

Algorithmic Engineering of Clustering and Cluster Validity with Applications to Web Usage Mining

Jianhua Yang

(B.Eng., Engineering Design, Hebei Institute of Technology, China, 1984

M.Eng., Computer Graphics & CAD, Dalian University of Technology, China, 1990)

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy



School of Electrical Engineering and Computer Science
The University of Newcastle
Callaghan NSW 2308
Australia

November 2002

Certificate of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree at any other University or Institution.

(Signed) _____
Jianhua Yang

© Copyright 2003

Jianhua Yang

Acknowledgements

I would like to thank my supervisor, Associate Professor Vladimir Estivill-Castro, for giving me invaluable guidance and encouragement during the preparation of this thesis and throughout my research. His knowledge, experience, motivation, and enthusiasm make my research worthwhile. His support towards my participation in international conferences, and commitment to an environment with adequate equipments and facilities is appreciated.

I wish to thank the other members of our Data Mining and Knowledge Discovery Research Group, in particular Dr. Ickjai Lee, Mr. Rodolfo Torres-Velázquez, Mr. Ozgur T. Pusatli, and Ms. Kyungmi Lee, for our wonderful discussions and collaboration. The interaction with them both socially and academically has been a highlight of my life. Thanks also goes to all members of Machine Learning Journal Club for discussions on current issues and on technical advances in machine intelligence, where Dr. Stephan Chalup has inspired and influenced me over the years.

I would like to express my sincere appreciation to the University of Newcastle for offering me a UNRS scholarship throughout my doctoral program. I would also like to thank the faculty, staff and fellow postgraduates in the School of Electrical Engineering and Computer Science for their hospitality and kindness. They provided a friendly and enjoyable environment during my time here. They were highly supportive and helpful.

Last but not least, I am deeply grateful to my wife, Daijun Yu, and my son, Foning Yang. They gave me inspiration and put off family activities so that I could focus on this thesis.

To my wife, **Daijun Yu** and
my son, **Foning**.

List of publications arising from this thesis

1. Estivill-Castro V., Yang J., Cluster Validity Using Support Vector Machines, Submitted to the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-03). Seoul, Korea, April 30 - May 2, 2003.
2. Yang J., Estivill-Castro V., & Chalup S. K., Support Vector Clustering through Proximity Graph Modeling, *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02)*. Orchid Country Club, Singapore, November 18-22, 2002.
3. Estivill-Castro V., Yang J., Categorizing Web Visitors Dynamically by Fast and Robust Clustering of Access Logs, *Proceedings of the First Asia-Pacific Conference on Web Intelligence (WI'2001)*, Lecture Notes in Artificial Intelligence Vol. 2198, pp 498-507, Ning Zhong and Yiyu Yao (eds) Springer-Verlag, Maebashi City, Japan, October 23-26, 2001.
4. Estivill-Castro V., Yang J., Non-crisp Clustering by Fast, Convergent and Robust Algorithms, *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, Lecture Notes in Artificial Intelligence Vol. 2168, pp 103-114, L. De Raedt and A. Siebes (eds) Springer-Verlag, Freiburg, Germany, September 3-7, 2001.
5. Estivill-Castro V., Yang J., Fast and Robust General Purpose Clustering Algorithms, *Proceedings of the Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)*, Lecture Notes in Artificial Intelligence Vol. 1886, pp 208-218, R. Mizoguchi and J. Slaney (eds) Springer-Verlag, Melbourne, Australia, August 29-September 1, 2000.

6. Estivill-Castro V., Yang J., Clustering Web Visitors by Fast, Robust and Convergent Algorithms, *Special Issue of IJFCS (International Journal of Foundations of Computer Science) on Mining the Web*, Vol. 13, No. 4, pp 497-520, C. X. Ling and N. Cercone (eds), 2002.
7. Estivill-Castro V., Yang J., Fast and Robust General Purpose Clustering Algorithms, *Journal of Data Mining and Knowledge Discovery*. To appear.

Contents

List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Data mining and knowledge discovery	1
1.1.1 Why data mining?	1
1.1.2 What is data mining?	2
1.1.3 The methodology of data mining	3
1.2 Cluster analysis	3
1.2.1 Clustering tasks	4
1.2.2 Clustering applications	5
1.3 Motivation	6
1.3.1 The components of data mining algorithms	6
1.3.2 Algorithmic engineering of clustering	7
1.4 Contribution	10
1.5 Outline of the thesis	12
2 Basic Concepts and Related Work	15
2.1 The clustering process	15
2.2 Feature selection	17
2.3 Similarity measures	19
2.3.1 Euclidean distance and Minkowski metrics	20
2.3.2 Cosine dissimilarity measure	20

2.3.3	Arbitrary distances	21
2.3.4	Proximity matrix	21
2.3.5	Differences in scales and domains of features	22
2.4	Cluster information modeling and clustering criteria	22
2.4.1	Probabilistic model	23
2.4.2	Representative-based model	24
2.4.3	Model based on support vectors	26
2.4.4	Model based on proximity graphs	27
2.4.5	Other models	29
2.5	Clustering algorithms	29
2.5.1	Partitional vs. hierarchical clustering	30
2.5.2	Based on all candidates vs. sampling-based	35
2.5.3	Hard vs. fuzzy clustering	36
2.5.4	Graph-based vs. density-based	37
2.6	Cluster validity	41
2.6.1	Problem specification	41
2.6.2	Fundamental concepts of cluster validity	42
3	Representative Clustering on Vector Spaces	44
3.1	How k -MEANS works	45
3.1.1	The optimization problem	45
3.1.2	Actual implementations	47
3.1.3	k -MEANS vs. EXPECTATION MAXIMIZATION	48
3.1.4	Interpretation of k -MEANS	51
3.2	Generalization of representative-based clustering	54
3.2.1	The optimization problem	55
3.2.2	Representative prototypes: continuous vs. discrete	55
3.2.3	Induction principles: squared error vs. absolute error minimization	56
3.2.4	Summary and comparison	58
3.3	Fast and robust general purpose clustering algorithms	58
3.3.1	General purpose clustering algorithms	58

3.3.2	Design of algorithms	59
3.3.3	Discrete gravity solution (DGP)	60
3.3.4	Continuous Fermat-Weber solution (CFW)	60
3.3.5	Discrete Fermat-Weber solution (DFW)	62
3.3.6	Other solutions	66
3.4	Performance evaluation	68
3.4.1	Time complexity	68
3.4.2	Resistance to noise	70
3.4.3	3D mixture data test	72
3.5	Remarks	75
4	Representative Clustering on Metric Spaces	77
4.1	Formal background of sequential data	78
4.1.1	Definitions of sequential data	78
4.1.2	Statistics of sequences	79
4.1.3	Similarity measures of sequential data	79
4.2	Mining sequential patterns: tasks and challenges	82
4.3	Existing clustering algorithms for sequences	84
4.3.1	Matrix-based clustering	84
4.3.2	Pattern-based clustering	85
4.3.3	Clustering algorithms on vector spaces	86
4.4	Representative clustering on metric spaces	87
4.4.1	Requirements of clustering algorithms for sequences	87
4.4.2	Design of clustering algorithms for sequences	89
4.5	Implementations and analyses of randomized algorithms	91
4.5.1	The EXPECTATION MAXIMIZATION type	92
4.5.2	The discrete hill-climber type	99
4.6	Remarks	101
5	Support Vector Clustering	102
5.1	Density estimation and domain description	102
5.2	Novelty detection using Support Vector Machines	104

5.2.1	Support Vector based Novelty Detection (SVND)	104
5.2.2	Formalization of SVND using kernel methods	105
5.3	Support vector clustering	107
5.3.1	Working principle	107
5.3.2	SVM training	108
5.3.3	Cluster labeling	108
5.4	Model enhancement using proximity graph	111
5.5	An efficient approach to SVC	112
5.5.1	Cluster labeling guided by proximity graphs	113
5.5.2	Cluster harvest	114
5.6	Performance evaluation	114
5.6.1	Time complexity	115
5.6.2	Clustering quality	117
5.7	Remarks	119
6	Cluster Validity	120
6.1	Cluster validity tasks	121
6.1.1	Clustering tendency	121
6.1.2	Parameter tuning	121
6.1.3	Choosing best-fit algorithms	123
6.1.4	Robustness of clustering output	123
6.1.5	Measuring significance of the revealed structure	124
6.2	Cluster validity methods	124
6.2.1	Statistical methods	125
6.2.2	Information-based methods	126
6.2.3	Formal validation	126
6.2.4	Empirical evaluation	126
6.2.5	Visualization-based evaluation	127
6.2.6	Generic discussion	127
6.3	Classification using SVMs	129
6.3.1	Linear hypothesis space	129

6.3.2	Non-linear hypothesis space	130
6.3.3	The ν -SVM approach	131
6.4	Cluster validity using SVMs	133
6.4.1	Separation measure	135
6.4.2	Compactness measure	136
6.5	Performance evaluation	138
6.5.1	Separation test: a normal case	138
6.5.2	Separation test: other cases	139
6.5.3	Outliers test: a general case	140
6.5.4	Outliers test: repeatable effects	140
6.5.5	Examples of 2D data	142
6.5.6	Examples of 3D data	145
6.6	Remarks	147
7	Case Studies: Web Usage Mining and Text Mining	148
7.1	Case study 1: Web usage mining	149
7.1.1	Preprocessing for Web usage mining	150
7.1.2	Clustering methods	151
7.1.3	On the scalability of the methods	151
7.1.4	On the quality of clustering	152
7.1.5	Discussion	154
7.2	Case study 2: Text mining	156
7.2.1	Data preparation	157
7.2.2	Clustering methods	158
7.2.3	Validity indices	158
7.2.4	Experimental results	159
8	Conclusions and research directions	161
8.1	Summary	161
8.2	Implementation issue	164
8.3	Research directions	166

CONTENTS

xii

Bibliography

171

List of Figures

2.1	The process of cluster analysis.	16
2.2	Data modeling and structuring with representatives	25
2.3	Data modeling and structuring with SVs	26
2.4	Data modeling with the Voronoi and Delaunay diagrams	28
2.5	Hard clustering and fuzzy clustering.	36
3.1	Generic iterative structure of representative-based clustering.	47
3.2	A function $\text{GRAVITY}(\vec{x})$ and its level curves	53
3.3	A Fermat-Weber function and its level curves	57
3.4	Trace of iteration towards the CONTINUOUS FW CENTER.	61
3.5	Fermat-Weber level curve and its normal and tangent	63
3.6	Comparison of four different types of centers (estimators of location) on the same data set	67
3.7	Comparison of CPU times for R. N. Neal examples for Bayesian mix- ture models.	69
3.8	Data is uniform if projected to either axis, but has a clear pattern. . .	75
4.1	Body of the iteration alternating between finding a classification for the data given a model, and finding a model given classified data. . .	89
5.1	Controlling cluster structures using training parameters	109
5.2	Various proximity modeling graphs	112
5.3	Procedure of SVC using DD modeling	116
5.4	CPU-time comparison of various cluster labeling mechanisms	117
5.5	Comparison of clustering results using different labeling methods . . .	118

6.1	The working principles of SVMs for classification tasks	130
6.2	Illustration of cluster validity using SVMs.	134
6.3	Problems avoided by measurements in feature space	136
6.4	Checking separation between clusters (a normal case)	139
6.5	Checking separation between clusters (other cases)	140
6.6	Checking outliers (a general case)	141
6.7	Checking outliers for estimating the compactness of clustering results (compact case)	142
6.8	Checking outliers for estimating the compactness of clustering results (non-compact case)	143
6.9	A 2D example of cluster validity through SMVs approach	144
6.10	A 3D example of cluster validity through SMVs approach	146
7.1	CPU-times comparison of Matrix-based algorithms and representative- based algorithms	152
7.2	CPU-times of different versions of representative-based algorithms . .	156
8.1	GUI of k -group clustering analyzer.	165
8.2	GUI of Web usage miner.	166
8.3	GUI of cluster validity detector.	167

List of Tables

3.1	Summary of representative-based optimization problems.	58
3.2	Misclassification with 95% confidence intervals (one data set).	71
3.3	Misclassification with 95% confidence intervals (10 data sets).	72
3.4	Results for 3-D mixture of normals with 20% noise.	73
3.5	Results for 10 data sets each consisting of a 3-D mixture of normals .	74
5.1	Types of kernel functions	107
7.1	Error rates of clustering results using Matrix-based algorithms and representative-based algorithms	154
7.2	Error rates of clustering results using different versions of representative- based algorithms	155
7.3	The proportion of discrepancies in clustering	157
7.4	Algorithms ranked by quality of the clustering on Reuters data sets .	159

Abstract

While many clustering methods have been suggested within data mining, they provide few clues towards the effective design of these methods at a fundamental algorithmic level. We argue that successfully designing and evaluating clustering methods must take into account a solid understanding of the algorithmic components of clustering methods, namely models, induction principles and search strategies. The main objectives of this research are to provide solid scientific foundations and empirical exercises for designing fast and robust clustering algorithms, and to investigate computability and complexity issues that arise naturally in the study of clustering algorithms. To this end, we take the route of the algorithmic engineering approach: 1) we analyze the requirements of clustering algorithms; 2) we design models and induction principles that reflect these requirements and then employ efficient heuristic search strategies and data structures accordingly; 3) we examine the algorithmic complexity of the designed algorithms mathematically whenever possible; 4) we implement the designed algorithms in high level languages and evaluate the algorithms and the quality of their clustering results by well understood validity processes; 5) we conduct case studies where we transfer advanced algorithmic techniques into important applications.

This thesis discusses representative-based clustering on vector spaces. We propose thoroughly engineered clustering algorithms that improve on the performance of their most similar approach, namely, the k -MEANS approach. These novel algorithms model cluster information using so-called discrete prototypes. The thesis formalizes the algorithms' induction principles differently, so the resulting optimization criteria evaluate the absolute loss. The heuristic search we propose employs a filtering strategy to improve efficiency. Such a clustering framework is then extended to cluster data objects in arbitrary metric spaces, where the algorithms generalize fuzzy membership in cluster assignment and utilize a randomized strategy to efficiently search for the solutions.

The thesis also studies support vector clustering, where extreme vectors are used to

describe a cluster structure. This is in contrast to representative-based modeling. To improve the efficiency of support vector clustering, we incorporate proximity graphs to enhance data modeling. The combination of models with extreme vectors and proximity graphs shows a promising improvement on time requirements and clustering quality.

Interpretability and validity of clustering results are of great importance. The thesis addresses the issue of cluster validity, and proposes a novel approach to evaluate clustering quality. This approach is motivated by support vector machines. By learning clustering results, this method gains an insight in the cluster structure inherent in the data. It provides both analytical measurements and geometric understandings. It is convenient for validity tasks within complex settings, such as high dimensional data, clusters with arbitrary shapes and sharp density fluctuations.

Algorithmic engineering involves implementation and thorough testing towards practical problem solving. Thus, we also bring the power of algorithmic engineering into the experiments carried out in this research to validate clustering performance. This serves as a useful support to assess improvements of algorithms, to verify theoretical developments, to investigate ideas that theoretical techniques fail to analyze precisely, and to give a new insight into possible fundamental solutions. For this purpose, we develop software packages that implement our algorithms with Object-Oriented framework and friendly GUIs. We conduct two case studies (text mining and Web usage mining) to illustrate real-world applications.

Notation

Terms and definitions

The following terms and notation are used throughout this thesis.

- pattern* a single data item used by the clustering algorithms. In vector spaces, patterns are often referred as to feature vectors or data points. In arbitrary metric spaces, patterns are often referred as to (symbolic) objects.
- mean* observed average value. The mean of $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$ is the location that minimizes $\text{GRAVITY}(\vec{x}) = \sum_{i=1}^{n_j} \text{EUCLID}^2(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} (\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i)$.
- median* the Fermat-Weber center of $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$ that minimizes $\text{FW}(\vec{x}) = \sum_{i=1}^{n_j} \text{EUCLID}(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} \sqrt{(\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i)}$, with an optional constraint [subject to $\vec{x} \in \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$].
- centroid* the centroid of $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$ is the continuous location that minimizes $M^a(\vec{x}) = \sum_{i=1}^{n_j} d(\vec{x}, \vec{x}_i)^a$.
- medoid* the medoid of $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$ is the discrete data point that minimizes $M^a(\vec{x}) = \sum_{i=1}^{n_j} d(\vec{x}, \vec{x}_i)^a$, subject to $\vec{x} \in \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{n_j}\}$.
- center* a centroid or a medoid.

Mathematical symbols

- $\Re^D$ a D -dimensional real vector space
- $\mathfrak{S}$ a feature space
- X a set of data points in $\Re^D$
- U a set of data objects in pseudo-metric spaces
- S an eventset
- $\mathcal{C}$ a cluster structure of X or U
- $\mathcal{C}$ a set of representatives of X or U
- $\mathcal{P}$ a partition of X or U
- P a sampled subset of X or U

$\vec{x}_i$	the i -th data point in X
u_i	the i -th data object in U
$d(\cdot, \cdot)$	the dissimilarity or distance between two data objects
$Sim(\cdot, \cdot)$	the similarity between two data objects
$M()$	an objective function
$AE()$	the absolute error criterion
$SE()$	the squared error criterion
$p()$	a probability density function
A	a matrix
$\mathcal{A}$	a clustering algorithm
G	a graph with vertices and edges
CG	the complete graph
SVG	the support vector graph
$VD, VD(X)$	the Voronoi diagram of X
$DD, DD(X)$	the Delaunay diagram of X
$MST, MST(X)$	the Minimum Spanning Tree of X
$k - NN, k - NN(X)$	the k -nearest neighbor graph of X
$e_{i,j}$	an edge connecting two vertices
p	a training parameter for SVM learning
q	width of the Gaussian kernels
k	the number of clusters
m, n	the number of data objects in a data set
R	radius
α	Lagrangian multiplier
γ	margin between a pair of clusters
λ	a scaling factor
ν	a training parameter for SVM learning
Δ	a gradient
$O(f(n))$	a function bounded from above by $f(n)$
$\Omega(f(n))$	a function bounded from below by $f(n)$
$\Theta(f(n))$	both $O(f(n))$ and $\Omega(f(n))$

Acronyms

<i>BEA</i>	Bond Energy Algorithm
<i>BSV</i>	Bounded Support Vector
<i>CDA</i>	Confirmatory Data Analysis
<i>CF</i>	Clustering Feature
<i>CF – Tree</i>	Clustering Feature Tree
<i>CFW</i>	Continuous Fermat-Weber Problem
<i>CGP</i>	Continuous Gravity Problem
<i>DDD</i>	Data Domain Description
<i>DFW</i>	Discrete Fermat-Weber Problem
<i>DGP</i>	Discrete Gravity Problem
<i>EDA</i>	Exploratory Data Analysis
<i>EM</i>	Expectation Maximization
<i>FW</i>	Fermat-Weber problem
<i>GIS</i>	Geographic Information Systems
<i>GPS</i>	Global Positioning System
<i>KDD</i>	Knowledge Discovery in Databases
<i>LEDA</i>	Library of Efficient Data Types and Algorithms
<i>ML</i>	Maximum Likelihood
<i>MLE</i>	Maximum Likelihood Estimate
<i>MDL</i>	Minimum Description Length
<i>MML</i>	Minimum Message Length
<i>SV</i>	Support Vector
<i>SVC</i>	Support Vector Clustering
<i>SVND</i>	Support Vector Novelty Detection
<i>SVMs</i>	Support Vector Machines
<i>VDA</i>	Visual Data Analysis
<i>VQ</i>	Vector Quantization

Chapter 1

Introduction

1.1 Data mining and knowledge discovery

1.1.1 Why data mining?

Across a wide variety of fields, data is being collected and accumulated at a dramatic pace. Powerful database systems for collecting and managing information are in use in virtually all large and mid-range organizations. Statistical studies demonstrate the explosive growth of data in information-intensive environments, such as the World Wide Web (WWW), Geographic Information Systems (GIS), and data warehouses. The rapid emergence of electronic data management leads to the “digital age”. For example, the Web exceeds 800 million HTML pages with six terabytes of data on about three million servers. Almost a million pages are added daily, and several hundred gigabytes change every month [30]. In GIS, the data gathering process has become faster and cheaper by the aid of the advances in data acquiring technologies such as Global Positioning System (GPS), digital remote sensing, aerial photography, and automatic or semi-automatic scanners. Geo-spatial information on the Web is getting bigger and even easier to obtain [115]. Commercial databases are growing

at unprecedented rates. As a submarket of data warehouse, the market for Packaged Business Intelligence (PBI) grew from \$602.3 million in 1998 to \$3.2 billion in 2003 [116]. This explosive growth in data cannot be matched by the current capabilities for analysis available in the literature. That is, the advances in our technology for collecting, managing and communicating data far exceed the technological capability to analyze it. There is an urgent need for a new generation of computational theories and tools to assist humans in extracting useful knowledge from the rapidly growing volume of digital data [64]. Consequently, data mining has become a research area with increasing importance [33].

1.1.2 What is data mining?

Data mining is sometimes regarded as a synonym for Knowledge Discovery in Databases (KDD). We adhere here to a well-established definition of KDD:

“KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data” [66].

KDD is a series of processes involving several interactive and iterative steps that may require quantitative inference and suggest unexpected patterns. The discovered unseen patterns are beneficial, understandable and applicable to new data sets with some degree of certainty. The KDD process consists of three main steps: pre-mining, data mining and post-mining [106]. Pre-mining selects a subset of data to mine, cleans this subset by removing redundancy and incomplete data, and transforms it to a suitable format for the algorithms at hand. Data mining is the core step in KDD. The application of pattern location algorithms is an exploratory technology that suggests highly plausible hypotheses as opposed to a confirmatory technology that evaluates hypotheses. Post-mining interprets and evaluates discovered patterns, possibly performing some validation. Thus, data mining enlarges the possibilities and complements Visual Data Analysis (VDA) and Exploratory Data Analysis (EDA)

while post-mining corresponds to Confirmatory Data Analysis (CDA) in traditional interactive and iterative data analysis [7].

1.1.3 The methodology of data mining

In data-rich environments, conventional data analysis that emphasizes evaluating suggested hypotheses is neither suitable nor applicable [99, 106, 120, 149]. Traditionally, generating hypotheses is based on visual analysis and classic statistical analysis guided by the experts' background knowledge and experience. This guided analysis generates hypotheses near expected patterns. In data-poor environments, the hypothesis space is relatively small and limited. Thus, comprehensive search for possible hypotheses is computationally feasible. As the volume of digital data grows explosively, the hypothesis space becomes huge, and thus the number of possible patterns increases dramatically. Some patterns may lie beyond the experts' imagination. Traditional approaches are unable to suggest totally unexpected patterns since they are not data-driven. Therefore, in data-rich environments, the emphasis must be placed on generating highly plausible hypotheses rather than evaluating them. Data mining is a suitable technology that summarizes massive volumes of data and suggests a reduced number of valuable patterns. Data mining re-examines the process of scientific formulation of knowledge [146]. The generic cycle of hypothesis formulation, experimental design, data collection and hypothesis refutation finds alternative paths, since much of the data is collected without any preconceived hypothesis [146].

1.2 Cluster analysis

With the advent of data mining, researchers have proposed various methods for discovering knowledge from large data sets. These methods strove to combine the areas like machine learning, statistics, databases and information retrieval [33]. Making sense of complex issues is naturally approached by breaking the subject into smaller

segments that can be each explained more simply. Cluster analysis is a central task in data mining processes; it aims at discovering smaller, more homogeneous groups and identifying interesting patterns from a large heterogeneous collection of data objects [14]. More accurately, cluster analysis consists of a series of processes that partition a given data set $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\} \subset \mathbb{R}^D$ into clusters such that the data points in a cluster are more similar to each other than points in different clusters [73]. For example, consider a retail database with records containing items purchased by customers. A clustering procedure could group the customers in such a way that customers with similar buying patterns are in the same cluster. Hence, the main concern in the clustering process is to reveal the organization of patterns into “sensible” groups, which allow us to discover similarities and differences, as well as to derive useful conclusions about them.

1.2.1 Clustering tasks

Here we summarize the main tasks of cluster analysis [109, 169]:

- **Data reduction.** Cluster analysis can contribute to compression of information. In many settings, the amount of available data is very large and its processing becomes very demanding. Clustering can be used to partition a data set into a number of “interesting” clusters. Then, instead of processing the data set entirely, we adopt the representatives of the defined clusters in our process. Consequently, we achieve data compression.
- **Hypothesis generation.** Cluster analysis is used here to infer some hypotheses concerning the data. For instance, we may find in a retail database that there are two significant groups of customers based on their age and the time of purchases. Then, we may infer some hypotheses for the data, that is, “young people go shopping in the evening”, and “old people go shopping in the morning”.
- **Hypothesis testing.** In this case, the cluster analysis is used for the verification

of the validity of a specific hypothesis. For example, we consider the following hypothesis: “Young people go shopping in the evening”. One way to verify its significance is to apply cluster analysis to a representative set of stores. Suppose that each store is represented by its customer’s details (age, job etc.) and the time of transactions. If, after applying cluster analysis, a cluster that corresponds to “young people buy in the evening” is formed, then the hypothesis is supported by cluster analysis.

- **Prediction based on groups.** Cluster analysis is applied to the data set and the resulting clusters are characterized by the features of the patterns belonging to these clusters. Then, unknown patterns can be classified into specified clusters based on their similarity to the clusters’ features. Assume, for example, that the cluster analysis is applied to a data set concerning patients infected by the same disease. The result is a number of clusters of patients, according to their reaction to specific drugs. Then, for new patients, we identify the clusters in which they can be classified. Based on the decision drawn from clustering results, the patients may be prescribed medication.

1.2.2 Clustering applications

Clustering applications have grown exponentially since all sorts of scientific disciplines are based on built or discovered classifications that structure knowledge. Some typical applications of the clustering occur in the following fields [83]:

- In business, clustering may help marketers discover significant groups in their customers’ databases and characterize them based on purchasing patterns.
- In biology, clustering can be used to define taxonomies, categorize genes with similar functionality, and gain an insight into structures inherent in populations.
- In GIS, clustering helps to automate the process of analyzing and understanding

spatial data. It is used to identify and extract interesting characteristics and patterns that may exist in large spatial databases.

- Web mining has been broadly classified into “Web content mining” and “Web usage mining” depending on whether the data used in the knowledge discovery process concerns the *content* of the Web or the *usage* of this content. For Web content mining, clustering is used to discover significant groups of documents on the Web. This classification of Web documents assists in information discovery. For Web usage mining, clustering is used to group the users of large Web sites into communities.

In general terms, clustering may serve as a preprocessing step for other algorithms, such as classification, which would then operate on the detected clusters.

1.3 Motivation

1.3.1 The components of data mining algorithms

To apply the general data mining methodology outlined above for construction of data mining algorithms, one can identify three primary components in any data mining algorithm [64]:

1. **Model representation.** In order to solve a data mining problem, some assumptions are naturally made to select a model to fit into the data. It is important that an algorithm designer fully comprehends which representational assumption is being made by a particular algorithm.
2. **Induction principle.** It is a mathematical formulation of how well a particular pattern meets the goals of the data mining process.

3. **Search method.** This consists of parameter search and model search. Once the model representation (or family of representation) and the induction principle are fixed, then the data mining problem is reduced to purely an optimization task: Find the parameters and models from the selected family that optimize the induction principle. In parameter search, the algorithm searches for the parameters that optimize the evaluation criteria, given observed data and a fixed model. With model search, the model representation is changed so that a family of models is considered.

In cluster analysis, the first two algorithmic components, model representation and induction principle together are referred to as a clustering context [52], which applies in a general manner to clustering algorithms.

1.3.2 Algorithmic engineering of clustering

Many clustering techniques have been suggested in data mining, statistics, artificial intelligence, databases, information retrieval. There is no clustering technique that is universally applicable in uncovering the variety of structures present in data sets, because clustering algorithms often contain implicit assumptions about the shape of clusters or about multiple-cluster configurations. Some of these implicit assumptions are based on similarity measures and grouping criteria. Different clustering methods have been shown to offer different capabilities, different application domains and different computational requirements [93].

Among the most significant requirements of cluster analysis is that it detects significant clusters efficiently and robustly. The technological basis underlying cluster analysis lies in designing and employing suitably fast and robust clustering algorithms. In order to explain the behavior of cluster analysis and enhance its performance, we will focus on design rules for efficient and robust implementations of clustering at a foundational algorithmic level. To this end, a deep understanding of clustering context

(model representation and induction principle) and exploration of search strategies are necessary prerequisites for the success of the cluster analysis.

We will investigate computability and complexity issues that arise naturally in the study of clustering algorithms. In particular, we would like to understand how the performance of a clustering algorithm is dependent on its model presentation, induction principle and search strategy. We would like also to upgrade current studies on cluster validity that appear in the practical literature. We expect that our research will be built on corresponding techniques geared towards more foundational problems for designing and analyzing clustering algorithms. In general, the significant features of a clustering algorithm will have to rely critically on the answers to the following questions:

- What are the assumptions on the inputs?
- What is the model representation adopted?
- What is the induction principle in an optimization formulation?
- What are the limitations and difficulties of this clustering algorithm?
- How is algorithmic structure designed?
- How is its performance measured?
- Does it converge by theoretical proofs or empirical (practical) evaluations?

Clearly, such characteristics will determine the entire discipline of clustering algorithms.

To pursue novel algorithmic design principles to enhance the robustness, efficiency and scalability of clustering algorithms, we are motivated to

- investigate various model representations and compare their characteristics, such as representative modeling, support vector modeling and proximity graph modeling.
- examine different induction principles to formulate optimization criteria, such as squared error and absolute error minimization.
- explore efficient algorithmic structures and employ novel search strategies to ensure lower bounds on the computational requirements of clustering algorithms.
- facilitate clustering algorithms that handle inputs in arbitrary metric spaces and guarantee convergence.
- address the issue of cluster validity.

In addition, we would like to take a new algorithmic engineering approach to the experiments carried out in our research to validate clustering performance. This will serve as a useful support to the assessment of algorithms and their improvement, to verify theoretical developments, to investigate ideas that theoretical techniques fail to analyze precisely, and to give a new insight into possible fundamental solutions. To this end, the algorithms will be implemented in high level language (such as C++, Java) with Object-Oriented design and friendly GUI. This software implementation will allow enough sizes of inputs, record running processes, and output descriptive results. Also, the software solution enables us to experiment easily with different settings, for example, different similarity metrics.

We transfer fast and robust clustering algorithms to real world applications. The Web harbors a large number of contents and communities (groups of visitors sharing a common interest). Our particular interest here is on clustering the communities in large Web sites based on the similarity of interests. With the rapid growth of online information, text categorization has become one of the key techniques in handling and organizing text data. Thus, we would like to conduct case studies on Web usage mining and text mining.

In a word, the motivation of this thesis is to carry out algorithmic research on the issues of efficiency and robustness in cluster analysis, and enhance the exploratory analysis capabilities of clustering. We provide appropriate algorithms that find structures to significantly subscribe the data and handle huge data sets.

1.4 Contribution

The core objective of this thesis is to develop effective and efficient clustering algorithms and increase the transfer of advanced algorithmic methods from research to applications. To this end, we deal with a data-rich environment of feature vectors, sequential patterns, or data objects in arbitrary metric spaces. By applying suitable data modeling and induction principles in cluster analysis, our clustering methods are capable of producing a descriptive and meaningful set of patterns. To describe the significance of clustering results, we discuss cluster validity issues and propose a new approach to evaluate clustering results geometrically by emphasizing the exploratory nature of KDD.

This thesis provides several advances:

1. Mathematical analysis of representative-based modeling to encode cluster information for vector data patterns. We systematically analyze the characteristics of centroid and medoid location estimation, and propose a filtering mechanism to efficiently locate a medoid.
2. Mathematical analysis of representative-based modeling to encode cluster information for data objects in arbitrary metric spaces. As an example, we take into account a sequential data domain, and examine challenges in clustering sequential data. We present a randomized strategy to efficiently find a medoid from a cluster of data objects in an arbitrary metric space.
3. Mathematical explanation of induction principles in a clustering context. We

discuss the optimizing criteria for distance-based clustering. A number of distance metrics are investigated to formulate the optimizing clustering criteria.

4. Mathematical and algorithmic explanation of the relationship between k -MEANS clustering and EXPECTATION MAXIMIZATION. This helps in understanding the algorithmic structure and implementation issue of distance-based clustering and probabilistic clustering.
5. Well engineered design of general purpose clustering algorithms on vector spaces. The design is derived from the k -MEANS structure, but with more robust location estimation in model representation and absolute error estimation in induction principle. Computational time requirements of these algorithms are linear or sub-quadratic on the number and dimension of the data objects. These algorithms are suitable for exploratory data analysis of a multi-dimensional data set.
6. Well engineered design of general purpose clustering algorithms on arbitrary metric spaces. The design inherits the k -MEANS structure, but with more robust location estimation in model representation and absolute error estimation in induction principle. The proposed algorithms generalize fuzzy membership in clustering assignment. Computational time requirements of these algorithms are sub-quadratic on the size of the data objects.
7. Extension of support vector clustering with modeling enhancement through proximity graphs. Essentially, the clusters are dense regions in an otherwise sparse graph. The combination of support vector modeling and proximity graph modeling greatly speeds up the cluster labeling phase in support vector clustering. A family of proximity graphs is compared in modeling data to obtain clusters in complex scenarios that include non-convex clusters, clusters in the presence of noise, clusters of arbitrary shapes, and clusters of heterogeneous densities.
8. Well engineered approach to cluster validity using Support Vector Machines

(SVMs). This new method can obtain an insight into the structure inherent in data through SVM training. More accurately, it is able to verify separation performance and potential outliers by analyzing the complexity of boundaries through support information. It allows evaluation of clustering results of multi-dimensional data with arbitrary shapes.

9. The development and implementation of a k -clustering program. This program implements our representative-based clustering algorithms on vector spaces, and other popular algorithms as well. The program's Object-Oriented design integrates clustering algorithms well, and its friendly GUI provides visualization of clustering results. Thus, it enhances the readability and usability of clusters.
10. The development of a Web usage mining program that implements our representative-based clustering algorithms on metric spaces, and other typical algorithms associated with sequential data mining such as matrix-based and pattern-based clustering.
11. The development and implementation of a program of cluster validity through SVMs and support vector clustering.
12. Two case studies for real-world applications of Web usage mining and text mining.

1.5 Outline of the thesis

The remainder of this thesis is organized as follows.

Chapter 2 presents background material on clustering. This chapter surveys the core concepts of cluster analysis, and reviews related work to the research in this thesis. We will describe clustering process, and explain “feature selection”, “proximity measure”, “clustering criterion”, “clustering context”, and “cluster validity”. In particular, we outline data modeling and induction principles of well-known families of clustering

methods and discuss their advantages and disadvantages. The clustering algorithms reviewed are those common amongst the data mining community.

Chapter 3 analyzes representative-based modeling and induction principles in clustering vector data patterns. We discuss characteristics of various representatives and different distance metrics. This chapter presents the design of fast and robust general purpose clustering algorithms. We explain the algorithmic structure of our clustering methods. The mathematical modeling and empirical illustration show the advantages of our clustering approaches.

Chapter 4 designs representative-based clustering algorithms to deal with data objects in arbitrary spaces. This extension is extremely useful, since in many cases it is not possible to model the data as feature vectors, or at least it is not easy to do so. This chapter uses the sequential data domain as an example, and examines the challenges in clustering data sequences. We discuss issues in modeling sequential data (e.g. similarity measures and the discrete nature). Then, we present the detailed algorithmic procedure of our clustering methods. We illustrate how to locate discrete representatives using randomized strategies and how to generalize fuzzy membership in cluster assignment. Experimental results are also presented to confirm the effectiveness (in terms of clustering quality) and efficiency (time requirements) of our clustering.

Chapter 5 presents another approach to modeling a cluster structure. Instead of using representatives, this approach uses support vectors to encode cluster information. We first review the fundamental knowledge of support vector machines. Then, we describe the optimization problems raised from support vector clustering. Next, we investigate the critical phase in support vector clustering — clustering labeling. We propose an efficient clustering labeling algorithm by combining support vector modeling and proximity graph modeling. Finally, we examine the flexibility of our clustering labeling strategy to a common family of proximity graphs.

Chapter 6 discusses cluster validity issues. First, we briefly review the common cluster validity tasks and methods. Then, we present a new approach to evaluating

clustering results, by which Support Vector Machines are used to detect the separation between clusters and filter outliers. This new approach can be used to compare clustering results from different algorithms or different runs of the same algorithm. This chapter details the cluster validity process using support vector machines. A set of experiments illustrates the steps of our approach.

Chapter 7 presents two case studies of applying our fast and robust representative-based clustering methods to Web usage mining and text mining. First, we model the optimization problems raised from Web visitors clustering and text classification. Then, we illustrate the experimental results of various clustering algorithms with different settings, such as similarity metrics.

Chapter 8 concludes this thesis with a summary, discussions on implementation issues and new research directions indicated by the advances of this thesis.

Chapter 2

Basic Concepts and Related Work

In this chapter, we review the theory and core concepts on which cluster analysis techniques are based. We survey major clustering methods proposed to efficiently handle large data sets, and outline the working principles of these clustering methods.

2.1 The clustering process

In the clustering process, there are no predefined classes and no examples that would show what kind of desirable relations should be valid among the data. That is why clustering is perceived as an unsupervised process [14]. In contrast, classification is a procedure of assigning a data object to a predefined set of categories [67]. Clustering produces initial categories in which objects in a data set are assigned during the classification process. Before we undertake a clustering task, there is a need of preprocessing to extract features from the data set. The clustering process may result in different partitions of a data set, depending on the specific criteria used for clustering. Post-clustering evaluates and interprets the significance of the detected cluster structure. Figure 2.1 presents the clustering process.

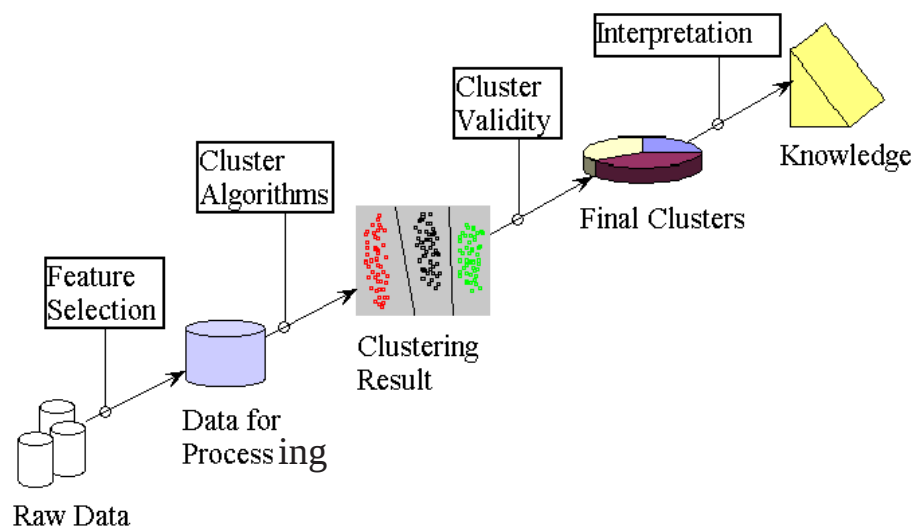


Figure 2.1: The process of cluster analysis.

We now briefly describe the basic steps to develop cluster analysis [67, 109].

- **Feature selection.** The goal is to select properly the features on which clustering is to be performed so as to encode as much information as possible concerning the task of our interest. Thus, preprocessing of data may be necessary prior to their utilization in clustering tasks.
- **Clustering algorithms.** This step refers to the choice of an algorithm that results in the definition of a good clustering scheme for a data set. Proximity measures and clustering criteria mainly characterize a clustering algorithm as well as its efficiency to define a clustering scheme that fits the data set.
- **Validation of the results.** Clustering algorithms define clusters that are not known a priori. Therefore, in many applications [139], the final partition of data requires some kind of evaluation. Analysts verify the significance of clustering results by using appropriate criteria and techniques.
- **Interpretation of the results.** In many cases, the experts in the application

area have to integrate the clustering results with other experimental evidences and analyses in order to draw suitable conclusions.

This clustering process reflects the exploratory (rather than confirmatory) nature of data mining. In many scenarios, data is collected without a preconceived hypothesis. However, the belief exists that this data captures or reflects input information about phenomena. Thus, the task for data mining is to explore the data and formulate hypotheses.

2.2 Feature selection

The first step in clustering is feature extraction and feature selection for pattern representation [93]. It is often valuable to isolate only the most descriptive and discriminatory features in the input set, and utilize those features exclusively in subsequent analyses. Feature extraction techniques compute new features from the original data set, while feature selection techniques identify a subset of the existing features for subsequent uses. In either case, the goal is to improve classification performance and computational efficiency.

There are no theoretical guidelines that suggest the appropriate features to use to represent patterns in a specific situation. Indeed, the pattern generation process is often not directly controllable; the user's role in the pattern representation process is to gather facts and conjectures about the data, optionally perform feature extraction and selection, and design the subsequent components of the clustering system. Because of the difficulties surrounding pattern representation, it is conveniently assumed that the pattern representation is available prior to clustering. Nonetheless, a careful investigation of the available features and any available transformation can yield significantly improved clustering results. A good pattern representation can often yield a relatively simple and easily understood clustering, whereas a poor pattern representation may yield a complex clustering whose true structure is difficult

or impossible to discern.

A pattern can measure either a physical object (e.g., a chair) or an abstract notion (e.g., a style of writing). Patterns are conventionally represented as multidimensional vectors, where each dimension represents a single feature [46]. These features can be either quantitative or qualitative:

- **Quantitative features.** They can be measured on a ratio scale (with a meaningful reference value), or nominal or ordinal scales. Some examples are:
 - continuous values (e.g. weight).
 - discrete values (e.g. the frequency of keywords).
 - interval values (e.g. the duration of an event).
- **Qualitative features.** They can be measured on a nominal scale (e.g. color), or on an ordinal scale (e.g. sound intensity “quiet” or “loud”).

In Chapter 7, we discuss a case study of text mining, where we extract two data sets from the original documents, and represent data patterns as feature vectors.

One can also use structured features [119], which can be represented as sequences, sets, or graphs. There exist many important databases that store categorical data sequences (sessions), where significant knowledge is hidden behind sequential dependencies between the data in sessions (credit card usage history, operating system logs, database redo logs, Web access logs, etc.) [2, 165]. These data patterns can be naturally modeled by sequences. Examples can be seen in our case study of Web usage mining in Chapter 7, where users’ navigation paths are extracted from Web access logs and represented as sequences. Suppose a user has visited Web pages 1, 2, 5, 15, and then 7, his navigation history can be modeled using a sequence $\langle \{1\}\{2\}\{5\}\{15\}\{7\} \rangle$. However, by ignoring the link information in the user sessions, we can model patterns as sets [61, 135].

A more generalized representation of a pattern is called a symbolic object [49]. Symbolic objects are defined by a logical conjunction of events. These events link values and features in which the features can take one or more values and not all the objects need to be defined on the same set of features.

Feature selection is a well-explored topic in statistical pattern recognition [46]. However, in a clustering context that lacks class labels for patterns, the feature selection process is of necessity ad hoc. Feature selection might involve a trial-and-error process where various subsets of features are selected, the resulting patterns clustered, and the output evaluated using a validity index. Some of the popular feature extraction processes (e.g., principal components analysis) do not depend on labeled data and can be used directly.

2.3 Similarity measures

Similarity measures quantify how “similar” two patterns are. In most cases we have to ensure that all selected features contribute equally to a similarity measure and there are no features that dominate others. Similarity is fundamental to the definition of a cluster; a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering procedures. It is most common to calculate the dissimilarity between two patterns using a distance measure defined on the feature space. Because of the variety of feature types and scales, the distance measures must be chosen carefully.

