
  - DBCLASD [XEKS 98]
  - OPTICS [ABKS 99]

Linkage -based Methods

(from Statistics) [Boc 74]

- Single Linkage (Connected components for distance d)

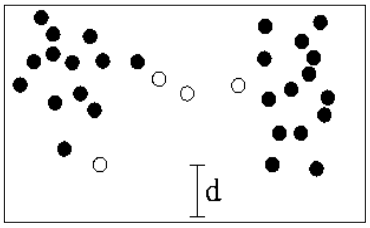


- Single Linkage + additional Stop Criterion

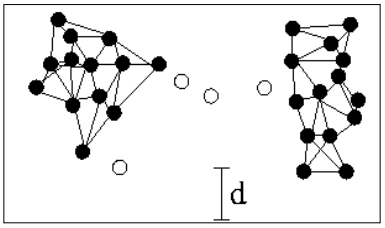
Linkage -based Methods

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

Reduce data set

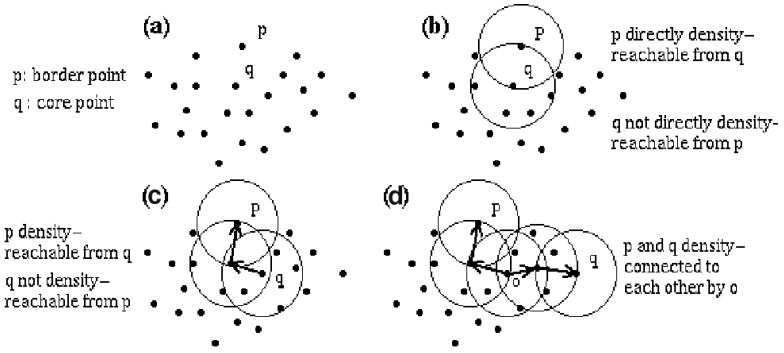


Apply Single Linkage



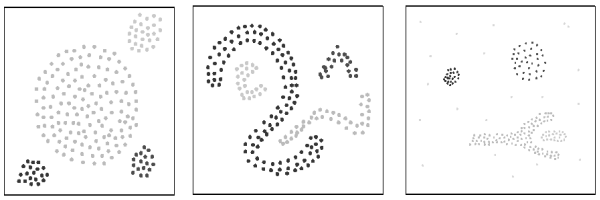
DBSCAN [EKSX 96]

- Clusters are defined as Density-Connected Sets (wrt. MinPts, ε)



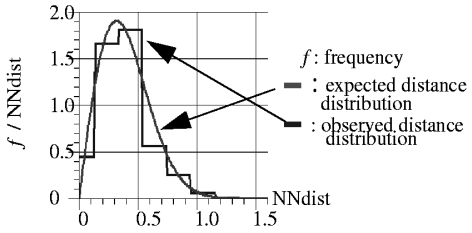
DBSCAN

- For each point, DBSCAN determines the ε-environment and checks, whether it contains more than MinPts data points
- DBSCAN uses index structures for determining the ε-environment
- Arbitrary shape clusters found by DBSCAN



DBCLASD [XEKS 98]

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



The expected and the observed distance distributions for cluster 1

DBCLASD

- Definition of a cluster *C* based on the distribution of the NN-distance *(NNDistSet)*:

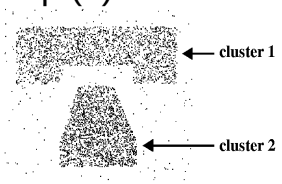
  (1) *NNDistSet(C)* has the expected distribution with a required confidence level.

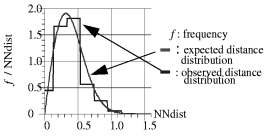
  (2) *C* is *maximal*, i.e. each extension of C by neighboring points does not fulfill condition (1). (maximality).

  (3) *C* is *connected*, i.e. for each pair of points (a,b) of the cluster there is a path of occupied grid cells connecting a and b (connectivity).

DBCLASD

- Step (1) uses the concept of the χ^2-test



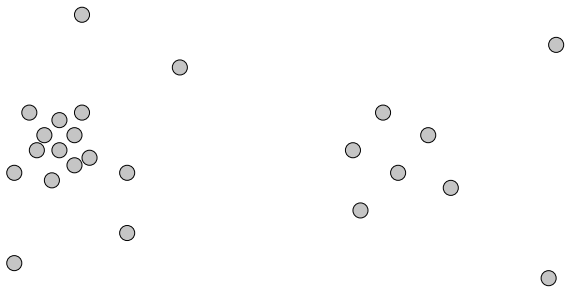


The expected and the observed distance distributions for cluster 1

- 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

Linkage-based Methods

- Single Linkage + additional Stop Criteria describes the border of the Clusters



OPTICS [ABKS 99]

- DBSCAN with variable ε, $0 \le \varepsilon \le \varepsilon_{MAX}$
- The result corresponds to the bottom of a hierarchy
- Ordering:
  - Reachability Distance:

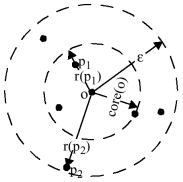
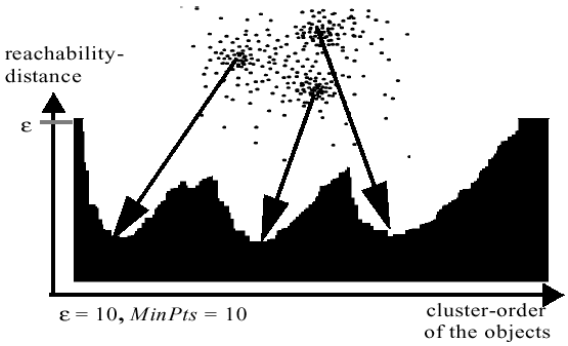


Figure 4. Core-distance(*o*), reachability-distances r(*p*1,*o*), r(*p*2,*o*) for *MinPts*=4

$$reach-dist(p,o) = \begin{cases} Undefined, if \left|N_{\varepsilon_{MAX}}(o)\right| < MinPTS \\ \max\{core-dist(o), dist(o,p)\}, else \end{cases}$$

OPTICS

- Breath First Search with Priority Queue



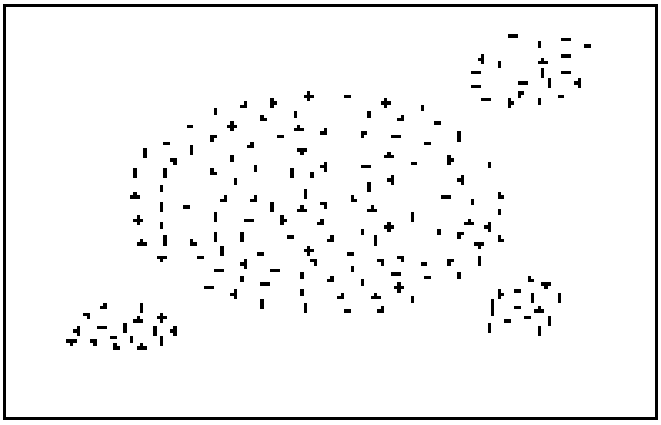
DBSCAN / DBCLASD/ OPTICS

- DBSCAN / DBCLASD / OPTICS use index structures to speed-up the ε-environment or nearest-neighbor search
- the index structures used are mainly the R-tree and variants

Density Based Methods

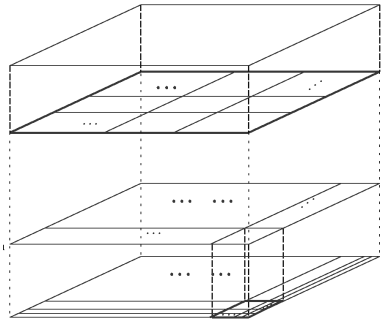
- STING, STING Plus
- Wave Cluster
- DENCLUE

Point Density



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

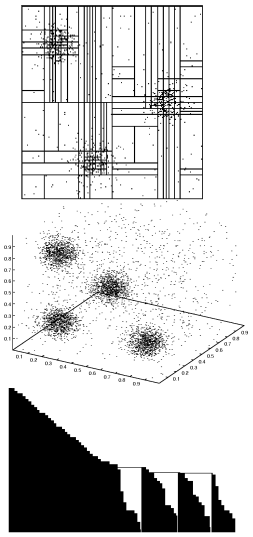


Hierarchical Grid Clustering [Sch 96]

- Organize the data space as a grid-file
- Sort the blocks by their density

$$DB = \frac{p_B}{V_B} \longrightarrow \langle B_{1'}, B_{2'}, \ldots B_{b'} \rangle$$

- Scan the blocks iteratively and merge blocks, which are adjacent over a (d-1)-dim. hyper plane.
- 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

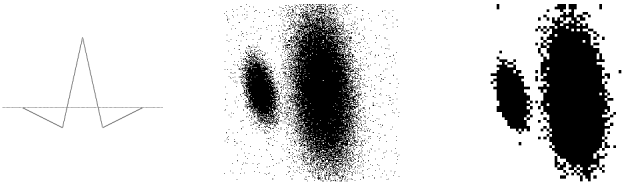
1. Quantize feature space, then assign objects to the units.
2. Apply wavelet transform on the feature space.
3. Find the connected components (clusters) in the subbands of transformed feature space, at different levels.
4. Assign label to the units.
5. Make the lookup table.
6. Map the objects to the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

WaveCluster

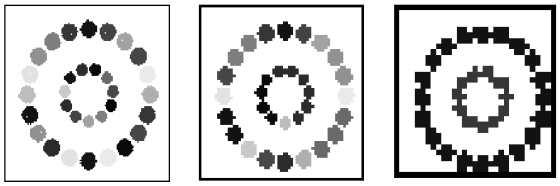
- Grid Approach
  - Partition the data space by a grid $\rightarrow$ reduce the number of data objects by making a small error
  - Apply the wavelet-transformation to the reduced feature space
  - Find the connected components as clusters
- Compression of the grid is crucial for the efficiency
- Does not work in high dimensional space!