2.3.1 Euclidean distance and Minkowski metrics

For patterns whose features are all continuous, the most popular metric is the Euclidean distance

$$d_2(\vec{x}_i, \vec{x}_j) = \left(\sum_{k=1}^D (x_{i,k} - x_{j,k})^2 \right)^{1/2} = \|\vec{x}_i - \vec{x}_j\|_2, \quad (2.3.1)$$

which is a special case ($a = 2$) of the Minkowski metrics

$$d_a(\vec{x}_i, \vec{x}_j) = \left(\sum_{k=1}^D |x_{i,k} - x_{j,k}|^a \right)^{1/a} = \|\vec{x}_i - \vec{x}_j\|_a. \quad (2.3.2)$$

The Euclidean distance has an intuitive appeal as it is commonly used to evaluate the proximity of objects in two or three-dimensional space. It works well when a data set has “compact” or “isolated” clusters [112]. One drawback in directly using the Minkowski metrics is the tendency of the largest-scaled feature to dominate the others. Solutions to this problem include normalization of the continuous features (to a common range or variance) or other weighting schemes. Linear correlation among features can also distort distance measures; this distortion can be alleviated by using the squared Mahalanobis distance

$$d_w(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j) \Sigma^{-1} (\vec{x}_i - \vec{x}_j)^T, \quad (2.3.3)$$

where the patterns $\vec{x}_i$ and $\vec{x}_j$ are assumed to be row vectors, and Σ is the sample covariance matrix of the patterns or the known covariance matrix of the pattern generation process.

2.3.2 Cosine dissimilarity measure

The similarity between patterns can also be measured by the cosine of patterns. The cosine of the angle between two vectors is identical to their correlation coefficient,

that is

$$\text{con}(\vec{x}_i, \vec{x}_j) = \frac{(\vec{x}_i \cdot \vec{x}_j)}{(\vec{x}_i \cdot \vec{x}_i)(\vec{x}_j \cdot \vec{x}_j)}. \quad (2.3.4)$$

2.3.3 Arbitrary distances

For arbitrary patterns where the features are not mapped to vectors but are structured mathematical objects like sequences, sets or graphs, we still require a measure of similarity. In the context of text categorization [136] and Web mining [183], the cosine-like dissimilarities are often calculated according to the occurrence, frequency or order of features. We present more details on this notion with specific domain information in Chapter 4.

Other “pseudo-distances” are also used to measure dissimilarity between patterns. For example, the Kullback-Leibler (KL) dissimilarity measure [136]

$$KL(T_1, T_2) = \sum_{F_j} f_2(F_j) \log\left(\frac{f_2(F_j)}{f_1(F_j)}\right), \quad (2.3.5)$$

and the x^2 dissimilarity

$$x^2(T_1, T_2) = \sum_{F_j} \frac{(f_1(F_j) - f_2(F_j))^2}{f_2(F_j)}, \quad (2.3.6)$$

where $f_i(F_j)$ is the frequency of the feature F_j in the pattern T_i .

2.3.4 Proximity matrix

In many settings, the similarity between patterns is not a function or a logical mathematical expression, but the clustering problem comes with a matrix of proximity values. This matrix provides in entry i, j the separation (distance or dissimilarity) between patterns i and j . In such a situation, all the pairwise distance values for the patterns are pre-computed and stored in a symmetric matrix [183]. This has implications for the evaluation of computational requirements of clustering algorithms.

When the data consists of n feature vectors for the n patterns and a closed formula to compute distance, the input size is $n \times D$ where D is the number of features. However, the explicit representation of all the pairwise distance may be the input size quadratic in n . Thus, quadratic algorithms in n are linear in the size of the input in this case.

2.3.5 Differences in scales and domains of features

Computation of distances between patterns with some or all features being non-continuous is problematic, since the different types of features are not comparable and (as an extreme example) the notion of proximity is effectively binary-valued for nominal-scaled features. Nonetheless, practitioners (especially those in machine learning, where mixed-type patterns are common) have developed proximity measures for heterogeneous type patterns. Cherkassky et al. [180] proposes a combination of a modified Minkowski metric for continuous features and a distance based on counts (population) for nominal attributes. Esposito et al. [50] have reported a variety of other metrics for computing the similarities between patterns represented using quantitative as well as qualitative features. Several algorithms proposed in this thesis work with an arbitrary distance, which could be the combination of metrics on continuous and nominal attributes.

2.4 Cluster information modeling and clustering criteria

In practice, a clustering problem is usually converted into a corresponding optimization problem associated with a clustering context consisting of the model representation and induction principle. Models are the structures one is willing to use to represent clusters, and induction principles are mathematical formalizations of what

researchers believe is a definition of cluster [52]. Clustering is a subjective process; it is in the eye of the beholder. Given a data set, any clustering is a hypothesis to suggest groups in the data. This hypothesis is a model to encode cluster information, and can potentially constitute a mechanism to classify instances of data. In order to discriminate one hypothesis over another, the corresponding clustering criterion is to be tested. A clustering criterion is the mathematical formulation of an induction principle. It can be expressed via a cost function or some other type of rule. “Good” models and clustering criteria lead to partitions that fit the data set well. Thus, researchers have proposed many models and induction principles.

2.4.1 Probabilistic model

Parametric statistics assumes that noise has been generated by an unknown number of qualitatively similar, stochastic processes. Each individual process is characterized by a probability density, which is regarded as a cluster of the data set. The density of the whole data set is described by a parameterized mixture model [25]. The common examples are the Gaussian mixtures. In this case, the data X consists of a fixed number k of clusters, each of which is distributed according to a multivariate normal distribution $N_{\vec{\mu}_i, \Sigma_i}$ and which are combined by a vector $\vec{\pi}^T = (\pi_1, \dots, \pi_k)$ ($\sum_{i=1}^k \pi_i = 1, \pi_i \geq 0$). Then, the probability distribution is given by $p(\vec{x}) = \sum_{i=1}^k \pi_i N_{\vec{\mu}_i, \Sigma_i}(\vec{x})$.

A probabilistic model encodes cluster information of the data in terms of the mixture parameters $\vec{\pi}$, $\vec{\mu}_i$ and Σ_i . But how does one estimate the set of parameters that best explains the data? A traditional approach is Maximum Likelihood (ML). Its induction principle says “choose the model that maximizes the probability of the data being generated by such model” [52]. This induction principle allows an explicit clustering criterion via the following optimization problem:

$$\text{Maximize } ML(\vec{\pi}, (\vec{\mu}_i, \Sigma_i)_{i=1, \dots, k}) = p(X | \vec{\pi}, (\vec{\mu}_i, \Sigma_i)_{i=1, \dots, k}). \quad (2.4.1)$$

Now the question is how to search for the solution to this multivariate optimization

problem. By equating the gradient of $\log ML$ to zero, sufficient conditions for the optima are obtained that lead to iterative algorithms. An efficient bootstrap solution to this problem is provided by the EXPECTATION MAXIMIZATION (EM) algorithm [41].

2.4.2 Representative-based model

The representative-based model employs another popularized modeling approach, Vector Quantization (VQ), to represent cluster information. VQ is a data compression method where a set of data patterns is encoded by a reduced set of reference vectors [25, 26], which are referred to as representatives. Of particular interest is the problem to partition the data X into k clusters with respect to a set of selected representatives and a given distance metric. A representative is typically chosen as the most central prototype of a cluster; it is also called the center of a cluster. Given a set of k centers $C = \{\vec{c}_1, \dots, \vec{c}_k\}$ and a distance metric $d(\vec{x}_i, \vec{x}_j)$, the representative-based model can be described as a vector quantizer

$$M(C) = \sum_{i=1}^n d(\vec{x}_i, \text{REP}[\vec{x}_i, C]), \quad (2.4.2)$$

where $\text{REP}[\vec{x}_i, C]$ is the closest center in C to $\vec{x}_i$ ($i = 1, \dots, n$). Furthermore, the model can be prototyped with an optional constraint of $C \subset X$. With this constraint, the selection of the representatives will be a set of discrete patterns in the given data set; otherwise, the representatives can be chosen from a continuous embedded space. Figure 2.2 demonstrates continuous prototypes (centroids) and discrete prototypes (medoids), which are chosen as representatives for the given data patterns. The distance metric can be Euclidean metric or non-Euclidean metric. For example, k -MEANS uses squared Euclidean distance.

Now it is straightforward to formulate the following clustering criterion (cost function) to detect the set of centers that best fit a cluster structure

$$\text{Minimize } M(C) = \sum_{i=1}^n d(\vec{x}_i, \text{REP}[\vec{x}_i, C]). \quad (2.4.3)$$

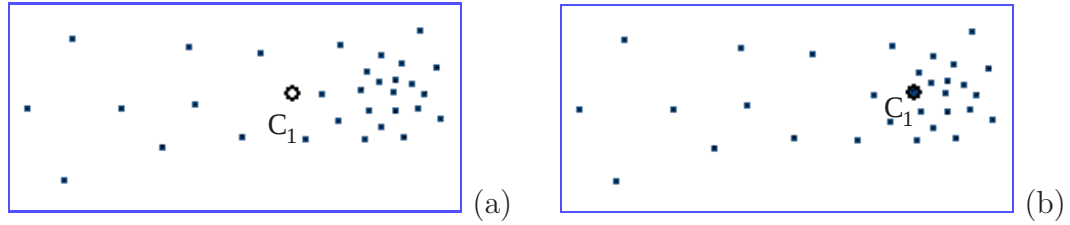


Figure 2.2: Data modeling and structuring with representatives. (a) A continuous prototype (centroid) is chosen as the representative of the given data patterns. (b) A discrete prototype (medoid) is chosen as the representative of the given data patterns.

This corresponds to the induction principle that says “pick the model (set of k representatives) that minimizes the sum of the distances of data patterns to their cluster representatives”. The resulting assignment problem is known as a deterministic NP-hard combinatorial optimization problem. However, the practical representative-based clustering algorithms typically produce approximate solutions to this optimization problem by iteratively refining the partition encoded by the representatives. We discuss algorithmic issues of representative-based clustering in Chapter 3.

The VQ approach can be related to the parametric statistics approach in two ways. First, a stochastic optimization strategy (like simulated annealing) is employed in VQ to search for solutions with low costs; second, the data set constitutes a randomly drawn instance of the combinatorial optimization problem of VQ. We present more details on this notion in Chapter 3.

A vector quantizer can also be modeled to encode each cluster by a certain number of representatives. In CURE [73], these representatives are generated by selecting well-scattered points and then shrinking them toward the cluster centroid by a specified fraction. It is also possible to model cluster information without using any representatives. For example, statistical principles suggest minimization of the total within-group distance (TWGD) as a robust criterion for spatial clustering [55].

2.4.3 Model based on support vectors

While formulations using representatives are widely used, description of data using boundary information has also attracted attentions. Support Vector Machines (SVMs) [173] make most use of boundary information to model a given data set in terms of a set of Support Vectors (SVs). More importantly, SVMs implement kernel methods to transform data into a feature space. SVMs have been widely adopted as supervised learning methods in classification with labeled data. They are also closely associated with an unsupervised learning task of Data Domain Description (DDD) [167], where the task is not to distinguish between classes of objects but to estimate the novel (abnormal) instances (outliers). There are many real-world problems that are better fit by a novelty detection paradigm than a supervised learning paradigm.

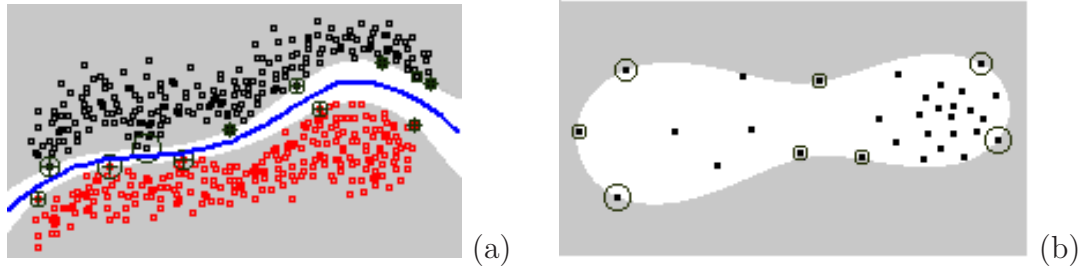


Figure 2.3: Data modeling and structuring with SVs. (a) SV modeling for classification task. The curve between two classes is the pre-image of a hyperplane that separates the two classes. Encircled points are SVs. (b) SV modeling for novelty detection task. A support region (white region) is estimated to cover most of the data patterns. Encircled points are SVs.

Figure 2.3 demonstrates that SVs are used to model classification and novelty detection tasks. For the case of classification, SVMs describe the data by constructing a set of hyperplanes in a feature space in terms of SVs such that each hyperplane separates a pair of classes, or one class from all others. The hyperplanes that best fit the data can be detected by maximizing the margins between each pair of classes, or one class against all others. In the case of novelty detection, SVMs typically describe the data by constructing a hypersphere in a feature space in terms of SVs such

that the hypersphere contains the data patterns of a given data set. The decision boundary that best describes the data can be obtained by minimizing the volume of the hypersphere. The objective functions derived from the above induction principles are quadratic optimization problems. In practice, researchers have proposed a variety of efficient algorithmic approaches to solve SVM problems. We address the issues of SV-based modeling further in Chapter 5 and Chapter 6.

2.4.4 Model based on proximity graphs

While cluster information can be encoded with continuous mathematical models, it can be also described using discrete structural models. One can consider the problem of clustering, the capacity to answer, for each pair of patterns $\vec{x}_i$ and $\vec{x}_j$, the question “are $\vec{x}_i$ and $\vec{x}_j$ in the same clusters?”. Answering this question for all pairs results in a graph whose nodes are the patterns and in which an edge exists between $\vec{x}_i$ and $\vec{x}_j$ if and only if their nodes are in the same cluster. The clusters are then the connected components of this graph. The intuition in clustering says that these connected edges are short as evaluated by the similarity measure in use. Thus, clustering can be regarded as extracting the connected components defined by short edges from the complete weighted graph that associates the weight $d(\vec{x}_i, \vec{x}_j)$ to the edge $\vec{x}_i, \vec{x}_j$.

Proximity graphs aim at representing the entire information available in the complete proximity matrix by some sparse graphs. Proximity graph modeling reduces redundancy and enhances integrity of data, so it is not excessive and remains informative. In fact, clustering is the ultimate process of summarizing the information encoded by proximity graphs.

Since proximity graphs effectively encode both proximity and topological relationships between data patterns, they have been widely used for modeling data, in particular for spatial data patterns. A well-known proximity graph is the Voronoi diagram, in which two spatial patterns are neighbors if and only if their territories share a common Voronoi boundary. The explicit representation of near-by neighborhood

relationship is the Delaunay diagram, which is the dual graph of the Voronoi diagram. It can be obtained by connecting two neighboring patterns in the Voronoi diagram (but avoiding possible cross edges) [57]. In 2D, the Delaunay diagram approximates the complete graph with linear size. Moreover, it includes many other proximity graphs (like the Gabriel Graph, the Relative Neighbor Graph and the Minimum Spanning Tree). Figure 2.4 demonstrates data modeling by the Voronoi diagram and the Delaunay diagram.

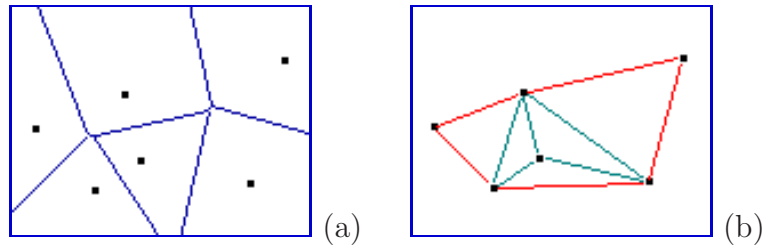


Figure 2.4: Data modeling with the Voronoi and Delaunay diagrams. (a) The Voronoi diagram. (b) The Delaunay diagram.

In addition, argument-dependent proximity graphs [59] like k -Nearest Neighbors are also widely adopted to model data patterns. For arbitrary patterns, a hypergraph model can be employed to map the relationship present in the data into a hypergraph [82]. A hyperedge represents the affinity between data patterns (or among subsets of data) and the weight of the hyperedge reflects the strength of the affinity.

Proximity graph clustering detects a partition of vertices such that the corresponding data patterns in each group are highly related to each other and the weight of the edges cut by the partitioning criterion is minimized. Researchers have proposed different clustering criteria following different working principles, such as the short-long criterion [59] and novelty detection rules [11]. In Chapter 5, we discuss modeling enhancement through proximity graphs for SVC.

2.4.5 Other models

Many implicit mathematical models are hidden under clustering algorithms. For instance, BIRCH [188] encodes cluster information using compact summary statistics. More accurately, it maintains a tree of nodes with estimators of location and scatter. DENCLUE [85], a condensation-based approach, is based on kernel models.

The clustering context (model and induction principle) is an essential algorithmic component of a clustering method. It impacts heavily on the quality of clustering results. The formulation of a clustering context is a critical issue to design clustering algorithms. For example, crisp membership of items to a cluster versus non-crisp membership is simply an issue of modeling. So is the case of whole data modeling versus random sampling. Researchers have created a direction that considers the complexity of the model in the formulation of the induction principle. Minimum encoding inductions MDL [140] and MML [128] are pioneer approaches in this regard.

2.5 Clustering algorithms

A multitude of clustering methods are proposed in the literature. In a general manner to clustering algorithms, the model and induction principle constitute the context of a clustering algorithm. Thus, clustering algorithms can be naturally classified according to cluster encoding prototypes (e.g., representative-based, support vector based, proximity graph based), or to the clustering criteria (e.g., minimum squared error or absolute loss estimation, maximum likelihood, maximum separate margin). Moreover, how to formulate a clustering context is heavily associated with algorithmic design and determines algorithmic performance. Once a clustering context is fixed, the clustering problem becomes to an optimization task of searching for models that optimize the induction principle. From algorithmic engineering perspective, such an optimization task is often under some limitations. For example, one limitation may give rise to NP-completeness, which can partially be circumvented by approximation

algorithms with heuristics. Therefore, thoroughly understanding a clustering context (algorithmic components) is a key step to successful clustering. In what follows next, we survey major clustering methods. We examine their problem formulations and algorithmic characteristics.

2.5.1 Partitional vs. hierarchical clustering

Partitional clustering

For a given parameter k , partitional clustering aims at segmenting a data set into k clusters. Thus, it is also called k -groups clustering or k -clustering [63]. Typically, a partitional clustering algorithm employs representative-based models and distance-based induction principles, and it obtains a single partition of the data (a set of representatives of the discovered clusters) instead of a hierarchical clustering structure (like the dendrogram produced by a hierarchical technique).

For a data set $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$, the partitional clustering problem can be formulated as the cost function in Equation (2.4.3). This corresponds to representative-based modeling. With regard to the clustering criteria, the algorithmic objective is to minimize the distance of the objects within a cluster from the representative of this cluster. Note that combinatorial search of the set of possible labelings for an optimum value of a criterion is computationally prohibitive (NP-Hard) [70] or even intractable. In practice, therefore, the need to use approximation algorithms (hill-climbers) arises. Typically, the algorithm starts with an initial partition with k clusters, and then iteratively optimizes the cost function by reassigning some objects to their new representatives of clusters, until some criteria are satisfied. In this sense, partitional clustering finds a local optimum as opposed to a global optimum. Note that, the partitional clustering problem can also be formulated as the optimization function in Equation (2.4.1), which corresponds to probabilistic modeling. This approach detects the best-fitting model by estimating the likelihood function.

k -MEANS [110], PAM [97] and k -modes [91] are well-known representative-based algorithms. k -MEANS models cluster information using centroids and formulates optimization function using squared Euclidean distance, while PAM encodes cluster information with k medoids and its cost function minimizes the sum of the dissimilarities of the objects to their closest representatives. k -modes extends the k -means approach to categorical data sets. It replaces the means of clusters with the modes and uses different dissimilarity measures to update the modes. It computes the total number of mismatches in the corresponding attribute categories. The effectiveness and efficiency of partitional algorithms vary with clustering context. For example, the discrete feature of medoids makes them more informative than means. Medoids are robust estimators of locations in the statistical sense, while means are not. A famous probability-based clustering method is EXPECTATION MAXIMIZATION, which typically uses the Gaussian probability distributions. In Chapter 3, we examine the intrinsic relationship between EXPECTATION MAXIMIZATION and k -MEANS clustering approaches.

The partitional clustering can detect k clusters by optimizing a criterion function defined either on a subset of the data (sampling), or over all the data [93, 109]. It can define hard boundaries between clusters, or allow for fuzzy membership with a probability function. Partitional methods have advantages in applications involving large data sets for which the construction of a dendrogram is computationally prohibitive. A problem accompanying the use of a partitional algorithm is the choice of the number of desired output clusters. A seminal paper [44, 128] provides guidance on this key design decision. Another problem of partitional algorithms is that they are unable to handle noise and outliers and they are not suitable for discovering clusters with non-convex shapes. More algorithmic issues of k -clustering are further addressed in Chapter 3 and Chapter 4.

Hierarchical clustering

In hierarchical clustering, the nested decomposition is represented by a tree diagram (dendrogram), which shows how the clusters are related. By cutting the dendrogram at a desired level, a clustering of the data items into disjoint groups is obtained. This is intuitively a reasonable procedure. Hierarchical clustering algorithms, according to the methods that produce clusters, can further be divided into divisive and agglomerative [169].

Divisive algorithms are based on a top-down approach. This approach produces a sequence of clustering schemes with an increasing number of clusters at each step. It begins with X in a single cluster representing the root in a dendrogram, and produces at each step results from the previous one by splitting a cluster into two subclusters until a termination condition is met.

Agglomerative algorithms are based on a bottom-up approach. This approach produces a sequence of clustering schemes with a decreasing number of clusters at each step. To start with, each object in X constitutes a distinct cluster. Contrary to the top-down approach, it proceeds successively by merging smaller clusters into larger ones based on some similarity measures, until a stopping criterion is satisfied.

Five widely adopted similarity measures between two clusters are as follows, where C_i , C_j and C_k are clusters of X ; C_k is the union of C_i and C_j ; $\vec{x}$ and $\vec{x}'$ are objects in X ; n_i is the number of objects in C_i ; and $\vec{m}_i$ is the mean of C_i .

- Single-linkage: $S_{sing}(C_i, C_j) = \min_{\vec{x} \in C_i, \vec{x}' \in C_j} \{d(\vec{x}, \vec{x}')\}$.
- Complete-linkage: $S_{comp}(C_i, C_j) = \max_{\vec{x} \in C_i, \vec{x}' \in C_j} \{d(\vec{x}, \vec{x}')\}$.
- Average-linkage: $S_{aver}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\vec{x} \in C_i} \sum_{\vec{x}' \in C_j} \{d(\vec{x}, \vec{x}')\}$.
- Mean-linkage: $S_{mean}(C_i, C_j) = d(\vec{m}_i, \vec{m}_j)$.
- Ward's linkage: $S_{Ward}(C_i, C_j) = \sum_{\vec{x} \in C_k = C_i \cup C_j} d(\vec{x}, \vec{m}_k)^2 - (\sum_{\vec{x} \in C_k} d(\vec{x}, \vec{m}_k))^2$.

The most popular dissimilarity measure, $d(\vec{x}, \vec{x}')$, is the Euclidean distance. The merge in the single-linkage and complete-linkage approaches is solely based on the extreme pairs. The single-linkage approach [157] merges clusters that contain the closest pair of data objects. Thus, it is robust to non-globular clusters. However, the resulting clusters tend to be long and elongated. The complete-linkage approach merges clusters that have the closest farthest pair of objects. The average-linkage approach attempts to incorporate more information to merge. This approach computes the average of dissimilarities between clusters. Here, every point within a cluster is equally important. Thus, this approach is extremely sensitive to noise points and outliers. The mean-linkage approach is a hybrid of the extreme value approach and the average-linkage approach. Since the means are the representatives of clusters, resulting clusters tend to be globular, large clusters tend to be broken into meaningless groups, and parts belonging to close neighboring clusters tend to be merged. The Ward's method optimizes the within-groups sum of squares. This method merges the two clusters resulting in the smallest increase in the within-groups sum of squares. Thus, this approach tends to partition X into globular clusters with similar sizes.

One advantage of hierarchical clustering is that it does not necessitate the number of clusters as an input argument. It allows the user to choose the best decomposition of X from the dendrogram. However, where to stop partitioning or merging must be specified. This termination condition is not easy to determine. Another drawback of this approach is that it is rigid in the merge or split process [83]. This rigidity may cause erroneous clustering when the merge or split process is not optimal. The computational inefficiency of this clustering is also a drawback. Algorithms of this type may easily require $O(n^2)$ [124] time, since they typically need $O(n)$ merging or splitting operations where the decision on what to merge or split is proportional to the current number of parts. In addition, dendrograms become very large with large data sets and their expert interpretation becomes extremely complex and challenging. One algorithmic engineering direction for overcoming the drawbacks in hierarchical clustering is to integrate hierarchical methods with other clustering techniques. A few such methods are summarized as follows:

- **BIRCH** [188]. It uses a hierarchical data structure called Cluster Feature Tree (CF-Tree) for partitioning the incoming data objects in an incremental and dynamic way. CF-Tree is a height-balanced tree that stores the Clustering Features (CFs), and it is based on two parameters: branching factor and threshold. A CF is a triplet summarizing the information about a cluster. These two parameters influence the size of a CF-Tree and thus the effectiveness of clustering. BIRCH uses compact summary statistics instead of large data sets themselves to improve efficiency. It can typically find a good clustering with a single scan of the data and improve the quality further with a few additional scans. It can handle noise effectively. However, it is order-sensitive. Moreover, it does not always correspond to a natural cluster, since each node in the CF-Tree can hold a limited number of entries due to its size.
- **CURE** [73]. This bottom-up hierarchical clustering method models each cluster using multiple representatives. These representatives are well-scattered objects randomly generated for each cluster to approximate the physical shape and geometry of the cluster. To reduce the influence of outliers, CURE introduces the shrinking operation on the representatives through a shrinking factor α (from 0 to 1). The shrinking factor α controls the compactness of clusters. The use of smaller α favors clusters with arbitrary shapes, while the use of larger α favors compact clusters. After shrinking, CURE merges two clusters with the closest pair of representatives belonging to different clusters. Despite its flexibility and effectiveness, CURE requires $O(n^2 \log n)$ time for higher dimensional data and $O(n^2)$ for lower dimensions, where n is the number of objects. To handle massive data sets, it uses a combination of random sampling and partition clustering.
- **ROCK** [74]. This algorithm is a robust clustering algorithm for Boolean and categorical data. It introduces two new concepts: an object's neighbors and its links. Links between two objects represent the number of common neighbors between the two objects. The similarity between a pair of clusters is based on the number of objects from different clusters who have neighbors in common.

ROCK first constructs a sparse graph from a given data similarity matrix using a similarity threshold and the concept of shared neighbors. It then performs a hierarchical clustering algorithm on the sparse graph.

2.5.2 Based on all candidates vs. sampling-based

Modeling cluster information by considering each object in the data set is a general case of a clustering algorithm. However, a significant number of clustering methods are not able to scale well for large data sets due to their efficiency. For example, a typical k -medoids partitioning algorithm like PAM works effectively only for small data sets. To deal with large data sets, sampling-based methods are often a better choice. CLARA [97] and CLARANS [126] are clustering methods that employ sampling strategies to model cluster structures.

- **CLARA.** This method is based on the PAM clustering approach. However, it considers samples of the data set on which clustering is applied, and as a consequence it may deal with larger data sets. More specifically, CLARA draws multiple samples of the data set and it applies PAM on each sample. Then it gives the best clustering as the output. The problem of this approach is that its efficiency depends on the sample size. Also, the clustering results are produced based only on samples of a data set. Clearly, if a sample is biased, a good clustering based on samples will not necessarily represent a good clustering of the whole data set.
- **CLARANS.** This is a mixture of PAM and CLARA. CLARANS searches a subset of the data set in order to define clusters. The subsets are drawn with some randomness in each step of the search, in contrast to CLARA that has a fixed sample at every stage. This has the benefit of not confining a search to a localized area. In general terms, CLARANS is more efficient and scalable than both CLARA and PAM.

2.5.3 Hard vs. fuzzy clustering

Hard clustering (crisp clustering) assigns each point in data set X to a single cluster, while in fuzzy clustering (non-crisp clustering) [87], a data point may belong to multiple clusters, having a degree of membership value in each cluster. Typically, a degree of membership is a real value from zero to one. Therefore, clusters are not mutually exclusive, but overlap in fuzzy clustering. Fuzzy clustering is especially useful when there are no clear boundaries between clusters. Figure 2.5 demonstrates the working principles of these two clustering approaches. Two hard clusters H_1 ($\vec{x}_1, \vec{x}_2$ and $\vec{x}_3$) and H_2 ($\vec{x}_4, \vec{x}_5, \vec{x}_6$ and $\vec{x}_7$) are presented by rectangles. On the other hand, two fuzzy clusters F_1 and F_2 are denoted by ellipses with overlap.

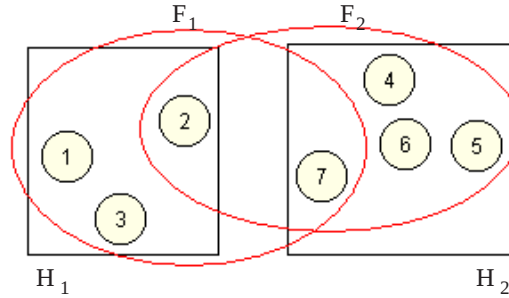


Figure 2.5: Hard clustering and fuzzy clustering.

It is straightforward to convert fuzzy clustering to hard clustering by allocating each object to the cluster exhibiting the greatest degree of membership. Fuzzy c -means and k -harmonic-means are well-known fuzzy clustering methods.

- **Fuzzy c -means** [16]. Consider a set of n objects $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ to be clustered into c groups of data. Fuzzy clustering of the objects can be represented by a fuzzy membership matrix called a fuzzy partition. The set of all $c \times n$ non-degenerate constrained fuzzy partition matrices is denoted by A_f . The clustering criterion used to define good clusters for fuzzy c -means

partitioning is the following function:

$$M_w(A, V) = \sum_{i=1}^c \sum_{j=1}^n (A_{ij})^w d_{ij}^2(\vec{x}_j, \vec{v}_i), \quad (2.5.1)$$

where $A \in A_f$ is a fuzzy partition matrix, $w \in [1, \infty)$ is the weighting exponent on each fuzzy membership, $V = \{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_c\}$ is a matrix of prototype parameters (a set of c cluster centers), and $d_{ij}(\vec{x}_j, \vec{v}_i)$ is a measure of the distance from $\vec{x}_j$ to the i th cluster prototype. The membership matrix plays an important role in fuzzy clustering. Thus, to define the membership matrix is a critical issue.

- **k -harmonic-means** [187]. This is also a variant of k -MEANS clustering. It attempts to overcome the sensitivity of k -MEANS to the initialization of centers. It computes the harmonic average of each cluster (that is the reciprocal of the arithmetic average of the reciprocals of the numbers in a cluster) and uses it as the optimization criterion. Logically, the same algorithmic procedure of k -MEANS clustering can be used in such a generalization. Both k -MEANS and k -harmonic-means converge to local optima.

2.5.4 Graph-based vs. density-based

Graph-based clustering

Graph-based algorithms are built on structural models. Graph-based clustering starts by constructing a proximity graph to encode the relationship between data objects. Once the proximity graph is constructed, it can be used to instruct clustering operations. Naturally, the objective is to learn a cut-off criterion, and remove uninteresting edges against the cut-off criterion by cutting nodes or edges. The connected components remaining are the most elementary clusters.

Graph-based clustering has some promising characteristics that make it effective and efficient. On one hand, it is able to detect non-convex clusters and is robust to noise.

On the other hand, it is typically linear in the number of edges in an underlying proximity graph. Thus, if the number of edges is not excessive but sufficient, the graph-based clustering is computationally efficient. However, graph-based clustering requires extra time to construct an underlying graph. Since graph-based clustering focuses only on short edges, sparse clusters linked by relatively homogeneous edges may remain undetected. The following clustering methods are of this category.

- **Zahn’s clustering** [185]. Zahn proposes a graph-theoretic clustering method that employs the Minimum Spanning Tree (MST) to capture similarity in perceptual groupings of a data set X . To detect clusters, this approach removes inconsistent edges from the resulting graph $MST(X)$. If the length of an edge $e_{i,j} \in MST(X)$ is significantly larger than the average of the lengths of nearby edges incident to both nodes of $e_{i,j}$, the edge becomes an insistent edge. That is, the criterion for determining the consistence of $e_{i,j}$ depends on relatively proximity and varies over X . Thus, this approach may be able to detect clusters of different densities. However, this notion of inconsistent edges works well with data sets where separations between clusters are apparent and densities of clusters vary smoothly. For complex situations, Zahn introduces several case-specific heuristics, but it is a difficult task to choose an adequate heuristic when the distribution of X is unknown.
- **AUTOCLUST**. Recently, Estivill-Castro and Lee [58, 59] proposed argument-free clustering algorithms for geographical data mining using proximity graph modeling. The main idea behind this approach is constructing cluster boundaries by detecting sharp density changes. By making use of both proximity and density information encoded by underlying proximity graphs, AUTOCLUST learns a Short-Long criterion. This data-driven criterion adapts to the density changes dynamically. Thus, it is able to identify the relative differences in edges and eliminate uninteresting edges.

Density-based clustering

Density-based algorithms typically regard clusters as dense regions of objects in the data space that are separated by regions of low density. In general, this approach first defines a *measure template* with a certain shape (usually circle or grid), size and a density threshold; then it detects the density distribution of the data against the predefined measure template. If the density of a test region is greater than or equal to the pre-specified density threshold, then the region belongs to the same cluster. Otherwise, it is treated as noise. In this sense, typical density-based algorithms are also based on structural models. Such a structural model can be mapped to a graph G . This graph represents the relation “ $\vec{x}_i$ is in the same cluster as $\vec{x}_j$ ” with the predicate “there is a path in G from $\vec{x}_i$ to $\vec{x}_j$ ” [52]. In other words, these algorithms search for the graph that best fits the data.

The effectiveness and efficiency of the clustering depend on the measure template specified. Clearly, the efficiency of grid-based clustering heavily depends on the granularity of the grid, because grid-based clustering uses statistical summaries of grids to perform clustering. When the grid is coarse, the clustering runs fast. However, coarse tessellation lowers the quality of clustering results. It is difficult to determine the best resolution for raster-like modeling. In general, circle-based clustering produces better clustering; but it is more expensive in terms of time efficiency than grid-based. This is because circles tessellate the data space with non-crisp feature, but grids do not. Density-based clustering can handle noise and discover arbitrary clusters, but it is not able to handle data sets with large density changes. While it can identify high density clusters, it fails to detect low peaks forming sparse clusters. The widely known algorithms of this category are as follows:

- **DBSCAN** [51]. The basic ideas of DBSCAN involve a number of new definitions. We intuitively present these definitions:
 1. The neighborhood within a radius ϵ of a given object is called the ϵ -*neighborhood* of the object.

2. If the ϵ -neighborhood of an object contains at least a minimum number, $MinPts$, of objects, then the object is called a *core object*.
3. An object $\vec{x}_k$ becomes *density-reachable* from a core object $\vec{x}_i$, if there is a list of objects $\vec{x}_i, \dots, \vec{x}_k$ such that $\vec{x}_{j+1} \in \epsilon\text{-neighborhood}(\vec{x}_j)$ for $i \leq j < k$.

Since all objects within a cluster are density-reachable from any core object within the cluster, DBSCAN first chooses a core point, then expands it by searching for all density-reachable objects to detect a cluster. The set of objects not belonging to any cluster becomes noise. DBSCAN keeps searching and expanding until no object is left.

DBSCAN can handle noise and discover clusters of arbitrary shape. It uses R*-tree to efficiently retrieve density-reachable points and to prune the search space. For a small ϵ , it requires subquadratic time. However, it is sensitive to the user-defined parameters.

- **STING** [178]. This grid-based clustering method uses a hierarchical structure to efficiently detect regions of high density. The hierarchical structure consists of several levels of different resolution. Each grid cell in a certain level is divided into four regular grids in the next level similar to quadtree data structure. Each cell in the hierarchical structure stores statistical information regarding the attributes in this cell (such as the number of objects, the mean, the standard deviation, and the extreme values). These summary statistics represent each cell and are used to determine whether the density of the cell is significant or not. STING first needs to build the hierarchical structure in bottom-up manner. Once it obtains the hierarchical structure, it performs top-down approach to find multi-level clusters through the query-answering process. Instead of looking at all the children of cells in the next lower level, STING only examines the children of the relevant cells in the previous level to prune the search space. It continues this process until the bottom level is reached.

Although STING is appealing for its efficiency, it suffers from the drawbacks of grid-based clustering mentioned above. It fails to detect clusters of different

densities. Another problem of STING is that it does not use information in the neighboring cells.

- **DENCLUE** [85]. This algorithm introduces a new approach to cluster large multimedia databases. This method is built on mathematical kernel models based on the following ideas:
 1. The influence of each data object can be formally modeled using a mathematical function, called an *influence function*, which describes the impact of a data object within its neighborhood.
 2. The overall density of the data space can be modeled analytically as the sum of the influence functions of all data objects in the space.
 3. Clusters can then be identified by determining *density attractors*, which are local maxima of the overall density function.

The main advantages of DENCLUE are that it has good clustering properties in data sets with large amounts of noise and it allows a compact mathematical description of arbitrary shaped clusters in high-dimensional data sets. However, this method requires careful selection of the density parameter and noise threshold, as the selection of such parameters may significantly influence the quality of clustering results.

2.6 Cluster validity

2.6.1 Problem specification

The objective of the clustering methods is to discover significant groups present in a data set. Cluster validity defines a certain amount of confidence that reflects how well the revealed cluster structure fits the data. That is, the hypothetical structure postulated as the result of a clustering algorithm must be tested to gain confidence

that it actually exists in the data. A variety of tasks are known under the term of cluster validity, such as clustering tendency test, parameter tuning, selecting best-fit algorithms, and measuring the significance of the revealed structure. These problems have motivated several research efforts [16, 44, 130, 169, 184]. Because there is no general definition of “cluster”, cluster validity is a challenging issue. For example, in clustering tendency tests, the concept of “no structure in the data set” is far from clear. It seems that it is impossible to formulate a theoretical null hypothesis which could be used to substantiate the validity of algorithmically suggested clusters. Despite the regular structure within a family of data sets, any clustering algorithm will be able to test only a finite set of hypotheses in finite time. Thus, many data sets in a family appear to have no structure [52].

2.6.2 Fundamental concepts of cluster validity

In general, clustering algorithms should search for clusters whose members are more homogeneous (in other words have a high degree of similarity) and well separated. Thus, researchers use compactness and separation to measure the significance of clustering results and select an optimal clustering scheme [14].

- **Compactness.** The members of each cluster should be as close to each other as possible. A common measure of compactness is the variance, which should be minimized.
- **Separation.** The clusters themselves should be widely spaced. Separation between two different clusters can be measured by inter-cluster distances, such as single-linkage.

A number of validity indices have been defined and proposed in literature to measure the significance of clustering results [17, 80, 169, 184]. Typically, cluster validity methods calculate formal indices for estimation of clustering quality, by employing the following three algorithmic processes [109, 169]:

- **External process.** This implies that we evaluate the results of a clustering algorithm against a pre-specified structure, which is imposed on a data set and reflects our intuitions about the clustering structure of the data set.
- **Internal process.** We may also evaluate the clustering results in terms of quantities that involve the patterns of the data set themselves (e.g. proximity matrix).
- **Relative process.** The basic idea is that a clustering structure is evaluated by comparing it with other clustering schemes resulting from different algorithms, or the same algorithm but with different parameter values.

External and internal validity processes are based on statistical tests. Their major drawback is high computational cost. Moreover, the indices used in these processes aim at measuring the degree to which a data set confirms an a priori specified scheme. In contrast, relative validity processes aim at finding the best clustering scheme for clustering algorithms under certain assumptions and parameters. We further address this issue in Chapter 6.

Chapter 3

Representative Clustering on Vector Spaces

The notions of models, induction principles, and clustering algorithms, apply in a general manner to clustering methods. Representative-based clustering models a cluster structure using representatives. It typically minimizes the sum of the distances of data objects to their cluster representatives. The optimization problems guided by this induction principle are typically solved heuristically for efficient computation. The most popular algorithm of this category is k -MEANS. Thus, Section 3.1 reviews k -MEANS and its alternatives. In Section 3.2, we generalize representative-based clustering methods. We investigate the characteristics of two different kinds of representatives, continuous centers (centroids) and discrete centers (medoids). We mathematically formalize the relevant objective functions based on various distance measures, for example, absolute metrics and squared distances. Based on the analysis, in Section 3.3, we propose fast and robust general purpose clustering algorithms. These algorithms inherit the basic structure of iterative methods as in k -MEANS, but minimize different loss functions. They remain very efficient and generally applicable but are more robust to noise and outliers. Section 3.4 describes a series of experiments that illustrate the efficiency of our algorithms and compare them with

alternative methods like k -HARMONIC MEANS, EXPECTATION MAXIMIZATION and GIBBS sampling. This section also includes experiments that demonstrate the robustness of our methods and the quality of their results with large data sets.

In this chapter, we assume the data are presented in vector patterns (feature vectors), which are referred to as data points. This is in contrast to the data presented in sequential patterns that we discuss in the next chapter.

3.1 How k -Means works

A distinct characteristic of data mining applications is the huge size of the data files involved. Thus, besides k -MEANS program-code simplicity, the attractiveness of k -MEANS is due to its computational efficiency. It offers practically no limitation on the size of data sets, because it typically requires only linear time to obtain an approximate solution to a hard optimization problem. More accurately, it requires $O(tDkn)$ time, where t is the number of iterations over the entire data set, D is the dimension, k is the number of clusters, and n is the number of data points. As $t, D, k \ll n$ for data mining applications, k -MEANS requires $O(n)$ time. The fascination with k -MEANS's speed has motivated its adaptation for efficient processing of large sets with both numeric and categorical attributes [77,91]. While the standard hierarchical clustering methods can handle data with numeric and categorical values using complex similarity measures [86,94], their $O(n^2)$ computational cost makes most of them unacceptable for clustering large data sets.

3.1.1 The optimization problem

By iteratively improving an initial clustering (perhaps a random clustering), the k -MEANS method produces an approximate solution to the following optimization prob-

lem:

$$\text{Minimize } SE(C) = \sum_{i=1}^n w_i \text{EUCLID}^2(\vec{x}_i, \text{REP}[\vec{x}_i, C]), \quad (3.1.1)$$

where

1. $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ is a set of n data points in D -dimensional real space $\mathbb{R}^D$;
2. the weight $w_i > 0$ may reflect relevance of the observation $\vec{x}_i$, and $\text{EUCLID}(\vec{x}, \vec{y}) = (\sum_{j=1}^D |x_j - y_j|^2)^{1/2}$ is the Euclidean metric;
3. $C = \{\vec{c}_1, \dots, \vec{c}_k\}$ is a set of k *continuous centers*, or representative points of $\mathbb{R}^D$; and
4. $\text{REP}[\vec{x}_i, C]$ is the closest point in C to $\vec{x}_i$; that is,

$$\text{EUCLID}(\vec{x}_i, \text{REP}[\vec{x}_i, C]) = \min_{j \in \{1, \dots, k\}} \text{EUCLID}(\vec{x}_i, \vec{c}_j).$$

The partition into clusters is defined by assigning each $\vec{x}_i$ to its representative $\text{REP}[\vec{x}_i, C]$. Those data points assigned to the same representative are deemed to be in the same cluster; thus, the k centers encode the partition $X = C_1 \mid \dots \mid C_k$ of the data. That is, $C_j = \{\vec{x}_i \in X \mid \text{EUCLID}(\vec{x}_i, \vec{c}_j) \leq \text{EUCLID}(\vec{x}_i, \vec{c}_q) \forall \vec{c}_q \in C \setminus \{\vec{c}_j\}\}$.

k -MEANS iteratively refines a partition alternating a minimization step and a classification step. In the minimization step, for each cluster in the partition, a new representative is computed. In k -MEANS, the *weighted arithmetic mean* of the cluster's points is a “continuous center” that minimizes the sum of squared errors between the center and the points in the cluster. Next, using the new representatives, a classification step obtains new clusters. These steps are repeated until an iteration occurs in which the clustering does not change; refer to Figure 3.1. This conceptual iteration of k -MEANS is illustrated in Figure 3.1 to highlight its similarity with EXPECTATION MAXIMIZATION.

<i>k</i> -MEANS type	EXPECTATION MAXIMIZATION type
(1) Construct initial set of representatives.	(1) Construct initial set of representatives.
(2) Iterate.	(2) Iterate.
a) Classification: (Find new clusters) Assign each observation to its nearest representative.	a) Expectation: (Find new ‘complete’ data $\tilde{Y}$) Evaluate $E[l(\tilde{Y}; \tilde{\theta}^{(t)})]$.
b) Minimization: (Find new representatives) For each cluster, find a new “center”.	b) Maximization: (Find new parameters for model) Find $\tilde{\theta}^{(t+1)}$ to maximize $E[l(\tilde{Y}; \tilde{\theta}^{(t)})]$.

Figure 3.1: Generic iterative structure of representative-based clustering.

3.1.2 Actual implementations

We highlight that the conceptual pseudo code of Figure 3.1 is not how *k*-MEANS or EXPECTATION MAXIMIZATION should actually be implemented. This is because this conceptual pseudo code implies two passes over the data. But in both cases, the two conceptual passes can be carried out per data point in an implementation that performs only one pass on the data per iteration (and obtain exactly the same result as the two-pass version).

There are a number of variants of *k*-MEANS. These may

- use random or other methods to obtain an initial partition;
- recompute the assignment of clusters each time a new representative is computed (the so-called ‘combinatorial’ reclassification [3]), rather than waiting until the new representatives of all clusters have been determined (‘non-combinatorial’ reclassification).

Basic ISO-data [46] is one variant of *k*-MEANS that uses random initialization and non-combinatorial reclassification. According to Aldenderfer and Blashfield [3], Basic ISO-data was the procedure inside the software package CLUSTAN [182].

Combinatorial reclassification has been favored in Data Mining applications [90].

Commercial software typically includes some version of k -MEANS for clustering, for example, DBMiner^{TM1}, Clementine^{TM2} and MineSet^{TM3}.

A third popular variant of k -MEANS is a combinatorial version in which the update of a representative takes place only if that update would result in a decrease in $SE(C)$. This variant goes by several names: hill-climber [3], basic minimum squared error [46], and cluster-swapping [150]. Other variants of k -MEANS also appear in the literature of vector quantization [34].

3.1.3 k -Means vs. Expectation Maximization

Disadvantages of k -Means

Despite its efficiency, k -MEANS has several disadvantages derived from its statistical simplicity. Drawbacks of k -MEANS variants are well documented in the literature:

1. From an optimization point of view, it often converges to a local optimum of poor quality [24, 68].
2. k -MEANS favors hyper-spherical clusters and it is sensitive to scaling or similar transformations [3, 46].
3. Because k central vectors are means of cluster points, they are commonly adopted as representatives of the data points of the clusters. However, it is possible for the arithmetic mean to have no valid interpretation; for example, the average of the coordinates of a group of schools may indicate that the representative school lies in the middle of a lake.
4. k -MEANS is very sensitive to the presence of noise and outliers, as well as to the initial random clustering [97]. In particular, much effort has been focused

¹A trademark of DBMiner Technology Inc.

²A trademark of SPSS Inc.

³A trademark of Silicon Graphics, Inc.

on the sensitivity of k -MEANS on the set of representatives used to initialize the search [3, 23, 65]. Some initialization mechanisms need to approximately solve NP-hard problems as well, or use dynamic programming algorithms that require $\Omega(n^2)$ time.

5. The method is statistically biased. For parametric statisticians, this implies that even if provided with the exact number of distributions in a uniform family mixture (for example, all multivariate normal distributions), and large volumes of noiseless data, k -MEANS converges to the wrong parameter values. This has favored other statistical methods such as EXPECTATION MAXIMIZATION [41] (and probabilistic clustering [9, 29, 68]). k -MEANS is also statistically inconsistent. This has favored Bayesian approaches (e.g. Data Augmentation [166] and Gibbs sampling Markov chain Monte Carlo algorithms [5, 71, 159]) and Minimum Message Length (MML) methods [43, 177]. (In fact, Wallace's MML method pioneers this type of alternatives [175]. For an account of these methods and their relation to Minimum Description Length (MDL) [141], see the paper of Wallace and Dowe [176]). While these alternatives offer more statistical accuracy, robustness and less bias, they trade these for substantially more computational requirements and more detailed prior knowledge [113].

Expectation Maximization

The most popular alternative to k -MEANS for learning with mixtures is EXPECTATION MAXIMIZATION [9, 29, 68]. This approach adds to the k -MEANS a probabilistic assignment treating class labels as hidden variables. Its formal analysis is much more complex than k -MEANS, but EXPECTATION MAXIMIZATION is consistent (asymptotically unbiased). Convergence is slow near local maxima, so some implementations switch to conjugate gradient methods or other methods near a solution [118]. The foundations of this approach were originally developed for the exponential family of distributions [28], although it applies more generally [95]. There is also a concern in EXPECTATION MAXIMIZATION for its sensitivity to initialization [65], while studies

on its attempts to accelerate its performance on large data sets via summarization indicate that the tails of distributions are critical [147].

EXPECTATION MAXIMIZATION corresponds to a family of iterative methods for approximating the *maximum likelihood estimate* (MLE) $\hat{\vec{\theta}}$ in the likelihood function [41, 166, 170]. Some statisticians interpret mixture data as incomplete data. This interpretation allows them to frame EXPECTATION MAXIMIZATION as a procedure that starts with an initial approximation $\vec{\theta}^{(0)}$ and produces a sequence of estimates $\langle \vec{\theta}^{(t)} \rangle$. Each iteration consists of a expectation step and maximization step. The expectation step estimates the complete data from the incomplete data. The maximization step takes the “estimated” complete data and estimates $\vec{\theta}$ by Maximum Likelihood (ML) [166, 170].

Titterington et al. [170] show that often, (for example, if $k = 2$ or the components are assumed to be of the same type and belonging to an exponential density family), the maximization step is explicit, in the sense that the value that attains the maximum of $E[l(\vec{Y}; \vec{\theta}^{(t)})]$ can be found algebraically, without numerical approximation, or it is of no more difficulty than an ML exercise on ‘complete’ data.

The Expectation Maximization procedure has solid theoretical results regarding the sequence $\langle \vec{\theta}^{(t)} \rangle$ of approximations. In particular, the estimated parameters produce a sequence of likelihood values that is non-decreasing.

Unfortunately, the maximization step for $\vec{\theta}_j$ depends on the form of the part f_j of the mixture $\sum_{j=1}^k \pi_j f_j(\vec{\theta}_j)$. Thus, different EXPECTATION MAXIMIZATION methods update their parameters at each iteration slightly differently. In order to have analytical solutions, typically, it is assumed that each f_j is a multivariate Gaussian density $N(\vec{\mu}_j, \Sigma_j)$ (normal density) [9, 29, 68]. Further simplifications assume that all components have the same known covariance Σ , and thus, the only unknown parameter of each component f_j in the mixture is the mean $\vec{\mu}_j$. Delicate aspects of the iteration occur at different levels. For example, even in the simple case where the mixture is uni-dimensional and the components f_j are all normal densities, it is im-

portant to have $\Sigma_j^{(t)} \neq 0$, otherwise, the process converges to the singularities created by data points placed on a class by themselves. Thus, only the simplest versions of EXPECTATION MAXIMIZATION can compete in speed with k -MEANS. By contrast, k -MEANS is general but too simple, while EXPECTATION MAXIMIZATION is robust but too specific.

3.1.4 Interpretation of k -Means

k -MEANS can be considered a direct simplification of EXPECTATION MAXIMIZATION, in that it iteratively performs a simplified expectation step and a maximization step; refer to Figure 3.1. The minimization step of k -MEANS corresponds to the maximization step of EXPECTATION MAXIMIZATION, in the case that EXPECTATION MAXIMIZATION is dealing with a mixture of k multivariate normal distributions sharing a known common covariance matrix Σ and the only unknown parameters are the mean vectors $\vec{\mu}_j$ of the components. However, the expectation step is replaced by a classification step that also has some origins in maximum likelihood estimation. The arithmetic mean serves as an estimate for $\vec{\mu}_j$. The underlying intuition follows from the observation that the multivariate normal distribution $N_{\vec{\mu}_j, \Sigma}(\vec{x})$ is large when $\|\vec{x} - \vec{\mu}_j\|_{\Sigma^{-1}}^2$ is small, because the normal distribution has a peak at $\vec{\mu}_j$ with iso-lines (level contours) defined by the covariance matrix Σ . Thus, k -MEANS can be viewed as using the squared Euclidean distance $\|\vec{x} - \vec{\mu}_j\|_I^2 = (\vec{x} - \vec{\mu}_j)^T(\vec{x} - \vec{\mu}_j)$ (which is easier to compute) to approximate the squared Mahalanobis distance $\|\vec{x} - \vec{\mu}_j\|_{\Sigma^{-1}}^2 = (\vec{x} - \vec{\mu}_j)^T \Sigma^{-1}(\vec{x} - \vec{\mu}_j)$. The dependence on the squared Euclidean distance (or gravity model) [69] and not simply the Euclidean distance is a source of concern when using k -MEANS for non-Gaussian data [123].

The criterion in Equation (3.1.1) is, in fact, a special case of several based on the *total dispersion matrix* or *total scatter matrix*

$$A = \sum_{i=1}^n (\vec{x}_i - \bar{\mu})(\vec{x}_i - \bar{\mu})^T,$$

where $\bar{\mu}$ is the total observed mean vector (no weights in this case); that is, $\bar{\mu} = \sum_{i=1}^n \vec{x}_i/n$. The criteria occur often in statistics in multivariate analysis of variance, and their motivation derives from the corresponding statistical theory [79]. The entries of A are the sum of squares of cross products determined by the data, and as such do not vary. However, if $\mathcal{P}$ is a partition of the data (a clustering), then $A = W(\mathcal{P}) + B(\mathcal{P})$, where $W(\mathcal{P})$ is the pooled ‘within-group’ dispersion matrix and $B(\mathcal{P})$ is the ‘between-group’ dispersion matrix. Methods more general than k -MEANS attempt to find a partition that minimizes the size of $W(\mathcal{P})$ (implicitly maximizing the size of $B(\mathcal{P})$). The traditional way to measure size is the trace and the determinant of these matrices. The optimization problem of Equation (3.1.1) corresponds to minimizing the trace of $W(\mathcal{P})$. This shows that k -MEANS favors hyper-spherical clusters and that it is sensitive to scaling or similar transformations [3, 46].

It is very important to realize that k -MEANS is not only measuring dissimilarity by the Euclidean distance, but more importantly, it is a least squares approach. Recall that k -MEANS approximately solves the optimization problem of Equation (3.1.1) where the dissimilarity is squared. This aspect is usually unnoticed by practitioners because the pseudo-code or the descriptions of k -MEANS and its variants do not make this aspect explicit. In fact, nowhere are distances squared. However, when k -MEANS finds a representative for a cluster by computing the arithmetic mean, what has happened is that k -MEANS has found the minimum of a strictly convex function called the gravity function. More precisely, let $C_j = \{\vec{x}_1, \dots, \vec{x}_{n_j}\}$ be the points in cluster C_j (in what follows, we use $\vec{x}_i$ for a vector assigned to cluster C_j even when our description pertains to finding the representative of the one cluster C_j). Consider the objective function

$$\text{GRAVITY}(\vec{x}) = \sum_{i=1}^{n_j} w_i \text{EUCLID}^2(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} w_i (\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i). \quad (3.1.2)$$

An illustration of this function appears in Figure 3.2.

Because the function is strictly convex, it has a unique minimum. The *gradient* of $G(\vec{x})$ is a vector field whose components are the partial derivatives of $G(\vec{x})$ at $\vec{x}$. We denote

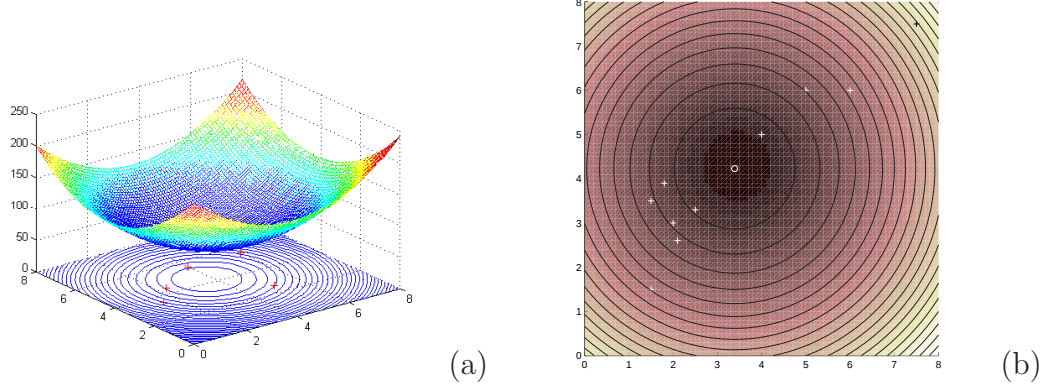


Figure 3.2: A function $\text{GRAVITY}(\vec{x})$ and its level curves. The minimum is the gravity center (shown with $\circ$).

by $\nabla G(\vec{x})$ the gradient of $G(\vec{x})$ and by definition $\nabla G(\vec{x}) = (\partial G/\partial x_1, \dots, \partial G/\partial x_D)$. Then, the unique minimum can be found by solving $\nabla G(\vec{x}) = \vec{0}$. So, for the d -th coordinate,

$$0 = \partial G/\partial x_d \sum_{i=1}^{n_j} w_i (\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i) = 2 \sum_{i=1}^{n_j} w_i (x_j - x_{i_d}).$$

Letting $W_j = \sum_{i=1}^{n_j} w_i$, this implies $\hat{x}_d = \sum_{i=1}^{n_j} w_i x_{i_d} / W_j$. That is, the minimum is the arithmetic mean $\hat{\vec{x}}^T = (\hat{x}_1, \dots, \hat{x}_D)$ of cluster C_j . It can be computed, as is typical in k -MEANS, in $O(n)$ time as $\hat{\vec{x}} = \sum_{i=1}^{n_j} w_i x_i / W_j$.

All variants of k -MEANS minimize $\text{GRAVITY}(\vec{x})$ for each cluster in their minimization step (see Figure 3.1). Thus, k -MEANS approximates the EXPECTATION MAXIMIZATION method where after each classification step, the maximization step maximizes the likelihood by minimizing $\text{GRAVITY}(\vec{x})$ within each cluster. Because of this, k -MEANS is a least squares error method where the sum of the squared discrepancies between the representative and each data point represented is minimized. Another view is that the error incurred by choosing the arithmetic mean $\hat{\vec{x}}$ as a representative for C_j is proportional to the total sum of squared discrepancies within cluster C_j .

That is, arithmetic manipulation shows that

$$\begin{aligned}
2W\text{GRAVITY}(\hat{\vec{x}}) &= 2W \left[\sum_{i=1}^{n_j} w_i (\vec{x}_i - \hat{\vec{x}})^T \cdot (\vec{x}_i - \hat{\vec{x}}) \right] \\
&= 2W \left[\left(\sum_{i=1}^{n_j} w_i \vec{x}_i^T \cdot \vec{x}_i \right) - 2W \hat{\vec{x}}^T \cdot \hat{\vec{x}} + W \hat{\vec{x}}^T \cdot \hat{\vec{x}} \right] \\
&= 2W \sum_{i=1}^{n_j} w_i \vec{x}_i^T \cdot \vec{x}_i - 2W^2 \hat{\vec{x}}^T \cdot \hat{\vec{x}} \\
&= \sum_{i=1}^{n_j} W w_i \vec{x}_i^T \cdot \vec{x}_i + \sum_{j=1}^{n_j} W w_j \vec{x}_j^T \cdot \vec{x}_j - 2 \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} w_i w_m \vec{x}_i^T \cdot \vec{x}_m \\
&= \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} w_i \vec{x}_i^T \cdot \vec{x}_i + \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} w_i \vec{x}_i^T \cdot \vec{x}_i - 2 \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} w_i w_m \vec{x}_i^T \cdot \vec{x}_m \\
&= \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} w_i w_m \text{EUCLID}^2(\vec{x}_i, \vec{x}_m). \tag{3.1.3}
\end{aligned}$$

3.2 Generalization of representative-based clustering

Berry et al. define clustering using a top-down view, where the aim is to partition a large data set of heterogeneous objects into more homogeneous classes [14]. Because we are to partition a set into more homogeneous clusters, we need to assess homogeneity. The degree of homogeneity in a group is a criterion for evaluating that the objects in one cluster are more like one another than like objects in other clusters. This criterion is then made explicit if there is a distance between objects. Representative-based clustering takes the vector quantization approach to model a cluster structure. It results in vector quantizers that encode each cluster by its representative, which is the most centrally located point of the cluster. The representatives are a reduced set of reference vectors of a data set that compresses the large reference set.

3.2.1 The optimization problem

A more general objective function to illustrate representative-based clustering can be described as

$$\begin{aligned} \text{Minimize } M^a(C) &= \sum_{i=1}^n w_i d(\vec{x}_i, \text{REP}[\vec{x}_i, C])^a \\ &[\text{subject to } C \subset X], \end{aligned} \tag{3.2.1}$$

where $a \geq 1$ is a constant, $C \subset X$ is an optional constraint. In comparison with Equation (3.1.1), the squared Euclidean metric in Equation (3.1.1) is replaced by a general form of distance $d(\vec{x}_i, \text{REP}[\vec{x}_i, C])^a$, and an optional constraint $C \subset X$ is added. With regard to the clustering context, $d(\cdot, \cdot)$ is a similarity metric. Variants of the partitioning problem arise as we see different measures of similarities and also different model representations. Then, within one variant of the problem, several algorithms are possible. For example, the case $a = 2$ defines the objective function that is the result of applying an analysis of variance using total sum error squares as the loss function. The case $a = 1$ replaces the squared loss function by the absolute loss function (the total absolute error). On the other hand, with the optional constraint $C \subset X$, we allow the cluster representatives to be either continuous (without the constraint) or discrete (with the constraint) centers.

3.2.2 Representative prototypes: continuous vs. discrete

A representative point may or may not be a data object of the data set. When chosen as a representative, a continuous center (centroid) needs not be an object of the data set of which it is a representative. For example, the k -MEANS algorithm inherently represents the clusters by their respective means, which are characterized with continuous feature. In contrast, a discrete center (medoid) must be chosen from the set of data points, that is, with the additional restriction $\text{REP}[\vec{x}_i, C] \in X$. The problem with centroids is that they are not robust estimators of central tendency [148]. Centroids are very sensitive to noise and outliers. Medoids represent better a typical value in

skew distributions and are invariant under monotonic transformations of the random variable. Centroids are invariant only under linear transformations. However, the medoid of a distribution is much less tractable from the mathematical point of view than the centroid. This is the main reason why traditional statistics usually chooses the centroid rather than the medoid to describe the “center” of a distribution [27].

3.2.3 Induction principles: squared error vs. absolute error minimization

Equation (3.1.1) represents what statisticians call a squared loss functional [148]. There is an impact on complexity and robustness if one switches the distance metric. Thus, an immediate alternative is to use an error evaluation that measures the sum of absolute errors rather than the sum of squared errors (gravity model). This criterion results in the Fermat-Weber (FW) clustering criterion [102],

$$\begin{aligned} \text{Minimize } AE(C) &= \sum_{i=1}^n w_i \text{EUCLID}(\vec{x}_i, \text{REP}[\vec{x}_i, C]) \\ &[\text{subject to } C \subset X]. \end{aligned} \quad (3.2.2)$$

Note that the squared Euclidean metric in Equation (3.1.1) is replaced by simply the absolute Euclidean metric.

Solving Equation (3.2.2) without $\text{REP}[x_i, C] \in X$ implies that, when we find the representative for cluster $C_j = \{\vec{x}_1, \dots, \vec{x}_{n_j}\}$, we find the minimum for the following function:

$$\text{FW}(\vec{x}) = \sum_{i=1}^{n_j} w_i \text{EUCLID}(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} w_i \sqrt{(\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i)}. \quad (3.2.3)$$

An illustration of this function appears in Figure 3.3. Because the function is strictly convex, it has a unique minimum that is the so called Fermat-Weber (FW) center.

However, some difficulties remain in using the Fermat-Weber center. For instance, the *gradient* of the objective function is discontinuous in the field (the *gradient* of $\text{FW}(\vec{x})$

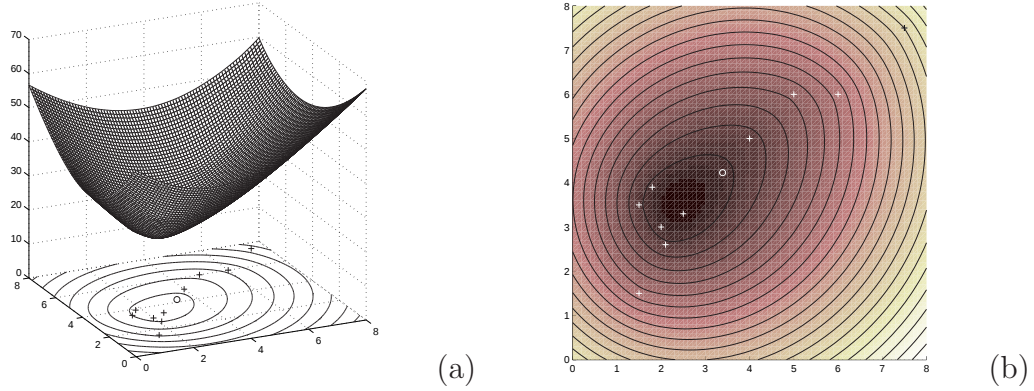


Figure 3.3: A Fermat-Weber function and its level curves. The arithmetic mean is shown with $\circ$.

is not defined for data points $\vec{x} = \vec{x}_i$). We emphasize that minimizing the function $FW(\vec{x})$ of Equation (3.2.3) for cluster $C_j = \{\vec{x}_1, \dots, \vec{x}_{n_j}\}$ is a continuous optimization problem. There is no algorithm to compute the exact coordinates of the FW center on a digital computer [8]. Other results have been mainly theoretical [6]. However, a practical approach can be developed. The use of the extended gradient [104] results in an iterative approximation algorithm [69]. Nevertheless, its convergence or divergence depends on initialization [103], and when converging, it is slow [129].

Solving Equation (3.2.2) with $\text{REP}[x_i, C] \in X$ results in medoids clustering. Obtaining optimal solutions (that we call *Discrete FW Centers*) is an NP-complete problem, because medoids clustering is equivalent to the p -medians problem (the optimization literature uses p rather than k for the number of representatives). However, several heuristics have been suggested to obtain medoid-based clustering [56,60,126]. Medoid-based clustering is much more robust than k -MEANS with respect to multiplicative or additive noise, thus resulting in clustering of much better quality, but the heuristics are still slower than k -MEANS and fundamentally applicable only for spatial data, in particular, the bi-dimensional case.

Table 3.1: Summary of representative-based optimization problems.

Optimization problems		Representative prototypes	
		Continuous centers	Discrete centers
Induction principles	Gravity distance (Squared error)	• Continuous Gravity Problem (CGP)	• Discrete Gravity Problem (DGP)
	Euclidean distance (Absolute error)	• Continuous FW Problem (CFW)	• Discrete FW Problem (DFW)

3.2.4 Summary and comparison

Table 3.1 summaries the representative-based optimization problems with respect to representative prototypes and induction principles. In previous sections, we studied CGP problem as in k -MEANS. The next section discusses DGP, CFW and DFW problems in details.

3.3 Fast and robust general purpose clustering algorithms

3.3.1 General purpose clustering algorithms

General purpose and highly applicable clustering methods are usually required during the early stages of knowledge discovery exercises to investigate potential for data mining. A general purpose clustering algorithm should satisfy basic requirements [22, 23, 65]:

- High efficiency and simplicity.
- Meaningful results with reasonable quality.

- Scalability to high-dimensional and large data sets.
- Capability to work with arbitrary metric spaces.
- Stoppable and resumable (the capacity to obtain a clustering solution at any time, and to be able to improve on the quality of the solution, given more computational resources).
- Easy interface (window or access methods) to databases and data warehouses.

In what follows next, we propose fast and robust general purpose algorithms applicable to situations in which k -MEANS is applied. While just slightly slower than k -MEANS, they offer robustness to additive and multiplicative noise. Our methods are faster than the next level of sophistication (namely, EXPECTATION MAXIMIZATION) and remain conceptually simple. For example, our methods do not demand the selection of a family of models for a mixture, the provision of good estimates of variance, or the provision of prior probabilities. Thus, they are simple to use.

3.3.2 Design of algorithms

We now illustrate the design of fast and robust general purpose clustering algorithms. To achieve the goal of efficiency and effectiveness, our algorithms derive from the basic structure of iterative methods, where subtle changes produce very different optimization problems. We employ two stages of improvement in our algorithms. First, we use medoids rather than centroids as estimators for the centers of clusters, which correspond to the optimization problems DGP and DFW in Table 3.1. Second, we use absolute error minimization rather than squared error minimization as the loss functions to learn representatives of clusters, which correspond to the optimization problems CFW and also DFW in Table 3.1. In particular, the DFW approach models clusters with respect to discrete FW centers in the classification phase of an iterative algorithm. It uses the absolute loss function in the minimization phase.

3.3.3 Discrete gravity solution (DGP)

The arithmetic means in k -MEANS (CGP) may have no valid interpretation, so one step towards robustness is to find the point in the data set that minimizes Equation (3.1.2). It is simple to compute. For any convex function f , the set $L(c) = \{\vec{x} \in \mathbb{R}^D \mid f(\vec{x}) \leq c\}$ is a convex set, for all $c \geq 0$. However, for $\text{GRAVITY}(\vec{x})$, it is not hard to show that $L(c)$ is a solid sphere [69] (in the case $D = 2$ illustrated in Figure 3.2 we see that the level curves are circles). Thus, the data point that minimizes $\text{GRAVITY}(\vec{x})$ can be found in $O(n)$ time simply by finding the center of all these spheres (the arithmetic mean $\hat{\vec{x}}$) in $O(n)$ time as before, and then, finding the nearest data point to the arithmetic mean in $O(n)$ time. Unfortunately, this variant to compute an estimator of location is only slightly more robust. However, it does point out that the finding of new centers can be achieved by other methods.

3.3.4 Continuous Fermat-Weber solution (CFW)

The CFW algorithm is to very closely locate the continuous FW center in each cluster during the maximization step. We call this algorithm k -C-FM-CENTERS. For approximating the continuous FW center, we use Kuhn's algorithm [104]. This algorithm is derived from the necessary and sufficient conditions for the optimum (equations that define it as a fixed point) [179]. Most applications of this algorithm for spatial median location have been in the context of small location problems, although the convergence may be slow. Figure 3.4 illustrates five data points and the trace of iteration towards the continuous FW center. For comparison, the figure also shows the location of the arithmetic mean.

In order to evaluate whether the approximation using Kuhn's algorithm is fast and accurate enough for data mining purposes, we conduct experiments using algorithmic engineering techniques. We perform two types of experiments. In our first set of experiments, we take a setting of data points $(\vec{x}_1, \dots, \vec{x}_m)$ where we know the

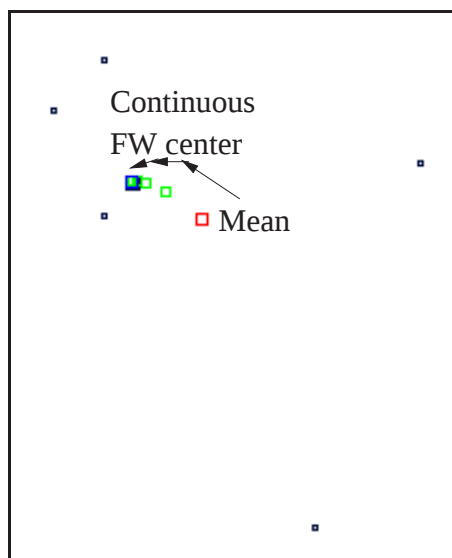


Figure 3.4: Trace of iteration towards the CONTINUOUS FW CENTER.

continuous *FW* center. We translate the data points by a vector $\vec{x}$, and we apply Kuhn's algorithm to the translated set. Then, we translate the solution obtained back by adding $-\vec{x}$. We found that the results from Kuhn's algorithm are accurate to the relative precision of the floating point systems used. In our second set of experiments, we distribute data points on a circle and compute the continuous FW center. For comparison, we repeat this experiment with various radii of the circles but keep the same relative positions of the data points on each circle; then we change the density of points distributed on the circles and compare the obtained results. This set of experiments gave similar results to the first experiment. Together these two experiments show that despite the errors introduced by the floating-point system, the results from Kuhn's algorithm are satisfactory for data mining. Moreover, no divergent cases happened with our implementation.

3.3.5 Discrete Fermat-Weber solution (DFW)

The DFW approach is to find the data point in each cluster that minimizes $\text{FW}(\vec{x})$. That is, we solve a discrete 1-median problem for each cluster. Again, we minimize Equation (3.2.3), but now with the additional restriction that the estimator of location be in C_j . This clustering algorithm (referred to as k -D-FM-CENTERS) has the same structure for k -MEANS presented in Figure 3.1, but the new center of each cluster C_j is the discrete 1-median of the points in C_j . This can be solved superficially in $O(n_j^2)$ time by evaluating $\text{FW}(x_i)$ (for $i = 1, \dots, n_j$) and returning the data point that resulted in the minimum value. However, this results in an overall clustering method requiring quadratic time. In what follows, we use our algorithmic engineering approach to present a novel filtering strategy to obtain the discrete 1-median problem for each cluster in $O(n_j \log n_j)$ time. Our strategy does not impose any restrictions on the dimension of the data.

The extended gradient as a filter

We now show that the extended gradient is very useful as a filter.

Lemma 3.3.5.1 (Kuhn [103]) *For $m = 1, \dots, n_j$, let*

$$\text{FW}_{-m}(\vec{x}) = \sum_{i=1, i \neq m}^n \text{EUCLID}(\vec{x}, \vec{x}_i)$$

be an objective function where the m -th point in C_j is excluded from consideration. Let $\nabla \text{FW}(\vec{x})$ be the gradient of $\text{FW}(\vec{x})$ defined in $\mathbb{R}^D \setminus C_j$. Let the extended gradient of $\text{FW}(\vec{x})$ be denoted by $\nabla_E \text{FW}(\vec{x})$ and defined by

$$\begin{aligned} & \nabla_E \text{FW}(\vec{x}) \\ &= \begin{cases} \nabla \text{FW}(\vec{x}) & \text{if } \vec{x} \notin C_j, \\ \max \left\{ 1 - \frac{1}{\|\nabla \text{FW}_{-m}(\vec{x})\|}, 0 \right\} \nabla \text{FW}_{-m}(\vec{x}_m) & \text{if } \vec{x} = \vec{x}_m \in C_j. \end{cases} \end{aligned}$$

Then, $\nabla_E \text{FW}(\vec{x})$ is defined in $\mathbb{R}^D$ and the point $\mathbf{f}$ minimizes $\text{FW}(\vec{x})$ if and only if $\nabla_E \text{FW}(\mathbf{f}) = \vec{0}$.

The extended gradient has the properties of the gradient with respect to the level curves. For our purposes the following is most important [4].

Property 3.3.5.1 *The extended gradient vector $\nabla \text{FW}_E(\vec{x})$ is normal to the tangent hyper-plane at $\vec{x}$ to $L(\text{FW}(\vec{x})) = \{\vec{y} \in \mathbb{R}^D \mid \text{FW}(\vec{y}) \leq \text{FW}(\vec{x})\}$, for all $\vec{x} \in \mathbb{R}^D$.*

For an illustration in two dimensions see Figure 3.5.

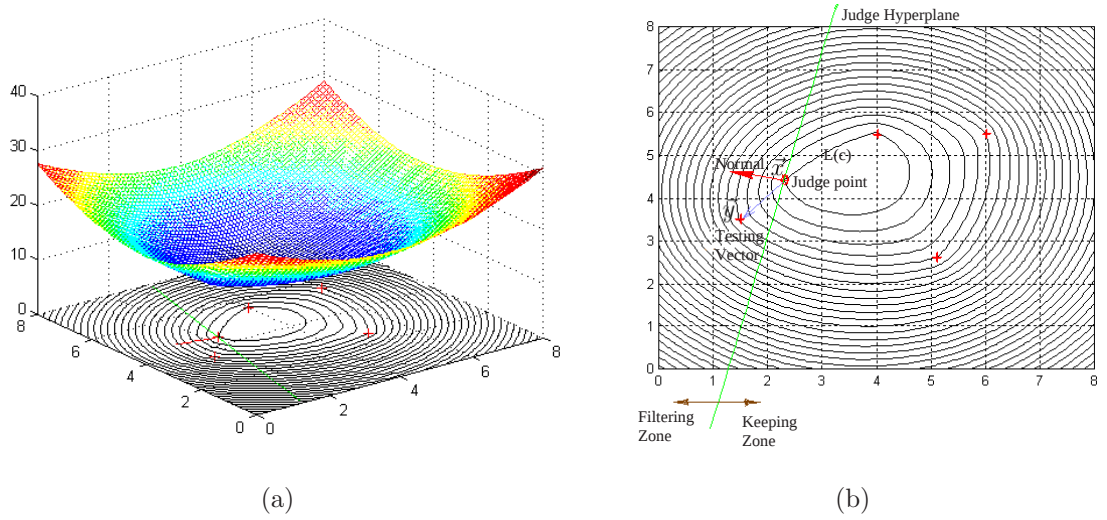


Figure 3.5: Fermat-Weber level curve and its normal and tangent.

Thus, the extended gradient can be used as a filtering mechanism to eliminate data points that cannot possibly be the discrete 1-median.

Consider a point $\vec{x}$ on a level curve bounding $L(c)$; thus, $c = \text{FW}(\vec{x})$. For illustration, refer to Figure 3.5. The tangent hyper-plane of $L(c)$ at $\vec{x}$ will divide the space $\mathbb{R}^D$ into two parts. One half-space contains the set $L(c)$. We call this half space the *keeping zone*. For any point $\vec{y}$ in the other half space, $\text{FW}(\vec{y}) > c = \text{FW}(\vec{x})$. We call the other half space the *filtering zone*. Thus, any point in the filtering zone can be discarded from being the discrete 1-median if there is a data point $\vec{x}_i \in C_j$ in the keeping zone with $\text{FW}(\vec{x}_i) \leq c = \text{FW}(\vec{x})$. In addition, we call the dividing hyper-plane between the keeping zone and the filtering zone the *judge hyper-plane*. In particular, the point $\vec{x}$ is called the *judge point*.

Note also that after computation of the extended gradient we have the judge hyper-plane of $L(c)$ encoded by its normal vector $\vec{n}_{\vec{x}}$. Moreover, testing whether a point is in the filtering zone can be done in constant time (with respect to $n_j = \|C_j\|$). Namely, for any point $\vec{y}$ in $\mathbb{R}^D$, if the dot product between the normal vector $\vec{n}_{\vec{x}}$ and $\vec{y} - \vec{x}$ is not negative, $\vec{y}$ is in the filtering zone (this is because if we let $\alpha_{\vec{x}\vec{y}}$ denote the angle between $\vec{n}_{\vec{x}}$ and $\vec{y} - \vec{x}$, then $\cos(\alpha_{\vec{x}\vec{y}})$ is non-negative if and only if $\vec{y}$ is in the filtering zone).

Our algorithm for 1-medoid repeatedly finds a hyper-plane and filters the data points. In order for the algorithm to be effective, we need a data point $\vec{x}_i$ in the keeping zone with $\text{FW}(\vec{x}_i) \leq c = \text{FW}(\vec{x})$. This is easily achieved if we select $\vec{x}$ to be some data point $\vec{x}_i$. Note the importance of the extended gradient (selecting $\vec{x} = \vec{x}_i$ is impossible with the standard gradient).

However, we also need computational efficiency. Filtering passes must remove many points from further consideration because we can only afford $O(n)$ passes for a sub-quadratic clustering method.

Each judge point halves the list of candidates

We now describe our algorithmic engineering strategy for choosing hyper-planes that remove half of the candidates in each filtering step, thus reducing the n_j candidates in $\log n_j$ steps to a very small number where the discrete 1-median can be found in $O(n_j)$ time. The halving strategy will result in a total of $O(n_j \log n_j)$ time to compute the discrete 1-median.

As we have already pointed out, the arithmetic mean corresponds to the center of mass. Also, by Equation (3.1.3) the mass is

$$\text{GRAVITY}(\hat{\vec{x}}) = \frac{1}{2W_j} \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \text{EUCLID}^2(\vec{x}_i, \vec{x}_m),$$

where $W_j = \sum_{\vec{x}_i \in C_j} w_i$. Moreover, the level curves are spheres. Thus any hyper-plane

on the arithmetic mean, independently of direction, will divide the mass in half. Thus, a procedure that repeatedly uses the center of mass as the judge point will require

$$O(n_j \log \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \text{EUCLID}^2(\vec{x}_i, \vec{x}_m) / 2W)$$

time in the worst case to reduce the number of candidates to a small constant. Since with $O(n_j)$ time we can make sure that

$$\sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \text{EUCLID}^2(\vec{x}_i, \vec{x}_m) / 2W = O(n_j^l)$$

with l a small constant, we have a total of $O(n_j \log n_j)$ time to filter n_j items.

The problem with this procedure is that the arithmetic mean can not be simply selected as a judge point. While performing judgement at the arithmetic mean, the discrete 1-median may actually be on the filtering zone. Thus, to keep the discrete FW center from being filtered, we need to select a point $\vec{x} = \vec{x}_i \in C_j$. For an efficient filtering processing, our choice is to repeatedly select the $\vec{x}_i$ in the list of candidates closest to the arithmetic mean of the list of candidates. Eventually, we select the u ($u \geq 1$ a small integer, e.g. $u = \max(D \log(n_j), 1)$) nearest neighbors in C_j to the arithmetic mean of C_j as judge points. We also ensure that the hyper-planes cut a large proportion of the mass. There are very constrained configurations where this strategy will fail to remove a fraction of the candidates (for example, when the data points are bi-dimensional and constitute the vertices of a regular polygon). We monitor the number of candidates filtered, if the u judging hyper-planes filter less than a fourth of the candidates, we just pick the discrete gravity center (the nearest neighbor to the arithmetic mean) to approximate the discrete FW center of this cluster. This maintains the $O(n_j \log n_j)$ time bound. Convergence of the overall clustering method results from its maximization/classification structure [155].

Thus, our algorithm for finding the discrete FW center of cluster C_j is as follows.

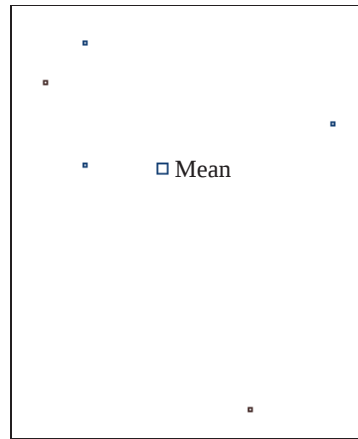
Step 0. Set the list of candidates to the data points in C_j .

- Step 1.** Calculate the arithmetic mean $\hat{x}$ of a list of candidates.
- Step 2.** Find u points in the candidate set nearest to the arithmetic mean.
- Step 3.** If for one of the u -points, $\vec{x}_m$, $\text{FW}(\vec{x}_m) < \text{FW}(\hat{x})$, use $\hat{x}$ as another judge point.
- Step 4.** Compute $\nabla_E \text{FW}(\vec{x}_m)$ and construct the normal vector with $\vec{x}_m$ as the judge point.
- Step 5.** Remove from the candidate list the points in u filtering zones.
- Step 6.** If at least a quarter of the candidates was filtered, repeat step 1 to step 5. Otherwise, return the point nearest to the arithmetic mean.
- Step 7.** When the list of candidates has $2u$ or less points, perform a brute force search for the discrete 1-median.

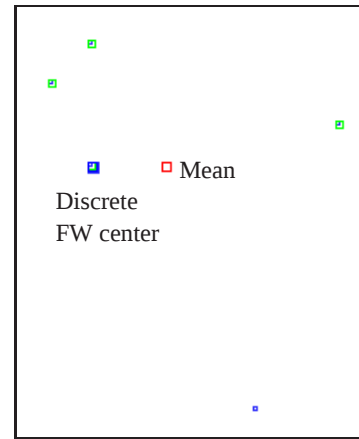
3.3.6 Other solutions

To show the difference among a group of representative-based clustering methods, we also implemented k -HARMONIC MEANS [187] and k -C-L1-CENTERS [24]. Figure 3.6 demonstrates five points and centers of different clustering methods. k -HARMONIC MEANS is a variant of k -MEANS, but computes the harmonic average of each cluster as the representative. k -C-L1-CENTERS is a similar method [24] to our algorithm k -C-FM-CENTERS, but it switches similarity measure to the 1-norm to avoid the Fermat-Weber problem. Although it confirms that continuous $L1$ centers offer more robust iterative representative-based clustering than arithmetic means, the change of similarity metric has problems because, beyond 2 dimensions, the 1-norm median can be outside the convex hull of the cluster of points. Moreover, this alternative was proposed [24] with the use of a search for continuous 1-norm centers that formulates a bilinear program. This bilinear program is solved iteratively in closed form.

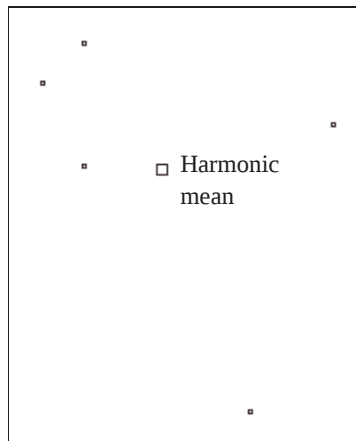
However, a much simpler implementation is possible since the continuous 1-norm center can be obtained as the median of the projections on each coordinate [69]. Our implementation of k -C-L1-CENTERS uses this much simpler and faster algorithm.



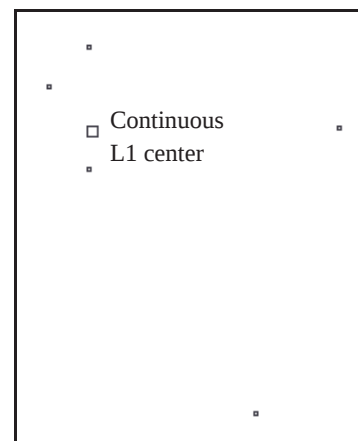
(a) Arithmetic mean



(b) Discrete FW center



(c) Harmonic mean



(d) Continuous L1 center

Figure 3.6: Comparison of four different types of centers (estimators of location) on the same data set.

3.4 Performance evaluation

3.4.1 Time complexity

We first present results that evaluate the efficiency of our algorithms discussed in Section 3.3. In particular, our C implementation of k -D-FM-CENTERS (see Section 3.3.5) is compared with our C implementation of k -MEANS and EXPECTATION MAXIMIZATION. For EXPECTATION MAXIMIZATION we used the assumption that the covariance matrix Σ_j of each component, although unknown, is diagonal [128]. The assumption that Σ_j is diagonal implies that the component densities are aligned with the axes. In a sense, the clouds of data points that are the clusters are ellipsoids with axes parallel to the coordinate axes. If the assumption on diagonal form is removed, then other assumptions are required to manage the maximization step explicitly. Thus, this is the most flexible EXPECTATION MAXIMIZATION with efficiency comparable with k -MEANS. We also compared with GIBBS sampling. We used the 1999 release (also in C language) of the *fbm-software* by R. N. Neal [125]⁴ for the GIBBS sampling and the generation of data in two examples for Bayesian mixture models.