WaveCluster

- Signal transformation using wavelets

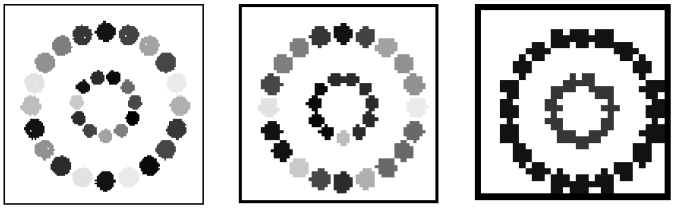


- Arbitrary shape clusters found by WaveCluster

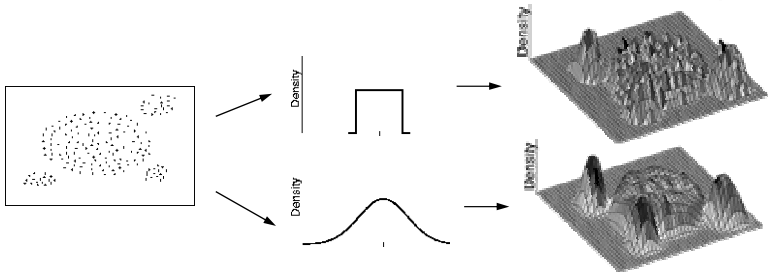


Hierarchical Variant of WaveCluster

- WaveCluster can be used to perform multi-resolution clustering
- Using coarser grids, cluster start to merge



Kernel Density Estimation



Data Set ***Influence Function*** ***Density Function***

Influence Function: Influence of a data point in its neighborhood

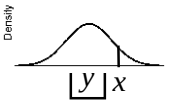
Density Function: Sum of the influences of all data points

Kernel Density Estimation

Influence Function

The influence of a data point *y* at a point *x* in the data space is modeled by a function $f_B^y : F^d \rightarrow \Re$,

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

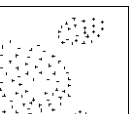


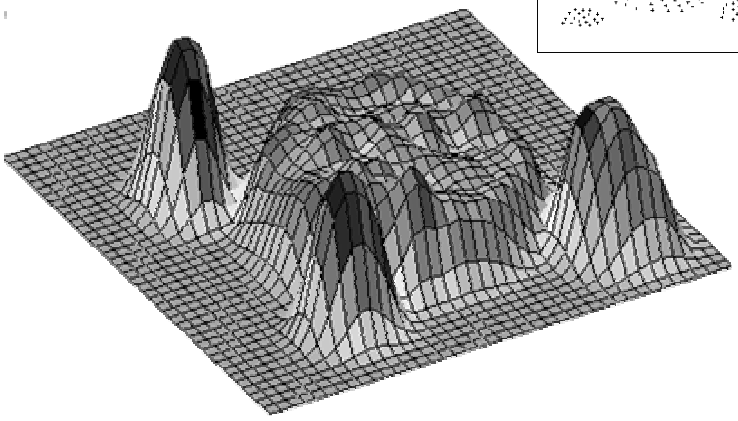
Density Function

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.

$$f_B^D(x) = \sum_{i=1}^{N} f_B^{x_i}(x)$$

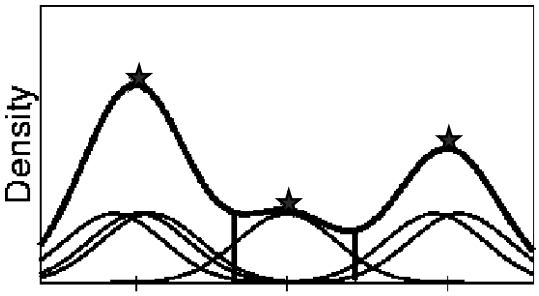
Kernel Density Estimation





DENCLUE [HK 98]

Definitions of Clusters



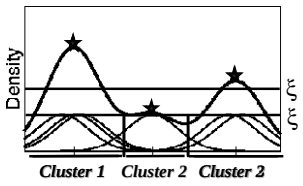
Density Attractor/Density-Attracted Points (★)

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

DENCLUE

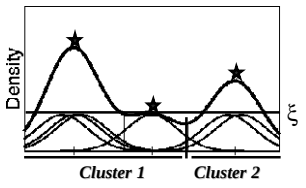
Center-Defined Cluster

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 ξ.



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:

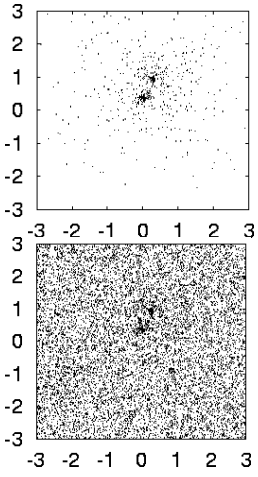
- partition density function into signal and noise

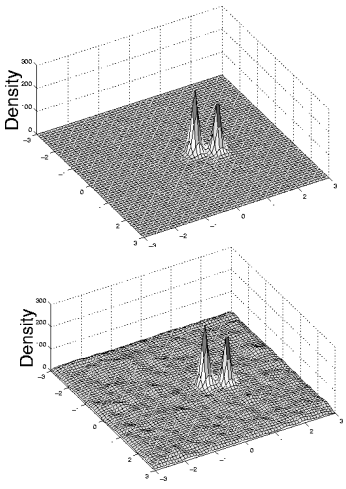
$$f^{D}(x) = f^{D_C}(x) + f^{N}(x)$$

- density function of noise approximates a constant ($f^{N}(x) \approx const.$)

DENCLUE

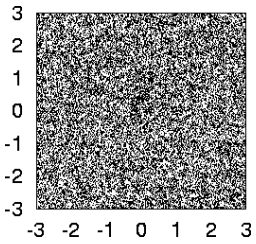
Noise Invariance

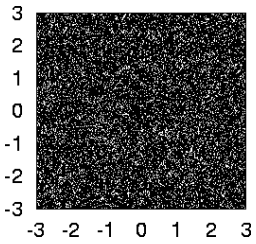


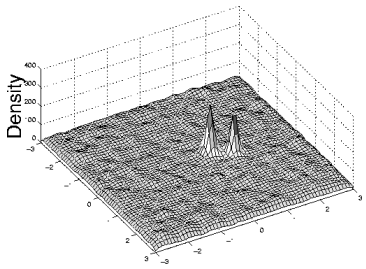


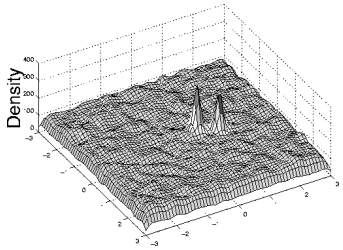
DENCLUE

Noise Invariance









DENCLUE

Local Density Function

Definition

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

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

Lemma (Error Bound)

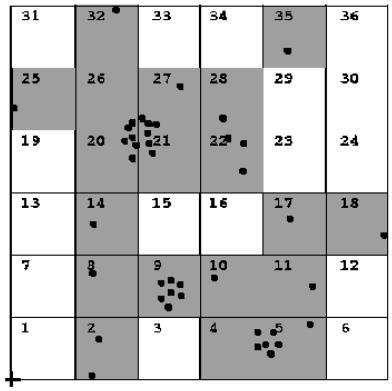
If $near(x) = \{x_i \in D \mid d(x, x_i) \leq k\sigma\}$, the error is bound by:

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

Multi-Dimensional Grids

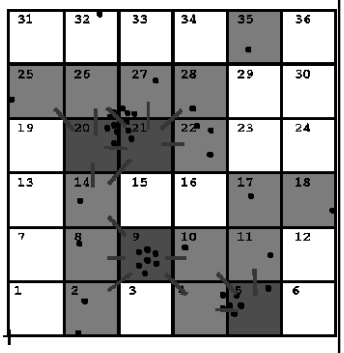
- Connecting Grid Cells:
  - the number of neighboring cells grows exponentially with the dimensionality
    ➡ Test if the cell is used is prohibitive
  - Connect only the highly populated cells
    drawback: highly populated cells are unlikely in high dimensional spaces

CubeMap



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

DENCLUE Algorithm



DENCLUE (D, σ, ξ)

(a) $MBR \leftarrow DetermineMBR(D)$

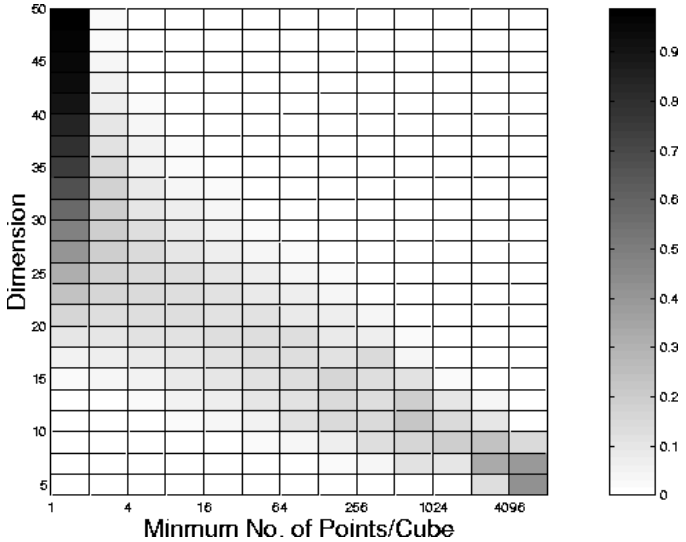
(b) $C_p \leftarrow DetPopCubes(D, MBR, \sigma)$

$C_{sp} \leftarrow DetHighlyPopCubes(C_p, \xi_c)$

(c) $map, C_r \leftarrow ConnectMap(C_p, C_{sp}, \sigma)$

(d) $clusters \leftarrow DetDensAttractors(map, C_r, \sigma, \xi)$

Multi-Dimensional Grids



Advanced Topics in Clustering

- Clustering with Constrains
- Projected Clustering
- Outlier Detection
- Database Technology for Clustering
- Categorical Clustering
- Interactive Clustering

Clustering with Constrains

- Constraint-based Clustering [TNLH01]
- Spatial Clustering with Obstacles
  [THH 01] [ECL 01]

Constraint-based Clustering

- Constraint Clustering:
  Extension of k-means by a set of Constraints C, so that the error function is minimized and each cluster satisfies the constraints C.
- Taxonomy of Constraints:
  - Constraints on individual objects
  - Obstacle objects as constraints
  - Clustering parameters as constraints
  - Constraints imposed on each individual cluster

Constraints under Consideration

[TNLH01]

- SQL Aggregate Constraints
  $(i)\ agg(\{O_i[A_j] \mid O_i \in Cl\})\,\theta\, c$
  $(ii)\ count(Cl)\,\theta\, c$
- Existential Constraints
  Pivot Objects (can be any subset) $W \subset D$
  $count(\{O_i \mid O_i \in Cl, O_i \in W\}) \geq c$

Data Set D, m Attributes $\{A_1,\ldots,A_m\}$
Value of an Attribute A_j of an Object $O_i : O_i[A_j]$
$agg \in \{\max(), \min(), \text{avg}(), \text{sum}()\}$; $\theta \in \{<, \leq, =, \neq, >, \geq\}$
$c \in \mathbf{R}$, Cl is a Cluster

Constraint-based Clustering

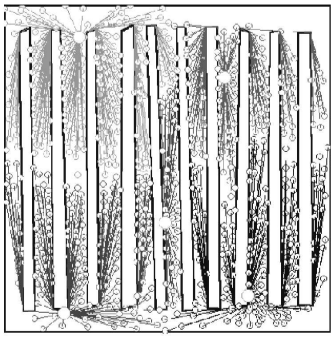
- Graph-based Approach
  - Model the Solution Space as Graph, similar to Clarans
  - Nodes are valid k-clusterings, which satisfy the existential constraints
  - An edge means the two clusterings differ only by one pivot object
- The problem to find a valid clustering with minimal error is NP-complete.
- Heuristics are used to find a good solution
  - Mirco-Cluster Sharing: the set objects is compressed to a set of micro-clusters (BIRCH)

Spatial Clustering with Obstacles

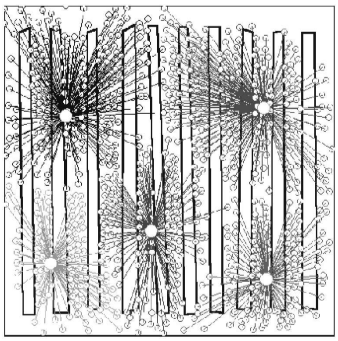
[THH 01]

- Extension of the CLARANS Algorithm
- A new distance function is used, which considers obstacles (Polygons in R^2)
- Used Concepts to improve COD-CLARANS
  - BSP Tree
    -> a visibility graph of the vertices of the obstacles
  - Spatial join indices
  - Micro-Clusters to compress the data set
  - Use Euclidian distance a lower bound of COD

Spatial Clustering with Obstacles



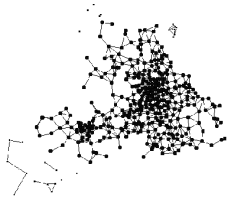
COD-CLARNS



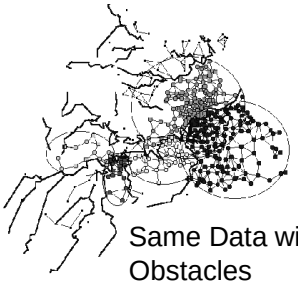
CLARANS

Fast Spatial Clustering with Obstacles [ECL01]

- Use Delaunay triangulation to determine clusters, different metrics are possible
- After Clusters are built, obstacles may split clusters.