The first example is a bivariate density estimation problem. We used the *fbm-software* to generate data sets from 500 to 100,000 points and measured the CPU time of the algorithms. The Bayesian GIBBS sampling (with two components in the mixture) remains linear because of a bound in the number of iterations, but requires more than 15 minutes of CPU for 100,000 records. It requires several meta-parameters and prior probabilities but provides much more information on termination. EXPECTATION MAXIMIZATION requires 5 minutes of CPU for the same 100,000 points while k -D-FM-CENTERS needs only 1 minute and k -MEANS only 15 seconds. Thus, our k -D-FM-CENTERS algorithm is significantly faster than EXPECTATION MAXIMIZATION. In fact, time requirements are a fifth of EXPECTATION MAXIMIZATION and 5 times more than k -MEANS for this Bayesian model. All methods achieved equivalent quality

⁴In October 1991, we retrieved it from <http://www.cs.toronto.edu/~radford/fbm.software.html>

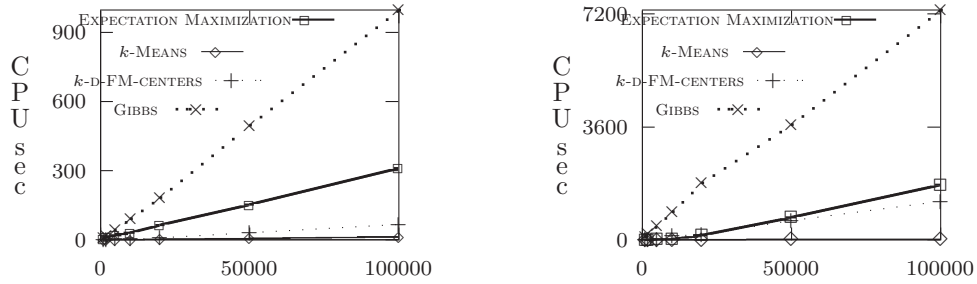


Figure 3.7: Comparison of CPU times for R. N. Neal examples for Bayesian mixture models.

classification with respect to GIBBS sampling with $k = 2$. Thus, the only difference for these data sets was speed. In fact, for most numerical data, specially from mixtures of Gaussians with noise, our methods was significantly faster than EXPECTATION MAXIMIZATION and just slightly slower than k -MEANS. The slowest remained GIBBS sampling.

The second example by R. N. Neal consists of 10 dimensional data where all attributes are categorical (and boolean). This data set is the most difficult for our type of method, because, when data items are regarded as vectors in $\mathbb{R}^{10}$, they all are placed in the vertices of the Hilbert cube. Thus, they are all in the vertices of a convex body and clustering really is a result of repetitions. When using the *fbm-software* to generate 100,000 records, only 1024 are different. The only method of those tested designed for this type of data is GIBBS sampling which took just over 2 hrs of CPU time (using the default stopping criteria in the *fbm-software*). However, all methods obtained the 4 patterns corresponding to the 4 centers of the clusters. The only exception was k -MEANS, which resulted in poorer clusters for files of 20,000 records or more, while for 500 records or less k -D-FM-CENTERS occasionally missed one center. Thus, for these data sets, quality of clustering was not an issue. However, our method becomes slower, about the same as EXPECTATION MAXIMIZATION, while k -MEANS remains extremely fast (only half a minute of CPU time for 100,000 items). After extensive search, we believe that this type of data set constitutes the worst case for our method, and we are pleased to see that it remains comparable to EXPECTATION

MAXIMIZATION. We should remark that we experimented with enlarging the categorical domain of attributes (from two to 5 or 10 values) and found that k -MEANS' performance deteriorates and ours improves.

3.4.2 Resistance to noise

To contrast the quality of clustering in the presence of noise, we present an experimental illustration contrasting our algorithm k -D-FM-CENTERS (see Section 3.3.5) with k -MEANS and EXPECTATION MAXIMIZATION. We first used a simple GENERATOR [56] of bi-dimensional data sets and mechanisms to regulate noise. The mixture of GENERATOR produces a random vector $\vec{x}_i^T = (x_{i1}, x_{i2}) \in \mathbb{R}^2$ with a probability density function given by

$$p(\vec{x}) = \frac{1-\phi}{k}P(\vec{c}_1 + r) + \dots + \frac{1-\phi}{k}P(\vec{c}_k + r) + \phi \mathcal{U}([0, 1] \times [0, 1]), \quad (3.4.1)$$

where $\mathcal{U}([0, 1] \times [0, 1])$ denotes the uniform distribution over the unit square and $P(\vec{c} + r)$ denotes a peak distribution over the circle centered at $\vec{c}$ and radius r .

In this GENERATOR, noise is modeled as the additive term in the finite mixture model corresponding to uniform distribution on the unit square. Additive noise are points that do not belong to any cluster. Our experiments compare clustering algorithms with different levels of additive noise (by increasing values of ϕ). For this purpose, the GENERATOR has an option modified to initially produce a data set with no noise. A later option can be used to perform a pass over the first data set and the GENERATOR repeatedly generates a random number ρ uniformly in $[0, 1]$. If $\rho > \phi$, it copies a data point from its input to the output. If $\rho \leq \phi$, then a point in $\mathcal{U}([0, 1] \times [0, 1])$ is produced. As performed in [56], for the same data set and level of additive noise we compare with different levels of multiplicative noise — the term for it would appear multiplying in the finite mixture model. A second program takes each data point and perturbs it slightly. The idea here is to test the numerical robustness (or robustness to multiplicative noise) of the algorithms. The model by which we perturb each point

Table 3.2: Misclassification with 95% confidence intervals (one data set).

$n = 300$ $k = 10$ $u = 1$		One data set (10 runs per set)			
		Algorithm			
		k -MEANS		k -D-FM-CENTERS	EM
Noise	Start	Random	MST	Random	Random
ψ	ϕ				
0	0	39% $\pm$ 7	8%	16% $\pm$ 4	25% $\pm$ 5 %
	0.1	30% $\pm$ 4	27%	16% $\pm$ 4	30% $\pm$ 4
	0.2	30% $\pm$ 5	30%	22% $\pm$ 5	39% $\pm$ 3
0.5	0	29% $\pm$ 5	12% $\pm$ 4	14% $\pm$ 4	24% $\pm$ 4
	0.1	30% $\pm$ 4	30% $\pm$ 6	14% $\pm$ 4	38% $\pm$ 4
	0.2	30% $\pm$ 5	31% $\pm$ 3	18% $\pm$ 5	38% $\pm$ 4
1.0	0	33% $\pm$ 5	19% $\pm$ 4	18% $\pm$ 4	22% $\pm$ 3
	0.1	26% $\pm$ 6	36% $\pm$ 9	17% $\pm$ 4	39% $\pm$ 4
	0.2	35% $\pm$ 6	30% $\pm$ 5	15% $\pm$ 5	38% $\pm$ 4
1.5	0	34% $\pm$ 4	22% $\pm$ 3	18% $\pm$ 4	30% $\pm$ 5
	0.1	30% $\pm$ 4	31% $\pm$ 6	14% $\pm$ 5	37% $\pm$ 10
	0.2	34% $\pm$ 4	30% $\pm$ 5	20% $\pm$ 5	34% $\pm$ 4

is to scan a data set and for each point $\vec{x}_i$, use it as a center of a peak distribution $P(\vec{x}_i + \psi r)$ to generate a replacement point no further away than ψr from $\vec{x}_i$. The levels of multiplicative noise are regulated by the parameter ψ . The case $\psi = 0$ replaces each point by itself introducing no multiplicative noise. As ψ grows, the original circle clusters become less dense and more overlap occurs. The quality of a partition is the percentage of non-noise points that are labeled correctly by the clustering algorithm. The fewer the mislabeled points, the higher the quality of the partition. Table 3.2 shows comparisons of the algorithms for one data set of 300 data items generated by Equation (3.4.1). The algorithms compared are k -MEANS initialized by the results of single-linkage clustering (found in $O(n \log n)$ time by Minimum Spanning Tree computation), our k -D-FM-CENTERS and EXPECTATION

Table 3.3: Misclassification with 95% confidence intervals (10 data sets).

$n = 300$		10 data sets (10 runs per set)			
$k = 10$		Algorithm			
$u = 1$		k -MEANS		k -D-FM-CENTERS	EM
Noise	Start	Random	MST	Random	Random
ψ	ϕ				
0	0	31% $\pm$ 4	7% $\pm$ 2	16% $\pm$ 5	24% $\pm$ 6
	0.1	30% $\pm$ 4	23% $\pm$ 5	18% $\pm$ 5	42% $\pm$ 6
	0.2	30% $\pm$ 5	30% $\pm$ 3	20% $\pm$ 4	40 % $\pm$ 6
0.5	0	30% $\pm$ 4	12% $\pm$ 4	19% $\pm$ 7	24% $\pm$ 4
	0.1	30% $\pm$ 4	30% $\pm$ 4	19% $\pm$ 7	37% $\pm$ 6
	0.2	30% $\pm$ 5	30% $\pm$ 5	18% $\pm$ 6	39% $\pm$ 7
1.0	0	31% $\pm$ 5	20% $\pm$ 4	22% $\pm$ 6	22% $\pm$ 4
	0.1	30% $\pm$ 6	32% $\pm$ 4	20% $\pm$ 4	38% $\pm$ 7
	0.2	31% $\pm$ 6	30% $\pm$ 5	19% $\pm$ 5	40% $\pm$ 8
1.5	0	33% $\pm$ 4	22% $\pm$ 4	18% $\pm$ 5	31% $\pm$ 6
	0.1	32% $\pm$ 4	31% $\pm$ 5	20% $\pm$ 4	38% $\pm$ 9
	0.2	32% $\pm$ 4	31% $\pm$ 5	20% $\pm$ 5	36% $\pm$ 8

MAXIMIZATION (with diagonal covariance matrices). Table 3.3 combines the results from 10 different data sets, each generated by the model in Equation (3.4.1). The results clearly show the robustness of our algorithm to both types of (additive and multiplicative) noise.

3.4.3 3D mixture data test

This experiment consists of generating data with respect to a mixture of 3-dimensional (multivariate) normal distributions with noise. Thus, data was generated with the form $p(\vec{x}) = \pi_1 N_{\vec{\mu}_1, \Sigma_1}(\vec{x}) + \dots + \pi_k N_{\vec{\mu}_k, \Sigma_k}(\vec{x}) + \pi_{k+1} \mathcal{U}(\vec{x})$ where each component $N_{\vec{\mu}_j, \Sigma_j}(\vec{x})$ is a multivariate normal distributions with mean $\vec{\mu}_j$ and the covariance

matrix Σ_j and $\mathcal{U}(\vec{x})$ is the uniform distribution in a box that bounds all $\vec{\mu}_j$. Again we compared k -MEANS, EXPECTATION MAXIMIZATION, and our algorithms: our implementations of k -C-L1-CENTERS (see Section 3.3.6) and k -D-FM-CENTERS. We used 20% noise, $k = 3$ and $\pi_j = .8/k$ for $j = 1, \dots, k = 3$. Moreover, the three covariance matrices Σ_j were set to the identity in order to create data sets as favorable as possible to k -MEANS and EXPECTATION MAXIMIZATION.

We evaluated the quality of the clustering results by the sum of the norms between the original $\vec{\mu}_j$ and the approximations $\hat{\vec{\mu}}_j$ obtained for the algorithms. Data sets with $n = 2,000$ were generated by selecting three points $\vec{\mu}_j$ at random in $[0, 20.0] \times [0, 20.0] \times [0, 20.0]$. For example, a typical data set had $\vec{\mu}_1^T = (13.1, 7.6, 6.9)$, $\vec{\mu}_2^T = (2.6, 7.1, 14.5)$ and $\vec{\mu}_3^T = (16.6, 9.3, 14.9)$. Table 3.4 shows the results for one data set.

Table 3.4: Results for 3-D mixture of normals with 20% noise.

Algorithm	Estimated $\hat{\vec{\mu}}_j^T$	$\sum_{j=1}^3 \ \hat{\vec{\mu}}_j - \vec{\mu}_j\ $	CPU time
k -MEANS	$\vec{\mu}_1^T = (12.7, 8.4, 6.3)$	2.83	96 sec
	$\vec{\mu}_2^T = (3.1, 7.8, 13.5)$		
	$\vec{\mu}_3^T = (16.3, 9.6, 14.8)$		
EXPECTATION MAXIMIZATION	$\vec{\mu}_1^T = (10.1, 9.7, 9.4)$	7.77	5 sec
	$\vec{\mu}_2^T = (2.7, 7.1, 14.4)$		
	$\vec{\mu}_3^T = (15.1, 8.6, 11.3)$		
k -D-FM-CENTERS	$\vec{\mu}_1^T = (12.8, 8.0, 6.9)$	0.97	0.7 sec
	$\vec{\mu}_2^T = (2.8, 7.1, 14.4)$		
	$\vec{\mu}_3^T = (16.6, 9.6, 14.8)$		
k -C-L1-CENTERS	$\vec{\mu}_1^T = (7.8, 16.0, 6.0)$	16.8	0.4 sec
	$\vec{\mu}_2^T = (3.8, 7.1, 14.4)$		
	$\vec{\mu}_3^T = (15.0, 8.5, 9.8)$		

Typically, the sum of discrepancies between norms was twice as large for k -MEANS than for k -D-FM-CENTERS. k -D-FM-CENTERS consistently outperformed the others with respect to quality. On average, EXPECTATION MAXIMIZATION is closer than the results in Table 3.4 but still surpassed by k -D-FM-CENTERS. The fastest is k -C-L1-CENTERS, followed by k -D-FM-CENTERS. k -MEANS has problems detecting convergence and sometimes is the slowest. Both EXPECTATION MAXIMIZATION and k -MEANS end up being several times slower than k -D-FM-CENTERS and k -C-L1-CENTERS. k -C-L1-CENTERS can produce some poor results, as illustrated by

Table 3.4.

Because the results of these algorithms depend on their random initialization, we have compared them when they are executed 10 times, but for each execution, all algorithms start with the same initial configuration. Using 10 different 3D mixtures (only the true means are different), the first 4 columns of Table 3.5 show the results of 4 algorithms starting 10 times with common random initialization.

Table 3.5: Results for 10 data sets each consisting of a 3-D mixture of normals with 20% noise.

Data set	10 common start configurations				Multistart Winner
	Error measured as $\sum_{i=1}^3 \ \hat{\mu}_j - \bar{\mu}_j\ $				
	EXPECTATION MAXIMIZATION	<i>k</i> -MEANS	FUZZY- <i>c</i> -MEANS	<i>k</i> -D-FM-CENTERS	
1	4.75 ±0	5.65 ±4	1.92 ±0	13.8 ±10	<i>k</i> -D-FM-CENTERS
2	3.96 ±3	6.17 ±3	1.94 ±0	10.90 ±5	<i>k</i> -D-FM-CENTERS
3	9.84 ±3	15.78 ±2	6.21 ±3	7.77 ±6	<i>k</i> -D-FM-CENTERS
4	10.59 ±5	4.93 ±4	1.57 ±0	5.12 ±6	<i>k</i> -D-FM-CENTERS
5	11.16 ±3	5.80 ±4	2.03 ±0	7.54 ±5	<i>k</i> -D-FM-CENTERS
6	15.09 ±0	13.04 ±5	1.80 ±0	10.09 ±6	<i>k</i> -D-FM-CENTERS
7	10.88±2	6.29 ±4	1.07 ±0	5.60 ±5	<i>k</i> -D-FM-CENTERS
8	3.3 ±0	5.31 ±4	4.17 ±6	5.51 ±6	<i>k</i> -D-FM-CENTERS
9	13.35 ±3	9.52 ±4	2.72 ±0	7.13 ±5	<i>k</i> -D-FM-CENTERS
10	9.11 ±2	6.79 ±4	2.21 ±0	4.54 ±5	<i>k</i> -D-FM-CENTERS

Clearly EXPECTATION MAXIMIZATION and *k*-MEANS have large error and small variance. FUZZY-*c*-MEANS [16] has small error on average and nil variance. This shows that FUZZY-*c*-MEANS is the least subject to the initialization configuration. In practical settings, all these algorithms should be executed several times and the best solution of all (measured by the criteria they are optimizing) should be adopted as the answer from such algorithm. This multi-start version eliminates the dependency on the initial configuration. We declared an algorithm a multi-start winner, if it produced the best result in each of 10 multi-starts, each consisting of 10 random initial configurations. The last column of Table 3.5 shows that our algorithm was the winner for the 10 data sets. Occasionally our methods provide poor results, on some initial configurations. But their multi-start version is superior to all others.

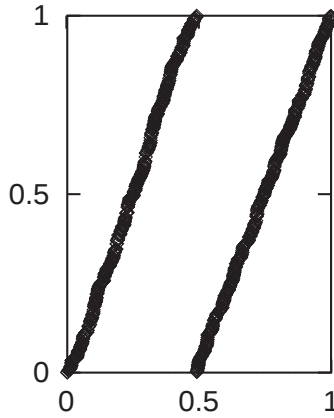


Figure 3.8: Data is uniform if projected to either axis, but has a clear pattern.

k -C-L1-CENTERS is faster than k -D-FM-CENTERS, but one should be aware that k -C-L1-CENTERS performs poorly when the clusters are the synergy (interaction) of the attributes. Figure 3.8 shows two 2D clusters. They have the very distinctive pattern: they are lines. However, for each coordinate, the data is a uniform distribution. The algorithm k -C-L1-CENTERS finds clusters with over 40% error 90% of the time, and 10% of the time the result is totally wrong (providing two clusters separated by the line $y = 0.5$). By contrast, k -D-FM-CENTERS performs very well in this data set, 90% of the time the misclassification is only 10%.

The statistical literature has rejected the L1-metric optimization because the estimator of location can be outside the convex hull of the cloud of points for which it is estimating a center [148]. A simple example is the 3 point set $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ in 3D. The $L1$ center is $(0,0,0)$ which is outside the convex hull.

3.5 Remarks

Many clustering methods exist to partition a data set by some natural measures of similarity. While there is no widely accepted definition of a cluster, many algorithms have been recently developed to suit specific domains. In this chapter, we generalized representative-based clustering methods with respect to models and induction prin-

ciples. We proposed fast and robust general purpose clustering algorithms. These algorithms are suitable for exploratory data analysis. They produce clusterings with improved results. They do not depend on the order of the data, as some variants of k -MEANS do, and they do not demand detailed initialization. Their uses bring an insight into the structure of a large multidimensional data set. Because they are faster than EXPECTATION MAXIMIZATION, they can be applied in processes (in combination with prespecified criteria) for determining the number k of clusters. Recall that the most common process to find a robust estimate of k is to repeat clustering with different values of k [126,128]. We report more about this issue when discussing validity indices in Chapter 6.

Chapter 4

Representative Clustering on Metric Spaces

Many scientific and commercial domains have seen an enormous growth of data that have inherent sequential nature. Examples for such data-providing environments include World Wide Web (WWW), where users, when seeking for information of interest, travel from one object to another via the corresponding facilities (i.e. hyperlinks and URLs) provided [33]. Clustering sequential data is useful for various purposes. For example, understanding Web visitor access patterns will not only help improving the system design, but also be able to lead to better marketing decisions. Over the years, many methods have been developed for clustering sequential objects according to their similarities. There is a demand for fast and robust clustering algorithms on sequential data. Section 4.1 describes sequential data and measuring similarities between sequences. Section 4.2 discusses tasks and challenges of sequential data mining. In Section 4.3, we provide a brief overview of existing methods to cluster sequential data. Section 4.4 presents a generally applicable approach to clustering sequences. Our approach implements medoid-based algorithms and generalizes fuzzy cluster assignments. This new approach does not require an all-against-all analysis on data patterns, but calls on the computation of the dissimilarity only when necessary. It

requires sub-quadratic time by using randomization strategies.

4.1 Formal background of sequential data

In recent years, there has been an enormous growth in the amount of available commercial and scientific data. Data from domains such as retail transactions, Web access logs, intrusion detection, and protein sequences have an inherent sequential nature [78].

4.1.1 Definitions of sequential data

In the field of data mining, a two-level mathematical modeling of sequential patterns has emerged [2]. For completeness, we describe this rich mathematical structure and illustrate many incarnations that it has with specific similarity measures.

Let $V = \{v_1, \dots, v_m\}$ be a set of literals, called *events*. An *eventset* is a non-empty multiset¹ of events. We denote an eventset S by $\{s_1, s_2, \dots, s_l\}$, where an element $s_j \in V$ is an event. A *sequence* is an ordered list of eventsets. We denote a sequence u by $\langle S_1 S_2 \dots S_m \rangle$, where S_i is an eventset. We also call S_i a member of the sequence. An eventset is considered to be a sequence with a single member.

A sequence $\langle S_1 S_2 \dots S_l \rangle$ is a subsequence of another sequence $\langle S'_1 S'_2 \dots S'_m \rangle$ if there exist integers $i_1 < i_2 < \dots < i_l$ such that $S_1 \subseteq S'_{i_1}, S_2 \subseteq S'_{i_2}, \dots, S_l \subseteq S'_{i_l}$. For example, the sequence $\langle \{3\}\{4, 5\}\{8\} \rangle$ is a subsequence of $\langle \{7\}\{3, 8\}\{9\}\{4, 5, 6\}\{8\} \rangle$, since $\{3\} \subseteq \{3, 8\}$, $\{4, 5\} \subseteq \{4, 5, 6\}$ and $\{8\} \subseteq \{8\}$. However, the sequence $\langle \{3\}\{5\} \rangle$ is not a subsequence of $\langle \{5\}\{3\} \rangle$ (and vice versa).

¹In set theory, $x \in S$ is a boolean predicate for S , a set; and its value is true if x is an element of S . In classical set theory, an event belongs only once to a set, so $\{1, 2\} \cup \{1, 3\} = \{1, 2, 3\}$. An extension that allows us to consider repetitions, but maintains that order is not important, is the notion of multiset. Here, $\{1, 1, 2\} = \{1, 2, 1\} = \{1^{(2)}, 2\}$ and also $\{1, 2\} \cup \{1, 3\} = \{1^{(2)}, 2, 3\}$, where the superscript (2) indicates that the element 1 occurs twice.

4.1.2 Statistics of sequences

- **Length of a sequence.** The length of a sequence u (denoted as $length(u)$) is the number of members in the sequence. Thus, $\langle S_1 S_2 \cdots S_m \rangle$ has length m . The event-count of a sequence is the sum of cardinalities of its members (if an event occurs multiple times in the members of a sequence, each occurrence contributes to the count). Thus, $\langle \{1, 2\} \{1, 2\} \{1, 2, 3\} \{8\} \rangle$ has event-count 8. We also refer to a sequence as an r -event-sequence if it has event-count r .
- **Transaction time of events.** For each event in an eventset, it is possible to record its accompanied transaction time.
- **Frequency of events.** Because an eventset is a multiset, an event may occur multiple times in an eventset. The number of occurrence of the event is used as a statistic indicating its frequency (transaction time of the event can also be added up accordingly). In some computations, it is simpler to ignore the notion of a sequence and just use the notion of an eventset (specially in cases where order is not important); and a sequence can be treated as a union set of its members. Further, if the notion of frequency is not important, an eventset is treated as a set (rather than a multiset).
- **Support of sequences.** For the purposes of mining association rules in sequences, a notion of support is necessary to filter frequent item sets as those that have at least minimum support. The support for a sequence u is defined as a fraction of total data sequences that “contain” this sequence. In the absence of sliding windows and time constraints, a data sequence u “contains” a sequence u' if u' is a subsequence of the data sequence [2].

4.1.3 Similarity measures of sequential data

Clustering consists of grouping together a heterogeneous set by some natural measure of similarity. An issue that lurks behind sequential data mining is to find natural

similarity measures [78, 183].

Similarity measures between eventsets

For two eventsets, S and S' , the following similarity measures between them are typically defined:

- **Access-based similarity measure [183].** Here, eventsets are treated as classical sets. Such a similarity measure is computed by counting the number of events that are common between the two eventsets and scaling the count so that the similarity is always between 0 and 1, resulting in the following measure:

$$Sim_{Access}(S, S') = \frac{|S \cap S'|}{\sqrt{|S| |S'|}}. \quad (4.1.1)$$

- **Frequency-based similarity measure [183].** Eventsets are treated as multisets, and the similarity is measured by counting the number of occurrences of events that are common between the two eventsets, and scaling the count so that the similarity is always between 0 and 1, resulting in the following measure:

$$Sim_{Frequency}(S, S') = \frac{\sum_j (f(s_j, S) \cdot f(s_j, S'))}{\sqrt{\sum_j (f(s_j, S))^2 \cdot \sum_j (f(s_j, S'))^2}}, \quad (4.1.2)$$

where $f(s_j, S)$ is the frequency of event s_j as it occurred in the eventset S .

- **Time-based similarity measure [183].** Eventsets are treated as multisets with this measure. The similarity is computed to take into account the actual transaction time spent on each event. In this case, the similarity between two eventsets can be expressed as:

$$Sim_{Time}(S, S') = \frac{\sum_j (t(s_j, S) \cdot t(s_j, S'))}{\sqrt{\sum_j (t(s_j, S))^2 \cdot \sum_j (t(s_j, S'))^2}}, \quad (4.1.3)$$

where $t(s_j, S)$ is the transaction time of event s_j when it occurred in the eventset S .

Similarity measures between sequences

Given two sequences, $u = \langle S_1 S_2 \cdots S_n \rangle$ and $u' = \langle S'_1 S'_2 \cdots S'_m \rangle$, their similarity can be measured by taking into account their elements with or without considering the order of these elements. It is typically computed in the following approaches:

- **Merge-based [183].** This approach does not consider the order of the members of given sequences. Rather, it converts each sequence to an eventset by merging all members of the sequence; then it computes the similarity between two sequences according to the converted eventsets. Let S and S' be the converted eventsets of sequences u and u' respectively, then the similarity between the two sequences can be defined as:

$$\text{Sim}(u, u') = \text{Sim}(S, S'). \quad (4.1.4)$$

- **Order-based [183].** In some applications, the transaction order of events plays an important role in similarity measures, and the notion of sequences must be considered. Let $u = \langle S_1 S_2 \cdots S_m \rangle$ be a given sequence. Define $u(q)$ as the set of q -hop sub-sequences ($q \leq m$) of u , namely, $u(q) = \{ \langle S_i S_{i+1} \cdots S_{i+q-1} \rangle \mid i = 1, 2, \dots, m - q + 1 \}$. The order-based similarity measure is defined as:

$$\text{Sim}(u, u') = \frac{\langle u, u' \rangle_l}{\sqrt{\langle u, u \rangle_l \cdot \langle u', u' \rangle_l}}, \quad (4.1.5)$$

where $l = \min(\text{length}(u), \text{length}(u'))$, and $\langle u, u' \rangle_l = \sum_{i=1}^l \sum_{s \in u(i) \cap u'(i)} \text{length}(s) \cdot \text{length}(s)$. Based on this definition, the similarity between two sequences will be 1 if they have the same members in the exact same order, and 0 if they have no common members at all.

- **Alignment-based [78].** The idea behind this approach is to align two sequences against each other so that they maximize the similarity between the two sequences. Let $u^{(\hat{A})}$ and $u'^{(\hat{A})}$ be two sequences of length l obtained after aligning u against u' , by inserting empty eventsets either inside, at the beginning, or at the ends of the two sequences, so that every eventset (including

empty) in either sequence is opposite a unique eventset in the other sequences. The score of such alignment $\dot{A}$ can be defined as

$$score(\dot{A}) = \sum_{i=1}^l Sim(S_i^{(\dot{A})}, S_i^{(\dot{A})}).$$

Let $\dot{A}^*$ be the optimal alignment of two sequences that maximize the alignment score $score(\dot{A}^*)$, the similarity between sequence can be computed as following:

$$Sim(u, u') = score(\dot{A}^*)/l. \quad (4.1.6)$$

In the following discussion, we suppose we are given a data set U , the data objects of which are modeled in eventsets or sequences.

4.2 Mining sequential patterns: tasks and challenges

The problem of discovering sequential patterns is to find inter-transaction patterns such that the presence of a set of events is followed by another in the time-stamp ordered transaction logs [35]. Despite the recent attention for mining sequential data, most work has concentrated on association rule extraction [1, 20, 131] and less attention has been given to clustering. Clustering is a central process inside data mining; it is a task of identifying groups in a data set by some natural criteria of similarity. Clustering of sequential data sets is useful for various purposes. For example, clustering of sequences from commercial data sets helps marketers identify different customer groups based upon their purchasing patterns, grouping protein sequences that share structure helps in identifying sequences with similar functionality [78]. However, working with conventional clustering methods designed for vector spaces faces some challenges. A vector space $\mathbb{R}^D$ demands that the information for an object be supplied as an attribute vector where each entry is a real-valued feature, and that these objects be distinguished by metrics like the Euclidean distance. This demands

the identification of D features, and the use of a distance in $\mathbb{R}^D$. This mapping to $\mathbb{R}^D$ allows the use of data structures and vector operations to create fast clustering methods. However, it raises the following modeling concerns.

- **Sequences are discrete structures.** The discrete nature of transaction sequences and dissimilarity measures remove vector-based operations, like averages (means). The continuous center of mass of a cluster, as an attribute vector, may not have the value of a prototypical object for that cluster. Thus, while it is possible to compute the average of two feature vectors, it is not clear that the result is the feature vector mapped by a sequence (a sequence with such a feature vector may actually be infeasible given the links between events).
- **Vector representation.** A straightforward way of achieving the feature space is to represent each data sequence as an D -dimensional vector of zeros and ones, with ones corresponding to all the features that are supported by the particular sequence. However, the pseudo-metric spaces defined by dissimilarity measures are different from the Euclidean spaces, since for all feature vectors $\vec{v}$, the similarity between $\vec{v}$ and itself is maximum. However, it is also maximum between $\vec{v}$ and $\lambda\vec{v}$, for all constants $\lambda > 0$. Not being able to treat attribute vectors numerically rules out algorithms like k -MEANS [110], EXPECTATION MAXIMIZATION [41], or its variants like fuzzy- c -means [81] or k -HARMONIC MEANS [187].
- **High dimensional feature space.** The feature space should be complete, in the sense that all features involved should be contained in the transformed space. More serious is the fact that for measuring similarity between vectors, the resulting dimension D may be very large. The common mapping to attribute vectors could be to transform a data sequence into a vector of usage of elements, frequency count of events, or the time spent on events. For the access similarity measure, the dimension of data sequences involved is the number of events in the sequences, typically a number higher than 10 and usually much larger. Considering frequency-feature, time-feature or order-feature of the

events in sequences, makes D at least proportional to the number of events in a set of transactions. Other more robust similarity measures are more costly to evaluate and imply feature vectors in even higher dimensions. Thus, for clustering sequential data, D is too large. This rules out the use of data structures like R -trees (the base of DBSCAN [51]) or CF-Trees (as in BIRCH [188]). Also, it may not be possible to use meaningfully statistical indicators (like vectorial means as in STING [178]).

4.3 Existing clustering algorithms for sequences

There are a number of clustering algorithms developed at sequential data. In our opinion, all previous attempts for clustering data sequences are ineffective. For sequential data, most methods tend to have a computational complexity of at least quadratic time on the number of sequences. The reason is that these methods typically require an initial analysis over all pairs of sequences [78]. An example of such an analysis is the construction of a similarity matrix as in matrix-based clustering methods.

4.3.1 Matrix-based clustering

Instead of projecting sequences into vector spaces, this clustering method attempts to work with similarities between sequences directly. This clustering approach maintains a similarity matrix of each pair of sequences, and faces the problem of infeasible computational resources. For the applications like clustering visitors of a Web site, similarity computation demands inspection of access logs for the patterns of visitation and comparisons that require time at least proportional to the pattern of visitation. Even if we assume that the similarity can be computed in constant time, this method needs to compute the similarity between all pairs of patterns, which implies that it will require at least quadratic space and quadratic time to get started [133, 156, 183, 186].

Matrix-based clustering of sequences can be modeled as a Longest Hamiltonian Path Problem [70]. Then, the preferred algorithm named Bond Energy Algorithm (BEA) [114] can be applied to this problem. BEA is a constructive approach that works on a non-negative $m \times n$ matrix A . Each cell in A stores a value of similarity between objects. The mechanism of BEA is guided by a measure of effectiveness e which reaches large values when the input matrix possesses dense clumps of numerically large elements when compared to the same matrix with large elements more uniformly distributed throughout the matrix [172]. Xiao et al. [183] propose to cluster a matrix which represents similarities between Web visitation patterns using a variant of BEA.

The computation of the similarity matrix makes this method far from scalable and its proponents report experiments with 500 sequences, or at most around 1,000 sequences. These sizes are not what is expected for transaction analysis where we have large and very large transaction logs to be clustered. In fact, Torres-Velázquez [171] has found that there are many implementations of the hill-climbing heuristic named BEA and typically they require quadratic time.

4.3.2 Pattern-based clustering

Morzy et al. [165] present a pattern-oriented algorithm to cluster data sequences. It works on the basis of discovering frequent patterns. The problem was introduced in [2] and then it was generalized in [162]. This algorithm employs the idea of agglomerative hierarchical clustering, which consists of merging pairs of similar clusters to form new large clusters. It can be decomposed into the following steps:

1. **Pattern extraction.** In this step, a set of frequent patterns will be extracted using the methods in [162]. The time required to discover frequent patterns depends linearly on the size n of a given data set.
2. **Initialization phase.** First, an initial set of clusters is created by mapping each frequent pattern into a cluster. The frequent patterns here are used as

cluster descriptions. Then, the algorithm distributes data sequences into the initial clusters by matching these data sequences with each cluster's description. Meanwhile an initial similarity matrix storing the values of inter-cluster similarity is completed. The time required for the computation in this phase is linearly on the size n of a given data set and the number m of frequent patterns.

3. **Cluster merging.** The algorithm merges together a pair of clusters according to their similarity values. Two most similar clusters are replaced by a new one.
4. **Similarity re-evaluation.** The algorithm re-evaluates the similarity values concerned with the newly created cluster.
5. **Loop.** The algorithm repeats steps 3 and 4 until the number of clusters reaches the required number, or there is no such pair of clusters whose similarity is greater than the minimal similarity threshold.

The time requirements for the loop processing to merge clusters and reevaluate similarities vary with the cube of the number m of frequent patterns [165] (the maximal possible number of iterations in the cluster merging phase is equal to m decreased by 1; in each iteration, the inter-cluster similarity matrix has to be scanned, where the initial size of the matrix is equal to m^2). When m is smaller, the algorithm will work efficiently. However, a smaller m may lower the clustering quality.

4.3.3 Clustering algorithms on vector spaces

It is possible to extend conventional clustering algorithms on vector spaces to the environment of arbitrary metric spaces. As we indicated, hierarchical clustering is ineffective in that it typically requires quadratic time in the size of the input [124]. Perkowitz and Etzioni report [133] that one algorithm from this family for Web clustering resulted in CPU-time requirements three orders of magnitude more than any other algorithm. Many clustering methods are built around the partitioning approach. Centroid-based techniques are suitable only for data in vector spaces in which it is

possible to compute the centroid for a given set of feature vectors. Because it is computationally hard to compute centroids in the space of data sequences, medoid-based approaches are better suited for clustering sequential data sets. On the other hand, medoids are a more robust estimator of location than means are [148]. However, medoids are less tractable. Han and Ng proposed an algorithm CLARANS for this optimization that is a randomized interchange hill-climber in order to obtain subquadratic algorithmic complexity. CLARANS can not guarantee local optimality. The best interchange hill-climber [60] is the Teitz and Bart heuristic [168] which requires quadratic time.

4.4 Representative clustering on metric spaces

In this section, we present generally applicable methods that are fast and robust with respect to an arbitrary measure of similarity (like those defined on sequential data). We first analyze requirements and design rules of our algorithms. Then, we explain how our algorithms generalize fuzzy membership for cluster assignments. Also, we prove that our clustering methods converge; we derive mathematically the worst-case complexity, and show they are sub-quadratic in n and linear in the computation of the dissimilarity.

4.4.1 Requirements of clustering algorithms for sequences

In Chapter 3, we discussed representative clustering on vector spaces systematically. We now propose methods to cluster data objects in arbitrary pseudo-metric spaces. Given a data set $U = \{u_1, u_2, \dots, u_n\}$ in a pseudo-metric space, Equation (3.2.1) can be rewritten for the problem of clustering the data set U :

$$\begin{aligned} \text{Minimize } M^a(C) &= \sum_{i=1}^n d(u_i, \text{REP}[u_i, C])^a \\ \text{subject to } C &\subset U, \end{aligned} \tag{4.4.1}$$

where $a \geq 1$ is a constant; $C \subset U$ is a set of k representatives, $\text{REP}[u_i, C]$ is the most similar representative (in C) to u_i ; $d(\cdot, \cdot)$ is a measure of dissimilarity. We underline that the constraint $C \subset U$ reflects the discrete nature of the data objects in U , and $d(\cdot, \cdot)$ does not need to be a distance satisfying the axioms of a metric. In particular, while $d(u_i, u_i) = 0$, we do not require that $d(u_i, u_j) = 0$ implies $u_i = u_j$. Also, we do not require the triangle inequality, but, we do expect symmetry, that is $d(u_i, u_j) = d(u_j, u_i)$. Note that, the similarity functions $\text{Sim}(\cdot, \cdot)$ introduced in Section 4.1.3 have a range in $[0, 1]$ and the corresponding dissimilarities are $d(\cdot, \cdot) = 1 - \text{Sim}(\cdot, \cdot)$, also in the range $[0, 1]$. The type of clustering that we find here attempts to detect the partition that optimizes a given homogeneity criterion defined in terms of distances. Optimization criteria in Equation (4.4.1) attempts to minimize the heterogeneity in each group with respect to the group representative.

To design general applicable methods working on arbitrary pseudo-metric spaces, we take into account two basic requirements. First, the clustering algorithms must be scalable to both the size n of objects to cluster and the computational requirements of the similarity measure. Second, the clustering results must be of high quality in the presence of noise and outliers.

Efficient clustering is a fundamental task in data mining where the goal is to discover patterns with large data sets that are also high dimensional. An exhaustive enumeration of all possible partitions to find the optimal one is infeasible. So our algorithms adopt an iterative optimization paradigm as found in k -MEANS, EXPECTATION MAXIMIZATION, FUZZY- c -MEANS and k -HARMONIC MEANS (refer to Figure 4.1). The iteration alternates classification of data from a current model (classification step) and model refinement from classified data (reconstruction step), where the algorithms call on the computation of the dissimilarity only when they need it.

For the purpose of high quality clustering, instead of using means, we use medoids to model a cluster structure; that is, the estimator of location for a cluster shall be a member of the data. This not only makes the clustering results robust, but also reflects the discrete nature of sequential data. For further improvement, we minimize

```

ITERATIVE_STEP( $C = \{c_1, \dots, c_k\} \subset U$ )
  CLASSIFICATION_STEP( $C$ )
    For  $j = 1, \dots, k$ : find  $U_j = \{u_i \in U \mid d(u_i, c_j) \leq d(u_i, c_{j'}) \mid j' = 1, \dots, k\}$ 
  new  $C \leftarrow \text{RECONSTRUCTION\_STEP}$ 
    For  $j = 1, \dots, k$ : new  $c_j \leftarrow$  new estimator of median for  $U_j$ 

```

Figure 4.1: Body of the iteration alternating between finding a classification for the data given a model, and finding a model given classified data.

the absolute loss function, rather than the sum of squared dissimilarities. In fact, the squaring seems to be introduced historically in order to allow differentiation of the objective function. Although the squaring methods are the result of numerical optimization from the iteration of the necessary conditions for optimality in differentiation, not all of them have been proven to converge (and those for which such a proof exists, it usually involves rather daunting mathematics).

However, the filtering mechanism proposed in Chapter 3 can not be applicable, since it works only on vector spaces. Recently, several variants of clustering algorithms for optimizing instances of the criteria in absolute error estimation have emerged by using randomization [54]. These variants are sub-quadratic robust clustering algorithms. These randomized algorithms have been shown mathematically and empirically to provide robust clustering results [54]. This is because randomization is not sampling. Sampling reduces the CPU requirements by using a very small part of the data, and the accuracy suffers directly with the size of the sample. Randomization uses all the data available.

4.4.2 Design of clustering algorithms for sequences

Before we go through the implementation issues and the analysis of our algorithms, we present a brief description of the design of our representative-based algorithms.

The algorithms will be implemented in two versions, namely, the crisp and non-crisp classifications. The next section discusses randomized strategies to implement these medoid-based algorithms.

Crisp classification

Crisp classification computes the representative $\text{REP}[u_i, C]$ of each data object u_i and each data object belongs to only the cluster of its representative. It requires a simple pass through the data and $O(nk)$ computations of $d(\cdot, \cdot)$. These result in a temporary partition of U into k clusters $U_1, \dots, U_k$. We can generalize crisp classification by assigning a vector $\vec{\pi}_i \in [0, 1]^k$ to each u_i . We consider that the j -th entry $\vec{\pi}_{ij}$ denotes the degree by which u_i belongs to the j -th cluster. In parallel with EXPECTATION MAXIMIZATION and FUZZY- c -MEANS, we will normalize the values so that $\sum_{j=1}^k \vec{\pi}_{ij} = 1$. Thus, crisp classification means the closest representative has its entry in $\vec{\pi}$ set to 1 while all other clusters have their entry set to zero.

Non-crisp classification

It seems that robustness to initialization (as it happens in k -HARMONIC MEANS [187]) comes from a combining relaxation of crisp classification with a boosting technique. To achieve this, each data object should be able to belong to different clusters with a degree of membership (the degree could be a probability as in EXPECTATION MAXIMIZATION, or a fuzzy membership as in FUZZY- c -MEANS). Non-crisp classification will mean that in $\vec{\pi}$ more than one cluster has its entries different from zero.

To detail more our approach we need to introduce some notation. Given a vector $\vec{x} \in \mathbb{R}^k$, let $\text{SORT}(\vec{x}) \in \mathbb{R}^k$ denote the vector of sorted entries from $\vec{x}$ in non-decreasing order (entries with equal values are arranged arbitrarily). Thus, if $j < j'$ then $\text{SORT}(\vec{x})_j \leq \text{SORT}(\vec{x})_{j'}$. We use the notation e_j^T to denote the j -th canonical vector $(0, \dots, 0, 1, 0, \dots, 0)$ that has only the j -th entry equal to 1 and all others

equal to zero. This notation allows us to rewrite the loss function in Equation (4.4.1) as $M^a(C) = \sum_{i=1}^n e_1^T \cdot \text{SORT}(d(u_i, c_1), d(u_i, c_2), \dots, d(u_i, c_k))$, because the minimum operator in $\text{REP}[u_i, C]$ can be replaced by $e_1^T \cdot \text{SORT}(\cdot)$. Moreover, we can also describe a harmonic set of weights as the degrees of membership. Let $K = \sum_{j=1}^k 1/j$, the harmonic loss function is then $M^H(C) = \frac{1}{K} \sum_{i=1}^n (1, 1/2, \dots, 1/j, \dots, 1/k)^T \cdot \text{SORT}(d(u_i, c_1), d(u_i, c_2), \dots, d(u_i, c_k))$. Moreover, we propose here an even more general approach, and let $\omega \in \Re^k$ be any vector with non-negative entries and entries in descending order; then, the non-crisp classification loss function with respect to ω is

$$M^\omega(C) = \frac{1}{\|\omega\|_1} \sum_{i=1}^n \omega^T \cdot \text{SORT}(d(u_i, c_1), d(u_i, c_2), \dots, d(u_i, c_k)). \quad (4.4.2)$$

(where we denote the 1-norm of a vector $\vec{x}$ as $\|\vec{x}\|_1$ and it equals $\sum_{j=1}^k |\vec{x}_j|$). Note that the first canonical vector e_1 and the Harmonic vector with $1/j$ in its j -th entry are special cases for ω . The vector $\vec{\pi}_i$ giving the degree of membership of u_i is defined by $\pi_{ij} = d(u_i, c_j) \omega_j / \|\omega\|_1$.

4.5 Implementations and analyses of randomized algorithms

In this section, we take the route of randomized strategies to implement the algorithms we have designed. We produce here a general family of algorithms that optimize Equation (4.4.2) and are more robust to initialization. We also analyze their convergence properties and other advances. Namely:

1. We present these algorithms requiring sub-quadratic time by using randomization.
2. We show our algorithms are generic (the degree of membership could be a fuzzy-membership function, a harmonic distribution, or even revert to the case of crisp-membership).

3. We probe mathematically that they converge.

4.5.1 The Expectation Maximization type

Our first family of randomized algorithms carries out an iterative improvement as found in k -MEANS, EXPECTATION MAXIMIZATION, FUZZY- c -MEANS and k -HARMONIC MEANS. We say that it is an EXPECTATION MAXIMIZATION type because it follows the alternation between Expectation and Maximization. That is, Expectation because in the statistical sense we estimate the hidden information (the membership to clusters of the data objects). Maximization, because we use some criteria (like Maximum Likelihood) to revise the description (model) of each cluster.

Our task now is to describe the iterative algorithms that minimize the crisp and non-crisp classification loss function with respect to a vector ω .

Implementations of the crisp version

To implement this algorithm, we must describe how a new representative c_j^{t+1} is computed for each subset $U_j^t \subset U$. The algorithm we use is a randomized approximation of the discrete median that has complexity $O(\|U_j^t\| \sqrt{\|U_j^t\|})$ [54]. For this subproblem of approximating the discrete median we denote with $P = \{p_1, \dots, p_n\}$ the set of objects (during the t -th iteration of the algorithm, when applying it to the j -th cluster, P is actually U_j^t). We also denote by $\text{OLD_MED}(P)$ the previous approximation to the discrete median of P (during the t -th iteration, $\text{OLD_MED}(P)$ is actually c_j^t).

First, recall that the discrete median of a set $P = \{p_1, \dots, p_n\}$ is the object $u = \text{med}.d(P) \in P$ such that $M(u, P) = \sum_{i=1}^n d(u, p_i)$ is minimum. Clearly, the discrete median $\text{med}.d(P)$ can be computed in $O(\|P\|^2)$ computations of the dissimilarity $d(\cdot, \cdot)$ by simply computing $M(u, P)$ for $u = p_1, \dots, u = p_n$ and returning the u that results in the smallest value. We will refer to this algorithm as EXHAUSTIVE. It

must be used carefully because it has quadratic complexity on the size $\|P\| = n$ of the data. However, it has linear complexity of $\phi(d)$, the time to compute $d(\cdot, \cdot)$. The randomized algorithm works as follows:

The first step consists of obtaining a random partition of P into approximately $r = \sqrt{n}$ subsets $P_1, \dots, P_r$ each of approximately $n/r \approx \sqrt{n}$ objects. Then, algorithm EXHAUSTIVE is applied to each of these subsets to obtain $m_i = \text{med}_d(P_i)$, $i = 1, \dots, r$. These r objects constitute candidates for the median of P . We compute $M(m_i, P)$ for $i = 1, \dots, r$ and also $M(\text{OLD_MED}(P), P)$. The object that provides the smallest amongst these (at most $r + 1$) objects is returned as the new approximation to the discrete median. The algorithm has complexity $O(\phi(d)\|P\|\sqrt{\|P\|})$ because EXHAUSTIVE is applied to $\Theta(\sqrt{\|P\|})$ sets, each of size $\Theta(\sqrt{\|P\|})$; thus this requires $O(\phi(d)\|P\|\sqrt{\|P\|})$ time. Finally, $M(m_i, P)$ requires $O(\phi(d)\|P\|)$ time and is performed $O(\sqrt{\|P\|})$ times. This is also $O(\phi(d)\|P\|\sqrt{\|P\|})$ time. These algorithms have been shown mathematically and empirically to provide robust estimators of location [54]. This is because randomization is not sampling.

Analyses of convergence properties of the crisp version

We enhance the fundamental results of iterative clustering algorithms by proving that our algorithm converges. We prove this by showing that both steps, the classification step and the reconstruction step, never increase the value of the objective function in Equation (4.4.2).

Lemma 4.5.1.1 *Let $U_i^t, \dots, U_k^t$ be the partition of U after the classification step in the t -th iteration of the algorithm. Let $M(U_i^t, \dots, U_k^t) = \sum_{j=1}^k \sum_{u_i \in U_j^t} d(u_i, c_j^t)$.*

Then, the value of $\sum_{j=1}^k \sum_{u_i \in U_j^t} d(u_i, c_j^{t+1})$ of the objective function after the reconstruction step is no larger than $M(U_i^t, \dots, U_k^t)$.

Proof: Since for each j , we have $\sum_{u_i \in U_j^t} d(u_i, c_j^t) = M(c_j^t, U_j^t)$, and the randomized algorithm for approximating the discrete median considers the old median estimate c_j^t amongst other $\sqrt{\|U_j^t\|}$ candidates, the new median estimate results in a smaller sum $\sum_{u_i \in U_j^t} d(u_i, c_j^{t+1})$ $\square$

Lemma 4.5.1.2 *The value $M(C^{t+1} = \{c_1^{t+1}, \dots, c_k^{t+1}\}) = \sum_{i=1}^n d(u_i, \text{rep}[u_i, C^{t+1}])$ after a classification step is no larger than the value $\sum_{j=1}^k \sum_{u_i \in U_j^t} d(u_i, c_j^{t+1})$ before such classification step.*

Proof: Note that for data object u_i , its contribution to the sum before the classification step is $d(u_i, c_j^{t+1})$. The contribution of u_i to $M(C)$ after the classification step is $d(u_i, \text{rep}[u_i, C^{t+1}])$. But $\text{rep}[u_i, C^{t+1}]$ is the object in C^{t+1} that produces the smallest value of $d(u_i, c_{j'}^{t+1})$ for $j' = 1, \dots, k$. Thus, $d(u_i, \text{rep}[u_i, C^{t+1}]) \leq d(u_i, c_j^{t+1})$, from which the claim follows. $\square$

Theorem 4.5.1.1 *Our algorithm converges.*

Proof: The domain of the objective function has size $\binom{k}{n}$, since it consists of all subsets of size k of U . Thus, the objective function has a finite range. The algorithm can not decrease the value of the objective function continuously. $\square$

This result is in contrast to the problem of the continuous center in dimensions $D \geq 2$ and the Euclidean metric (the continuous Fermat-Weber problem) where fundamental results show that it is impossible to obtain an algorithm to converge [8] (numerical algorithms usually halt because of the finite precision of digital computers).

Implementations of the non-crisp version

The algorithm has the generic structure of iterative algorithms. It will start with a random set C^0 . The t -th iteration proceeds as follows. First we note that non-crisp classification is computationally as costly as classification in the crisp algorithms. It essentially requires us to compute SORT of the vector of distances of the data object u_i under consideration to each member of the current set of representatives C^t . This requires computations of $d(\cdot, \cdot)$ for the k representatives. Although sorting of k objects requires $k \log k$ comparisons, it does not require any more dissimilarity computations and in the applications we have in mind, the computations of dissimilarity values is far more costly than the time required to sort the k values. Thus, classification is performed by a simple pass through the data and $O(nk)$ dissimilarity computations. This results in a labeling of all data objects in U by a vector ranking the degrees of membership to the k clusters. That is, for each u_i , we record the rank of $d(u_i, c_j^t)$, where c_j^t is the j -th current representative. We let $\text{RANK}_i^t[j]$ denote the rank of u_i with respect to the j -th representative at the t -th classification. For example, if the first current representative is the 3rd nearest representative to u_i then $\text{RANK}_i^t[1] = 3$.

The reconstruction step computes new representatives. For each old representative, we have a degree of membership of every data object. Thus, for each $j = 1, \dots, k$, we seek a new representative that minimizes $f_j^t(u) = \sum_{i=1}^n e_{\text{RANK}_i^t[j]}^T \cdot \omega d(u, u_i) = \sum_{i=1}^n \omega_{\text{RANK}_i^t[j]} d(u, u_i)$. Thus, for example, if an object u_i was ranked first with respect to the j -th representative because it was its nearest representative, then the distance $d(u, u_i)$ in Equation 4.5.1 is multiplied by the largest weight in ω (which is the value in the first entry). Also, note that in the case of crisp classification, all weights are zero except the largest (and the largest weight equals 1). Thus, the minimization of $f_j^t(u)$ above just corresponds to finding the median amongst the data objects in the j -th cluster. Because of this, the data object u_i such that $f_j^t(u_i)$ is smallest will be called the j -th discrete weighted median.

To detail our algorithm further, we must describe how a new representative c_j^{t+1} is

computed by minimization of $f^t(u)$ amongst the data objects u_i . The algorithm we introduce for this task is a randomized approximation inspired in a sub-quadratic randomized algorithm for computation of the discrete median [54]. For this subproblem of approximating the discrete median we note that $U = \{u_1, \dots, u_n\}$ is the set of candidates (during the t -th iteration of the algorithm) for finding the minimum of each function $f_j^t(u)$, for $j = 1, \dots, k$.

However, for simplicity of the notation, in what follows we drop the super-index t , since it is understood we are dealing with the current iteration. We will also assume that we know which representative is being revised and we denote by $\text{OLD_MED}(j)$ the previous approximation to the data object that minimizes $f_j(u)$ (during the t -th iteration, $\text{OLD_MED}(j)$ is actually c_j^t).

Clearly, the discrete weighted median $\text{MED}(j)$ can be computed in $O(\|U\|^2)$ computations of the dissimilarity $d(\cdot, \cdot)$ by simply computing $f(u)$ for $u = u_1, \dots, u = u_n$ and returning the u that results in the smallest value. We will also refer to this algorithm as **EXHAUSTIVE**. Again, it must be used carefully because it has quadratic complexity on the size $\|U\| = n$ of the data. However, it has linear complexity of $\phi(d)$, the time to compute $d(\cdot, \cdot)$. Thus, our use of randomization.

The first step consists of obtaining a random partition of U into approximately $r = \sqrt{n}$ subsets $U_1, \dots, U_r$ each of approximately $n/r \approx \sqrt{n}$ objects. Then, algorithm **EXHAUSTIVE** is applied to each of these subsets to obtain $m(j)_s = \text{med}_d(U_s)$, $s = 1, \dots, r$. These r objects constitute candidates for the j -th discrete weighted median of U . We compute $f_j(m(j)_s)$ for $s = 1, \dots, r$ and also $f(\text{OLD_MED}(j))$. The object that provides the smallest amongst these (at most $r + 1$) objects is returned as the new approximation to the discrete median. The algorithm has complexity $O(\phi(d)\|U\|\sqrt{\|U\|})$ because **EXHAUSTIVE** is applied to $\Theta(\sqrt{\|U\|})$ sets, each of size $\Theta(\sqrt{\|U\|})$; thus this requires $O(\phi(d)\|U\|\sqrt{\|U\|})$ time. Finally, $f_j(m(j)_s)$ requires $O(\phi(d)\|U\|)$ time and is performed $O(\sqrt{\|U\|})$ times. This is also $O(\phi(d)\|U\|\sqrt{\|U\|})$ time.

Analyses of convergence properties of the non-crisp version

We prove this by showing that both steps, the non-crisp classification step and the reconstruction step never increase the value of the objective function in Equation (4.4.2).

Lemma 4.5.1.3 *Let RANK_i^t be the rank vector for each $u_i \in U$ ($i = 1, \dots, n$) after the non-crisp classification step in the t -th iteration of the algorithm. Let*

$$M^\omega(C^t) = \frac{1}{\|\omega\|_1} \sum_{i=1}^n \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(u_i, c_j^t). \quad (4.5.1)$$

Then, the value of $\frac{1}{\|\omega\|_1} \sum_{i=1}^n \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(u_i, c_j^{t+1})$. of the objective function after the reconstruction step is no larger than $M^\omega(C^t)$.

Proof: First note that, by expanding the dot product $\omega^T \text{SORT}(\cdot)$ and using the RANK vector we have shown that Equation (4.4.2) and Equation (4.5.1) are the same. Then, we can reverse the order of the summation signs and also note that $1/\|\omega\|_1$ is a constant. Thus, the objective function is simply $M^\omega(C^t) = \frac{1}{\|\omega\|_1} \sum_{j=1}^k f_j^t(c_j^t)$. The reconstruction step finds new c_j^{t+1} such that $f_j^t(c_j^{t+1}) \leq f_j^t(c_j^t)$, for $j = 1, \dots, k$ (because the previous discrete median is considered among the candidates for a new weighted discrete median). Thus,

$$M^\omega(C^t) \geq \frac{1}{\|\omega\|_1} \sum_{i=1}^n \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(u_i, c_j^{t+1})$$

as required. □

Lemma 4.5.1.4 *The value*

$$\begin{aligned} & M^\omega(C^{t+1} = \{c_1^{t+1}, \dots, c_k^{t+1}\}) \\ &= \frac{1}{\|\omega\|_1} \sum_{i=1}^n \omega^T \text{SORT}(d(u_i, c_1^{t+1}), d(u_i, c_2^{t+1}), \dots, d(u_i, c_k^{t+1})) \end{aligned}$$

after a classification step is no larger than the value

$$\frac{1}{\|\omega\|_1} \sum_{i=1}^n \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(u_i, c_j^{t+1})$$

resulting in the previous reconstruction step.

Proof: Note that

$$\frac{1}{\|\omega\|_1} \sum_{j=1}^k f_j^t(c_j^{t+1}) = \sum_{i=1}^n \frac{1}{\|\omega\|_1} \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(c_j^{t+1}, u_i).$$

Thus, we can say that the contribution of u_i to the objective function before the next classification is $\frac{1}{\|\omega\|_1} \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(c_j^{t+1}, u_i)$. Now, we know that after classification, this contribution by u_i is

$$\frac{1}{\|\omega\|_1} \omega^{\text{T SORT}}(d(u_i, c_1^{t+1}), d(u_i, c_2^{t+1}), \dots, d(u_i, c_k^{t+1}),).$$

Thus, the terms $d(u_i, c_j^{t+1})$ involved before and after non-crisp classification are the same. It is just that after classification they are in non-decreasing order. Since ω^{T} has entries in descending order we claim that

$$\omega^{\text{T SORT}}(d(u_i, c_1^{t+1}), d(u_i, c_2^{t+1}), \dots, d(u_i, c_k^{t+1}),) \leq \sum_{j=1}^k \omega_{\text{RANK}_i^t[j]} d(c_j^{t+1}, u_i)$$

whatever the order of the set $\{d(u_i, c_1^{t+1}), d(u_i, c_2^{t+1}), \dots, d(u_i, c_k^{t+1}), \}$ given by the permutation encoded in $\text{RANK}_i^t[j]$ (note that this rank is the one that resulted in the previous classification, and we are about to perform the $(t+1)$ -th classification).

We prove this claim by showing that if $j < j'$ and $d(u_i, c_j^{t+1}) > d(u_i, c_{j'}^{t+1})$, but $d(u_i, c_j^{t+1})$ is ranked earlier than $d(u_i, c_{j'}^{t+1})$ in the t -th ranking, then we can reduce the value of the contribution of u_i by swapping the order in the permutation of j and j' .

This is because if $j < j'$ we have $\omega_j > \omega_{j'}$ and $d(u_i, c_j^{t+1}) > d(u_i, c_{j'}^{t+1})$ implies

$$\omega_j(d(u_i, c_j^{t+1}) - d(u_i, c_{j'}^{t+1})) > \omega_{j'}(d(u_i, c_j^{t+1}) - d(u_i, c_{j'}^{t+1})).$$

Thus,

$$\omega_j d(u_i, c_j^{t+1}) + \omega_{j'} d(u_i, c_{j'}^{t+1}) > \omega_{j'} d(u_i, c_j^{t+1}) + \omega_j d(u_i, c_{j'}^{t+1}).$$

Thus, swapping j and j' reduces the contribution of u_i . This proves that the sorted array of values reduces the contribution of u_i , and this is for all u_i . Thus, the new non-crisp classification produces a ranking that can not increase the value of the objective function. $\square$

Theorem 4.5.1.2 *Our algorithm converges.*

Proof: The domain of the objective function has size $\binom{k}{n}$, since it consists of all subsets of size k of U . Thus, the objective function has a finite range. The algorithm can not decrease the value of the objective function continuously. $\square$

Again, this result is in contrast to the problem of the continuous center in dimensions $D \geq 2$ and the Euclidean metric where fundamental results show that it is impossible to obtain an algorithm to converge [8]. Other algorithms of for non-crisp classification, like Harmonic- k -means have not been shown to converge.

4.5.2 The discrete hill-climber type

We now present an alternative algorithm to optimizing Equation (4.4.2). The algorithm here can be composed with the algorithm in the previous section (for example, the first can be the initialization of the latter). The result is an even more robust algorithm, still with complexity $O(n\sqrt{n})$ similarity computations.

The algorithm in this section is an interchange heuristics based on a hill-climbing search strategy. However, we need to adapt this to non-crisp classification since all previous versions [42, 53, 60, 88, 121, 122, 126, 168] are for the crisp classification case.

We will first present a quadratic-time version of our algorithm, which we will name non-crisp TAB in honor of Teitz and Bart [168] and their original heuristic. Later, we will use randomization to achieve subquadratic time complexity, as in the previous section.

Our algorithms explore the space of subsets $C \subset U$ of size k . Non-crisp TAB will start with a random set C^0 . Then, the data objects $u_i \in U$ are organized in a circular list. Whenever the turn belonging to a data object u_i comes up, if $u_i \notin C^t$, it is used to test at most k subsets $C_j = (C^t \cup \{u_i\}) \setminus \{c_j^t\}$. That is, if u_i is currently not a representative (medoid), it is swapped with each of the k current medoids. The objective function M^ω in Equation (4.4.2) is evaluated in $M^\omega(C_j)$, for $j = 1, \dots, k$ and if any of these values is less than the current $M^\omega(C^t)$, then the swap with the best improvement $M^\omega(C_{j_0})$ is accepted and $C^{t+1} = C_{j_0}$. In this case, or if $u_i \in C^t$ or if no C_j improves C^t , the data object u_i is placed at the end of the circular list. The turn passes to the next data object and the algorithm halts when a pass through the circular list produces no swap.

The algorithm requires $O(kn^2)$ dissimilarity evaluations because evaluating $M^\omega(C)$ requires $O(n)$ distance evaluations and at least one pass is made through the circular list to halt.

Our randomized TAB also partitions U into approximately $r = \sqrt{n}$ subsets $U_1, \dots, U_r$ each of approximately $n/r \approx \sqrt{n}$ objects. The non-crisp TAB is applied to each of $U_1, \dots, U_r$. Thus each U_i is clustered into k groups. This requires $O(kn\sqrt{n})$ distance calculations and results in r sets of k representatives. Then, all the resulting rk representatives are placed in a circular list. Then non-crisp TAB is applied in this list but the evaluation of $M^\omega(C)$ is performed with respect to the entire data set U . Since the circular list has length $k\sqrt{n}$, the integration achieved by the last execution of non-crisp TAB requires $O(k^2n\sqrt{n})$. If having k^2 in the complexity is a problem, then we can simply choose $r = \sqrt{n}/k$, and obtain an algorithm with linear complexity in k as well.

Clearly, this algorithm also converges.

4.6 Remarks

In this chapter, we have discussed sequential data mining problem. We have provided a generic framework of medoid-based, sub-quadratic algorithms on arbitrary metric spaces, where the objects are not necessarily feature vectors. These algorithm can thus be applied to cluster sequences. Because our EM-methods are generic on the vector ω , they offer a range of diversity for the computational requirements in this regard. For example, one can imagine a k' -nearest neighbor classification with $k' < k$. The vector ω would have entries equal to zero from the $(k' + 1)$ -th entry onwards. The nonzero entries can then be a harmonic average of the k' nearest neighbors or some other combination. Thus, this is a classification that incorporates the supervised-learning technique of nearest-neighbors [34] and reduces smoothly to crisp-classification as k' is closer to 1. In parallel to k -MEANS, EXPECTATION MAXIMIZATION, and k -HARMONIC MEANS, our harmonic EM type algorithm may place two representatives very close together attracted by the same peak in frequency. However, the theoretical foundation provided here allows us also to detect this and apply the “boosting” techniques as suggested for k -HARMONIC MEANS [187]. We also point out there is a possibility of hybridization between the discrete hill-climber type and the EM type. Later in Chapter 7, we evaluate the performance of these algorithms in Web usage mining.

Chapter 5

Support Vector Clustering

Data can be modeled in various prototypes to carry out clustering analysis. From a statistical perspective, knowledge of extreme values (also called support vectors) may be useful in data modeling. An example is building a data model to detect events which are outside our expected data extremes. Different from representative-based models discussed in previous chapters, where the central information is used to encode a cluster structure, the prototypes with extreme values make use of boundary information to describe a data set. In this chapter, we present the working principles of data description. Based on this, we discuss data modeling techniques in terms of support vectors. We investigate the effective enhancement of data modeling through proximity graphs. We propose a new clustering algorithm by combining the support vector modeling and proximity graph modeling.

5.1 Density estimation and domain description

Density is a natural principle in data clustering. Clearly, knowledge of density would allow us to solve whatever problem can be solved on the basis of data [154]. Density-based clustering is a direct application of density estimation. Another density-driven

application is data domain description [142,167], where the task is to give a description of a set of data objects. This description should be able to cover the class of given objects within a data set, and ideally reject all other possible objects in the object space. A probability density of the available data is first estimated, and new test objects that are under some density threshold will be rejected. Thus, the domain description tasks can be characterized as estimating functions of the data which tell us something interesting about the underlying distribution.

Note that the objectives of data domain description vary with the situations of data sets. When an underlying statistical law for the domain is assumed, this underlying distribution should be estimated. It is necessary to find a real-valued function for estimating the densities. When nothing about the domain knowledge can be assumed (or an insufficient domain knowledge is available), only the *novelty detection* of the target data can be made. Novelty detection [19,144,145,148] refers to the detection of objects that differ significantly in some sense from the rest of the data set. In the case of novelty detection, we need only to compute a binary function, which is supposed to capture regions in the input data space where the probability density lives. More accurately, most of the data will live in the region where the function is non-zero (its support). Thus, novelty detection is applicable in a situation where the density of data distribution is not well-defined, and this is the typical case in domain description.

The functions for novelty detection can be naturally formalized in terms of extreme vectors. This favors using Support Vector Machines (SVMs) for novelty detection, since SVMs are able to effectively extract extreme vectors (support vectors) in boundary regions.

5.2 Novelty detection using Support Vector Machines

The foundations of Support Vector Machines (SVMs) were developed by Vapnik [173]. They are gaining popularity because of many attractive features, and promising empirical performance [75]. SVMs are systematic, reproducible, and motivated by statistical learning theory. In addition to their accuracy, a key characteristic of SVMs is their mathematical tractability and geometric interpretation. SVMs exhibit desirable properties, such as invariance under symmetries and robustness in the presence of outliers [11, 13, 173]. The training formulation embodies optimization of a convex cost function, so all local minima are the global minimum in the learning process [13]. SVMs and related kernel methods provide good generalization performance on machine learning tasks without incorporating problem domain knowledge; they have been successfully applied to basic classification tasks and extended to handle regression, operator inversion, as well as novelty detection. In this section, we present the fundamentals of SVMs and focus on Support Vector based Novelty Detection (SVND), which is the basis of Support Vector Clustering (SVC). In the next chapter, we concentrate in support vector modeling for classification tasks, which forms the basis of our cluster validity approach.

5.2.1 Support Vector based Novelty Detection (SVND)

SVND [151, 167] is an unsupervised learning task using SVMs. Novelty detection inspired by SVMs overcomes the drawbacks of other density-based novelty detection methods. For example, Tax [167] reports that the method of Ritter [142] can suffer from large variance in data density. One approach to SVND is constructing a hypersphere with minimum volume (or minimum radius) containing most of the data objects. Novel instances are then identified as those that lie outside the boundary of this hypersphere. This modeling corresponds to the so-called “one-class” problem,

and eventually results in a binary descriptive function in terms of a set of Support Vectors (SVs). SVs are extreme vectors lying on boundaries, where the density of data changes sharply.

5.2.2 Formalization of SVND using kernel methods

More importantly, SVND implements kernel methods to transform data into a feature space [153]. Assume we are given a set of n data points $\{\vec{x}_i\} \subseteq X$, with $X \subseteq \mathbb{R}^D$, being the data space. To formulate a support vector description of this data set, a nonlinear mapping ϕ is employed to map X into some high dimensional feature space. The next step is to find the smallest enclosing hypersphere:

$$\|\phi(\vec{x}_i) - \vec{a}\|^2 \leq R^2 + \xi_i \quad (\xi_i \geq 0) \quad \forall i, \quad (5.2.1)$$

where R is the radius, $\vec{a}$ is the center and ξ_i are some slack variables allowing for soft boundaries (some data points can be allowed to lie outside the sphere). The problem (5.2.1) is usually solved in its dual by introducing the Lagrangian and a regularization constant C in the penalty term,

$$\begin{aligned} M(R, \vec{a}, \alpha_i, \xi_i) = & R^2 - \sum_i (R^2 + \xi_i - \|\phi(\vec{x}_i) - \vec{a}\|^2) \alpha_i \\ & - \sum \xi_i \mu_i + C \sum \xi_i. \end{aligned} \quad (5.2.2)$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are Lagrangian multipliers, and $C \sum \xi_i$ is a penalty term. Also the Karush-Kuhn-Tucker condition allows the problem to be rewritten as

$$\begin{aligned} \text{Maximize } M = & \sum_i \alpha_i \phi(\vec{x}_i)^2 - \sum_{i,j} \alpha_i \alpha_j \phi(\vec{x}_i) \phi(\vec{x}_j) \\ \text{subject to } & 0 \leq \alpha_i \leq C, \sum \alpha_i = 1, i = 1, \dots, n. \end{aligned} \quad (5.2.3)$$

Following the SVM method, we use a *kernel* representation $k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$. Equation (5.2.3) is now written as:

$$\begin{aligned} \text{Maximize } M = & \sum_i \alpha_i k(\vec{x}_i, \vec{x}_i) - \sum_{i,j} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) \\ \text{subject to } & 0 \leq \alpha_i \leq C, \sum \alpha_i = 1, i = 1, \dots, n. \end{aligned} \quad (5.2.4)$$

The Lagrangian multipliers α_i can be obtained by optimizing Equation (5.2.4). Only those points with non-zero α_i satisfy Equation (6.3.1) with equality. These points lie outside or on the boundary of the hypersphere. All these are called *Support Vectors* (SVs), hence the name “Support Vector Machines”. Points with $\alpha_i = C$ have hit the upper bound for the radius and lie outside the sphere. These points are called *Bounded Support Vectors* (BSVs), and are treated as noise. In conceptual terms, the SVs are those data points that lie closest to the decision surface (hypersphere) and are the most difficult for novelty detection. As such, they have a direct bearing on the optimum location of the decision surface [84], and they play a prominent role in the operation of this class of learning machines.

Support vectors can be used to describe the hypersphere in feature space. For each point $\vec{x}$, the distance of its image $\phi(\vec{x})$ to the center of the hypersphere is given by

$$R^2(\vec{x}) = k(\vec{x}, \vec{x}) - 2 \sum_i \alpha_i k(\vec{x}, \vec{x}_i) + \sum_{i,j} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j). \quad (5.2.5)$$

In a practical implementation, the radius R of the hypersphere can be approximated by calculating the radii $R(\vec{x}_i)$ for all support vectors $\vec{x}_i$. Then the average of these radii can be used.

$$R \approx \frac{\sum R(\vec{x}_i)}{\|\{\vec{x}_i \mid \vec{x}_i \text{ is a support vector}\}\|}. \quad (5.2.6)$$

One of the key features of kernel methods is that they do not require an explicit calculation of the feature map ϕ but only use the values of the dot products between mapped patterns. That is, SVMs get around this issue through the use of *kernels*:

$$k(\vec{x}, \vec{x}') = \phi(\vec{x})^T \cdot \phi(\vec{x}') \quad (5.2.7)$$

SVMs can realize polynomial, multi-layer perceptron classifiers and radial basis function (RBF). Typical kernel functions are shown in Table. 5.1.

For clustering purposes, researchers [11,12] usually use the Gaussian kernels $k_q(\vec{x}_i, \vec{x}_j) = e^{q\|\vec{x}_i - \vec{x}_j\|^2}$ with width parameter $q = -1/(2\sigma^2)$. According to Tax et al. [167], polynomial kernels do not yield tight boundaries, while the Gaussian kernels yield an

Table 5.1: Types of kernel functions.

Summarized Types		Detailed Types	
Kernel Functions	Expression	Kernel Functions	Expression
Inner Product	$f(\vec{x}^T \cdot \vec{x}')$	Polynomial	$(\vec{x}^T \cdot \vec{x}' + 1)^p, p = 1, 2, \dots$
		Sigmoid	$\tanh(\alpha_0 \vec{x}^T \cdot \vec{x}' + \alpha_1), \alpha_0$ and α_1 are decided by the user
Radial	$f(\frac{1}{2}\ \vec{x} - \vec{x}'\ ^2)$	Gaussian RBF	$\exp(-\frac{1}{2\sigma^2}\ \vec{x} - \vec{x}'\ ^2),$ σ is decided by the user

appropriate tight contour representations of clusters [11]. Moreover, for a kernel $k(\vec{x}, \vec{x}')$ that only depends on $\vec{x} - \vec{x}'$, $k(\vec{x}, \vec{x})$ is constant, and the linear term in the dual target function is constant. This simplifies computation.

5.3 Support vector clustering

5.3.1 Working principle

Recent studies [11, 12] employed SVND for the development of Support Vector Clustering (SVC). SVND describes the data distribution in terms of support information. One point of view is that clustering is about model detection [52]. SVND provides a straightforward modeling prototype for SVC. The main idea behind SVC is that the support vectors extracted for novelty detection can be used to construct the boundaries of clusters in a data set. From its SVND nature, the SVC is able to detect clusters with arbitrary shapes and different density distributions. It works with high dimensional data. Moreover, it provides a way to deal with outliers.

Based on its working principle, SVC is carried out in two main phases [12], namely SVM training and cluster labeling. We now explain these phases.

5.3.2 SVM training

The SVM training part in SVC is responsible for determining the cluster structure. The SVC [11, 12] is a boundary-based clustering method. When the hypersphere is mapped back to the original data space, this method produces a set of disjoint contours that enclose the data points. These contours can be interpreted as cluster boundaries, and linkages between each pair of data items can be estimated. Recalling Equation (5.2.5) and Equation (5.2.6), cluster boundaries can be constructed by a set of contours $\{\vec{x} \mid R(\vec{x}) = R\}$, which encloses the points in data space. Thus, SVs lie on cluster boundaries, BSVs are outside, and all other points lie inside clusters.

Clearly, the number of SVs and BSVs affects the cluster structure, which therefore can be controlled by the SVM training parameters q and C . As the width q of the Gaussian kernels increases, the number of SVs (n_{sv}) increases, the shape of the cluster boundaries becomes rougher, and the contours tend to split up (cf. the boundaries of white regions in Fig. 5.1(a)-(d) below). On the other hand, the number of BSVs (n_{bsv}) can be controlled by C , more precisely by $n_{bsv} < 1/C$. That is, if $C \geq 1$, there are no BSVs. To allow for BSVs, one should set $C < 1$. Instead of using C , it is more natural to work with the parameter $p = 1/(nC)$, which represents an upper bound for the fraction of BSVs. The parameter q determines the scale at which the data is probed, and p decides the softness of the boundaries [11]. Figure 5.1 shows a 285-points data set and the changes of cluster boundaries in dependence on different settings of q and p , which are selected experimentally as in [11].

5.3.3 Cluster labeling

The cluster description itself does not differentiate between points that belong to different clusters. The cluster labeling part checks the connectivity for each pair of points based on cut-off criteria (the radius R of the hypersphere) obtained from the trained SVMs. To do this, an adjacency matrix A_{ij} is defined based on geometric

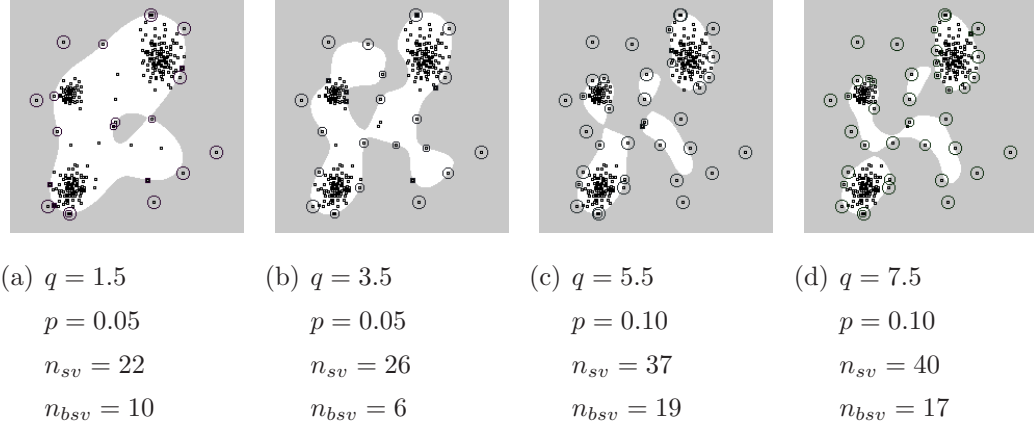


Figure 5.1: Controlling cluster structures using training parameters. The clusters are represented by white regions. Their boundaries and number vary with q and p . Encircled points are the resulting SVs and BSVs.

observation: given a pair of data points that belong to different clusters, for any path in data space connecting them, the corresponding path in feature space must have an intersection with the outside of the hypersphere. For each pair of points $\vec{x}_i$ and $\vec{x}_j$, A_{ij} takes a binary value.

$$A_{ij} = \begin{cases} 1, & \text{if } R(\vec{x}_i + \lambda(\vec{x}_j - \vec{x}_i)) \leq R, \forall \lambda \in [0, 1]; \\ 0, & \text{otherwise.} \end{cases} \quad (5.3.1)$$

Clusters are now defined as the connected components of the graph induced by A . Calculating A_{ij} for points $\vec{x}_i$ and $\vec{x}_j$ is implemented by sampling a number of, say m points on the line segment between these two points (where m is usually chosen between 10 and 20). The two cluster labeling strategies of Ben-Hur et al. [11, 12] are described in Cluster Labeling Strategy 1 and 2 below.

Cluster Labeling Strategy 1 (CG Labeling)

Calculate A_{ij} for each pair of points $\vec{x}_i$ and $\vec{x}_j$ in data space. This results in using the complete graph, denoted by CG, to model the adjacency matrix A_{ij} . It takes $O(n^2m)$ time, where n is the number of data items, and m the number of sampling points between each pair of data points.

Cluster Labeling Strategy 2 (SVG Labeling)

Calculate A_{ij} only for pairs of points $\vec{x}_i$ and $\vec{x}_j$, where $\vec{x}_i$ or $\vec{x}_j$ is a support vector. This results in a subgraph of CG, which is referred to as SVG. It takes $O((n - n_{bsv})n_{sv}^2m)$ time, where n_{bsv} is the number of BSVs and n_{sv} is the number of free SVs.

With our SVC implementation, we found that the SVM training part takes much less time than the cluster labeling part. Therefore, the time complexity of the SVM training part is not the main concern of SVC, while the computation of the cluster labeling part is the critical issue. The cluster labeling has extremely demanding CPU-requirements for large data sets. Since the clusters are the connected components induced by A_{ij} that can be represented by spanning trees, the explicit use of CG in Cluster Labeling Strategy 1 results in many redundant edges. Cluster Labeling Strategy 2 uses a heuristic. This approach does not check the linkages between all pairs of data items, but only those between points and support vectors. Thus, it improves the efficiency of the cluster labeling part. While Cluster Labeling Strategy 2 is faster than Strategy 1, there are two issues with Strategy 2 to be considered. First, when n_{sv} is greater than $0.05n - 0.1n$, its observed CPU-time complexity is just a small constant from the original quadratic time. Second, the heuristic in this strategy for linkage estimation can sometimes fail. The observation is that points in data space that are close neighbors are more likely to belong to the same cluster. Adjacencies between points are only kept within *neighborhoods* in the same cluster in feature space. However, checking only linkages to support vectors does not necessarily take this into account. Thus, Cluster Labeling Strategy 2 often yields trivial clusters, and even produces a number of unlinked data points of close proximity (we refer to these as *false negatives*). This can make clustering results less meaningful.

We propose an algorithmic engineering approach to improve the efficiency and quality of cluster labeling in SVC. We first describe the enhanced modeling of discrete data points in Section 5.4. Then, in Section 5.5, we present our algorithmic approach that employs proximity graphs.

5.4 Model enhancement using proximity graph

Typically, clustering methods use a similarity concept (e.g., Euclidean distance) to measure proximity between data objects. Even in high-dimensions, proximity is critical to cluster analysis. Any particular clustering solution of a data set X can be presented in the form of an $n \times n$ cluster connectivity matrix A_{ij} , defined by

$$A_{ij} = \begin{cases} 1, & \text{if points } \vec{x}_i \text{ and } \vec{x}_j \text{ belong to the same cluster;} \\ 0, & \text{otherwise.} \end{cases}$$

The matrix model can be mapped to a subgraph of a proximity graph. In proximity graphs, vertices represent data points and edges connect pairs of points to model their proximity and adjacency. The main principle of proximity graph modeling is encoding proximity and topology between data points. With a proximity graph, points are connected by edges if they are close to each other according to some proximity measure. Near-by points are naturally more likely to be in the same cluster than points that are far away. Thus, cluster labeling with a proximity graph strategy is a good heuristic to reduce the time of testing linkages.

Recently, Estivill-Castro and Lee [57, 59] proposed boundary-based clustering methods using proximity graph modeling. The main idea they employed is to detect the sharp density changes at potential cluster boundaries. Proximity and density information modeling of 2D point data using Delaunay Diagrams has shown its power in the development of exploratory and argument-free clustering algorithms for geographical data mining [57]. To best describe proximity between data points, a common family of proximity graphs was investigated and compared for different modeling considerations [59]. These proximity graphs include, for example, Delaunay Diagrams (DD), Minimum Spanning Trees (MST) and k -Nearest Neighbors (k -NN). They can be derived by considering different aspects of proximity and topology. DD represents a “is-neighbor” relationship. The MST is based on the local closeness of data points. It is a subgraph of DD, and encodes less proximity information. k -NN is based on distance concepts. Figure 5.2 shows the construction of a variety of proximity graphs for

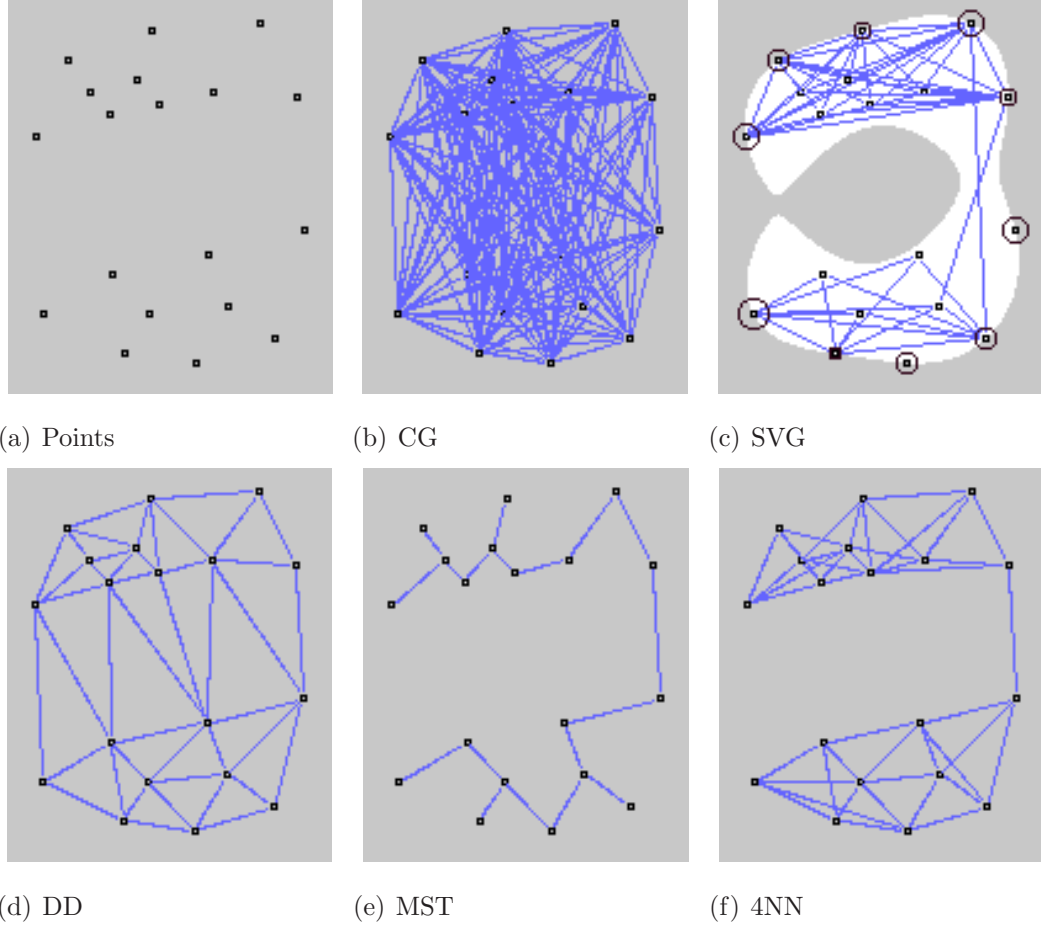


Figure 5.2: Various proximity modeling graphs. In (c), encircled points are SVs and the white region is the pre-image of the hypersphere in feature space.

cluster assignments. For comparison, we also plotted CG and SVG in this figure. We used the LEDA [117] and ANN [158] libraries in constructing these proximity graphs. By choosing appropriate underlying proximity graphs the expected time complexity is sub-quadratic for data of all dimensions.

5.5 An efficient approach to SVC

In this section, we propose an efficient SVC approach through proximity graph modeling. This approach extends the previously proposed SVC method of [11, 12] with

proximity graph modeling. It consists of four phases:

1. SVM training for the cluster structure.
2. Model enhancement through proximity graphs.
3. Cluster labeling for cluster assignment.
4. Cluster harvest and cleaning up.

Since we have already discussed most of the work related to the first two phases, here we only concentrate on a new cluster labeling algorithm and cluster harvest process.

5.5.1 Cluster labeling guided by proximity graphs

Once we obtain the cluster structure through SVM training, the radius R of the hypersphere can be used as a cut-off criterion to check the connectivities among data points. Instead of using Cluster Labeling Strategy 1 or 2 due to their drawbacks, we propose the Cluster Labeling Strategy 3.

Cluster Labeling Strategy 3 (Proximity Graph Labeling)

Modeling the data with an appropriate proximity graph that reflects the data distribution and incorporates proximity and topology information. We calculate coefficients of the adjacency matrix A_{ij} only for pairs of $\vec{x}_i$ and $\vec{x}_j$, where $\vec{x}_i$ and $\vec{x}_j$ form an edge e_{ij} in a proximity graph. While estimating the edges of a proximity graph with a cut-off criteria (i.e., R), we perform the same sampling strategy for computation of A_{ij} as in [11, 12]. For simplicity, all edges in the current proximity graph are called *candidate edges*. We refer to an edge e_{ij} as *active edge* if $A_{ij} = 1$, and as *passive edge* if $A_{ij} = 0$. An *active path* in the current proximity graph will be formed if every edge in the path is an active edge.

Cluster Labeling Strategy 3 avoids redundant checks in a complete graph and also avoids the loss of neighborhood information as it can occur when only estimating

adjacencies to support vectors. The time complexity of our labeling strategy is only $O(mn \log n)$, where n is the size of data set and m is the number of sampling points on each edge. This is further analyzed in Section 5.6.

5.5.2 Cluster harvest

Without going into the mathematical details, the cluster harvest procedure is justified by the empirical observation that clusters correspond to connected components of edges, which are active paths (for the notion, see Cluster Labeling Strategy 3). Passive edges are not of interest, and they will be removed from the proximity graph. After the removal of passive edges, the task of cluster harvest lies in recognizing all active paths formed. Once active edges have been determined, a classical algorithm like Depth-First-Search (DFS) can be used for collecting connected components. Note that DFS has complexity proportional to the number of edges in the proximity graph. Pseudo code for cluster harvest is shown in Algorithm 4. To avoid trivial clusters, some clean-up work is necessary. For example, BSVs can eventually be included in the closest cluster (as in our experiments), or discarded. Also, we treat a cluster with a number of points under a threshold (say 3) as noise. Figure 5.3 demonstrates the whole procedure of SVM clustering for the case of the DD modeling.