Single Cluster without Obstacles



Same Data with Water Obstacles

Projected Clustering

- k-Means like Approaches
  + local dim. Reduction
  - ProClus [APW+99]
  - ORClus [AY00]
- Search Strategies for the Dimension-Lattice
  - CLIQUE [AGGR98]
  - OptiGrid [HK99]
  - DOC [PJAM02]

Projected Clustering

- Motivation
  - Attributes do not equally contribute to all clusters
  - High-dimensional real data has a lower intrinsic dimensionality
  - Feature selection and dimensionality reduction ***can not*** capture these patterns as the reduced dimensionality applies to the whole data instead of a subset only.

Projected Clustering

- K-Means like Approaches
- General Idea
  - Represent the clustering by centroids or medoids
  - Start with (randomly) init. configuration
  - Reduce the set of active dimensions and improve the point assignments iteratively to
  - lower the Approximation Error
- Differences in the dim. Reduction
  - ProClus: axis-parallel projections based on variance
  - ORClus: arbitrary linear projections based on Eigen-vectors of the covariance matrix

PROCLUS [APW+99]

Algorithm similar to k-means

Algorithm PROCLUS(*No. of Clusters: k, Avg. Dimensions: l*

{1. Initialization Phase}

{2. Iterative Phase}

$BestObjective = \infty$

$M_{current}$ = Random set of medoids $\{m_1, m_2, \ldots m_k\} \subset \mathcal{M}$

repeat

{ Approximate the optimal set of dimensions }

for each medoid $m_i \in M_{current}$ **do**

begin

Let δ_i be distance to nearest medoid from m_i

$\mathcal{L}_i$ = Points in sphere centered at m_i with radius δ_i

end;

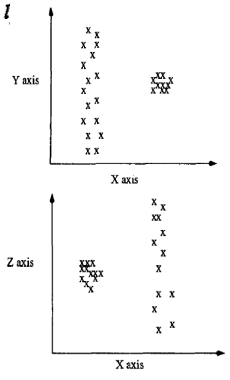
$\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_k\}$

$(\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_k) = \text{FindDimensions}(k, l, \mathcal{L})$

{ Form the clusters }

until (*termination_criterion*)

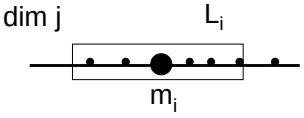
{3. Refinement Phase}



Shorted Version of the Alg.

PROCLUS

- How to pick the Dimensions?



Problem:

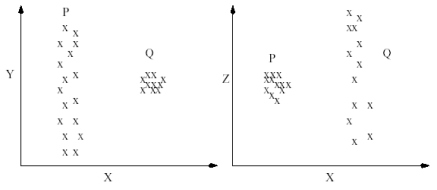
The total number of picked Dimensions is fixed to *k*l*

How to specify *l* ?

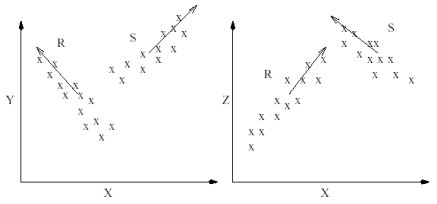
Algorithm FindDimensions($k, l, \mathcal{L}$)
begin
{ d is the total number of dimensions }
{ $X_{i,j}$ is the average distance from the points in $\mathcal{L}_i$ to medoid m_i, along dimension j}
for each medoid i **do**
begin
$Y_i = \frac{\sum_{j=1}^{d} X_{i,j}}{d}$
$\mathcal{D}_i = \emptyset$
$\sigma_i = \sqrt{\frac{\sum_{j=1}^{d}(X_{i,j}-Y_i)^2}{d-1}}$
for each dimension j **do** $Z_{i,j} = (X_{i,j} - Y_i)/\sigma_i$
end
Pick the $k \cdot l$ numbers with the least (most negative) values of $Z_{i,j}$ subject to the constraint that there are at least 2 dimensions for each cluster
if $Z_{i,j}$ is picked **then** add dimension j to $\mathcal{D}_i$
return($\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_k$)
end

ORClus [AY00]

Algorithm ORCLUS(*Number of Clusters: k, Number of Dimensions: l*)
{ $\mathcal{C}_i$ is the current cluster i }
{ $\mathcal{E}_i$ is the set of vectors defining subspace for cluster $\mathcal{C}_i$ }
{ $k_c \Rightarrow$ current number of seeds; $l_c \Rightarrow$ current dimensionality associated with each seed }
{ $S = \{s_1, s_2 \ldots s_{k_c}\}$ is the current set of seeds }
{ k_0 is the number of seeds that we begin with }
begin
Pick $k_0 > k$ points from the database and denote by S; $\{S = (s_1, \ldots s_{k_c})\}$
$k_c = k_0$; $l_c = d$;
for each i set $\mathcal{E}_i = \mathcal{D}$;
{ Initially, $\mathcal{E}_i$ is the original axis-system }
$\alpha = 0.5$; $\beta = e^{-\log(d/l)\cdot\log(1/\alpha)/\log(k_0/k)}$;
while $(k_c > k)$ **do**
begin
{ Find partitioning induced by the seeds }
$(s_1, \ldots s_{k_c}, \mathcal{C}_1, \ldots \mathcal{C}_{k_c}) = Assign(s_1, \ldots s_{k_c}, \mathcal{E}_1, \ldots \mathcal{E}_{k_c})$;
{ Determine current subspace associated with each cluster $\mathcal{C}_i$ }
for $i = 1$ to k_{new} **do** $\mathcal{E}_i = FindVectors(\mathcal{C}_i, l_{new})$;
{ Reduce number of seeds and dimensionality associated with each seed }
$k_{new} = \max\{k, k_c \cdot \alpha\}$; $l_{new} = \max\{l, l_c \cdot \beta\}$;
$(s_1 \ldots s_{k_{new}}, \mathcal{C}_1, \ldots \mathcal{C}_{k_{new}}, \mathcal{E}_1 \ldots \mathcal{E}_{k_{new}}) = Merge(\mathcal{C}_1, \ldots \mathcal{C}_{k_c}, k_{new}, l_{new})$;
$k_c = k_{new}$; $l_c = l_{new}$;
end;
$(s_1, \ldots s_k, \mathcal{C}_1, \ldots \mathcal{C}_k) = Assign(s_1, \ldots s_k, \mathcal{E}_1, \ldots \mathcal{E}_k)$;
return($\mathcal{C}_1 \ldots \mathcal{C}_{k_c}$);
end;



old



new

ORClus

Algorithm ORCLUS(*Number of Clusters:* k, *Number of Dimensions:* l)
{ $\mathcal{C}_i$ is the current cluster i }
{ $\mathcal{E}_i$ is the set of vectors defining subspace for cluster $\mathcal{C}_i$ }
{ $k_c \Rightarrow$ current number of seeds; $l_c \Rightarrow$ current dimensionality associated with each seed }
{ $S = \{s_1, s_2 \ldots s_{k_c}\}$ is the current set of seeds }
{ k_0 is the number of seeds that we begin with }
begin
  Pick $k_0 > k$ points from the database and denote by S; $\{S = (s_1, \ldots s_{k_c})\}$
  $k_c = k_0$; $l_c = d$;
  for each i set $\mathcal{E}_i = \mathcal{D}$;
  { Initially, $\mathcal{E}_i$ is the original axis-system }
  $\alpha = 0.5$; $\beta = e^{-\log(d/l)\log(1/\alpha)/\log(k_0/k)}$;
  while $(k_c > k)$ **do**
  begin
    { Find partitioning induced by the seeds }
    $(s_1, \ldots s_{k_c}, \mathcal{C}_1, \ldots \mathcal{C}_{k_c}) = Assign(s_1, \ldots s_{k_c}, \mathcal{E}_1, \ldots \mathcal{E}_{k_c})$;
    { Determine current subspace associated with each cluster $\mathcal{C}_i$ }
    for $i = 1$ to k_{new} **do** $\mathcal{E}_i = FindVectors(\mathcal{C}_i, l_{new})$;
    { Reduce number of seeds and dimensionality associated with each seed }
    $k_{new} = \max\{k, k_c \cdot \alpha\}$; $l_{new} = \max\{l, l_c \cdot \beta\}$;
    $(s_1 \ldots s_{k_{new}}, \mathcal{C}_1 \ldots \mathcal{C}_{k_{new}}, \mathcal{E}_1 \ldots \mathcal{E}_{k_{new}}) =$
      $Merge(\mathcal{C}_1, \ldots \mathcal{C}_{k_c}, k_{new}, l_{new})$;
    $k_c = k_{new}$; $l_c = l_{new}$;
  end;
  $(s_1, \ldots s_k, \mathcal{C}_1, \ldots \mathcal{C}_k) = Assign(s_1, \ldots s_k, \mathcal{E}_1, \ldots \mathcal{E}_k)$;
  return$(\mathcal{C}_1 \ldots \mathcal{C}_{k_c})$;
end;

Algorithm *FindVectors*(Cluster of points:$\mathcal{C}$, Dimensionality of projection:q)
begin
  Determine the $d * d$ covariance matrix $\mathcal{M}$ for $\mathcal{C}$;
  Determine the eigenvectors of matrix $\mathcal{M}$;
  $\mathcal{E}$ = Set of eigenvectors corresponding to smallest q eigenvalues;
  return$(\mathcal{E})$;
end