5.6 Performance evaluation

In this section, we present results of experiments, which compare and evaluate the performance of different cluster labeling methods. We demonstrate empirically the robustness and efficiency of cluster labeling with proximity graphs. The LibSVM [31] library has been used in the performance evaluation.

Algorithm 4 (Cluster Harvest)

```

function clusterHarvest(G:Graph; P:Sets)
    {input G: Subgraph after cluster labeling}
    {output P: Set of clusters}
    var
        v: Node;
        Cj: Set; {A set of points in one cluster}
    begin
        while (v := G.chooseNode())
            Cj.clear();
            DFS(G, v, Cj); {v is moved from G to Cj}
            P.add(Cj);
        end while
    end

```

5.6.1 Time complexity

Given is a data set of n points $X = \{\vec{x}_i\}$, with $X \subseteq \Re^D$, the data space. We now analyze three types of proximity graphs, Delaunay Diagram (DD), Minimum Spanning Tree (MST), and k -Nearest Neighbors (k -NN). The number of edges in DD is linear in size n for 2D ($3n-6$ at most), but it is quadratic in size for 3D. Other proximity graphs are linear in size for all dimensions. The number of edges in MST is $n - 1$. For k -NN, $O(kn)$ is the upper bound. Thus, walking on edges for cluster assignment and cluster harvest takes $O(n)$ time. On the other hand, the field of computational geometry has developed $O(n \log n)$ time algorithms to construct DD, MST and k -NN in spaces of various dimensions. Therefore, all proximity graphs presented here can perform 2D cluster labeling and harvest in $O(n \log n)$ time. This is remarkably efficient compared to the quadratic time requirements of CG (used in Cluster Labeling Strategy 1) and of SVG (used in Cluster Labeling Strategy 2). The experimental results displayed in Figure 5.4 illustrate that Cluster Labeling Strategy 3

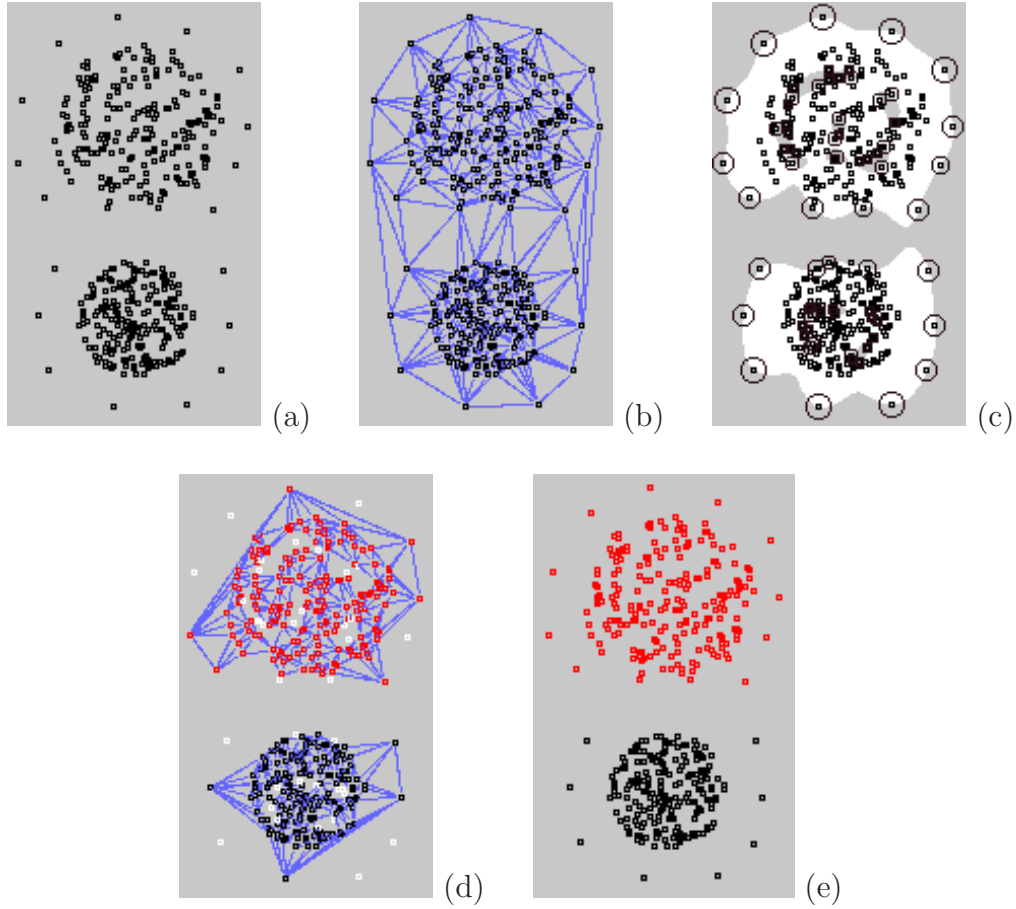


Figure 5.3: Procedure of SVC using DD modeling. (a) A set of points. (b) DD modeling. (c) SVMs training result. (d) Active paths after DD labeling. (e) Final result after cluster harvest.

outperforms Strategy 1 and Strategy 2. We first generated a data set with 1,000 points based on a mixture model. From this data set, we sampled five subsets with 100, 200, 400, 600 and 800 points, respectively. Then, we performed SVC on these subsets using various cluster labeling mechanisms. After empirical test with different values, we set parameters $q = 0.20$ and $p = 0.10$ to train the SVMs. The results shown in Figure 5.4 include the overall time requirements for proximity graph construction, cut-off criterion computation, cluster labeling and harvest.

When we move to high-dimensional data sets, k -NN and MST graphs become more attractive, since the number of edges of these graphs remains linear in n . This makes

proximity graph modeling scalable to high-dimensional data for support vector cluster analysis.

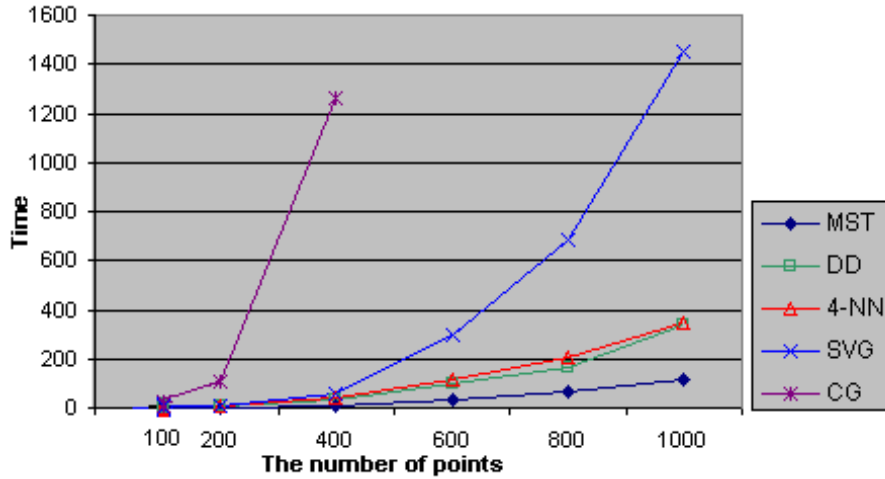


Figure 5.4: CPU-time comparison of various cluster labeling mechanisms. Slowest is the result using CG which is quadratic in the number of points n and fastest is the result using MST which is $n \log n$.

5.6.2 Clustering quality

When applying different proximity graphs to model data for support vector clustering, different results may be produced. We take the result of the CG as reference against which we compare the other cluster labeling methods. Our experiments show that the DD works as well as the CG. This is because, as a spanner [98,106]¹, the DD holds sufficient interconnections. In contrast, as a subgraph of the DD, the MST encodes much less proximity information. Some linkages between neighbors in the DD have been lost in the MST. Despite its speed, the MST is a fragile clustering method. Interestingly, a k -NN with appropriate k also reports good results. The experiments indicate that a k -NN (with $k > 3$) captures satisfactory proximity information for cluster labeling. Here the argument k does not need to be tuned as carefully as in [59], because the proximity graph modeling in our study is only for the purpose of cluster

¹A graph is a spanner if it encodes all proximity information approximately.

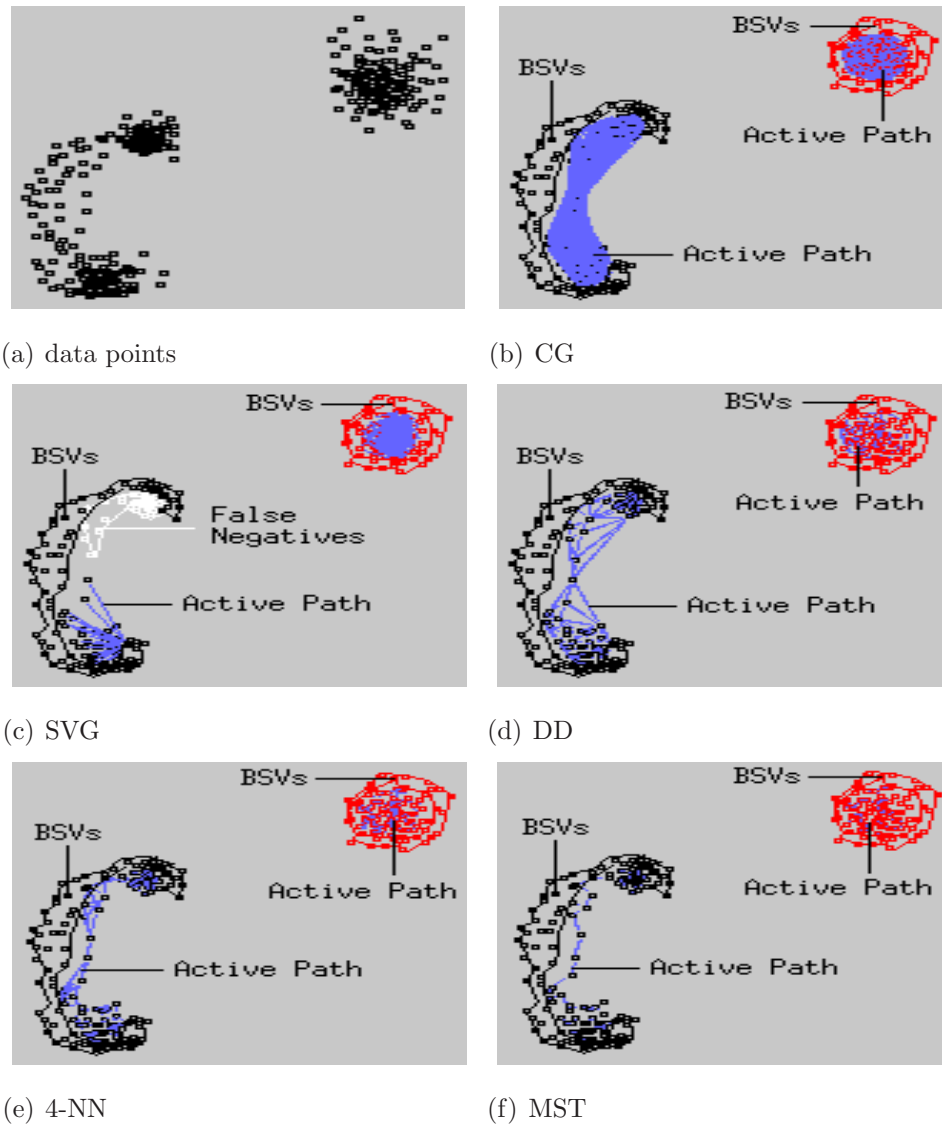


Figure 5.5: Comparison of clustering results using different labeling methods with parameters $p = 0.5$, $q = 2.75$ for SVM training. Note that, for better recognition, the two clouds of BSVs have been highlighted by additional boundary lines. The cloud in the upper right corner has approximately the shape of an annulus.

labeling, not for the computation of the cut-off criterion. As mentioned before, the SVG is a heuristic to produce a sparse graph to simplify cluster labeling.

However, it sometimes can produce false negatives and make the clustering result less meaningful. This situation can occur when the parameter p is high. In this case, most SVs turn out to be BSVs and there are only a few free SVs left. The data points, lacking support information (i.e. they can not connect to SVs), become false negatives. Figure 5.5 illustrates cluster labeling results from different labeling methods for a data set shown in Figure 5.5(a). Figure 5.5(b) is the result of the CG. Figure 5.5(c) shows that the SVG strategy produces false negatives. In contrast, with consideration of neighborhood relationship, the proximity graph modeling approaches, as shown in Figure 5.5(d, e, f), do not yield false negatives.

5.7 Remarks

In this chapter, we presented the support vector clustering method, and applied proximity graphs to enhance data modeling. Our experimental results demonstrate that the combination of SVC and proximity graph modeling robustly and efficiently performs cluster assignment. The impact of efficiency improvement, from quadratic to $n \log n$ time through proximity graph modeling, is also reflected if we attempt parallel implementation. For example, such parallel implementation of Cluster Labeling Strategy 3 would be logarithmic on n processors, while the Strategy 1 and Strategy 2 would only be linear on n processors.

Chapter 6

Cluster Validity

Clustering techniques are the main tools used for exploratory data analysis, where we are dealing with data with little internal structure, or no prior information is available. Clustering algorithms are expected to produce partitions that reflect the internal structure of the data and identify “natural” classes and hierarchies present in it. Lacking a priori information about structure in the data makes clustering analysis a challenging task. To solve clustering problems, some assumptions are naturally made to select a model to fit to the data. Hence, clustering depends significantly on the data modeling and the induction principle of the learning algorithm to reveal structure for the data. Since they do not know which assumptions are satisfied by the data, researchers may run into severe difficulties in interpreting results. The purpose of clustering validity is to increase the confidence about groups proposed by a clustering algorithm. Validity is a certain amount of confidence that the clusters found are actually somehow significant [45]. That is, the hypothetical structure postulated as the result of a clustering algorithm must be tested to gain confidence that it actually exists in the data. A fundamental way is to measure how “natural” are the resulting clusters. Here, formalizing how “natural” a partition is, implies fitting metrics between the clusters and the data structure [72]. The validity of results is of utmost importance, since patterns in data will be far from useful if they are invalid.

In Section 6.1, we summarize cluster validity tasks. In Section 6.2, we review the cluster validity methods that have been proposed so far. Then, we present an engineering-style cluster validity method, which is geometrically motivated through Support Vector Machines (SVMs).

6.1 Cluster validity tasks

6.1.1 Clustering tendency

Before running a clustering algorithm, we can check clustering tendency to ensure that meaningful clusters do exist. Clustering tendency is the problem of determining the presence or the absence of a clustering structure [109]. Usually, this task relies on statistical hypothesis testing on a sampling window. For a data set X , the sampling window is commonly defined as the compact convex support set for the underlying distribution of the vectors of X . We typically test the randomness hypothesis (H_0) and regularity hypothesis. These have been designed to carry out hypothesis testing to increase the confidence that there is an actual structure in the data (structure being understood as discrepancy from the null hypothesis). For the case of H_0 , we suppose the feature vectors of X are distributed uniformly at random throughout the sampling window; while for the case of regularity hypothesis, we suppose the feature vectors of X are regularly spaced in the sampling window. If the randomness or the regularity hypothesis is accepted, methods alternative to clustering analysis should be used for the interpretation of X .

6.1.2 Parameter tuning

Typically, applications of any cluster algorithm necessitate choosing specific values for some parameters. For example, k -clustering takes the expected number k of clusters

as part of their input. The results yielded by the clustering algorithm may depend strongly on the choice of parameters. Instead of involving statistical tests (as in the clustering tendency check), to seek for appropriate values of the parameters, we commonly use relative criteria. To this end, we conduct an exploratory procedure to tune the parameters according to a prespecified criterion. Let $\mathcal{T}$ be the set of parameters associated with a specific algorithm. The parameter tuning can be considered in the following cases [109]:

- **The set $\mathcal{T}$ of parameters does not contain the number k of clusters as a parameter.** The choice of the best values for this type of parameter is based on the assumption that *if X possesses a clustering structure, this structure is captured for a large range of values of the parameters in $\mathcal{T}$* [134]. Under this assumption, we run the algorithm for a wide range of values of its parameters. Intuitively, if we find the largest range of parameter values for which the clustering structure keeps consistent (for example k is constant), then we can choose parameter values in the middle of the range.
- **The set $\mathcal{T}$ of parameters contains the number k of clusters as a parameter.** In this case, a different procedure is followed. We first select a suitable performance index g , then we employ the following steps [21]:
 1. Compute different data partition for $k = 2, \dots, k_{max}$.
 2. At each convergence, measure the value of validity index g .
 3. Seek for the extremum of g (only if g is independent of k), or a significant change (“knee”) in g ; set the correspondent k and other parameters as appropriate values.

Note that algorithmic engineering of fast clustering methods is very important in parameter tuning processes, since these algorithms are going to be repeated many times with different settings. For example, if the time complexity of an algorithm is $O(n)$, executing this algorithm n times results in a total $O(n^2)$ time complexity;

but if the time complexity of an algorithm is $O(n^2)$, repeating this algorithm n times results in a total $O(n^3)$ time complexity.

6.1.3 Choosing best-fit algorithms

When applied to the same data set, different clustering algorithms often lead to markedly different results. In some cases, such differences are expected, since different algorithms make different assumptions (explicitly or implicitly) about the structure of the data [107]. For example, if the particular set consists of several clouds of data points, with each cloud spherically distributed about its center, methods that assume such structure (e.g. k -MEANS) will work well. However, choosing a best-fit algorithm for a particular problem can be a daunting task. A simple way to undertake this task is using relative process, where researchers set some indices and apply a set of given clustering algorithms to a data set. After running these algorithms in an exploratory way, we can choose the algorithm with ideal values of the predefined criteria.

6.1.4 Robustness of clustering output

One critical issue related to the structural quality of clusters is the robustness of clustering results. That is, if input data points deviate slightly from their current values, or the initial values to a clustering algorithm change, will we get the same clustering result? This is important in data analysis because usually there is noise in the data, or the initial values are subject to change when a clustering algorithm is repeated. A good clustering result should be insensitive to the noise and able to capture the stable structure in the data. To test the robustness of the results obtained from different algorithms, we can also use a relative process. Briefly, we make the data set perturbed by generating some additive or multiplicative noise of the same dimension. Then we can calculate the overall discrepancy rate for the clustering using statistical methods. This process is repeated many times and the

average overall discrepancy rate can be used as a criterion to measure the robustness of clustering results.

6.1.5 Measuring significance of the revealed structure

Although most researchers shy away from giving a formal definition of “clusters”, there is wide consensus that a cluster is a relatively compact region of high data density that is isolated, in the sense that it is separated from other clusters by regions of low data density (voids). Thus, two measures, *compactness* and *separation*, are proposed to quantify these qualitative descriptions [109]. Compactness means the members of each cluster should be as close to each other as possible. Separation means the clusters themselves should be widely spaced. Numerous studies suggest direct or indirect indices for evaluating compactness and separation of clusters. Some are statistically motivated, for example, comparing the within-cluster scattering with the between-cluster separation [40]. Some are based on geometrical motivation to estimate how compact and well separated the clusters are (an example is Dunn’s index [47]). Others are information-based [37], parametric estimation based [46], and more. However, a problem with these indices is that the best value of an index may not imply valid results. This is possibly because these methods do not detect the complexity of boundary regions and deal with outliers.

6.2 Cluster validity methods

Cluster validity methods usually calculate formal indexes for estimation of clustering quality based on external, internal, or relative processes [109]. External validity process is used either for comparison of a cluster structure $\mathcal{C}$ produced by a clustering algorithm, with a partition $\mathcal{P}$ of X drawn independently from $\mathcal{C}$, or for measuring the degree of agreement between the proximity matrix of X and a predefined partition $\mathcal{P}$. Typically, the partition $\mathcal{P}$ presents a ground truth classification of X . For example,

researchers often produce synthetic data with a specific structure, and clustering algorithms are evaluated on the amount of structure they recover. The internal validity process attempts to verify whether the clustering structure fits data, using only information inherent in the data. This is an unsupervised estimation manner, making use of nothing more than the available data. In the relative validity process, a set of clustering contexts is considered, and the goal is to choose the best one (e.g. a best-fit clustering algorithm or a best-fit parameter setting for a clustering algorithm) according to a prespecified criterion. In the following, we summarize methods and indicators that come under the name of cluster validity.

6.2.1 Statistical methods

Statistical tools provide clear and comprehensive description of cluster quality assessment [92, 109]. There are a number of statistical indices for various cluster validity requirements. For example, in order to measure the degree of a cluster structure $\mathcal{C}$ to a predefined partition $\mathcal{P}$, we can use popular external statistics like *Rand statistic*, *Jaccard coefficient*, *Fowlkes-Mallows index*, and *Hubert's Γ statistic* and *Normalized Γ statistic* [109]. As internal statistics, *cophenetic correlation coefficient* (CPCC) and *γ statistic* are often used to measure the degree of agreement between the cophenetic matrix A_c produced by a clustering algorithm, with the proximity matrix A_p of the data set X . In the case of relative validity, to compare different clustering contexts we can use *Davies-Bouldin (DB) index* [40], *Dunn's index* [47], and *partition coefficient* [15]. However, these mathematically defined indices face many difficulties. In almost all practical settings, this statistic-based methodology for validity faces challenging computation of the probability density function of indexes that complicates the hypothesis testing approach around the null hypothesis [109]. Bezdek [16] realized that it seemed impossible to formulate a theoretical null hypothesis used to substantiate or repudiate the validity of algorithmically suggested clusters.

6.2.2 Information-based methods

The information contained in data models can also be captured using concepts from information theory. Researchers recently developed clustering evaluation function using Renyi's entropy [72]. Entropy was originally introduced to measure the unexpectedness of the information contained in a message. For evaluating the effectiveness of clustering, a related measure, *information potential*, is derived from Renyi's entropy. Another measure for clustering evaluation using information theory, MML, is suggested by Oliver et al. [128]. The MML principle considers the message consisting of a description of a model, and encodes the data under the assumption that the model is true. It can be used for estimating model parameters, such as the number of clusters in a data set. The main difficulty of information-based methods is computation of the probability density function of indexes.

6.2.3 Formal validation

In specialized cases, like conceptual schema clustering, formal validation has been used for suggesting and verifying certain properties [181]. While formal validity guarantees the consistency of clustering operations in some special cases like information system modeling, it is not a general-purpose method. On the other hand, if the use of more sophisticated mathematics requires more specific assumptions about the model, and if these assumptions are not satisfied by the application, the performance of such validity test could degrade beyond usefulness.

6.2.4 Empirical evaluation

In addition to theoretical indexes, empirical evaluation methods [101, 137] are also used in some cases where sample data sets with similar known patterns are available. The major drawback of empirical evaluation is the lack of benchmarks and unified

methodology. In addition, in practice it is sometimes not so simple to get reliable and accurate ground truth.

6.2.5 Visualization-based evaluation

For settings where visualization is possible, intuitive verification of the clustering results is feasible. In fact, most researchers use visualization of 2D data sets when demonstrating clustering algorithms and the quality of their results. Obviously, the assumption is that quality would extrapolate to higher dimensions, because if a method displays better behavior in 2D than another, then it would be expected that it would retain the improved performance with larger dimensions. In the case of large multidimensional data sets, effective visualization of data is difficult. Moreover, the perception of clusters using available visualization tools is a difficult task for humans [109].

6.2.6 Generic discussion

Although there is no shortage of indices that measure some sort of grouping quality in clustering literature, evaluation of the validity of the output of clustering algorithms is a difficult problem in general. Perhaps, the major problem lies in embedding cluster validity into induction principles of clustering [138, 143]. In some studies, a clustering principle is referred to as cluster validity [143]. Note that, a cluster partition is the result of optimizing an induction rule, but validity is a function of the data set [52]. Such optimization should not be interpreted as obtaining valid clusters. For example, in probability-based clustering, the maximum likelihood of a supposed model can not draw the conclusion that a clustering result is valid. Another problem is that many of these published techniques are based on the same theme in that they compare inter-cluster versus intra-cluster variability and tend to favor configurations with ball-shaped well-separated clusters. Irregularly shaped clusters are problematic.

Actually, the real-world data are often distributed in more complicated structure with high-dimension, arbitrary shape, and large density change. Moreover, the above-mentioned methods are too weak to measure the margin between two clusters that are in arbitrary shape and non-linearly separable. In this situation, traditional measures, like single-linkage, are not able to detect the actual separation between two clusters. In addition, these methods lack a mechanism to deal with outliers.

In the following sections, we propose a new approach to cluster validity using Support Vector Machines (SVMs). Unlike SVC that is based on novelty detection (an unsupervised learning task for unlabeled data), cluster validity using SVMs is on the basis of classification (a supervised learning task for labeled data). Intuitively speaking, clustering results are useful, if there are well defined separations between clusters, and there is at least one dense core within each cluster. The present cluster validity approach is geometrically motivated. It attempts to confirm that the structure of clustering results is of some significance. Applying SVMs to clustering results and learning the boundary complexity between clusters provides a natural approach to detecting valleys and outliers. First, SVMs are able to measure margins analytically; this makes them favorable for verifying separations among the clusters of an algorithm's output. Second, SVMs' ability to detect outliers makes them applicable to reveal a stable compact structure through a controlled process. Our cluster validity approach can be used as an internal process to evaluate the quality of a clustering algorithm's outputs, or as a relative process to choose better clustering algorithms.

In Section 6.3, we discuss the classification aspects of SVMs. Based on this supervised learning scheme of SVMs, we propose an approach to clustering validity tasks. Section 6.4 describes the details of our approach. Section 6.5 presents experimental results to reinforce the method proposed here for increasing the confidence in a clustering result from an arbitrary clustering algorithm $\mathcal{A}$.

6.3 Classification using SVMs

6.3.1 Linear hypothesis space

Consider the problem of separating the set of training samples $\{(\vec{x}_i, y_i)\}_{i=1}^n$ belonging to two classes, where $\vec{x}_i$ is the input vector for the i th example and y_i is the target output. We assume that for the positive subset $y_i = +1$ while for the negative subset $y_i = -1$, and that positive and negative examples are “linearly separable”. The equation of a decision hyper-plane that separates is:

$$\vec{w}^T \vec{x} + b = 0 \quad (6.3.1)$$

where $\vec{x}$ is an input vector, $\vec{w}$ is an adjustable weight vector (the normal to the decision hyper-plane), and b is called the bias. There is an infinite number of separating hyper-planes that correctly classify linearly separable training data. For a given weight vector $\vec{w}$ and bias b , the distance from the hyper-plane to the origin is $|b|/\|\vec{w}\|$; the distance of a point $\vec{x}$ from the decision hyper-plane is $|\vec{w}^T \vec{x} + b|/\|\vec{w}\|$; the separation between the hyper-plane and the closest data point is called the *margin of separation* and is denoted by γ . The goal of SVMs is to choose the hyper-plane whose parameters $\vec{w}$ and b maximize γ .

Intuitively, we can construct a convex hull for each class in the training data and find the closest pair of points with each point in a respective convex hull. If a decision hyper-plane bisects these two closest points, the resulting classifier should be robust in some sense [13]. Figure 6.1(a) demonstrates this approach.

While this approach appears intuitively obvious, there is still a need to formally describe its properties. The closest pair of points in the respective convex hull lie on the hyper-planes $\vec{w}^T \vec{x} + b = \pm 1$. The margin γ is $1/\|\vec{w}\|$, and maximizing the margin is equivalent to the following problem. Given the training set $\{(\vec{x}_i, y_i)\}_{i=1}^n$, find the decision hyper-plane that minimizes the following quadratic program:

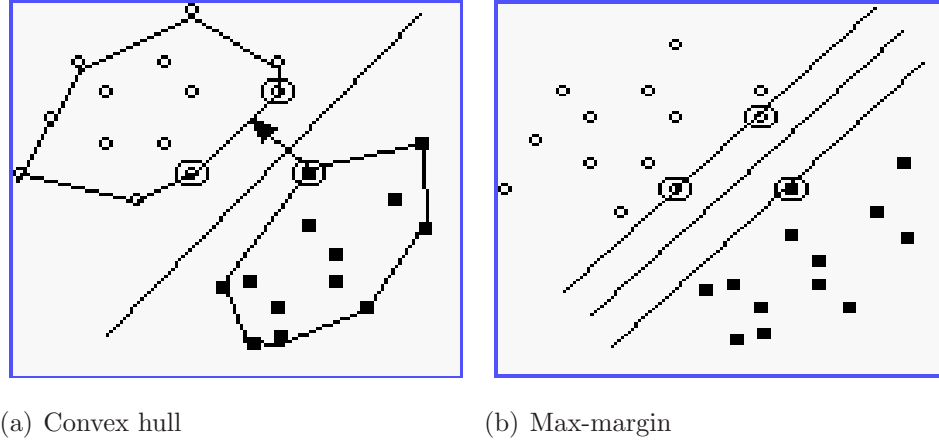


Figure 6.1: The working principles of SVMs for classification tasks. An optimal hyperplane bisects the closest pair of points in respective convex hulls and maximizes the margin between two linearly separable classes.

$$\begin{aligned}
 &\text{Minimize } M(\vec{w}, b) = \frac{1}{2} \|\vec{w}\|^2 \\
 &\text{subject to } y_i [\vec{w}^T \vec{x}_i + b] \geq 1.
 \end{aligned} \tag{6.3.2}$$

Under this condition, the decision surface is referred to as the *optimal hyper-plane*. Figure 6.1(b) illustrates the geometric construction of an optimal hyper-plane for a 2D input space. The particular data points $(\vec{x}_i, y_i)$ that satisfy Equation (6.3.2) with equality are SVs.

6.3.2 Non-linear hypothesis space

If the two classes are nonlinearly separable, the variants called ϕ -machines are classically used to map the input space $X = \{\vec{x}_1, \dots, \vec{x}_n\}$ into a high-dimensional feature space $\mathfrak{F} = \{\phi(\vec{x}) | i = 1, \dots, n\}$. By choosing an adequate mapping ϕ , the input samples become linearly or mostly linearly separable in feature space. However, to learn a nonlinear hypothesis, SVMs use kernel methods (see Equation (5.2.7)). That is, the training data will only be used in the form of dot products between vectors. By constructing a feature space nonlinearly related to input space, the SVMs can

find the hyper-plane in the nonlinear feature space, which separates the training data with the widest margin. Typical types of kernels are shown in Table 5.1. In our cluster validity approach, we use the Gaussian kernel $k_q(\vec{x}, \vec{x}') = e^{q\|\vec{x}-\vec{x}'\|^2}$ with width parameter $q = \frac{-1}{2\sigma^2}$ for its favorable features.

6.3.3 The ν -SVM approach

There are many existing algorithms for solving general-purpose quadratic problems concerned with SVMs (mostly involving slack variables and Lagrangian multipliers). For our cluster validity, we make most use of the features of ν -Support Vector Machine (ν -SVM).

To compare with regular C-SVM, the ν -SVM is a new class of SVM. It has the advantage of using a parameter ν in effectively controlling the number of SVs [32, 152, 154]. Again consider training vectors $\vec{x}_i \in \mathbb{R}^D, i = 1, \dots, l$ labeled in two classes by a label vector $\vec{y} \in \mathbb{R}^n$ such that $y_i \in \{1, -1\}$. As a primal problem for ν -Support Vector Classification (ν -SVC), we consider the following minimization:

$$\begin{aligned} & \text{Minimize } \frac{1}{2}\|\vec{w}\|^2 - \nu\rho + \frac{1}{n} \sum_{i=1}^n \xi_i \\ & \text{subject to } y_i(\vec{w}^T \phi(\vec{x}_i) + b) \geq \rho - \xi_i, \\ & \xi_i \geq 0, i = 1, \dots, n, \rho \geq 0, \end{aligned} \tag{6.3.3}$$

where

1. training vectors $\vec{x}_i$ are mapped into a higher dimensional feature space through the function ϕ , and
2. Non-negative slack variables ξ_i for soft margin control are penalized in the objective function.

The parameter ρ is such that when $\vec{\xi}^T = (\xi_1, \dots, \xi_n) = 0$, the *margin of separation*

is $\gamma = \rho/\|\vec{w}\|$. The parameter $\nu \in [0, 1]$ has been shown to be an upper bound of the fraction of margin errors and a lower bound of the fraction of SVs [32, 152]. In practice, the above prime problem is usually solved through its *dual* by introducing Lagrangian multipliers and incorporating kernels:

$$\begin{aligned}
& \text{Minimize } \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) \\
& \text{subject to } 0 \leq \alpha_i \leq 1/n, \quad i = 1, \dots, n \\
& \quad \sum_i \alpha_i y_i = 0 \\
& \quad \sum_i \alpha_i \geq \nu
\end{aligned} \tag{6.3.4}$$

or

$$\begin{aligned}
& \text{Minimize } \frac{1}{2} \vec{\alpha}^T (Q + \vec{y} \vec{y}^T) \vec{\alpha} \\
& \text{subject to } 0 \leq \alpha_i \leq 1/n, \quad i = 1, \dots, n \\
& \quad \vec{e}^T \vec{\alpha} \geq \nu
\end{aligned} \tag{6.3.5}$$

where Q is a positive semidefinite matrix, $Q_{ij} \equiv y_i y_j k(\vec{x}_i, \vec{x}_j)$, and $k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^T \cdot \phi(\vec{x}_j)$ is a kernel, $\vec{e}$ is a vector of all ones.

The context for solving this dual problem is presented in [32, 152]. Some conclusions in those works are useful for our cluster validity approach.

Proposition 6.3.3.1 *Suppose ν -SVC leads to $\rho > 0$, then regular C-SVC with parameter C set a priori to $1/\rho$, leads to the same decision function.*

Lemma 6.3.3.1 *Optimization problem (6.3.5) is feasible if and only if $\nu \leq \nu_{max}$, where $\nu_{max} = 2 \min(\#y_i = 1, \#y_i = -1)/n$, and $(\#y_i = 1)$, $(\#y_i = -1)$ denote the number of elements in the first and second classes respectively.*

Corollary 6.3.3.1 *If Q is positive definite, then the training data are separable.*

The proofs for the above conclusions can be found in [32]. In ν -SVM, the value νn is a lower bound of the number of SVs and an upper bound of the number of

misclassified training data. Here, these misclassified data are BSVs and treated as outliers. The larger the parameter ν , the more points are allowed to lie inside the margin; if ν is smaller, the total number of SVs decreases accordingly, and the margin becomes harder. The bound νn lies between the number of SVs and the number BSVs. Proposition 6.3.3.1 describes the relation between standard C-SVC and ν -SVC, and an interesting interpretation of the regularization parameter C . The increase of C in C-SVC is like the decrease of ν in ν -SVC. Lemma 6.3.3.1 shows that the size of ν_{max} depends on how balanced the training set is. If the numbers of positive and negative examples match, then $\nu_{max} = 1$. Corollary 6.3.3.1 tells whether a training problem under extent kernels is separable (it does not mean the training problem is well separated). In many situations, Q is positive definite. If the RBF kernel is used, Q is positive definite [32].

When we use ν -SVC and choose the Gaussian kernels to analyze clustering results, the situations of margins and outliers can be tested by controlling training parameters for SVMs formalism, namely ν and q . The number of SVs depends on both ν and q . When q increases, boundaries become very rough, since a large fraction of the data turns into SVs, especially those potential outliers that are broken off from *core* data points in the form of SVs. However, no outliers will be allowed, if $\nu = 0$. By increasing ν , more SVs will be turned into outliers or BSVs. For regular C-SVC, an upper bound on the fraction of BSVs is denoted to the parameter $p = 1/(Cn)$. Parameters ν and p will be used alternately in the following discussion.

6.4 Cluster validity using SVMs

When SVMs are applied to the output of clustering algorithms, they are able to learn the structure inherent in clustering results. By checking the complexity of boundaries, we are able to verify if there are somehow significant “valleys” between data clusters and how outliers are distributed. All these are readily computable from the data in a supervised manner through SVMs training. SVMs are capable of

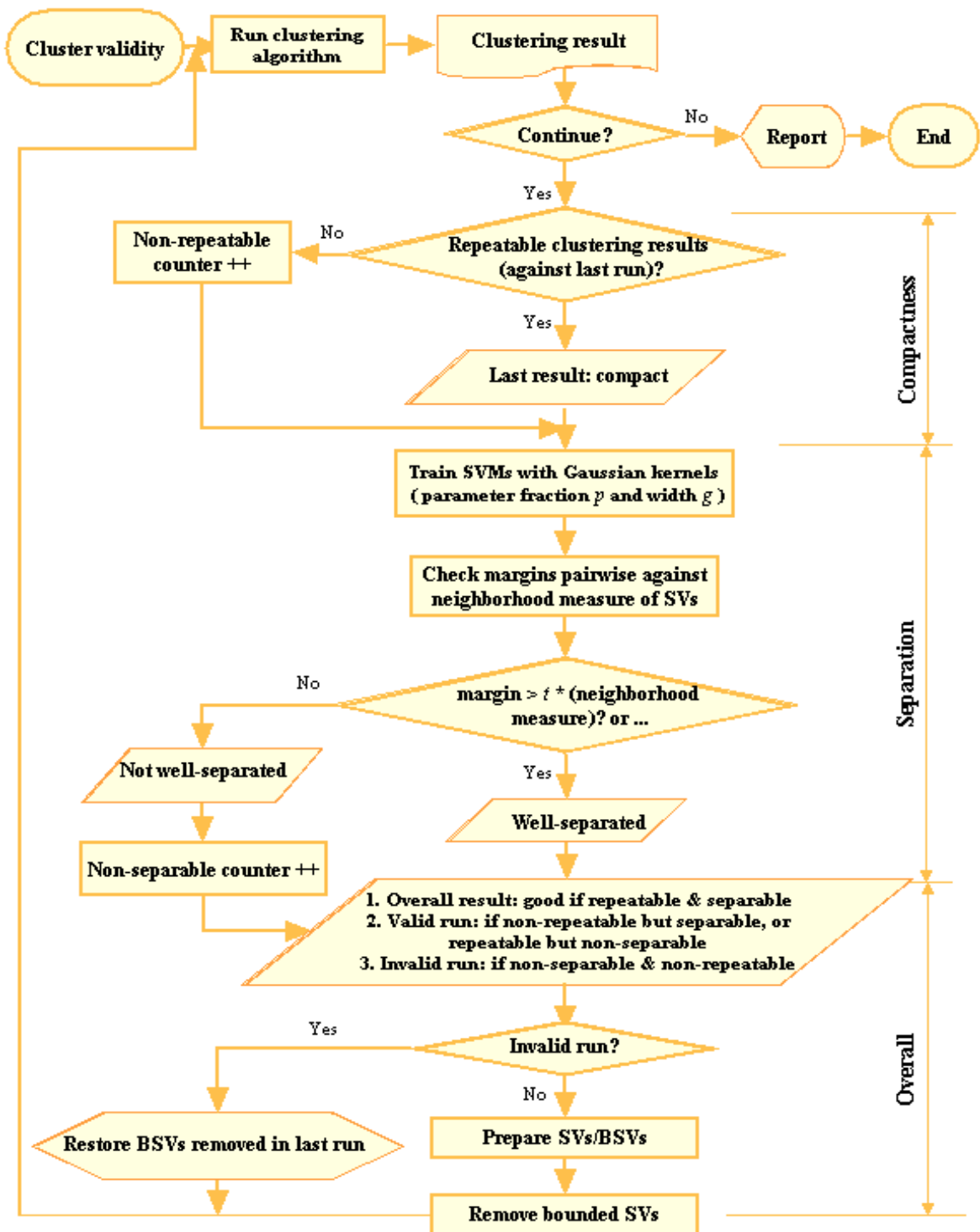


Figure 6.2: Illustration of cluster validity using SVMs.

providing good generalization for high dimensional training data, since the complexity of optimal hyperplanes can be carefully controlled independently of the number of dimensions [34]. SVMs can deal with arbitrary boundaries in data space, and are not limited to linear discriminants. These properties are appropriate to realistic data structures concerned with clustering applications.

Our cluster validity approach is shown in Figure 6.2. This approach is based on two observations of clustering outputs. First, good clustering results should separate clusters well (we should find isolated clusters). Second, there is at least a *core* in each cluster. That is, there should be high density concentration in the core of the cluster, and removing a few points in the core does not affect their shape. However, points in cluster boundaries are in sparse regions and perturbing them does change the shape of boundaries. The validity process that we propose performs pairwise comparisons for all clusters.

6.4.1 Separation measure

To verify the separation between a pair of clusters, we learn the *margin* γ through SVMs training; then we choose the top ranked SVs (say up to 5) from each class and their u (say 5) nearest neighbors. For each class, we measure the average distance of the chosen SVs from their neighbors in feature space. Distances are measured in projection along the normal of the optimal hyper-plane. We let their average be $\bar{\gamma}_1$ for the first class and we denote it as $\bar{\gamma}_2$ for the other class. We compare γ with $\bar{\gamma}_i$, $i = 1, 2$. Given scalars λ_1 and λ_2 , the relations between local measures and margin is evaluated by analyzing to find out if any of the following conditions holds.

$$\bar{\gamma}_1 < \lambda_1 \cdot \gamma \text{ or } \bar{\gamma}_2 < \lambda_1 \cdot \gamma \quad (6.4.1)$$

$$\bar{\gamma}_1 > \lambda_2 \cdot \gamma \text{ or } \bar{\gamma}_2 > \lambda_2 \cdot \gamma \quad (6.4.2)$$

If either of them holds for carefully selected control parameters λ_1 and λ_2 , the clusters are separable. Otherwise they are non-separable. These two parameters are tuned

in an algorithmic engineering approach. After analyzing a number of experimental instances under investigation, we found that $\lambda_1 \leq 0.5$ and $\lambda_2 \geq 2$ indicate a significant separation. We measured separation in feature space, because feature space normalizes the margin. Measurements in feature space avoid two difficulties with respect to original data space. First, if the SVs in data space are away from the region contrasting the two class, the measurement in data space is incorrect. An illustration of this situation is in Figure 6.3(a), for the SV labeled $\vec{x}$. Second, the margin in data space may be irregular. An illustration of this is in Figure 6.3(b).

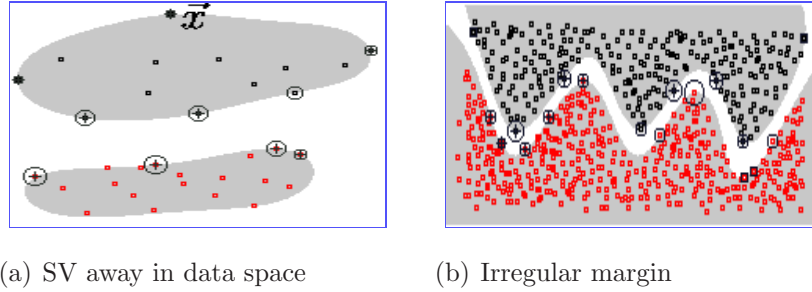


Figure 6.3: Problems avoided by measurements in feature space.

The separation assessment can discriminate between two results of a clustering algorithm. That is, when faced with two results, maybe because the algorithm is randomized or because two clustering methods are applied, we increase the confidence (and thus the preference to believe one is more valid than the other) by selecting the clustering result that shows fewer pairs of non-separable classes.

6.4.2 Compactness measure

To verify the compactness of each cluster, we control the number of SVs and BSVs. As mentioned before, the parameter q of the Gaussian kernel determines the scale at which the data is probed, and as it is increased, more SVs will result. Potential outliers especially tend to appear isolated as BSVs. However, to allow for BSVs, the parameter ν should be greater than 0. This parameter enables us to analyze noise.

Controlling q and ν provides us a mechanism for verifying compactness of clusters. We note that if clusters are compact, the cores will appear when outliers are removed. This can be verified by checking the stability of cluster assignment. After removing a fraction of BSVs, if reclustering results in repeatable assignments, we can conclude that the cores of classes exist and outliers have been detected.

Consider an arbitrary clustering algorithm $\mathcal{A}$. The idea behind our approach is to increase our confidence in the result in applying $\mathcal{A}$ to a data set. If the clustering result is repeatable (robust to our removal of BSVs and their nearest neighbors) and separable (in the sense of having a margin a fraction larger than the average distance between SVs), we can increase our confidence that the data does reflect this clustering and is not an artifact of the clustering algorithm. We say the clustering result has an increased sense of validity. On the other hand, if reclustering results are not quite repeatable but well separable, or repeatable but not quite separable, we call the current run a *valid* run. This may be due to situations where: (1) the results are separable but the removal of the BSVs may lead to different cluster assignment, or (2) there remain BSVs unremoved and these BSVs display clusters that are not well separated. However, if reclustering shows output that is neither separable nor repeatable, we call the current run an *invalid* run. In this case, the BSVs removed in the last run may not be outliers, and they should be recovered for a reclustering.

Valid runs or invalid runs can still be discriminated by repeating the analysis. After several rounds of the above validity process, if consecutive clustering results converge to a stable assignment (that is, the result from each run is repeatable and separable) we believe the potential outliers have been removed, and cores of clusters have emerged. If most of the repetitions produce invalid runs, that is clustering solutions differ across runs without good separation, the clustering results are not interesting.

6.5 Performance evaluation

The approach we propose provides a novel mechanism to address cluster validity problems for more elaborate analysis and intuitive interpretability. In this section, we demonstrate empirically our framework and the whole course for cluster validity, especially SVMs learning analysis. Following the diagram proposed, we show separation and compactness checking, and complete examples as well. The data sets used in our demonstration are in different shapes to ensure generality. For simplicity and because our approach checks each pair of clusters; that is, it works in a pairwise way, examples are usually shown with two clusters. The LibSVM [31] SVMs library has been used in our implementation of our cluster validity scheme.

6.5.1 Separation test: a normal case

First, we illustrate the evaluation of separation with results from experiments on boxed data. To accurately measure the margin between two clusters, namely to ensure the lower error bound, we use a *hard margin* training strategy by setting parameter ν with a value lower than 0.01. This allows for only a few BSVs. Figure 6.4 shows six data sets.

In each data set there is a pair of clusters and the margin is decreasing across data sets. The data in a box consist of 486 points uniformly and randomly generated. The training parameters are set to $\nu = 0.01$ and $q = 0.001$. To verify the separation of a pair of clusters, we calculate the average local measures around top ranked SVs in both clusters, (the values of $\bar{\gamma}_1$ and $\bar{\gamma}_2$). Our process then compares them with the margin γ and inspects the difference. The experiment illustrates that the larger the discrepancies between $\bar{\gamma}_1$ and γ (or $\bar{\gamma}_2$ and γ), the more separable the clusters are. In general, if $\bar{\gamma}_1 < 0.5\gamma$ or $\bar{\gamma}_2 < 0.5\gamma$, the two clusters are separable. Thus, the choice of values for λ_1 in our process.

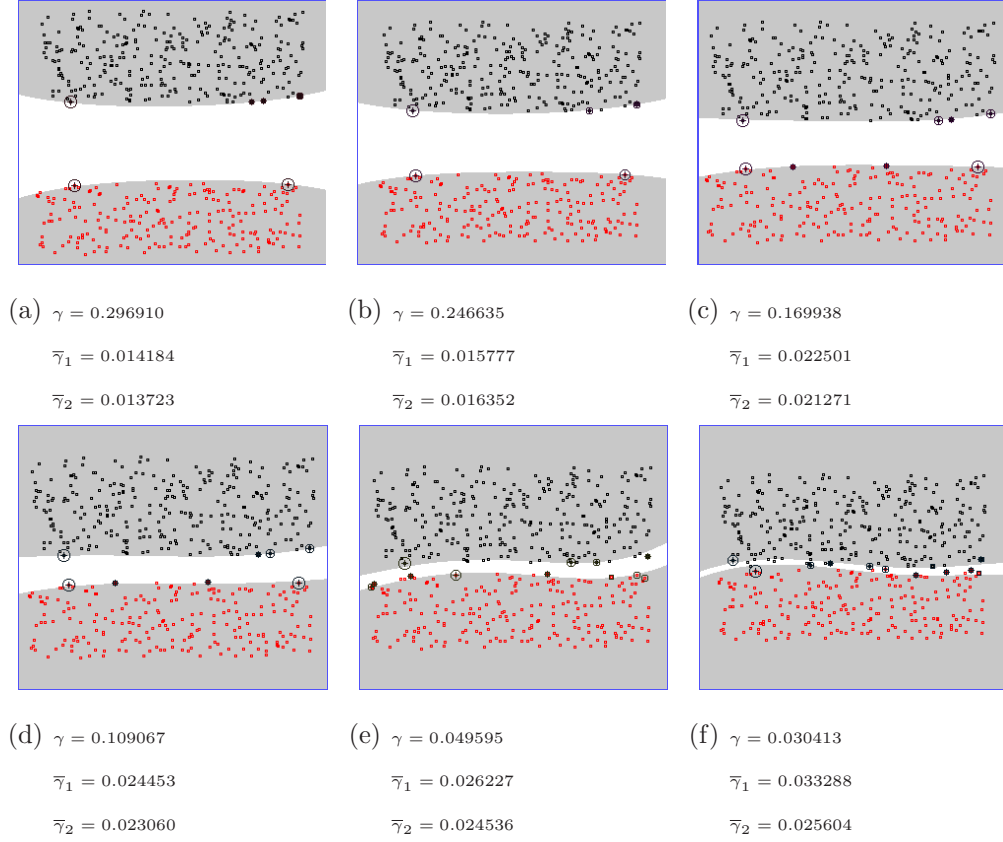


Figure 6.4: Checking separation between clusters (a normal case). Six pairs of clusters are plotted with margin decreasing. Training parameters $\nu = 0.01$, $q = 0.001$. Circled points are SVs.

6.5.2 Separation test: other cases

Experiments here demonstrate other possible cases of the separation test. In Figure 6.5(a), both $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are much larger than γ . Figure 6.5(b) does not show a large difference between $\bar{\gamma}_1$ and γ , but the difference between $\bar{\gamma}_2$ and γ is significant. The case in Figure 6.5(c) shows significant difference between $\bar{\gamma}_1$ and γ , although there is not much difference between $\bar{\gamma}_2$ and γ . Again, we set $\lambda_1 = 0.5$ and $\lambda_2 = 2$ for our test. Then, according to the verification rules of separation (in Equation (6.4.1) and Equation (6.4.2)) all of these examples are declared separable.

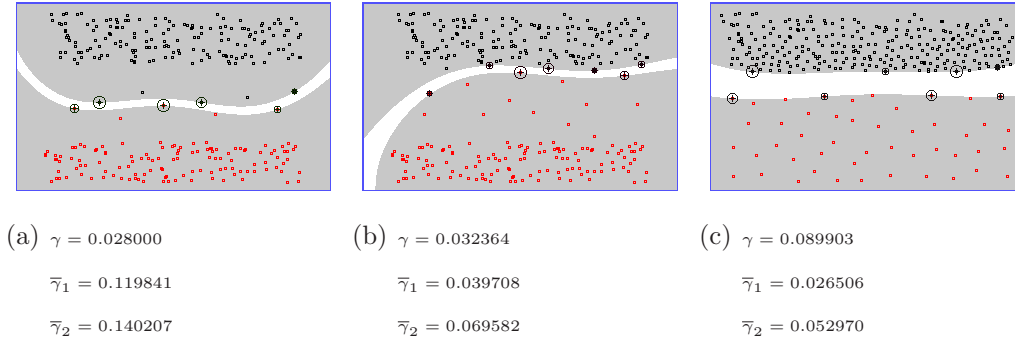


Figure 6.5: Checking separation between clusters (other cases). Training parameters $\nu = 0.01$, $q = 0.001$

6.5.3 Outliers test: a general case

Occasionally clustering results of an algorithm $\mathcal{A}$ might not accurately describe the groups in the data or are hard to interpret because noise is present and outliers may mask data models. When these potential outliers are tested and removed, the cores of clusters appear. In this case, our approach works as a filter and the structure that fits to the data becomes clearer. Figure 6.6 demonstrates such a process. Figure 6.6(a) presents 558 points in a ring shape data with many outliers. Figure 6.6(b) shows the result of a ν -SVC training with $\nu = 0.1$ and $q = 0.001$, where 51 BSVs are obtained. After filtering these BSVs (outliers are more likely to become BSVs), Figure 6.6(c) shows a clear data model that has two significantly isolated dense clusters. In contrast, if a ν -SVC is trained again with $\nu = 0.05$ and $q = 0.001$ on the clearer model, much fewer (17 BSVs) are generated as shown in Figure 6.6(d).

6.5.4 Outliers test: repeatable effects

Again, consider an arbitrary clustering algorithm. Because the existence of outliers complicates clustering results, reclustering results may be not repeatable after removing these outliers. The repeated performance of algorithm $\mathcal{A}$ depends on the previous clustering results. If these results have recognized compact clusters with cores, then

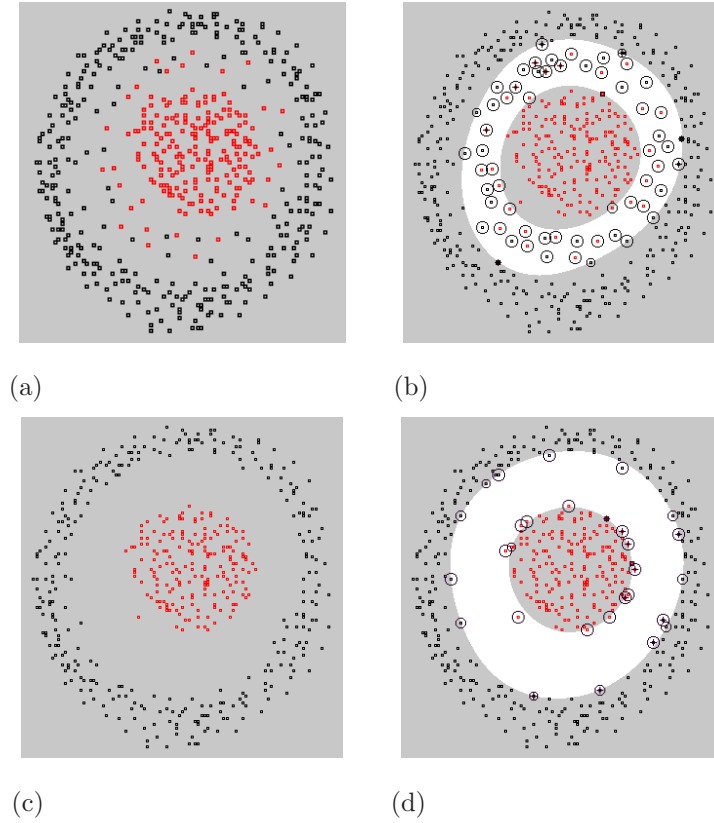


Figure 6.6: Checking outliers (a general case). Circled points are SVs.

they become robust to our removal of BSVs. There are two cases. In the first case, the last two consecutive runs of algorithm $\mathcal{A}$ (separated by an application of BSVs removal) are consistent. That is, the clustering results are repeatable. The alternative case is that reclustering with $\mathcal{A}$ after BSVs removal is not concordant with the previous result. Our check for repeated performance of clustering results verifies their cluster compactness. Figure 6.7 and Figure 6.8 illustrate these two different cases respectively, where 1000 points drawn from a mixture data model are used. Training parameters for ν -SVC are set to $\nu = 0.05$ and $q = 0.005$. In Figure 6.7, the reclustering results are completely repeatable, but in case of Figure 6.8, reclustering results give a significantly different model from the first run.

To measure the degree of repeated performance between clustering results of two dif-

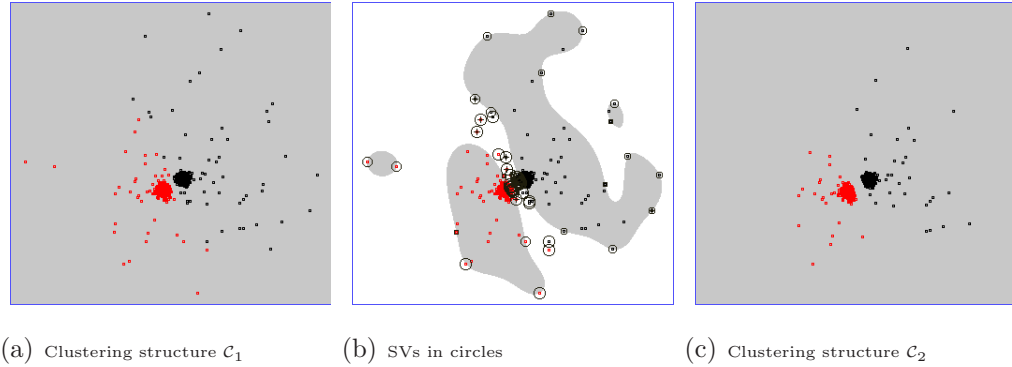


Figure 6.7: Checking outliers for estimating the compactness of clustering results (compact case). For compact data, reclustering results are repeated when outliers are removed. (a) Results of the first run. (b) Test for outliers. Circled points are SVs. (c) Reclustering results ($\dot{R} = 1.0$, $\dot{J} = 1.0$, $\dot{F} = 1.0$).

ferent runs, we can adopt indexes of external criteria used in cluster validity. External criteria are usually used for comparing a clustering structure $\mathcal{C}$ with a predetermined partition $\mathcal{P}$ for a given data set X . Instead of referring to a predetermined partition $\mathcal{P}$ of X , we measure the match degree between two clustering structures $\mathcal{C}_1$ and $\mathcal{C}_2$ using these indexes. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be consecutively produced from a clustering method working on data set X or its subset with outliers removed. The indexes we used for this purpose are the *rand statistic* $\dot{R}$, the *Jaccard coefficient* $\dot{J}$ and the *Fowlkes-Mallows index* $\dot{F}$ [101]. The values of these three statistics are between 0 and 1. The larger their value, the higher the degree to which $\mathcal{C}_1$ matches $\mathcal{C}_2$.

6.5.5 Examples of 2D data

We now provide a detailed illustration of our cluster validity testing using SVMs shown in Figure 6.9. The 2D data set is from a mixture model and consists of 1000 points. The k -memoids algorithm assigns two clusters.

As shown in the diagram for our method (Figure 6.2), the validity process will be conducted in several rounds. Each round consists of reclustering and SVMs analysis

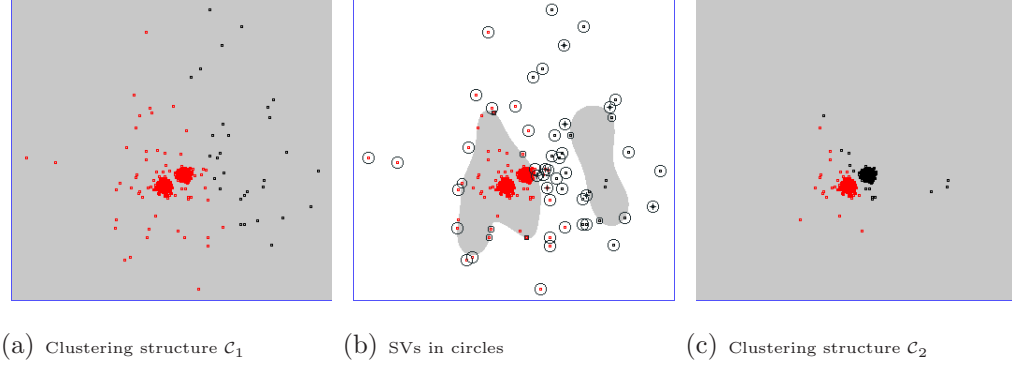


Figure 6.8: Checking outliers for estimating the compactness of clustering results (non-compact case). For non-compact data, reclustering results are not repeated when outliers are removed. (a) Results of the first run. (b) Test for outliers. (c) Reclustering results ($\dot{R} = 0.5077$, $\dot{J} = 0.3924$, $\dot{F} = 0.5637$).

(compactness checking, separation verification, and outliers splitting and filtering). The process stops when a clear clustering structure appears (this is identified because it is separable and repeatable), or after several rounds (say six). Several runs that do not suggest a valid result indicate that the clustering method $\mathcal{A}$ is not finding reasonable clusters in the data. For the separation test in this example, we train ν -SVC with parameters $\nu = 0.01$ and $q = 0.0005$. To filter potential outliers, we conduct ν -SVC with $\nu = 0.05$ but different q in every round. The first round starts with $q = 0.005$, and q will be doubled in each following round.

Figure 6.9(a) and Figure 6.9(b) show separation test and compactness evaluation respectively corresponding to the first round. We observed that the cluster results are separable. Figure 6.9(a) indicates $\bar{\gamma}_1 > 2\gamma$ and $\bar{\gamma}_2 > 2\gamma$. Figure 6.9(b) shows the SVs generated, where 39 BSVs will be filtered as potential outliers. We perform reclustering after filtering outliers, and match the current cluster structure to previous clustering structures. Indexes $\dot{R} = 1$ ($\dot{J} = 1$ and $\dot{F} = 1$) indicate compactness. Similarly, the second, third and fourth rounds also show repeatable and separable clustering structure. We conclude that the original cluster results can be considered valid.

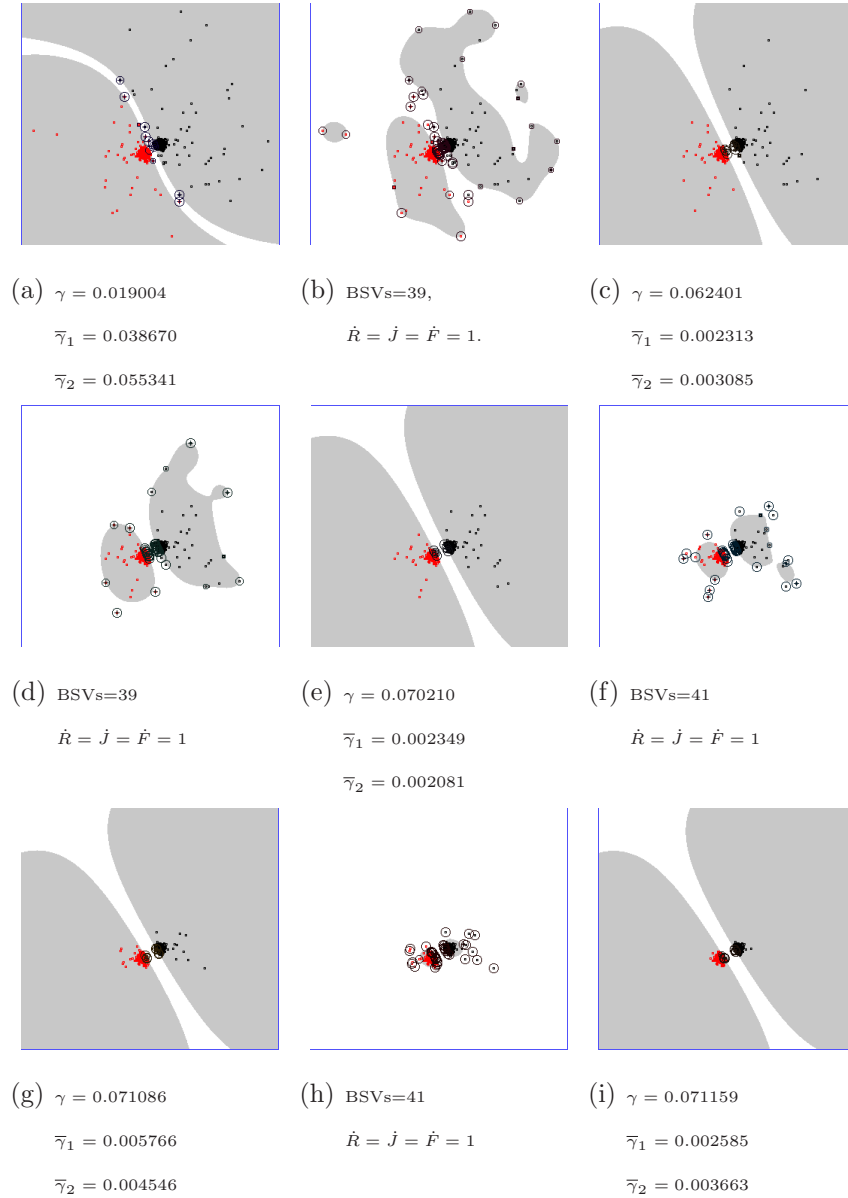


Figure 6.9: A 2D example of cluster validity through SMVs approach. Circled points are SVs. (a) and (b) demonstrate separation check and compactness verification of the first round. (c) and (d) demonstrate separation check and compactness verification of the second round. (e) and (f) demonstrate separation check and compactness verification of the third round. (g) and (h) demonstrate separation check and compactness verification of the fourth round. (i) shows a clearly separable and repeatable clustering structure.

6.5.6 Examples of 3D data

We now conduct our cluster validity testing using SVMs on a 3D data set shown in Figure 6.10. The data set is from a mixture model and consists of 2000 points. The algorithm DISCRETE FW CENTER assigns three clusters. The validity process is similar to that in the 2D example. After five rounds of reclustering and SVMs analysis, the validity process stops, and a clear clustering structure appears. For the separation test in this example, we train ν -SVC with parameters $\nu = 0.01$ and $q = 0.0005$. To filter potential outliers, we conduct ν -SVC with $\nu = 0.05$ but different q in every round. The first round starts with $q = 0.005$, and q will be doubled in each following round.

In each round, we demonstrate a 3D view of the data, followed by separation test and compactness verification. To give a clear 3D view effect, we construct convex hulls of clusters. For separation and compactness checking, we use projections along the z axis. Because of pairwise analysis, we denote by $\gamma_{i,j}$ the margin between clusters i and j , while $\bar{\gamma}_{i(i,j)}$ is the neighborhood dispersion measure of SVs in cluster i with respect to the pair of clusters i and j . Figure 6.10(a) illustrates a 3D view of original clustering result. Figure 6.10(b) and Figure 6.10(c) show separation test and compactness evaluation respectively corresponding to the first round. Figure 6.10(b) indicates $\bar{\gamma}_{1(1,2)}/\gamma_{1,2} = 6.8$, $\bar{\gamma}_{1(1,3)}/\gamma_{1,3} = 11.2$ and $\bar{\gamma}_{2(2,3)}/\gamma_{2,3} = 21.2$. Thus we conclude that the cluster results are separable in the first run. Figure 6.10(c) shows the SVs generated, where 63 BSVs will be filtered as potential outliers. We perform reclustering after filtering outliers, and match the current cluster structure to previous clustering structure. Index $\dot{R} = 1$ indicates the compactness of the result in previous run. Similarly, the second round up to the fifth round also show repeatable and separable clustering structures. Figure 6.10(d) shows a 3D view of the reclustering result in the second run. Figure 6.10(e) and Figure 6.10(f) demonstrate separation check and compactness verification of the second round. Figure 6.10(g) shows a 3D view of the reclustering result in the third run. Figure 6.10(h) and Figure 6.10(i) demonstrate separation check and compactness verification of the third round. Fig-

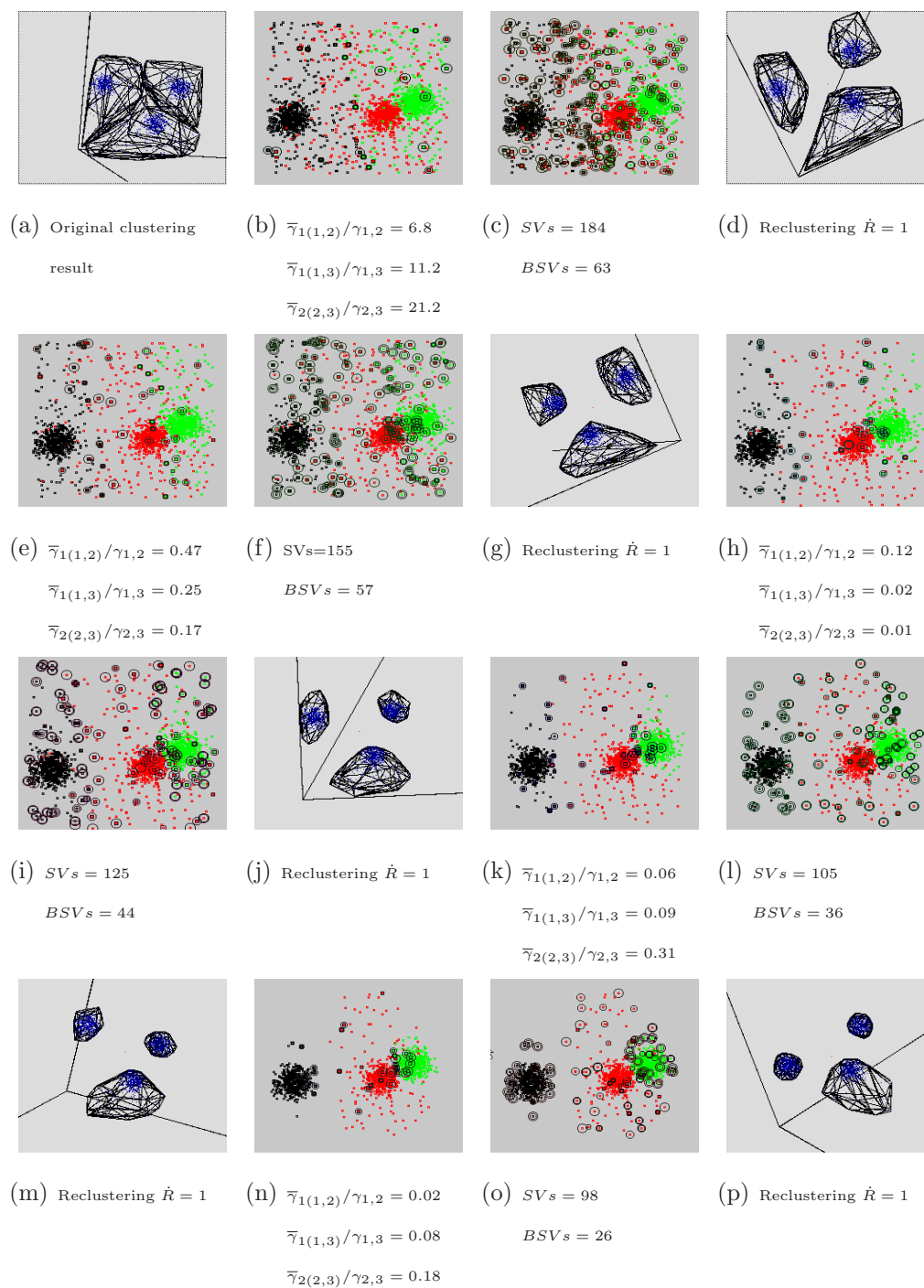


Figure 6.10: A 3D example of cluster validity through SMVs approach. Circled points are SVs.

Figure 6.10(j) shows a 3D view of the reclustering result in the fourth run. Figure 6.10(k) and Figure 6.10(l) demonstrate separation check and compactness verification of the fourth round. Figure 6.10(m) shows a 3D view of the reclustering result in the fifth run. Figure 6.10(n) and Figure 6.10(o) demonstrate separation check and compactness verification of the fifth round. Figure 6.10(p) shows a clearly separable and repeatable clustering structure. Thus the original cluster results can be considered valid.

6.6 Remarks

The nature of clustering is exploratory, rather than confirmatory. The task of data mining is to reveal novel patterns. Cluster validity is a certain amount of confidence that the detected cluster structure is significant. In this chapter, we have applied SVMs and related kernel methods to cluster validity. This engineering-style method evaluates clustering quality both analytically and geometrically. It is a good practice in transferring advanced algorithmic techniques into cluster validity applications. Intuitively, if clusters are isolated from each other and each cluster is compact, the clustering results are interpretable. Learning clustering results using SVMs can obtain an insight into the structure inherent in data. By analyzing the complexity of boundaries through support information, we can verify separation performance and potential outliers. After several rounds of reclustering and outlier filtering, we will obtain clearer clustering structures. Counting the number of valid runs and match results from different rounds in our process contributes to verifying the goodness of the clustering result.

Chapter 7

Case Studies: Web Usage Mining and Text Mining

With the rapid growth of the Internet, the volume of Web data increases daily and so does its usage. After the advent of data mining and its successful application on conventional data, Web-related information has been an appropriate and emerging target of knowledge discovery [100, 111, 160, 163]. Web mining is a methodology for the extraction of knowledge from Web data. Depending on whether the data used in the knowledge discovery process concerns the Web *content* itself or the *usage* of this content, Web mining has been broadly classified into “Web content mining” and “Web usage mining” [36]. Section 7.1 provides a case study of Web usage mining, where the representative clustering methods proposed in Chapter 4 are used to group Web visitors and analyze their behavior. Although we do not explicitly discuss Web content mining, we present a case study of text mining in Section 7.2. Since Web content mining concentrates on categorizing and analyzing documents on the Web, techniques of text mining can be used for Web content mining. We apply the clustering methods proposed in Chapter 3 to text mining.

7.1 Case study 1: Web usage mining

Capturing useful information from Web visitors is an important task for Web site design and evaluation [132, 156, 186]. There has been an increased demand for automatically understanding the behavior of Web users due to the expansion of the Web and the increased number of Web-based applications [133, 183]. Tracking users is useful to provide the elements for answering the most crucial questions for most organizations adapting to the Web [62, 161]: Does the Web really work? In other words, does the Web attract visitors? and if so, do those visitors turn into customers? So the Web-site owner needs to discover what users want in the site, what their likes and dislikes are.

In other media, statistical indicators are computed to answer these questions. Newspapers have circulation figures, radio has broadcast ranges, and television has Nielsen ratings. Surprisingly, few Web sites monitor access levels [62] although collecting data on visitation seems easier through the Web than any other medium (it is impossible to know exactly how many people are listening to radio). Access to demographic information is a key advantage of these other media over the Web [62]. Accurate demographics are much more readily available for these traditional media, because a representative sampling of the general population in geographical areas can be extrapolated meaningfully to apply to the whole audience. With Web, however, we have several problems in achieving this. First, people's self-selected visitations of a Web site are combined in complex ways with people's accidental arrivals. Thus, sampling the general population of Web visitors to obtain demographics may be simply impossible. Second, the international reach of the Web means that visitors from all over the world could be attracted, making it much harder to survey aspects of their likes and dislikes. Thus, there has been comparative less success at categorizing Web visitors than categorizing customers in transactional data. Both of these problems mean that the only way we could get accurate demographics would be while people are actually visiting the Web site. This is where the idea of collecting visitation information emerges.

Statistics analysis of Web site traffic based on users' access logs or client caches can collect some characteristics about how the Web site is being accessed. It can determine statistical indicators of behavioral activity by visitors [62]. With basic statistical methods, we can summarize the raw information from users into various useful categories, such as the number of hits by domain, by file path, by day, by hour. However we need to discover more knowledge by data mining techniques, and implement them for Web site analysis settings [132].

Web mining is a methodology for the extraction of knowledge from Web data. As a central process inside data mining, clustering is typically used to categorize the market. The goal of clustering here is to identify strong correlation among users' interests by grouping their navigation paths. The literature contains several illustrations in commercial applications where clustering discovers the types of customers [14, 18]. In addition, clustering analysis has demonstrated the extensive benefits and sophisticated applications emerging from identifying groups of visitors to a Web site. For example, when user-clustering reveals similar information needs, dynamic hypertext links in Web pages could be suggested for users in a common cluster. Another example is Web page pre-fetching. Pre-fetching Web pages can help users personalize their needs and reduce their waiting time. However, it is only effective when the right documents are identified and users' moves are correctly predicted. Capturing noticeable access patterns by clustering allows predictions with increased confidence. Thus, clustering visitation paths is to play a central role in identifying whether the site is being used as expected, optimizing the performance of a Web server, or discovering which products are being purchased by which visitors.

7.1.1 Preprocessing for Web usage mining

We will not argue on the techniques by which entries in an access log are converted to visitation paths. We adopt the conventional assumption [20, 96, 131, 133, 156, 183, 186], that each traversed link from page p_i to p_j , has been identified in a session and to a

visitor, and this is a component of a path. A transversal path u can be modeled as a sequence $\langle \{p_1\}\{p_2\}\cdots\{p_\rho\} \rangle$ that indicates a visitor starting page, and an ordered page trajectory by links. The length of the path, the access frequency, the access time interval, the order of the Web pages, and other aspects play a role in similarity measures, as we discussed in Chapter 4.

7.1.2 Clustering methods

In this section, we demonstrate empirically the virtues of the representative clustering methods we proposed in Chapter 4 with respect to Web usage data. We performed this experiment on implementations in Visual C++. First, we concentrate on illustrating the much improved use of CPU-time, and that the implementation of $O(n\sqrt{n})$ time results in a dramatic improvement in CPU-time resources. Later, we concentrate on showing that those clustering methods actually produce better clustering quality. Our contribution is then fast and robust algorithms for clustering visitation paths.

7.1.3 On the scalability of the methods

Our first experiment reproduces, to the best of our ability, the implementation of the clustering algorithm by Xiao et al. [183], which is an incarnation of the methods suggested by Shahabi et al. [156]. Here we use the simplest of the similarity measures, access-based (refer to Equation (4.1.1)). Since other measures require more computational time, our algorithms would outperform the algorithms based on similarity matrices by a larger margin.

We used the same data set of logs identified with visitors and sessions used by Xiao et al. [183]. That is, we used the Web access logs [38]¹ publicly available from the Computer Science Department, Boston University. Figure 7.1(a) displays the differ-

¹Data obtained in February 2001, from <ftp://cs-ftp.bu.edu/techreports/1995-010-www-client-traces.tar.gz>

ence in CPU time between the matrix-based algorithm detailed in Xiao et al. [183] and our algorithms. Our implementation is much faster than previous matrix-based algorithms. Just for the access-based similarity metric given in Equation (4.1.1), the matrix-based algorithm requires over 18,000 CPU seconds (5 hours!) with 590 users, while our algorithm in crisp mode requires only 83 seconds (just over a minute) and in harmonic mode it requires 961 second (16 minutes).

Figure 7.1(b) displays the difference in CPU time between the two algorithms again with synthetic data generated as it will be described in the next section. Again our algorithms outperform the matrix-based algorithm. This plot also displays 95% confidence intervals of the CPU time requirements over 5 runs for each algorithm.

Shahabi et al. [156] attempted to cluster the rows of the matrix by k -means; however, this makes the algorithm quadratic, since the dimension D of the vectors is also n .

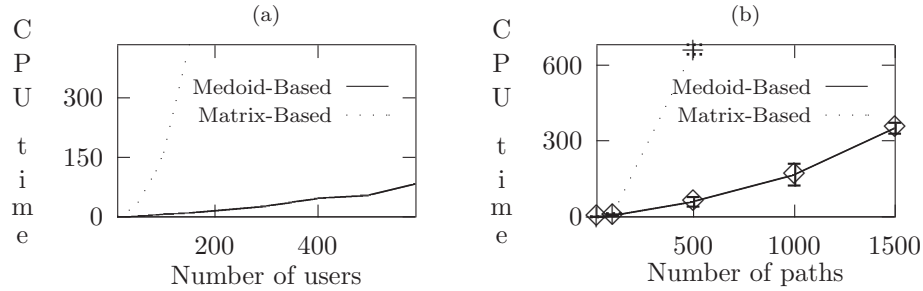


Figure 7.1: CPU-times comparison of Matrix-based algorithms and representative-based algorithms (as a function of the input size).

7.1.4 On the quality of clustering

We now evaluate the quality of the clustering by using an external process. A synthetic data set is useful in order to compare the results of the algorithms with the true clustering. Typically, synthetic data is generated from a mixture or from a set of k representatives by perturbing each slightly. The quality of the clustering is reflected by the proportion in which the clustering algorithm retrieves groups and identifies data items to their original group. We use the error rate in assigning items to their

original representative as an index of the quality of the clustering; methods with large errors on the same synthetic data set are considered poorer quality methods.

We reproduced, to the best of our ability, the synthetic generation suggested for the same task of clustering paths used by Shahabi et al. [156]. In this metaphor, the Web pages of a site are nodes in a hypergraph (the pages can have links to themselves, or there could be two pages connected by two different links). The pages (nodes) also have links going out of the site, or links arriving from the outside. In this graph we generate k random walks of length ρ (in our experiments we use $\rho = 8$ as Shahabi et al. [156]). These paths are the representatives and are called the *nucleus paths* [156]). The generation is also controlled by another parameter called the *branching factor* $br \in (0, 1)$. This encodes the probability that a user represented by a given nucleus path will choose a different link than the prototypical visitor path over nucleus path. The generation proceeds as follows. Repeatedly, one nucleus path is selected at random, and a visitation path is generated from it as a perturbation on it. While still following the visitation path, with probability $1 - br$, it continues to do so to the next node. With probability br , the path-generator branches randomly and uniformly to an adjacent node, and from there on it uniformly and randomly selects the next link for the current node (uniformly from those leaving the node) until completing a path of length ρ . Shahabi et al. [156] argue that with $br = 0.05$ and $\rho = 8$ a reasonable data set is constructed since approximately 34% of the paths are distortions of the nucleus and at different degrees of dissimilarity.

Table 7.1 shows that the crisp version of our randomized medoid-based algorithm recuperates the class correctly for approximately 73% of the paths across the range of data sizes tested. This is much better than the matrix-based algorithm. The Xiao et al. algorithm is a hill-climber where each iteration costs $O(n)$ and thus, the $O(n^2)$ version of the algorithm can only evaluate $O(n)$ swaps of columns in the dissimilarity matrix. Better quality is to be expected in the matrix-based algorithm if the hill-climber exhausts all swaps of columns, but then, the number of hill-climber iterations would be $\Theta(n^2)$ for a total cost of $O(n^3)$ of the algorithm, which is prohibitively large.

Table 7.1: Error rates of clustering results using Matrix-based algorithms and representative-based algorithms (with respect to the data generated around nucleus paths).

n	Matrix-based	Medoid-based (crisp version)
100	34%	27% ± 3.4
500	54%	27% ± 3.4
1000	n/a	27% ± 3.4
1500	n/a	27% ± 3.4

We did not compare our algorithms with the matrix-based algorithm of Perkowitz and Etzioni, because it also requires $O(n^2)$ and its goal is not to recuperate clusters, but to identify some high-density hot spots. This algorithm is thus incapable of recovering the components of a mixture.

Our crisp version has already been shown to provide much better results than matrix-based alternatives for the access-based similarity measure. In what follows, we report results comparing our harmonic EM type algorithm ($\omega^T = (1, 1/2, \dots, 1/k)$) and its crisp EM type version ($\omega^T = e_i^T$). We use the access-based similarity measure and two other measures, the frequency-based measure (refer to Equation (4.1.2)) and the order-based measure (refer to Equation (4.1.5)). The order-based measure is the same as the path measure [156]. Because our algorithms start with a random set of representatives, we ran each of them 5 times and report confidence intervals with 95% accuracy. Table 7.2 summarizes our results.

7.1.5 Discussion

There are several issues we would like to comment on in our results. The first is that we were surprised that what the literature has claimed on the issue of features from paths towards similarity measures is not reflected in our results. In fact, the more sophisticated the similarity measure, the poorer the results. Quality was much

Table 7.2: Error rates of clustering results using different versions of representative-based algorithms (with respect to generated data around nucleus paths).

Dissimilarity	n	Harmonic version	Crisp version
Access	100	24% $\pm$ 2.2	27% $\pm$ 3.4
	500	24% $\pm$ 2.2	27% $\pm$ 3.4
Frequency	100	30% $\pm$ 2.9	33% $\pm$ 4.4
	500	30% $\pm$ 2.9	33% $\pm$ 4.4
Order	100	26% $\pm$ 2.7	29% $\pm$ 4.4
	500	26% $\pm$ 2.7	29% $\pm$ 4.4

more affected by the similarity measure used than by the size of the data set or the clustering algorithm used. Our non-crisp algorithm does better, but we admit that for this data set the results are not dramatic. In fact, they are probably less impressive if we consider the CPU-time requirements. Figure 7.2 contrasts the CPU-time requirements of the harmonic and the crisp versions of our EM-type algorithms.

The harmonic version is slower. It requires a factor of $O(k)$ more space and $O(k \log k)$ administrative work. However, the observed CPU times confirm the $O(n \log n)$ nature of our algorithms and that similarity evaluation is the main cost. In fact, the access and frequency similarity functions can be computed in $O(\rho)$, where ρ is the average length of paths. However, the order-based similarity function requires $\Omega(\rho^2)$ time to be computed.

We note that the confidence intervals for the harmonic version are smaller than for the crisp version. This indicates that harmonic is more robust to initialization. To explore this issue further, we also report on some experiments regarding the robustness to initialization of our algorithms. The robustness is evaluated in a relative process that consists of estimating the discrepancy in clustering results between independent executions (with a different random seed) of the same algorithm. The validity tech-

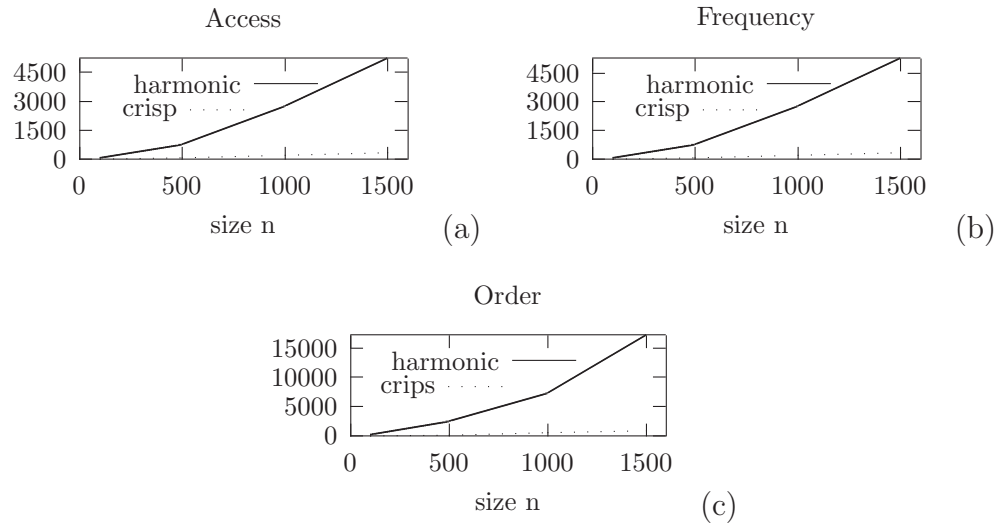


Figure 7.2: CPU-times of different versions of representative-based algorithms (crisp and harmonic versions of our EM-type algorithms).

nique used here is to measure how repeatable the results are. Table 7.3 summarizes our results. Clearly, there is much less variance with the harmonic version. Given that the initialization is random, this suggests that the algorithm finds high quality local optima with respect to its loss function. However, more thorough experimentation is required to confirm this suggestion. The experiments are not exhaustive, but they illustrate that there are benefits to be obtained with respect to quality with non-crisp classification. Moreover, they also reflect that there are serious trade-offs to be investigated between the complexity of the similarity function and its computational requirements.

7.2 Case study 2: Text mining

Text categorization is a problem that has received a lot of attention in information retrieval lately [30, 127], mainly due to the increased need for automated methods in clustering large volumes of text documents, in particular vast amounts of text data available through the Web. In this section, we conducted experiments on a

Table 7.3: The proportion of discrepancies in clustering (from independent runs of the same algorithm).

Dissimilarity	n	Harmonic version	Crisp version
Access	100	7 %	11 %
	500	7.12 %	7.35 %
Frequency	100	6.6 %	10.2 %
	500	6.01 %	7.20 %
Order	100	5.75 %	8.75 %
	500	4.15 %	6.34 %

well-known REUTERS-21578 data set [108]. This data set has been used before as a benchmark for representative-based clustering in data mining [22,65]. We reproduced the experiments to assess our algorithms proposed in Chapter 3.

7.2.1 Data preparation

We derived two data sets from the original text file `reut2-all.sgm` consisting of 21,578 SGML marked documents. Data patterns in these two data sets are feature vectors. Data set one (X1) has 302 dimensions, while data set two (X2) has 135 dimensions. The features (columns) in X1 correspond to the 302 most frequently used words (key words) in the entire collection. A record in X1 corresponds to a document and has the counts for these 302 key words as they appear in the document. The features in X2 correspond to the 135 topics in the entire collection. A record in X2 is a vector of counts for these 135 topics. Data sets X1 and X2 are used in our experiment as input data for clustering analysis.