Merge the closest clusters until k_new is reached

Problems of PROCLUS and ORClus

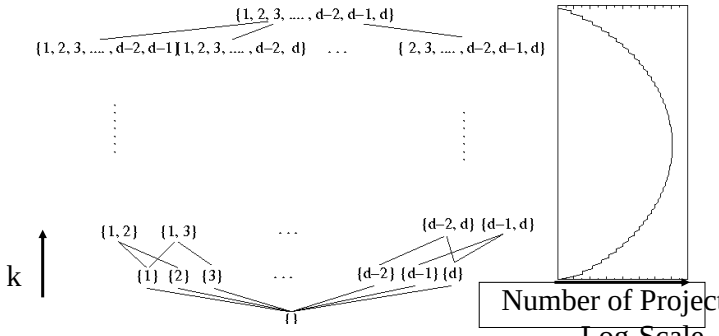
- The determined clustering is a partitioning of the data set.
  - For projected clustering a point may belong to multiple clusters defined in different projections at the same time.
  - Example: (longitude, latitude, salary, age)
- The number of independent components have to be estimated in advance.
  - k … number of cluster *
    l … number average relevant dimensions

Projected Clustering

Search Strategies in the Dimension Lattice

- k is the dimensionality of the projection
- d is the global dimensionality

$$\binom{d}{k} = \frac{d!}{(d-k)!k!}$$



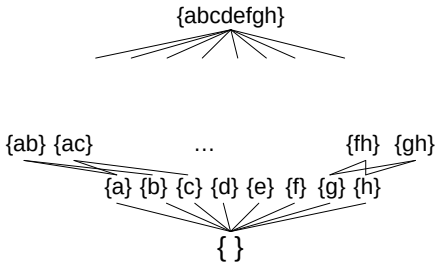
Projected Clustering

Search Strategies in the Dimension Lattice

- CLIQUE [AGGR98]
  - Application of the Apriory idea + DBSCAN
- OptiGrid [HK99]
  - Recursively combine interesting splits from different projections to grid
- DOC [PJAM02]
  - Optimize the tradeoff between #(relevant dimensions) versus cluster size with a Monte-Carlo strategy

Apriory for Mining Frequent Sets

- Input: a large number of item sets
  T1: a f h c d
  T2: b h j a
  T3: e g h f
- Find frequent item sets, which occur more often than min-sup times.



- Monotonicity: all subsets of a frequent item set are also frequent. => If a subset is not frequent, all supersets are also not frequent.

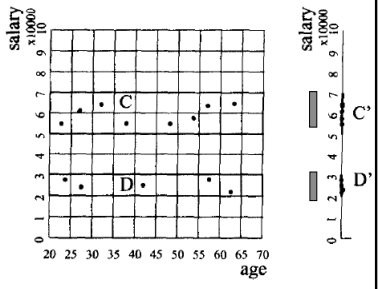
CLIQUE [AGGR98]

Algorithm

1. Identification of subspaces that contain clusters

   Discretize the numeric dimensions

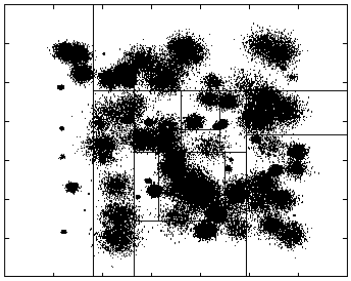
   Find frequent itemsets with Apriory
2. Identification of cluster
3. Generation of minimal descriptions for the clusters



Lemma 1 (Monotonicity): *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 projections of this space.*

OptiGrid [HK99]

- Idea
  - Search for good splits in the one dimensional projections
  - Combine the k best split planes to a multi-dimensional grid
  - Find frequent grid cells
  - Process theses cells recursively



x_2 / H_5	56	57	59	63
	24	25	27	31
H_4	8	9	11	15
H_3	0	1	3	7
origin	H_0		H_1	H_2 x_1

Orthogonal Grid

Numbering $e(x) = \sum_{i=1}^{k} 2^i \cdot H_i(x)$

First recursions for the special case for k=1

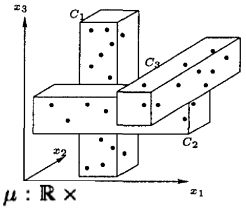
DOC [PJAM02]

DEFINITION 1. *Let S be a set of points in $\mathbb{R}^d$. For any $0 \leq \alpha \leq 1$ and $w \geq 0$, an* α-dense projective cluster *of width w in S is a pair $(\mathcal{C}, \mathcal{D})$, $\mathcal{C} \subseteq S$, $\mathcal{D} \subseteq [d]$, such that*

(1) $\mathcal{C}$ is α-dense, *i.e. $|\mathcal{C}| \geq \alpha|S|$;*

(2) $\forall i \in \mathcal{D}$, $\max_{p \in \mathcal{C}} p_i - \min_{q \in \mathcal{C}} q_i \leq w$;

(3) $\forall i \in [d] \setminus \mathcal{D}$, $\max_{p \in \mathcal{C}} p_i - \min_{q \in \mathcal{C}} q_i > w$.



DEFINITION 2. *Let S be a set of n points in $\mathbb{R}^d$. Let $\mu : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a function such that $\mu(0,0) = 0$ and μ is monotonically increasing in each argument. We define the* quality *of a projective cluster $(\mathcal{C}, \mathcal{D})$ to be $\mu(|\mathcal{C}|, |\mathcal{D}|)$. For any fixed $0 \leq \alpha \leq 1$, a projective cluster $(\mathcal{C}, \mathcal{D}) \in \mathcal{P}_\alpha$ is* μ-optimal *(or optimal for brevity) if it maximizes μ over $\mathcal{P}_\alpha$.*

Monte-Carlo techniques are used to find optimal projected clusters.

Advantages & Problems

- Advantages
  - Simple cluster descriptions
  - Clusters may overlap, that means a single object may belong to different clusters
- Problem of CLIQUE, OptiGrid and DOC
  - No good strategy to handle complex dependencies between the attributes

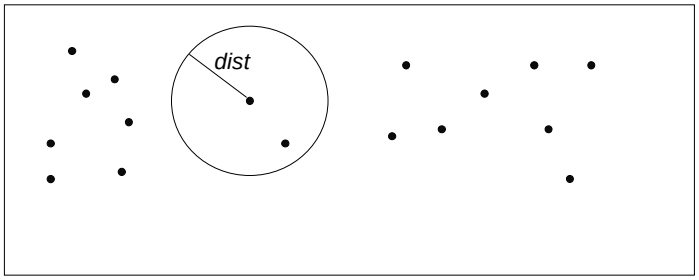
Outlier Detection

- Distance-Based Outliers
  - [KN98, KNT00],
  - [RRS00]
- Local Outliers
  - [BKNS99,BKNS00]
  - [JTH01]
  - [PKGF03]

Distance-based Outliers

[KN98, KNT00]

- Definition:
  - A object *o* is an outlier, iff there are at least *p* percent of the database with a distance larger than *dist*.



Problems

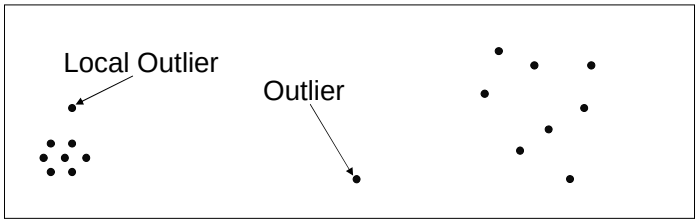
- Complexity of the Algorithms
  - Nested Loop Algorithm → $O(N^2)$
  - Cell-based Algorithm → $O(c^d N)$
- Difficult parameter specification of *dist* and *p*
- No Ranking of the outliers is provided

Distance-based Outliers
[RRS00]

- Definition
  - A object *o* is an outlier, iff at least *p* percent of the database has a smaller k-NN distance than *o.*
- Algorithms
  - Nested Loop Algorithm ⟶ $O(N^2)$
  - Index-based Algorithm: R*-Tree is used to speed up the comparison of the k-NN distance
  - Partition-based Algorithm: use a BIRCH pre-clustering to derive upper bounds for the k-NN dist.

Local Outlier
[BKNS00]

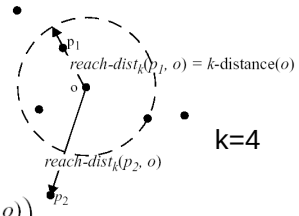
- Problem of distance based Outliers



- Concept of local Outliers
  - Outliers are exceptional with respect to their local neighborhood

Definition of local density-based Outliers

- Density is measured in terms of the k-NN dist.
- Local Reachability Distance



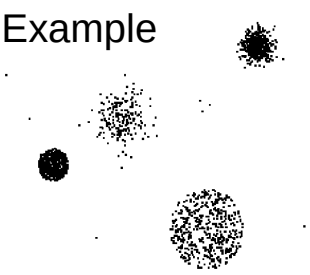
$$lrd_{MinPts}(p) = 1 / \left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts}(p)|} \right)$$

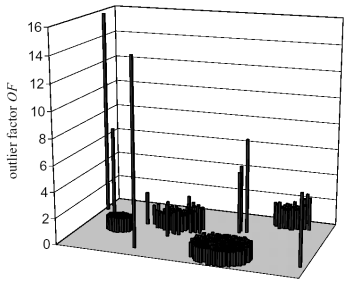
- Local Outlier Factor

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

Local Density based Outliers

- Example

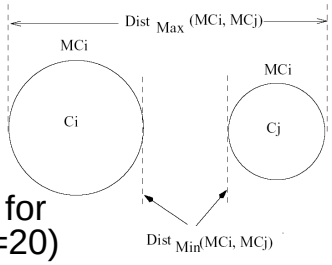




- Problems
  - Expensive determination of the k-NN dist. using a spatial index => large run time
  - Estimation of the parameters MinPts and eps

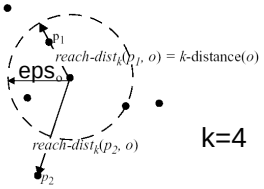
Fast Top-n local Outliers
[JTH01]

- Idea: Use micro-clusters to derive upper and lower bounds for LOF
- Micro-Clusters are determined with BIRCH and consist of a CF-vector
- Algorithm:
  - Preprocessing
  - Computing LOF bound for micro-clusters
  - Rank top-n local outliers
- Speed-up grows up to 10 for high-dimensional data (d=20)



LOCI [PKGF03]

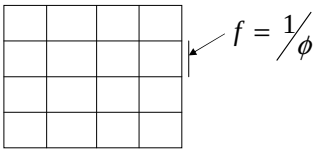
- Make local outliers more robust
- Eliminate parameter eps, which defines the local neighborhood



- Generalization: a object is an outlier, if the LOF value exceeds a given threshold for any eps-value in the range [eps_{min},…, eps_{max}]
- Fast implementation using a quadtree like data structure.

Projected Outliers [AY01]

- Idea: Outlier are defined by exceptionally low density regions in projections.



- Expectation: Points are uniformly distributed, then No. of Points in a grid cell is $N \cdot f^k$ with standard deviation $\sqrt{N \cdot f^k \cdot (1 - f^k)}$
- If the real number $n(D)$ of points in a *k*-dim. cube is much lower than the expected one, the points in D are considered as outliers.
- Sparsity coefficient: $\dfrac{n(D) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}}$

Projected Cluster

- Example: People (…,age,diabetes,…)
  - Many records with age<20
  - Many records with diabetes=true
  - But very few records with both
- Evolutionary algorithm
  - Structured search methods do not work
  - Try many projections and combine good ones
- Parameters: k and phi

Database Technology for Clustering

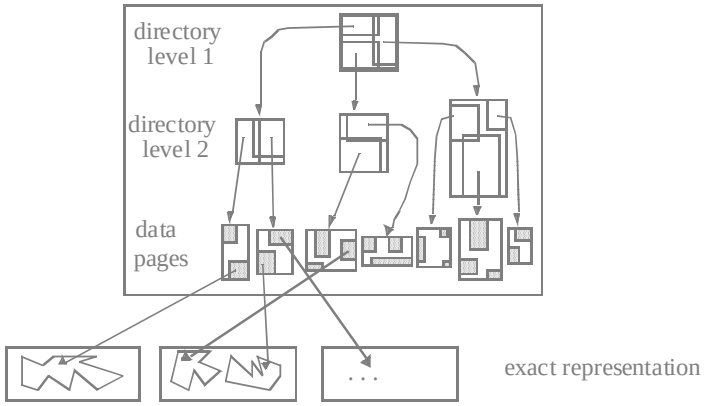
- Index Structures [BBK01]
- ATLAS SQL Extensions [WZ00]
- SQL-EM [OC00]
- Density Estimation in Projections [HLH03]
  Micro-Clusters, Data Bubbles

Indexing [BBK01]

- 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]

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]

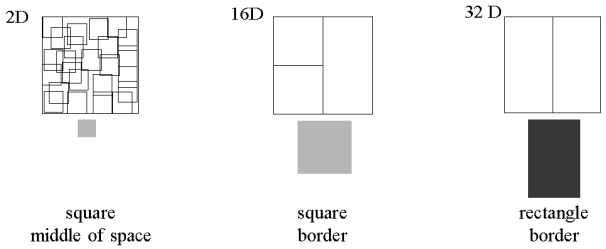
High-dimensional

- TV-Tree [LJF 94]
- X-Tree [BKK 96]
- SS-Tree [WJ 96]
- SR-Tree [KS 97]

Effects of High Dimensionality

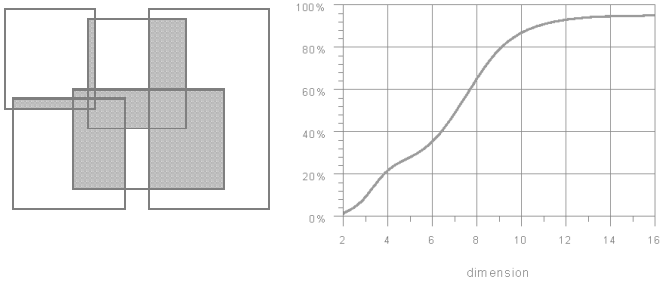
Location and Shape of Data Pages

- Data pages have large extensions
- Most data pages touch the surface of the data space on most sides



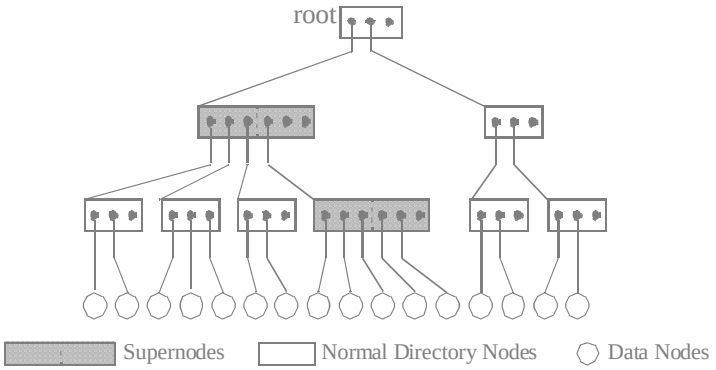
The X-Tree [BKK 96] (eXtended-Node Tree)

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

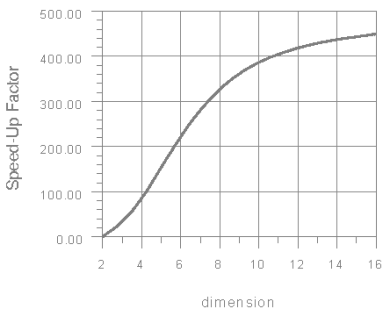


The X-Tree

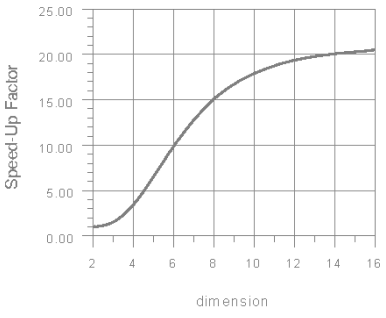
- X-tree avoids overlap in the directory by using
  - an overlap-free split
  - the concept of supernodes



Speed-Up of X-Tree over the R*-Tree



Point Query



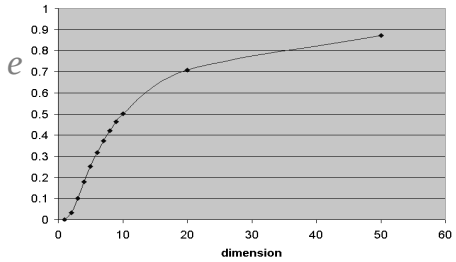
10 NN Query

Effects of High Dimensionality

Selectivity of Range Queries

- The selectivity depends on the volume of the query

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



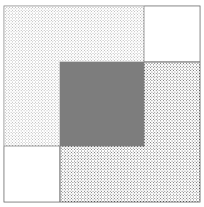
selectivity = 0.1 %

⇒ **no fixed ε-environment (as in DBSCAN)**

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)**

Efficiency of NN-Search [WSB98]

- **Assumptions:**
  - A cluster is characterized by a geometrical form (MBR) that covers all cluster points
  - The geometrical form is convex
  - Each Cluster contains at least two points
- **Theorem:** For any clustering and partitioning method there is a dimensionality d for which a sequential scan performs better.

VA File [WSB 98]

- Vector Approximation File:
  - Compressing Vector Data: each dimension of a vector is represented by some bits
    ➔ partitions the space into a grid
  - Filtering Step: scan the compressed vectors to derive an upper and lower bound for the NN-distance ➔ Candidate Set
  - Accessing the Vectors: test the Candidate Set

ATLaS SQL Extension

[WZ00]

- Extension of SQL with User Defined Aggregates, UDA
- UDA's definition is close to Standard SQL
- UDA prog.s are compiled on top of a DBS
- The language extension is Turing complete
- Many data mining algorithms can be reformulated in ATLaS with very small code
  - Apriory
  - DBSCAN
- Aggregation over streams is also possible

User Defined Aggregates (UDAs)

- Important for decision support, stream queries and other advanced database applications.
- UDA consists of 3 parts:
  - INITIALIZE
  - ITERATE
  - TERMINATE

Standard aggregate average


```
AGGREGATE myavg(Next Int) : Real
{   TABLE state(tsum Int, cnt Int);
    INITIALIZE : {
        INSERT INTO state VALUES (Next, 1);
    }
    ITERATE : {
        UPDATE state
            SET tsum=tsum+Next, cnt=cnt+1;
    }
    TERMINATE : {
        INSERT INTO RETURN
        SELECT tsum/cnt FROM state;
    }
}
```

Online aggregation

- Standard average aggregate returns results at TERMINATE state.
- Online aggregate: Get results before input all the tuples
  - e.g. return the average for every 200 input tuples.
- RETURN statements appear in ITERATE instead of TERMINATE.

Online averages


```
AGGREGATE online_avg(Next Int) : Real
{   TABLE state(tsum Int, cnt Int);
    INITIALIZE : {
        INSERT INTO state VALUES (Next, 1);
    }
    ITERATE: {
        UPDATE state
            SET tsum=tsum+Next, cnt=cnt+1;
        INSERT INTO RETURN
            SELECT sum/cnt FROM state
            WHERE cnt % 200 = 0;
    }
    TERMINATE : { }
}
```

DBSCAN with ATLaS (1)


```
table SetOfPoints (x real, y real, ClId int) RTREE;
/* meaning of ClId: -1: unclassified, 0: noise, 1,2,3...: cluster */
table nextId(ClusterId int);
table seeds (sx real, sy real);

insert into nextId values (1);

select ExpandCluster(x, y, ClusterId, Eps, Minpts)
from SetOfPoints, nextId
where ClId= -1 ;
```

DBSCAN with ATLaS (2)


```
aggregate ExpandCluster (x real, y real, ClusterId int, Eps real, MinPts
    int):Boolean
{   table seedssize (size int);
    initialize:
    iterate:
     {
      insert into seeds select regionQuery (x, y, Eps);
      insert into seedssize select count(*) from seeds;
      insert into return select False from seedssize where size<MinPts;
      update SetofPoints set ClId=ClusterId
            where exists (select * from seeds where sx=x and sy=y) and
    SQLCODE=0;
      update nextId as n set n.ClusterId=n.ClusterId+1 where SQLCODE=1;
      delete from seeds where sx=x and sy=y and SQLCODE=1;
      select changeClId (sx, sy, ClusterId, Eps, MinPts) from seeds and
    SQLCODE=1;
} }
```

DBSCAN with ATLaS (3)


```
aggregate changeClId (sx real, sy real, ClusterId int, Eps real, MinPts
    int):Boolean
{
 table result (rx real, ry real);
 table resultsize (size int);
 initialize:
 iterate:
  {
   insert into result select regionQuery(sx, sy, Eps);
   insert into resultsize select count(*) from result;
   insert into seeds select rx, ry from result
       where (select size from resultsize)>=Minpts
       and (select ClId from SetofPoints where x=rx and y=ry)=-1;
   update SetofPoints set ClId=ClusterId where SQLCODE=1
       and (x,y) in (select rx,ry from result) and (ClId=-1 or ClId=0);
   delete from seeds where seeds.sx=sx and seeds.sy=sy;
  }
}
```

DBSCAN with ATLaS (4)


```
aggregate regionQuery (qx real, qy real, Eps real):(real, real)
{
 initialize:
 iterate:
 terminate:
   {Insert into return select x,y from SetOfPoints where distance(x,
     y, qx, qy) <=Eps;
   }
}
```

SQL EM [OC00]

- An implementation of an EM iteration as SQL Queries
- Horizontal Approach
  - Compute the distance from the points to all centroids in a single scan
  - Drawback: very large query size as the distance function has to be expanded
- Vertical Approach
  - Pivot zed table => smaller statements but less performance

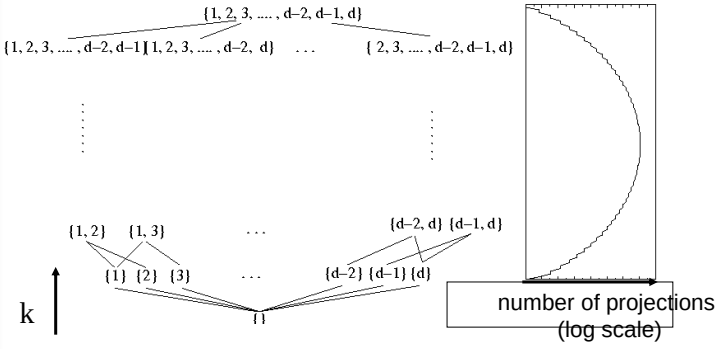
Combi Operator
[HLH03]

- Extend SQL by an additional OLAP operator
- Grouping in combination of attributes is a common task to many DM algorithms
- Scalable implementations of the operator
  - Memory based
  - Partition based
  - Sort based
- Speed up of 10 over simple Group By

Computing Projections

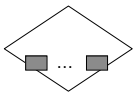
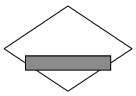
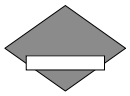
- Problem
  - efficiently compute histograms in low dimensional projections

$$\binom{d}{k} = \frac{d!}{(d-k)!k!}$$



Available Aggregation Mechanisms

- Fatal Disadvantages
  - simple GROUP-BY queries -> multiple scans
  - CUBE / ROLLUP / GROUPING SETS ?

	simple GROUP BY	GROUPING SETS	Cube
			
positive	multiple small queries	one single query	one single query
negative	- $\binom{d}{k}$ table scans - many single queries	large query statement (enumerate all combinations) internal implementation performs a group-by wrt. the aggregation base	large query statement (eliminate almost all grouping combinations)

Example: ...using GROUPING SETS

- Scenario
  - 2-dimensional projections within a 4-dimensional data set
  - GROUPING SETS() requires manual enumeration

```
SELECT A1, A2, A3, A4, COUNT(*) AS CNT
FROM ...
GROUP BY GROUPING SETS((A1,A2), (A1,A3),
(A1,A4),(A2,A3), (A2,A4), (A3,A4))
```

- Lesson learned
  - query size grows exponentially
  - no internal optimization

Example: ... using CUBE

- Scenario
  - 2-dimensional projections within a 4-dimensional data set
  - CUBE() requires the elimination of almost all computed combinations

```
SELECT A1, A2, A3, A4, COUNT(*) AS CNT
FROM ...
GROUP BY CUBE(A1,A2,A3,A4)
HAVING NOT(
 -- the 1-combination ...
    (GROUPING(A1)=1 AND GROUPING(A2)=1 AND
     GROUPING(A3)=1 AND GROUPING(A4)=1)
 -- all 3-combinations ...
    OR (GROUPING(A1)=1 AND GROUPING(A2)=1 AND GROUPING(A3)=1)
    OR   ...
 -- the 4-combination ...
    OR (GROUPING(A1)=0 AND GROUPING(A2)=0 AND
        GROUPING(A3)=0 AND GROUPING(A4)=0))
```

GROUPING COMBINATIONS

- Syntax with n grouping attributes

```
SELECT A1, A2, …, An, …
  FROM …
 GROUP BY GROUPING COMBINATIONS((A1,A2, ... ,An), k)
```

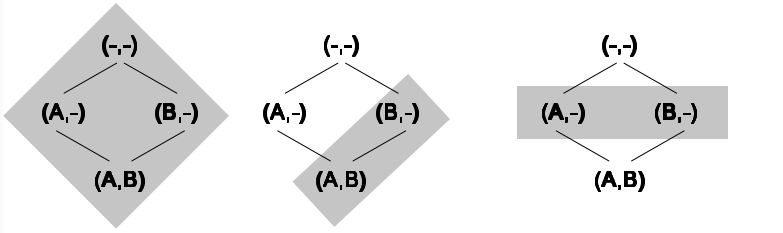
is equivalent to

```
SELECT A1, A2, …, An, …
  FROM …
 GROUP BY GROUPING SETS((A1,A2,...,Ak),
                        (A1,A2,...,Ak-1,Ak+1),
                        ...
                                              ))
```

- Semantics of the GROUPING COMBINATIONS operator
  - returns grouping combinations of size k
  - query size grows only linearly to the number of grouping attributes

Comparison with CUBE/ROLLUP

Example: grouping columns A, B



CUBE(A,B)

ROLLUP(A)

GROUPING COMBINATIONS((A,B),1)

Parameters of the COMBI operator
- **set of attributes**
- **size of the combination**

Main Memory-based Algorithm

- Assumption: the aggregation results for all combinations fit into the main memory at the same time
  - array-based implementation (based on the number of distinct values)
  - hash-based implementation (need to estimate the number of distinct combinations)

Algorithm 1 Main memory based algorithm for the COMBI operator.

Require: Relation $R(A_1, \ldots, A_n), k$ with $0 \leq k \leq n$,
for all tuple $t \in R$ **do**
  for all Grouping Combinations $c \in C$ **do**
    add the projection $c.P(t)$ to c.container
  end for
end for
Return C

Partition-based Algorithm

- Assumption
  - the aggregation results for **some** combinations fit into main memory at the same time
- Idea
  - compute p combinations within a single scan

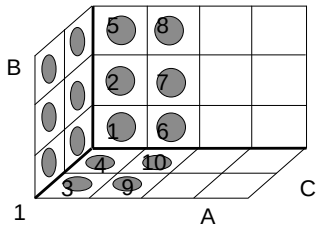
Algorithm 2 Partitioned algorithm for the COMBI operator.

Require: Relation $R(A_1, \ldots, A_n), k$ with $0 \leq k \leq n$,
for $i = 0; i < \lceil \binom{n}{k}/p \rceil; i++$ **do**
  for all tuple $t \in R$ **do**
    for all Grouping Combinations $c \in C$ **do**
      add the projection $c.P(t)$ to c.container
    end for
  end for
  write out C
end for

Sort-based Algorithm

- Assumption: data is sorted
  - sort order determined by number of distinct values
- Example
  - A,B,C
  - |A|=4
  - |B|=3
  - |C|=2

```
     A   B   C   Action  #
==============================
1: A1 B1 C1 ==> S(A1B1)
                S(A1C1)
                S(B1C1) 3
2: A1 B1 C2 ==> S(A1C2)
                S(B1C2)
                W(A1B1) 4
3: A1 B2 C1 ==> S(A1B2)
                S(B2C1) 6
```




```
     A   B   C    Action  #
==============================
 4: A1 B2 C2 ==> S(B2C2)
                 W(A1B2) 6
 5: A1 B3 C1 ==> S(A1B3)
                 S(B3C1)
                 W(A1C1) 7
 6: A1 B3 C2 ==> S(B3C2)
                 W(A1C2)
                 W(A1B3) 6
 7: A2 B1 C1 ==> S(A2B1)
                 S(A2C1) 8
 8: A2 B1 C2 ==> S(A2C2)
                 W(A2B1) 8
 9: A2 B2 C1 ==> S(A2B2) 9
10: A2 B2 C2 ==> W(A2B2) 8
11: A2 B3 C1 ==> S(A2B3) 9
12: A2 B3 C2 ==> W(A2B3)
                 W(A2C1)
                 W(A2C2) 6
13: A3 B1 C1 ==> ...
```

Sort-based Algorithm (cont.)

- Phase 1
  - compute combinations
- Phase 2
  - check for early output
- Benefit
  - sort operator is used to learn number of distinct values
  - COMBI is no longer a full pipeline breaker

Algorithm 3 Sort based algorithm

Require: Relation R
{G: set of grouping combinations}
{M: main memory slots}

```
1:  for all tuple t ∈ R do
2:    {Phase 1: compute all grouping combinations}
3:    for all grouping combinations g ∈ G do
4:      if (g(t) already allocated in main memory)
        then
5:        update S(g(t))
6:      else
7:        allocate new S(g(t))
8:      end if
9:    end for
10:   {Phase 2: check for partial results}
11:   for all grouping combinations g ∈ M do
12:     if (g() is ready for early output) then
13:       write to output W(g())
14:     end if
15:   end for
16: end for
```

Density Estimation, Data Bubbles & Micro-Clusters

- Techniques to get a raw estimation of the point density
  - CF Vectors [ZRL96] -> BIRCH
  - Micro-Clusters [THH01] -> CLARANS
  - Data Bubbles [BKKS01,SZ03] -> OPTICS
  - Histograms [PKGF03] -> LOCI

Categorical Clustering

- ROCK [GRS99]
- STIRR [GKR98]
- CACTUS [GGR99]

ROCK [GRS99]

Similarity functions for clustering

- Numerical data: L_P-metric
- Categorical data:
  - L_P-metric
  - Jaccard coefficient

These types of similarity functions do not work for categorical data

ROCK

Two points are called neighbors, if

$$sim(p_i, p_j) = p_i \cap p_j \,/\, p_i \cup p_j \geq \theta$$

To separate clusters better, define

link(p_i, p_j) = # common neighbors of p_i and p_j

Use this as a similarity function.

ROCK

Quality function - maximize

$$E = \sum_{i=1}^{k} n_i * \sum_{p,q \in C_i} \frac{link(p,q)}{n_i^{1+2f(\Theta)}}$$

A point belonging to cluster C_i has approximately $n_i^{f(\Theta)}$ neighbors in C_i

Example for f: $f(\Theta) = \frac{1-\Theta}{1+\Theta}$

ROCK

- Hierarchical Algorithm
- Merge Clusters C_i and C_j with maximal

$$g(C_i, C_j) = \frac{link(C_i, C_j)}{(n_i + n_j)^{1+2f(\Theta)} - n_i^{1+2f(\Theta)} - n_j^{1+2f(\Theta)}}$$

Details to make the algorithm efficient can be found in the paper.

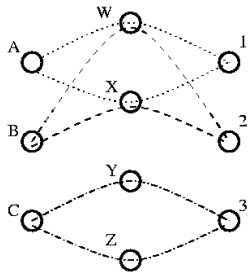
STIRR [GKR98]

- Methods based on hypergraph clustering
  - Nodes correspond to categorical values
  - Edges correspond to tuples
- Existing approaches encounter combinatorial problems (–> NP-completeness)
- STIRR: Generalization of spectral partitioning techniques to the problem of hypergraph clustering

STIRR

Tuple	Attribute a	b	c
1.	A	W	1
2.	A	X	1
3.	B	W	2
4.	B	X	2
5.	C	Y	3
6.	C	Z	3

is represented as



STIRR

- Based on a dynamic system
- Configuration is an assignment of a weight w_v to each node v
- Apply f iteratively until w = f(w)

STIRR

To update the weight w_v of node v:

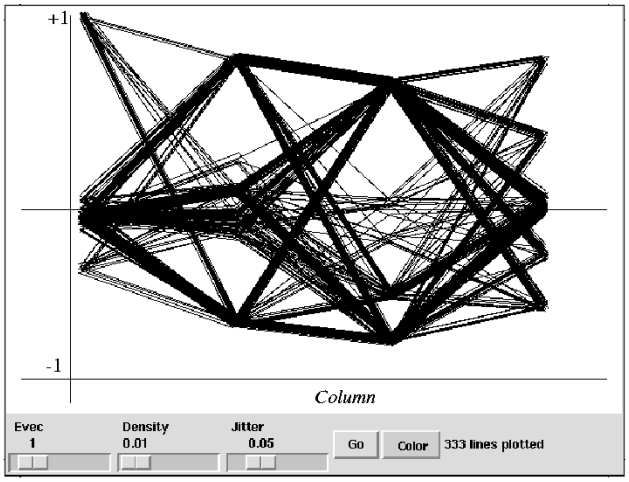
For each tuple $t = \{ v, u_1, \ldots, u_k \}$ containing v

$x_t \leftarrow \oplus(u_1, \ldots, u_k)$

$w_v \leftarrow \Sigma_t x_t$

Updating the weights for each node w_v gives the function f

STIRR



CACTUS [GGR99]

$C=(C_1, \ldots, C_n)$ is called a cluster if

- For all $i, j \in \{1, \ldots, n\}$, $i \neq j$, C_i and C_j are strongly connected
- For all $i \in \{1, \ldots, n\}$ C_i is maximal.
- The support fulfills $\sigma(C) > \alpha |D|$

CACTUS

Consider the co-occurrences of attribute values

Two sets of values $C_i \subseteq D_i$ and $C_j \subseteq D_j$ of different attributes are called strongly connected, if all pairs of values $a_i \in D_i$ and $a_j \in D_j$ occur more frequently than expected

CACTUS

The Algorithm:

1. Summarization Phase
   - Inter- and Intra-attribute summaries
   - Access the data set
2. Clustering Phase
   - Determine cluster candidates
3. Validation Phase
   - Determine actual clusters from the set of candidates

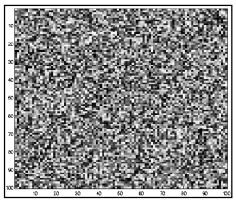
Interactive Clustering

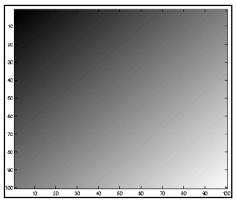
- Image is Everything [AKN01]
- HD-Eye [HWK99, HKW03]

Basic Idea of Pixel Validity

[AKN01]

- Suppose we have numeric attributes: X, Y, and Z
- Is there any dependency or interrelation between X, Y and Z?
  - ☞ Idea: Check the continuity of Z over (X,Y)!

Not dependent: 

Dependent: 

Pixel Validity: Background

- The statistical route:

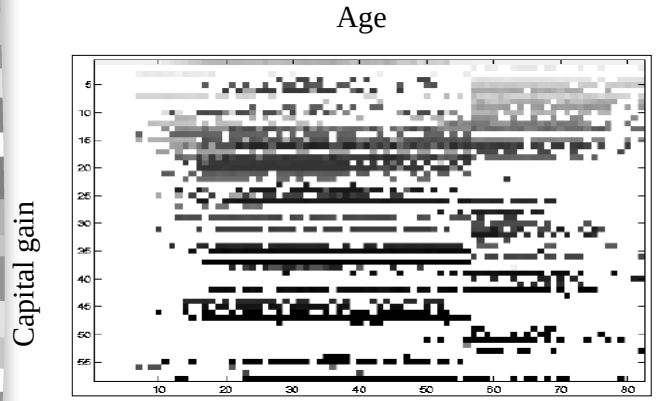
Hypothesis → Testing → Conclusion: Yes/No

- Statistical Tools:
  - Recognize interrelations in the entire data set. Much more difficult to find local interrelations.
  - The attributes to be explored must be pre-specified. It is very costly to check all hypotheses on all combinations of attributes.

Pixel Validity: Background

- The Inference of Statistics and Data Mining:
  - ☞ Statistics has little to offer in generating hypotheses, but a great deal to offer in evaluating the hypotheses.

Example: Census Data

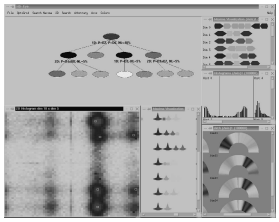


Age, CapGain ⇒ Adjusted Cross Income

Basic Idea of HD-Eye

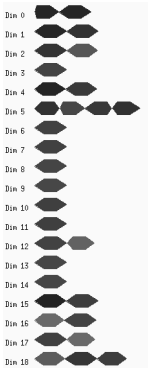
[HWK99, HKW03]

- Integration of Visual and Automated Data Mining
- Support the critical steps of clustering algorithms by visualization techniques

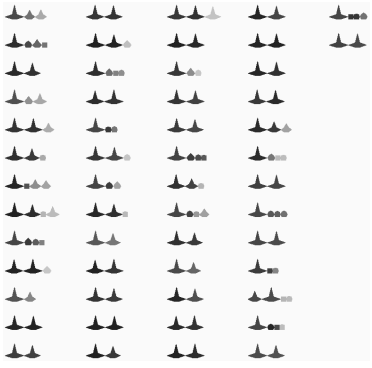


HD-Eye

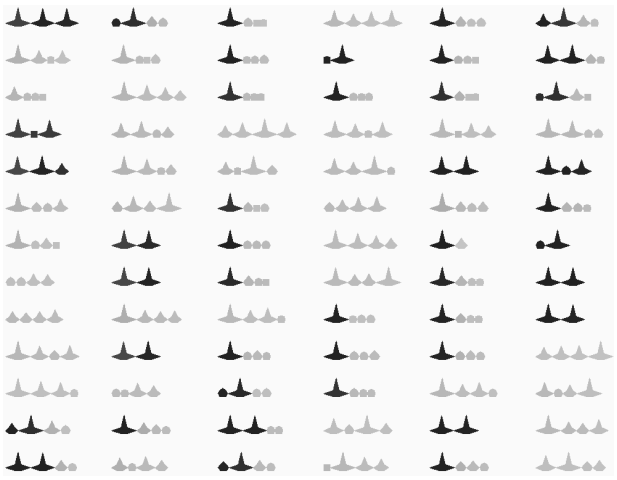
Icons for one-dim. Projections



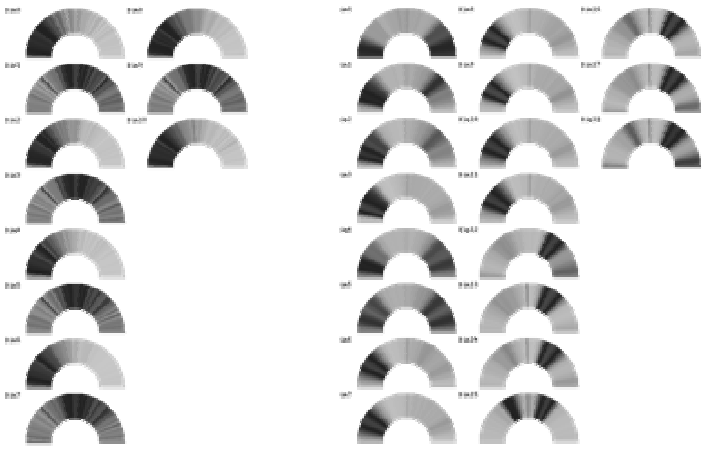
Icons for two-dim. Projections



HD-Eye

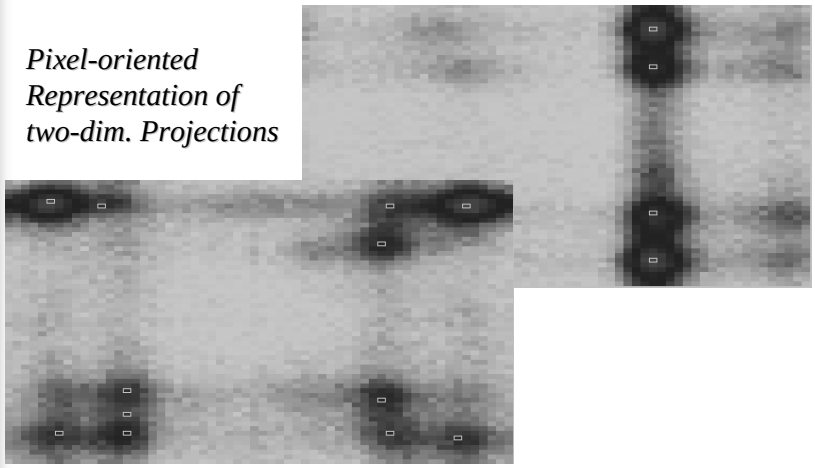


HD-Eye



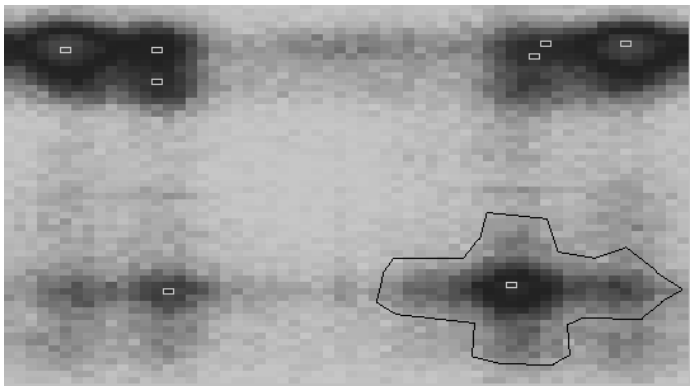
Pixel-oriented Representation of one-dimensional Projections

HD-Eye



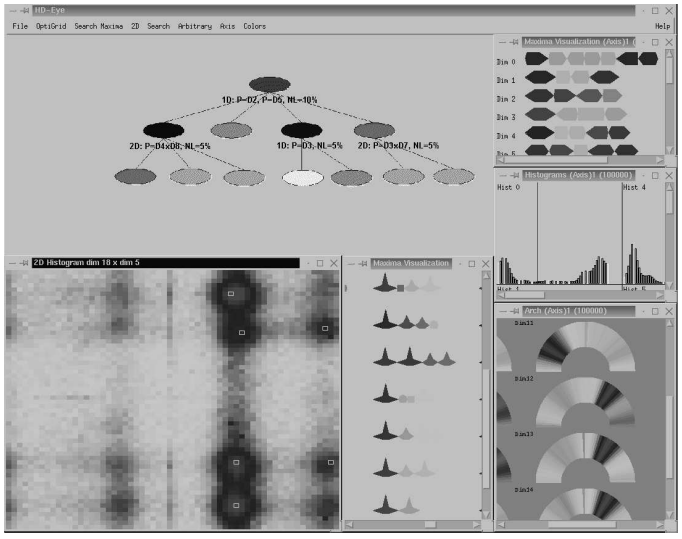
Pixel-oriented Representation of two-dim. Projections

HD-Eye



Interactive Specification of Cutting Planes in 2D Projections

The HD-Eye System



Applications of Clustering

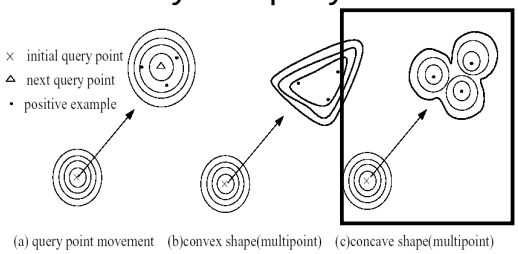
- Images
- Micro-Array Data from Bio-Informatics
- Geographical Data

Clustering for Image Retrieval
[KC03]

- Standard content based Image Retrieval:
  - Derive feature vectors from the images
  - Determine the top k nearest neighbors to a given query point
- Complex Similarity Queries
  - Determine the top k nearest neighbors to a set of query points (near to any query point)

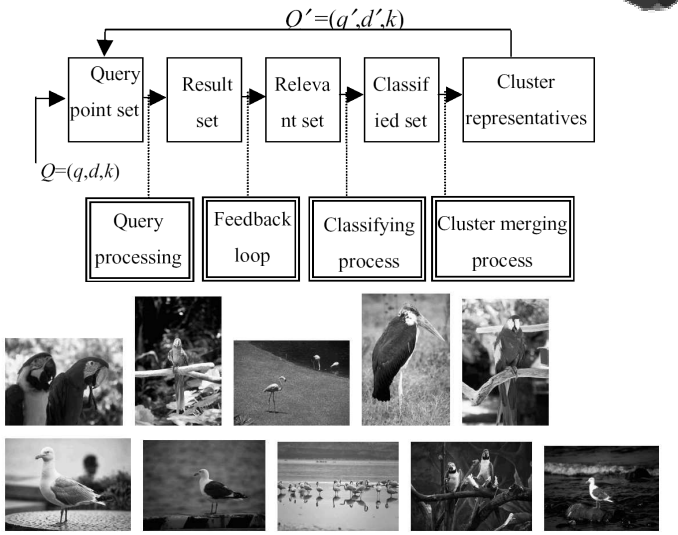
Usage of Relevance Feedback

- Iterative result refinement, the user picks relevant results => modify the query
- Concepts:



- Problem: many redundant query points after some iterations
- Solution: Clustering of intermediate query points

General Approach

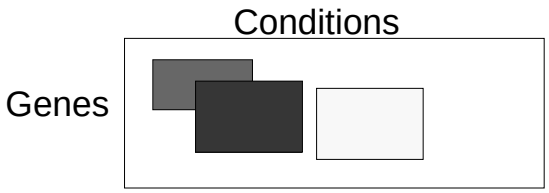


The Usage of Clustering

- Hierarchical Centroid Clustering
- Reduce the set of query points to a given size
- For this increase the effective radius of the clusters, which determine whether a new point is inside the cluster or not. (concept similar to BIRCH)

Clustering of Micro-Array Data

- Given:
  - matrix with genes and conditions as rows/cols
  - entries are the expression value of a particular gene/condition combination
- Problem:
  - Find a subgroup of genes which are co-regulated under a particular subset of conditions



Clustering of Micro-Array Expression Data

- Bi-Clustering [CC00]
- Enhanced Bi-Clustering, (delta Cluster) [YWWY03, YWWY02]
- Fuzzy k-means [GE02]

Bi-Clustering, Cluster Definition

- X set of Genes, Y set of Conditions, $a_{i,j}$ element of the Expression Matrix A
- The pair (I, J) with $I \subset X, J \subset Y$ is a bi-cluster
- Quality is measured by the mean squared residue score (MSRS)

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

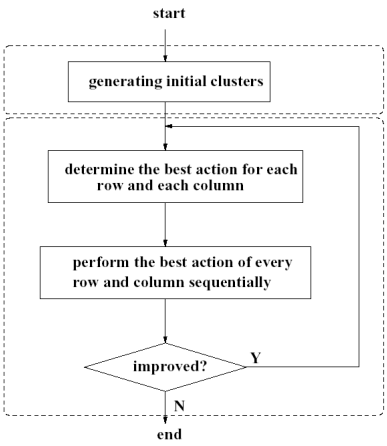
$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} \qquad a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \quad a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$$

- Invariants:
  - Trivial bi-cluster (nearly constant values for $a_{i,j}$)
  - Scaling, translation

Bi-Clustering, Cluster Definition

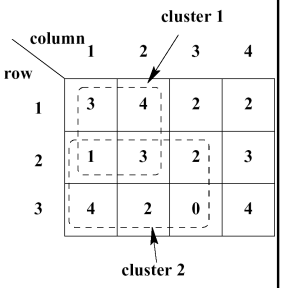
- Problem: Find the largest square δ bi-cluster with ($MSRS \leq \delta, |I| = |J|$)
- The Problem is NP-hard [CC00]
- Greedy Algorithms [CC00]
- FLOC (FLexibile Overlapped biClustering) [YWY02, YWY03a, YWY03b]
  - Iteratively move rows and columns to build good bi-clusters

FLOC Algorithm



Phase 1

Phase 2



Example

Fuzzy k-Means [GE02]

- Idea
  - Model overlapping clusters of genes by member-scores
  - A gene may have high score for multiple clusters
- Objective Function:
  - X_i … Expression Pattern of ith gene
  - V_j … centroid of cluster j
  - d_{XiVj} … Pearson distance
  - m_{XiVj} … Membership of Xi in Cluster j

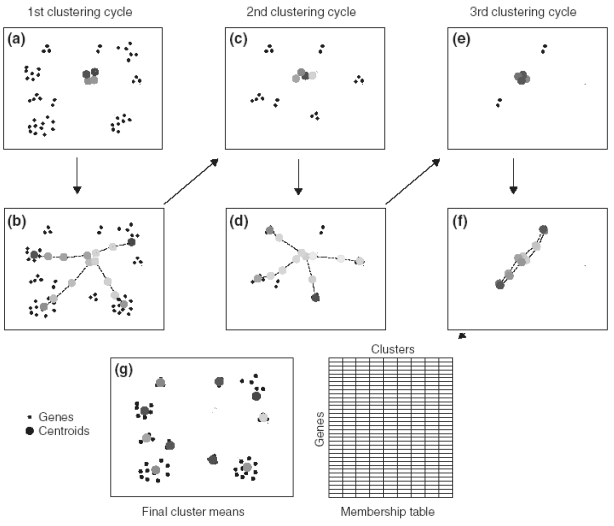
$$J(F,V) = \sum_{i=1}^{N}\sum_{j=1}^{K} m_{XiVj}^2 \, d_{XiVj}^2$$

$$m_{XiVj} = \frac{\frac{1}{d_{XiVj}^2}}{\sum_{j=1}^{K} \frac{1}{d_{XiVj}^2}}$$

Fuzzy k Means

- Heuristic to determine a good k
- Three Cycles
  1. Init centroids at Eigen vectors, determine membership and move centroids to weighted mean until covergence
  2. Merge close Centroids (Pearson corr. >0.9), eliminate obj. with Pearson corr. >0.7, add new centroids
  3. Repeat step 2
- Final Step: Determine the membership based on the set of identified centroids

Fuzzy k Means Example



Problems of all Approaches

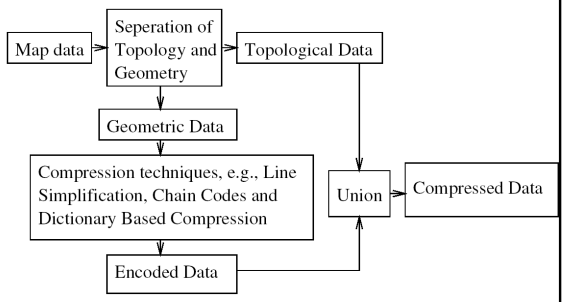
- Validation of the clusters
  - Many algorithms available
    => to many results to inspect them manually
  - Comparison of the clusters found by different methods
  - Comparison to published results
- Many different application scenarios of micro-array chips
  - Good results from one application may not be applicable to others

Clustering of Geographical Data

- Example Application:
  - Vector Map Compression [SHDZ02]
- Motivation:
  - Mobil devices require access to spatial data for location based services
  - Small representations of vector maps are needed

Vector Map Compression

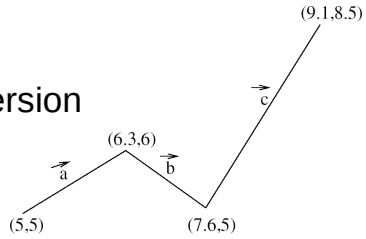
- Overview



- Representation of Line Segements
  - Convert lines (lists of 2D points) into base point + difference vectors
  - Two flavors: Delta(i,0) or Delta(i,i-1)

Vector Map Compression

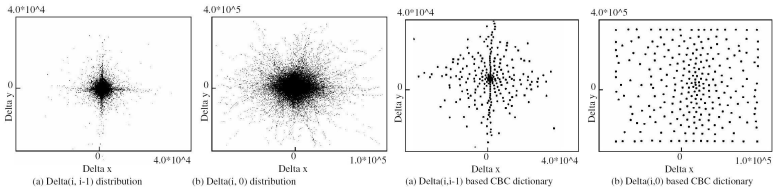
- Example for Line Conversion
  - Delta(i,i-1):
    (5,5)| (1.3,1) (1.3,-1) (1.5,3.5)
  - Delta(i,0):
    (5,5)| (1.3,1) (2.6,0) (4.1,3.5)

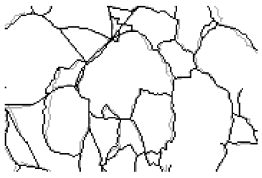


- The differential vectors are clustered with k-Means => Diff Vector Dictionary
- The compressed lines consist of the base points plus dictionary entries for each diff vector

Examples

- Data Distributions from US Road Maps





- Dark Lines: Original
- Light Lines: Compressed

Conclusions

- Open Problems & Future Research
  - Clustering on Data Streams
  - Specification of the Similarity Measure
  - Evaluation of Clustering
- New Concepts for new Applications
  - Clustering of Graphs

Clustering on Data Streams

- Related work from the machine learning community: Online Learning
- First approaches in Data Mining
  - Based on additivity of CF Vectors (Micro Clusters)
- More real applications with new cluster definitions can be expected

Specification of Similarity

- The right similarity measure is critical for the successful usage of clustering
- Often only standard metrics such as the Minkowski, Manhattan or Euclidian metrics are used
- Flexible scaling methods independent from the used metric are needed

Evaluation of Clustering

- Evaluation depends on application
- General Application Scenarios
  - Clustering for lossy data compression
    - Quantization Error
  - Clustering for finding more general categories
    - Depends on the interpretation in the application context
    - Flexible Frameworks are needed

New Applications

- Growing need to cluster complex objects
- Not only feature vectors but
  - graphs
  - matrices
  - sequence patterns (not only strings)
  - ...