The 302 key words are extracted from all nouns between the tag pairs ($\langle TEXT \rangle$ $\langle /TEXT \rangle$), ($\langle TITLE \rangle$ $\langle /TITLE \rangle$) and ($\langle BODY \rangle$ $\langle /BODY \rangle$)

except the stop words ². Abbreviations like *mln*, *pct* and *corp* are included.

There is a standard set of 135 topic words for the Reuters data set, but not all topics appear in the data set. Actually only 120 topic words appear. Although some documents have no topic, we do not exclude them in our experiment.

7.2.2 Clustering methods

Our experiment evaluates clustering results of five different clustering methods we discussed in Chapter 3. These are our two algorithms *k*-D-FM-CENTERS and *k*-C-FM-CENTERS, our novel implementation of *k*-C-L1-CENTERS and two fast versions of *k*-MEANS, a simple standard *k*-MEANS and also *k*-HARMONIC MEANS. We would like to identify which of these representative-based clustering algorithms provides the best clusters.

7.2.3 Validity indices

We compare a group of given algorithms through the relative process where the algorithms are assessed against some criteria. The algorithms are favorable if they are explicitly attempting to optimize such criteria. We use the total gravity as an index, because this is the criterion used by *k*-MEANS. We use this to show that *k*-MEANS has actually the poorest performance even for the criterion it is explicitly optimizing. The gravity of each cluster is the sum of squared Euclidean distance of all points in the cluster C_j to their representative, and the total gravity is the sum of gravity over all clusters C_j .

Since clustering aims at reducing diversity within a class, we also use entropy on the attribute vectors to measure the diversity in clustering results. For the entropy analysis, we first compute dissimilarity inside each cluster. For a given cluster C_j

²<http://swish-e.org/stopwords.txt>

Table 7.4: Algorithms ranked by quality of the clustering on Reuters data sets (consisting of 21,578 records).

$n = 21,578$	Data set X1			Data set X2		
	(dimensions $D = 302$)			(dimension $D = 135$)		
Algorithm	Entropy	Gravity	CPU s	Entropy	Gravity	CPU s
k -D-FM-CENTERS	440	2 622,467	3,521	40	3,742	312
k -C-FM-CENTERS	423	2 764,454	885	49	4,093	226
k -MEANS	505	2 782,003	288	62	5,713	72
k -HARMONIC MEANS	524	3 342,582	768	41	10,034	75
k -C-L1-CENTERS	553	3 024,238	2,066	68	10,983	739

with n_j elements and D dimensions, after normalizing each vector, we add all the values in each attribute. That is, we obtain the vector $\vec{e} = \sum_{\vec{x}_i \in C_j} \frac{\vec{x}_i}{\|\vec{x}_i\|}$. Then, the entropy of cluster C_j is given by

$$E(C_j) = - \sum_{i=0}^D \frac{e_i}{n_j} \log_2\left(\frac{e_i}{n_j}\right). \quad (7.2.1)$$

Finally, the entropy of a clustering is just the sum of the entropy in all clusters. The clustering method that has smaller entropy has reduced more diversity in the clusters.

7.2.4 Experimental results

We performed this experiment on implementations in Java. Table 7.4 shows a summary of the experimental results. Clearly, for both entropy and gravity assessment of the clusters, our algorithms obtain better clustering results. Also, our algorithms are scalable. They are linear in D (the dimension), and $O(n \log n)$ time is required, where n is the size of the data sets. The CPU-time measurements show that they are a constant factor of about 5 slower than fast versions of k -MEANS (this is across the arguments n and D). However, they compensate for this fixed overhead by much improved clustering results. The readers may note that once k -HARMONIC MEANS performed better with respect to the entropy index, but much worse with respect

to the gravity index. This indicates that the gravity is inherently a less effective clustering criterion, and is sensitive to outliers [123].

Chapter 8

Conclusions and research directions

This chapter summarizes the contents of this thesis, discusses implementation issues and also investigates new research directions indicated by the advances of the research in this thesis.

8.1 Summary

The increasing size and sophistication of databases and the World Wide Web highlight the demand of data mining and clustering techniques. While there are a number of activities in the clustering process, the main issues are: design and analysis of algorithms, assessment of their implementations and clustering results, and immediate applications. This thesis has discussed those objectives in previous chapters.

- **Algorithmic engineering of clustering.** Algorithmic engineering of clustering aims at designing algorithms that are theoretically efficient and are practical in solving problems arising in modern data engineering, and in applications re-

lated to complex resource management. This thesis concentrates on discovering new algorithmic concepts, identifies key algorithmic problems, deals with the solution of optimization problems in clustering analysis, and contributes to the accelerated transfer of advanced algorithmic techniques into application, with special emphasis on those involving large data sets. In large data sets, an algorithm of reduced theoretical complexity that can be applied in practical settings and produces equivalent or better quality, is of much value. Moreover, the larger the data sets, the more significant the impact of the better algorithms.

Designing efficient algorithms under limitations, such as the limitations on running time, on robustness, or on the features of the data, is a compelling subject in computer science. This thesis circumvents the constraints on clustering algorithms by examining the clustering context.

In the thesis, we investigated the characteristics of various data models like representative-based models, support vector based models, and graph based models. Representative-based modeling results in vector quantizers that encode each cluster by its representative, the most centrally located point of the cluster. Therefore, the representatives can be regarded as a reduced set of reference vectors of a data set, and thus they are used to compress large reference sets. The features of representatives play an important role in modeling clusters. Support vector modeling describes a cluster structure in terms of support vectors, which are the extreme vectors in boundaries. More importantly, support vector modeling implements kernel methods to transform data into a feature space. Graph modeling employs proximity graphs to reflect the proximity and topological relationships among data points. It greatly enhances the effectiveness of data modeling, and reduces the search space in detecting clusters.

We discussed the optimization criteria derived from induction principles. For distance-based approaches, we analyzed absolute error estimation and squared error estimation. For the SVND approach, we dealt with the volume of a hypersphere that covers the data points in feature space.

We designed clustering algorithms for two distinctive data patterns, vector data

and sequential data. We proposed fast and robust general purpose clustering algorithms for vector data patterns. These algorithms are suitable for exploratory data analysis. They produce clusters with improved results. They do not depend on the order of the data, as some variants of k -MEANS and they do not demand detailed initialization. Their use brings an insight into the structure of a large multidimensional data set. We also proposed a generic framework of medoid-based, sub-quadratic algorithms on arbitrary metric spaces, where we generalized fuzzy membership and employed sampling strategies. In support vector clustering, we applied proximity graphs to enhance data modelling. Our combination of support vector modeling and proximity graph modeling results in more robustness and higher efficiency for cluster assignment.

The discussion of algorithmic engineering of clustering is of both theoretical and practical significance. First, this research aims to contribute to the development of a unified and well defined theory, with respect to computation practice, for clustering algorithms. Second, it aims to exploit this theory in real world applications and to lead to practical algorithmic tools. The algorithms designed in this thesis have been shown to have the claimed complexity theoretically as well as practically (in several implementations). Also, the quality of their results has been illustrated experimentally (or theoretically when mathematical challenge has been available).

- **Performance analysis of clustering and cluster validity.** Interpretability and usability of clustering results are of fundamental importance. We summarized the cluster validity problems. We developed an analytical and computational method to describe the behavior of clustering algorithms and evaluate their outputs. Cluster validity is a certain amount of confidence that the detected cluster structure is significant. Intuitively, if clusters are isolated from each other and each cluster is compact, the clustering results are somehow significant. We applied SVMs and related kernel methods to cluster validity. SVM training based on clustering results can give an insight into the structure inherent in data. By analyzing the complexity of boundaries through support

information, we can verify separation performance and potential outliers. This provides a novel mechanism for cluster evaluation.

- **Immediate applications.** We intend to elaborate feasible and more efficient solutions to important applications. We have conducted two case studies. One is Web usage mining, and the other is text mining. For these applications, besides the core discipline algorithm design and algorithmic engineering, there is an interdisciplinary endeavor requiring a close cooperation between computer science and application domains, such as Web intelligence.

In Web usage mining, our goal is to apply clustering methods to group Web visitors and analyze their behavior based on their access history. Clustering visitation paths is to play a central role in identifying whether a Web site is being used as expected, in optimizing the performance of a Web server, or in discovering which products are being purchased by which visitors. We modeled the clustering optimization problems for Web usage mining, and solved them using clustering algorithms on metric space.

Text categorization presents unique challenges due to the large number of samples present in data sets, large number of attributes and attribute dependencies. In text mining, we derived two data sets from the original SGML file `reut2-all.sgm` in a well-known REUTERS-21578 data set. We carried out cluster analysis on the two derived data sets by utilizing representative clustering algorithms on vector space.

8.2 Implementation issue

In order to demonstrate and visualize our clustering methods, we have developed the following software packages in the Object-Oriented approach:

1. ***k*-group clustering analyzer.** This Java software package implements *k*-group clustering methods with models using different types of representatives, and

with induction principles using different distance metrics. It accepts multi-dimensional feature vectors as inputs, and provides 2D visualization and descriptive reports on outputs. The graphical user interface of this tool is shown in Figure 8.1.

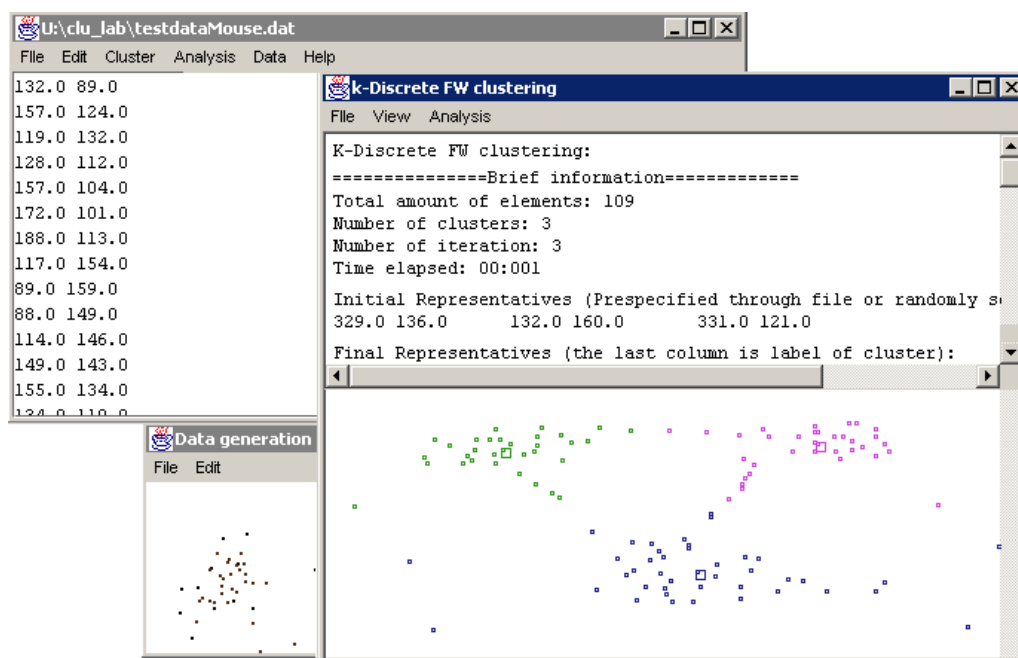


Figure 8.1: GUI of k -group clustering analyzer.

2. **Web usage miner.** This VC++ application implements methods for sequential data clustering. The software solution consists of representative-based, matrix-based and pattern-based algorithms. It provides analytical reports on clustering results, and it is featured with visualization and database connectivity. A snapshot of the graphical user interface is shown in Figure 8.2.
3. **Cluster validity detector.** This is developed in VC++ and integrated with LEDA version 4.2 [117] and ANN version 0.2¹ [158] libraries. It provides a number of useful features, such as analytical quality measures, 2D and 3D animation,

¹For documentation and code see <http://www.cs.sunysb.edu/~algorithm/implement/ANN>.

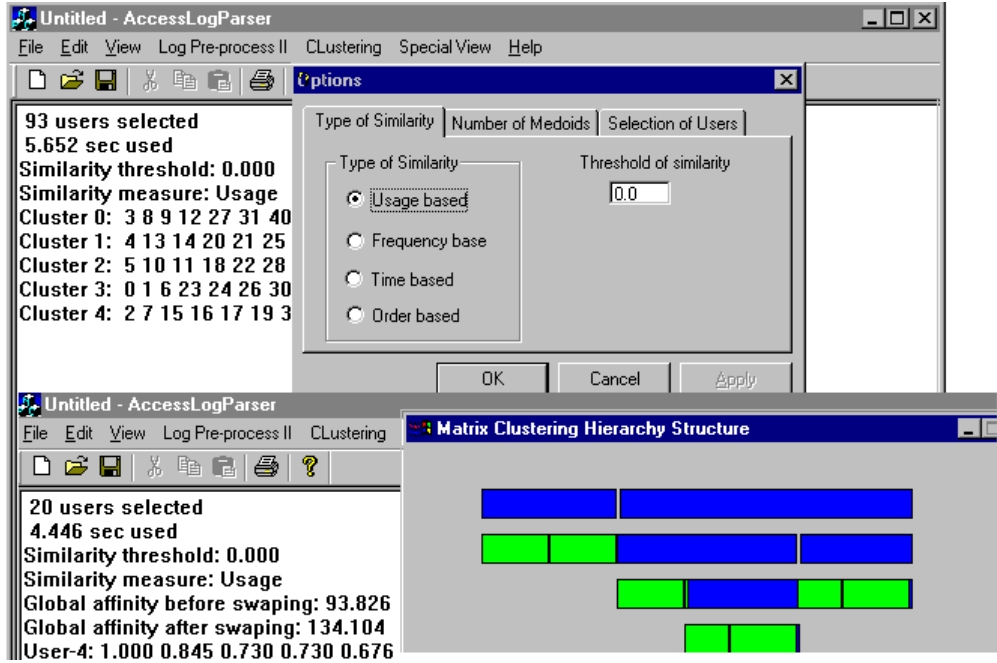


Figure 8.2: GUI of Web usage miner.

support vector information and the framework of proximity graph modeling. In addition, it implements SVC. Figure 8.3 shows a snapshot of the graphical user interface.

8.3 Research directions

Our experience with algorithmic engineering of clustering indicates that the clustering algorithms we have proposed are fast and robust, and can be used in important applications with massive data. New avenues opened by these results consist of:

- **Distance-based clustering of association rules.** Association rule mining is one of the most important procedures in data mining. In current industry applications, often more than 10,000 rules are discovered [76]. To allow manual inspection and support knowledge discovery, the number of rules has to be re-

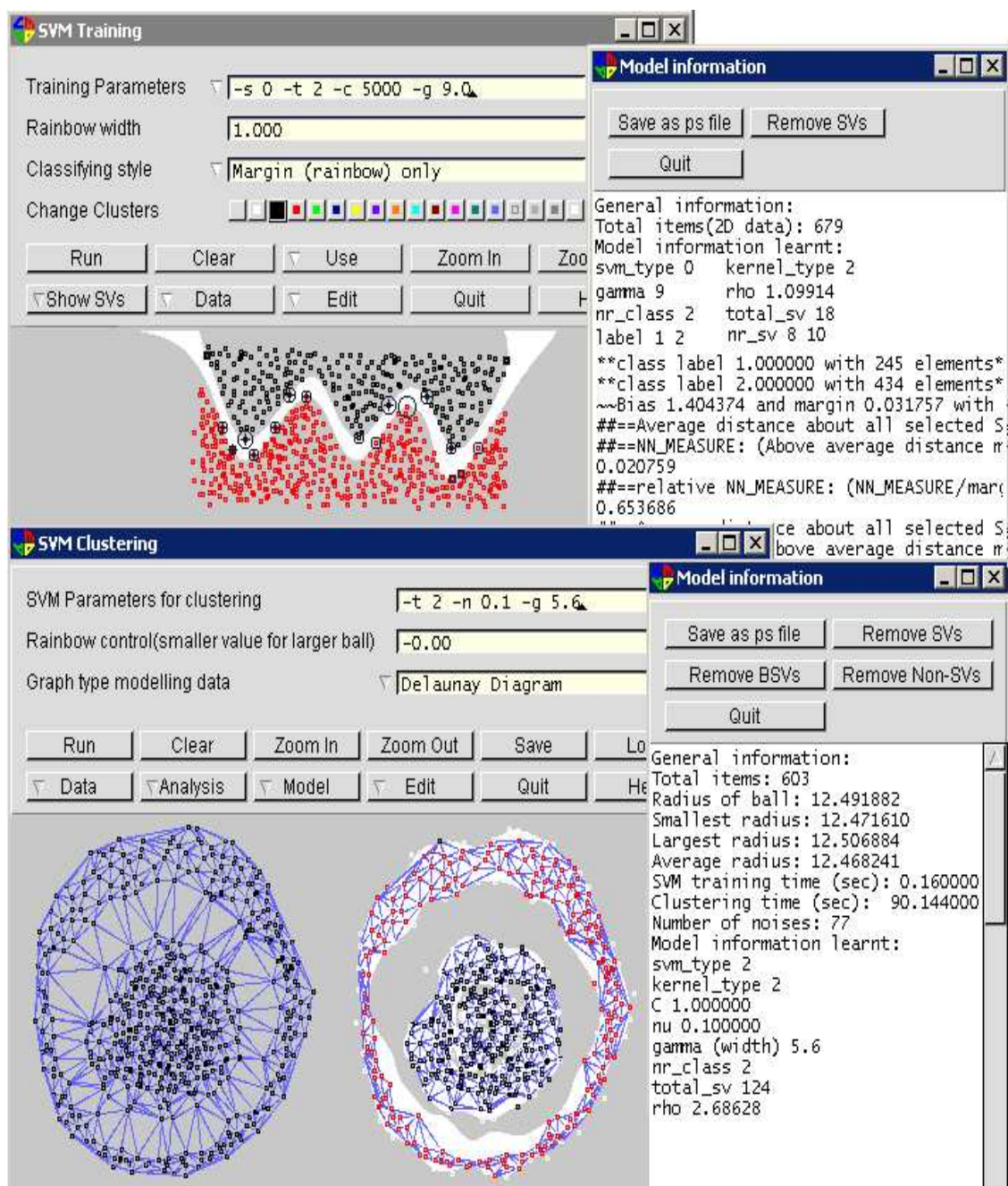


Figure 8.3: GUI of cluster validity detector.

duced significantly by techniques such as pruning and grouping. A key reason for the clustering of association rules is to obtain more concise and abstract descriptions on the data. Merging rules of the same cluster result in jointed meta-rules that may be used to support merging decisions. Our representative clustering methods on metric space can potentially be applied to group association rules, since the input objects required by these algorithms do not need to be feature vectors. Once obtaining a distance metric to measure the similarity between association rules in an embedding space, such a distance metric can be used in the optimizing criteria of these algorithms.

- **New labeling strategy of SVC.** Although our cluster labeling strategy dramatically improves the efficiency of SVC, there is still room for its improvement. We plan on designing a new labeling strategy of SVC, which merges the phases of cluster labeling and harvesting, and utilizes sophisticated data structures.

The time complexity of the cluster labeling can also be reduced by employing a parallel implementation of the algorithm. This can possibly be achieved because the cluster labeling of SVC is trivially parallelizable. The Message Passing Interface (MPI)² standard is a possibility to distribute parts of the process to different processors. Note that, in parallel formulations, load imbalance causes idling of processors. Also, parallel programming incurs communication overheads. Thus, the parallel algorithm for cluster labeling in SVC should be designed to achieve good speedups with as little overhead as possible.

- **Hypertext clustering and data management.** Hypertext classification poses new research challenges because of the rich representation of a document and the connectivity between documents [30]. Hyperlinks, meta-data and documents all provide rich sources of information for hypertext classification, which are not major concerns in traditional text classification. Given this, the question of how to effectively use such information becomes important in designing clustering algorithms. From the algorithmic engineering perspective, new requirements

²For documentation and tools see <http://www-unix.mcs.anl.gov/mpi/index.html>.

emerge during the designing of hypertext clustering algorithms. The algorithms should be powerful for discriminating informative links from noisy links; they should exploit meta-data semantics; they should be robust in dealing with a noisy vocabulary of hundreds of thousands of words.

Meta-data is an essential component of any data administration function. It is at the heart of data management by providing meaning and coherence to data objects. Designing fast and robust meta-data clustering algorithms should be a new research direction.

- **Clustering for information fusion.** Information fusion has been conceived and employed to investigate the benefits of combining the results of different information retrieval schemes to improve performance (qualitatively or quantitatively, in terms of accuracy, robustness etc.). Researchers have made an important observation that a different query expert can contribute to improved performance in the combined system even when it performs poorly on its own [10]. So meta-search with fusion technique turned out to be a wiser approach to improve the Internet retrieval performance, where the results from multiple search engines are combined to generate a unified, potentially relevant document in response to presentation of a single query to a number of query systems [174]. SavvySearch [89] and Inquirus [105] are examples of such meta-search engines. The majority of meta-search engines follow conventional document retrieval systems and return long lists of ranked documents. The information fusion techniques employed in these meta-search engines rank searched results by estimating their scores. Due to the low efficiency of ranking each individual result, grouping similar results and encoding a cluster of results with its representative is of much value in information fusion. Consequently, our representative-based clustering methods can be used in information fusion.
- **Visualization of clustering results.** The information in a clustering solution is extensive. Visualizing the clustering results can help to quickly assimilate this information and provide insights that support and complement textual de-

scriptions or statistical summaries. For example, we wish to know quickly how well the clusters are defined, how distinctive they are from each other, what their sizes are, and whether the observations belong strongly to the cluster or only marginally. Visualizing a clustering solution has many potential uses [39]. From the algorithmic engineering perspective, during iterative model building or testing processes, the analyst user can quickly obtain an insight from the visualization that suggests the adequacy of the solution, and also suggests further experiments to conduct. Alternatively, the business user can examine and query the final clustering solutions using the visualization. While most methods for visualizing clustering results are application dependent, describing a general framework to display the information in a clustering solution is of much value. Visualizing 2D or 3D spatial data is relatively straightforward; but for the case of large multidimensional data sets, effective visualization of data is more difficult. Although research in information visualization [48, 164] has provided many approaches to viewing data, including less obvious methods such as 3D scenes, cityscapes, nested boxes and cubes and alternative geometries, how to layout data to reflect a cluster structure is still a challenge. This indicates a new research direction of visualizing clustering results by sophisticated graph drawing techniques, where specialized drawing rules may be required.

Bibliography

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann Publishers, 12–15 1994.
- [2] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.
- [3] M. S. Aldenderfer and R. K. Blashfield. *Cluster Analysis*. Sage Publications, Beverly Hills, USA, 1984.
- [4] T. M. Apostol. *Calculus — Volume II*. John Wiley, New York, second edition, 1969.
- [5] S. F. Arnold. Gibbs sampling. In C.R. Rao, editor, *Handbook of Statistics 9*, pages 599–625, Amsterdam, 1993. North Holland.
- [6] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k -medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*. ACM Press, 1998.
- [7] T. C. Bailey and A. C. Gatrell. *Interactive Spatial Analysis*. Longman Scientific & Technical, Harlow, UK, 1995.

- [8] C. Bajaj. Proving geometric algorithm non-solvability: An application of factoring polynomials. *Journal of Symbolic Computation*, 2:99–102, 1986.
- [9] J. D. Banfield and A. E. Raftery. Model-Based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, September 1993.
- [10] N. J. Belkin, C. Cool, W. B. Croft, and J. P. Callan. Effect of multiple query representations on information retrieval system performance. In *Research and Development in Information Retrieval*, pages 339–346, 1993.
- [11] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.
- [12] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. A support vector method for hierarchical clustering. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.
- [13] K. P. Bennett and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.
- [14] M. J. A. Berry and G. S. Linoff. *Data Mining Techniques for Marketing, Sales and Customer Support*. John Wiley & Sons, Inc., New York, 1997.
- [15] J. C. Bezdek. Cluster validity with fuzzy sets. *Journal of Cybernetics*, 3(3):58–72, 1974.
- [16] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.
- [17] J. C. Bezdek. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):301–315, 1998.
- [18] J. P. Bigus. *Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support*. McGraw-Hill, NY, 1996.
- [19] C. Bishop. Novelty detection and neural network validation. *IEE Proceedings on Vision, Image & Signal Processing*, 141(4):217–222, 1994.

- [20] J. Borges and M. Levene. Mining association rules in hypertext databases. In R. Agrawal, editor, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 149–153, NY, August 27–31 1998.
- [21] N. Boujemaa. Generalized competitive clustering for image segmentation. In *19th International Meeting of the North American Fuzzy Information Processing Society NAFIPS*, Atlanta, USA, Jul 2000.
- [22] P. S. Bradley and U. Fayyad. Refining the initial points in k -means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99. Morgan Kaufmann Publishers, 1998.
- [23] P. S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15. AAAI Press, 1998.
- [24] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. *Advances in neural information processing systems*, 9:368–374, 1997.
- [25] J. Buhmann. Learning and data clustering. In M. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, volume ix. Bradford Books/MIT Press, Secaucus, NJ, 1995.
- [26] J. Buhmann. Empirical risk approximation: An induction principle for unsupervised learning. Technical Report IAI-TR-98-3, 3, 1998.
- [27] M. G. Bulmer. *Principles of Statistics*. Dover, NY, second edition, 1979.
- [28] G. Casella and R. L. Berger. *Statistical Inference*. Wadsworth & Brooks/Cole, Belmont, CA, 1990.
- [29] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793, 1995.

- [30] S. Chakrabarti. Data mining for hypertext: A tutorial survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, ACM, 1, 2000.
- [31] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [32] C. C. Chang and C. J. Lin. Training ν -support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001.
- [33] M.-S. Chen, J. Han, and P. S. Yu. Data mining: an overview from a database perspective. *IEEE Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.
- [34] V. Cherkassky and F. Mulier. *Learning from data: concepts, theory and methods*. John Wiley and Sons, 1998.
- [35] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the World Wide Web, 1997.
- [36] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining World Wide Web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
- [37] A. Culter and M. P. Windham. Information-based validity functionals for mixture analysis. In H. Bozdogan, editor, *Proceedings of the first US-Japan Conference on the Frontiers of Statistical Modeling*, pages 149–170, Amsterdam, 1993. Kluwer.
- [38] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of World Wide Web Client-based Traces. Technical Report BUCS-TR-1995-010, Computer Science Department, Boston University, April 1995. [Available at <http://www.cs.bu.edu/techreports/pdf/1995-010-www-client-traces.pdf>].
- [39] I. Davidson. Visualizing clustering results. In *Proceedings of the Second SIAM International Conference on Data Mining*, Arlington VA, USA, April 2002.

- [40] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.
- [41] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.
- [42] P. Densham and G. Rushton. A more efficient heuristic for solving large p -median problems. *Papers in Regional Science*, 71:307–329, 1992.
- [43] D. Dowe, R. A. Baxter, J. J. Oliver, and C. Wallace. Point estimation using the Kullback-Leibler loss function and MML. In X. Wu, R. Kotagiri, and K. K. Korb, editors, *Proceedings of Second Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD-98*, pages 87–95, Melbourne, Australia, 1998. Springer-Verlag Lecture Notes in Artificial Intelligence 1394.
- [44] R. C. Dubes. How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645–663, 1987.
- [45] R. C. Dubes. Cluster analysis and related issues. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 1.1, pages 3–32. World Scientific, Singapore, 1993.
- [46] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, Inc., New York, 1973.
- [47] J. C. Dunn. A fuzzy relative isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1974.
- [48] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181, 2000.
- [49] F. Esposito, D. Malerba, and V. Tamma. Dissimilarity measures in symbolic data analysis. In *Proceedings of Cladag-SIS' 99*, Rome, 5-6 July 1999.

- [50] F. Esposito, D. Malerba, V. Tamma, and H. Bock. Classical similarity and dissimilarity measures (section 8.1). In *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data, Series: Studies in Classification, Data Analysis and Knowledge Organisation*, volume 15, pages 139–152. Springer-Verlag, Berlin, 1999.
- [51] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Portland, OR, 1996.
- [52] V. Estivill-Castro. Why so many clustering algorithms - a position paper. *SIGKDD Explorations*, 4(1):65–75, 2002.
- [53] V. Estivill-Castro and M. E. Houle. Robust clustering of large geo-referenced data sets. In N. Zhong and L. Zhou, editors, *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, pages 327–337. Springer-Verlag Lecture Notes in Artificial Intelligence 1574, April 1999.
- [54] V. Estivill-Castro and M. E. Houle. Fast randomized algorithms for robust estimation of location. In J. Roddick and K. Hornsby, editors, *Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining - TSDM2000, in conjunction with the 4th European Conference on Principles and Practices of Knowledge Discovery and Databases*, pages 74–85, Lyon, France, 2000. Springer-Verlag Lecture Notes in Artificial Intelligence 2007. to appear.
- [55] V. Estivill-Castro and M. E. Houle. Data structures for minimization of total within-group distance for spatio-temporal clustering. *Lecture Notes in Artificial Intelligence*, (2168), 2001.
- [56] V. Estivill-Castro and M.E. Houle. Robust distance-based clustering with applications to spatial data mining. *Algorithmica*, 30(2):216–242, 2001.

- [57] V. Estivill-Castro and I. Lee. Amoeba: Hierarchical clustering based on spatial proximity using Delaunay diagram. In *Proc. of the 9th Int. Symposium on Spatial Data Handling*, pages 7a.26–7a.41, 2000.
- [58] V. Estivill-Castro and I. Lee. AUTOCLUST: Automatic clustering via boundary extraction for mining massive point-data sets. In J. Abraham and B. H. Carlisle, editors, *Proceedings of the 5th International Conference on Geocomputation*, Kent, UK, 2000.
- [59] V. Estivill-Castro, I. Lee, and A. T. Murray. Criteria on proximity graphs for boundary extraction and spatial clustering. *LNCS*, 2035:348–347, 2001.
- [60] V. Estivill-Castro and A. T. Murray. Discovering associations in spatial data - an efficient medoid based approach. In X. Wu, R. Kotagiri, and K. K. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98)*, pages 110–121, Melbourne, Australia, 1998. Springer-Verlag Lecture Notes in Artificial Intelligence 1394.
- [61] V. Estivill-Castro and J. Yang. Categorizing visitors dynamically by fast and robust clustering of access logs. *Lecture Notes in Computer Science*, 2198:498–507, 2001.
- [62] R. L. Berlin et al. *CGI Programming*. Sams.net, Indianapolis, IN 46290, 1996.
- [63] D. Fasulo. An analysis of recent work on clustering algorithms. Technical Report UW-CSE01-03-02, Department of Computer Science & Software Engineering, University of Washington, 1999. [Available at <http://www.cs.washington.edu/homes/dfasulo/clustering.ps>].
- [64] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.
- [65] U. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In R. Agrawal and P. Stolorz, editors, *Proceedings of the*

- Fourth International Conference on Knowledge Discovery and Data Mining*, pages 194–198. AAAI Press, 1998.
- [66] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the 2th International Conference on Knowledge Discovery and Data Mining*, pages 82–88, Portland, Oregon, 1996. AAAI Press.
- [67] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.). *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.
- [68] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.
- [69] R. L. Francis. *Facility Layout and Location*. Prentice-Hall, New Jersey, 1974.
- [70] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
- [71] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995.
- [72] E. Gokcay and J. Principe. A new clustering evaluation function using Renyi’s information potential. In J. Tian, R. G. Baraniuk, R. O. Wells, D. M. Tan, and H. R. Wu, editors, *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2000)*, pages 3490–3493, Istanbul, Turkey, 2000.
- [73] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large database. In *Proc. of the ACM SIGMOD Conference*, 1998.
- [74] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

- [75] S. Gunn. Support vector machines for classification and regression. Technical Report ISIS-1-98, Department of Electronics and Computer Science, University of Southampton, 1998.
- [76] G. Gupta, A. Strehl, and J. Ghosh. Distance based clustering of association rules. *Intelligent Engineering Systems Through Artificial Neural Networks*, 9:759–764, 1999.
- [77] S. K. Gupta, K. S. Rao, and V. Bhatnagar. K-means clustering algorithm for categorical attributes. In M. Mohania and A. M. Tjoa, editors, *Data Warehousing and Knowledge Discovery DaWaK-99*, pages 203–208, Florence, Italy, 1999. Springer-Verlag Lecture Notes in Computer Science 1676.
- [78] V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM 2001)*, pages 179–186, San Jose, California, USA, November 2001.
- [79] R. P. Haining. *Spatial data analysis in the social and environmental sciences*. Cambridge University Press, New York, 1990.
- [80] M. Halkidi, M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. In *Proceedings of Principles and Practice of Knowledge Discovery in Databases (PKDD'2000)*, Lyon, France, 2000.
- [81] L. O. Hall, I. B. Özyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, July 1999.
- [82] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, 1997.
- [83] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publisher, San Francisco, C.A., 2000.

- [84] S. S. Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall International, Upper Saddle River, N.J., 1999.
- [85] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.
- [86] A. Hitchinson. *Algorithmic Learning*. Clarendon Press, Oxford, UK, 1994. Graduate Texts in Computer Science.
- [87] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis : Methods for Classification, Data Analysis, and Image Recognition*. John Wiley, 1999.
- [88] M. Horn. Analysis and computation schemes for p -median heuristics. *Environment and Planning A*, 28:1699–1708, 1996.
- [89] A. E. Howe and D. Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.
- [90] Z. Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the First Asia-Pacific Conference on Knowledge Discovery and Data Mining*, Singapore, 1997. World Scientific.
- [91] Z. Huang. Extensions to the k -means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):2283–304, 1998.
- [92] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall International, New Jersey, 1998. Advanced Reference Series: Computer Science.
- [93] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.
- [94] M. Jambu and M.-O. Leebaux. *Cluster Analysis and Data Analysis*. North-Holland, Amsterdam, 1983.

- [95] M. I. Jordan and R. I. Jacobs. Supervised learning and divide-and-conquer: A statistical approach. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 159–166. Morgan Kaufmann Publisher, 1993.
- [96] H. Kato, T. Nakayama, and Y. Yamane. Navigation analysis tool based on the correlation between contents distribution and access patterns. In *Proceedings of the Web Mining for E-Commerce Workshop (WEBKDD-2000)*, Boston, MA, USA, August 20 2000.
- [97] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., New York, 1990.
- [98] J. M. Keil and C. A. Gutwin. The Delauney Triangulation Closely Approximates the Complete Euclidean Graph. In F. K. H. A. Denhe, J.-R. Sack, and N. Santoro, editors, *Proceedings of the First Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science 382, pages 47–56, Ottawa, Canada, 1989. Springer.
- [99] K. Koperski. *A Progressive Refinement Approach to Spatial Data Mining*. PhD thesis, Computing Science, Simon Fraser University, B.C., Canada, 1999.
- [100] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explorations*, 2(1):1–15, 2000.
- [101] R. Koschke and T. Eisenbarth. A framework for experimental evaluation of clustering techniques. In *Proceedings of the 8th International Workshop on Program Comprehension (IWPC'00)*, Limerick, Ireland, June 2000.
- [102] H. W. Kuhn. On a pair of dual non-linear problems. In J. Abadie and S. Vajda, editors, *Nonlinear programming*, page Chapter 3, New York, 1967. John Wiley.
- [103] H. W. Kuhn. A Note on Fremat's Problem. *Mathematical Programming*, 4:98–107, 1973.

- [104] H. W. Kuhn and R. E. Kuenne. An Efficient Algorithm for the Numerical Solution of the Generalized Weber Problem in Spatial Economics. *Journal of Regional Science*, 4(2):21–33, 1962.
- [105] S. Lawrence and C. Lee Giles. Inquirus, the NECI meta search engine. In *Seventh International World Wide Web Conference*, pages 95–105, Brisbane, Australia, 1998. Elsevier Science.
- [106] I. Lee. *Multi-Purpose Boundary-Based Clustering on Proximity Graphs for Geographical Data Mining*. PhD thesis, School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan, NSW 2308, Australia, February 2002.
- [107] E. Levine and E. Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13:2573–2593, 2001.
- [108] D. D. Lewis. *Reuters-21578, distribution 1.0*, 1997. Data available at <http://www.daviddlewis.com/resources/testcollections/reuters21578>.
- [109] M. Vazirgiannis M. Halkidi, Y. Batistakis. On clustering validation techniques. *Intelligent Information Systems Journal (Special Issue on Scientific and Statistical Database Management)*, 2001.
- [110] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. Le Cam and J. Neyman, editors, *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967. Volume 1.
- [111] S. K. Madria, S. S. Bhowmick, W. K. Ng, and E.-P. Lim. Research issues in Web data mining. In *Data Warehousing and Knowledge Discovery*, pages 303–312, 1999.
- [112] J. Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, 7(1):16–29, 1996.
- [113] S. Massa, M. Paolucci, and P. P. Puliafito. A new modelling technique based on Markov chains to mine behavioral patterns in event based time series. In

- M. Mohania and A. M. Tjoa, editors, *Data Warehousing and Knowledge Discovery DaWaK-99*, pages 331–342, Florence, Italy, 1999. Springer-Verlag Lecture Notes in Computer Science 1676.
- [114] W. T. McCormick and P. J. Schweitzer. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20:993–1009, 1972.
- [115] L. McKee. Geography Connects Cyberspace with the Real World. *GEO World*, February 2001.
- [116] W. McKnight. The crm-ready data warehouse: Packaged business intelligence (pbi). *DM Review*, February 2002. Stay Informed: DM Review Magazine Archived Article.
- [117] K. Mehlhorn and S. Näher. *LEDA A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK, 1999.
- [118] I. Meilijson. A fast improvement to the EM algorithm in its own terms. *Journal of the Royal Statistical Society B*, 51(1):127–138, 1989.
- [119] R. S. Michalski and R. Stepp. Automated construction of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:219–243, 1983.
- [120] H. J. Miller and J. Han. *Geographic Data Mining and Knowledge Discovery: An Overview*. Cambridge University Press, Cambridge, UK, 2001.
- [121] A. T. Murray. Spatial characteristics and comparisons of interaction and median clustering models. *Geographical Analysis*, 32(1):1–18, 2000.
- [122] A. T. Murray and R. L. Church. Applying simulated annealing to location-planning models. *Journal of Heuristics*, 2:31–53, 1996.
- [123] A. T. Murray and V. Estivill-Castro. Cluster discovery techniques for exploratory spatial data analysis. *ijgis*, 12(5):431–443, 1998.

- [124] F. Murtagh. Comments on parallel algorithms for hierarchical clustering and cluster validity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):1056–1057, 1992.
- [125] R. M. Neal. *Software for Flexible Bayesian Modeling, Version of 1999-03-13. Copyright (c) 1995-1999*, 1999. Documentation available at <http://www.cs.toronto.edu/~radford/fbm.1999-03-13.doc/index.html>.
- [126] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of 20th International Conference on Very Large Data Bases*, pages 144–155, Santiago, Chile, September 12–15 1994. Morgan Kaufmann Publishers.
- [127] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- [128] J. J. Oliver, R. A. Baxter, and C. S. Wallace. Unsupervised Learning Using MML. In *Machine Learning: Proceedings of the Thirteenth International Conference (ICML 96)*, pages 364–372. Morgan Kaufmann Publishers, 1996.
- [129] M. L. Overton. A Quadratically Convergent Method for Minimizing a Sum of Euclidean Norms. *Mathematical Programming*, 27:34–63, 1983.
- [130] N. R. Pal and J. Riswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6), 1997.
- [131] J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu. Mining Access Patterns Efficiently from Web Logs. In T. Terano, H. Liu, and A.L.P. Chen, editors, *Proceedings of the Pacific-Asia conference on knowledge discovery and data mining*, pages 396–407, Kyoto, Japan, April 4th 2000. Springer-Verlag.
- [132] M. Perkowitz and O. Etzioni. Adaptive Web sites: an AI challenge. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 16–23, Nagoya, Japan, August 1997. IJCAI.

- [133] M. Perkowitz and O. Etzioni. Adaptive Web sites: Automatically synthesizing Web pages. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 727–732, Madison, WI, July 1998. American Association for Artificial Intelligence, AAAI Press.
- [134] J. G. Postaire, R. D. Zhang, and C. Lecocq-Botte. Cluster analysis by binary morphology. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(2):170–180, 1993.
- [135] J. Punin, M. Krishnamoorthy, and M. Zaki. Logml: Log markup language for Web usage mining. In *Proceedings of the WEBKDD*, San Francisco, USA, August, 2001.
- [136] O. Teytaud R. Jalam. Kernel based text categorization. In *Proceedings of IJCNN 2001*, pages 1892–1897, 2001.
- [137] A. Rauber, J. Paralic, and E. Pampalk. Empirical evaluation of clustering algorithms. In M. Malekovic and A. Lorencic, editors, *Proceedings of the 11th International Conference on Information and Intelligent Systems (IIS'2000)*, Varazdin, Croatia, September 20. - 22. 2000. University of Zagreb.
- [138] S. Ray and R. H. Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In N. R. Pal, A. K. De, and J. Das, editors, *Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99)*, pages 137–143, New Delhi, India, December 1999.
- [139] M. R. Rezaee, B. P. F. Lelieveldt, and J. H. C. Reiber. A new cluster validity index for the fuzzy c-mean. *PRL*, 19(3-4):237–246, March 1998.
- [140] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [141] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49(3):223–239, 1987.

- [142] G. Ritter and M. T. Gallegos. Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, 18:525–539, 1997.
- [143] S. Roberts. Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognition*, 30(5):261–272, 1997.
- [144] S. J. Roberts. Novelty detection using extreme value statistics. *IEE Proceedings on Vision, Image & Signal Processing*, 146(3):124–129, 1999.
- [145] S. J. Roberts. Extreme value statistics for novelty detection in biomedical data processing. *Proc IEE: Science, Technology & Measurement*, 147(6):363–367, 2000.
- [146] J. F. Roddick and B. G. Lees. Paradigms for Spatial Data Warehousing for Geographic Knowledge Discovery. In H. J. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery: An Overview*. Taylor and Francis, 2001.
- [147] G. W. Rogers, B. C. Wallet, and E. J. Wegman. A mixed measure formulation of the EM algorithm for huge data set applications. In L. Billard and N. I. Fisher, editors, *Proceedings of the 28th Symposium on the Interface between Computer Science and Statistics*, pages 492–497, Sydney, Australia, July 1997. Interface Foundation of North America.
- [148] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley, New York, 1987.
- [149] J. Sander. *Generalized Density-Based Clustering for Spatial Data Mining*. PhD thesis, Computer Science, University of Munich, München, Germany, 1998.
- [150] R.J. Schalkoff. *Pattern Recognition — Statistical, Structural and Neural Approaches*. John Wiley & Sons, Inc., New York, 1992.
- [151] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1472, 2001.

- [152] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.
- [153] B. Schölkopf, A. J. Smola, and K.-R. Müller. Kernel principal component analysis. *LNCS*, 1327:583–588, 1997.
- [154] B. Schölkopf, R. C. Williamson, A. J. Smola, and J. Shawe-Taylor. SV estimation of a distribution’s support. In T. K. Leen S. A. Solla and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 1999.
- [155] S. Z. Selim and M. A. Ismail. k -means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–86, January 1984.
- [156] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users Web page navigation. In *Proceedings of the IEEE RIDE’97*, 1997.
- [157] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16:30–34, 1973.
- [158] S. S. Skiena. *The Algorithm Design Manual*. Telos/Springer-Verlag, New York, 1998.
- [159] A. F. M. Smith and G. O. Roberts. Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society B*, 55(1):2–23, 1993.
- [160] M. Spiliopoulou. The laborious way from data mining to Web log mining. *International Journal of Computer Systems Science and Engineering*, 14(2):113–125, 1999.
- [161] M. Spiliopoulou. Web usage mining for Web site evaluation. *Communication of the ACM*, 43(8):127–134, 2000.

- [162] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.
- [163] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from Web data. *SIGKDD Explorations*, 1(2):12–23, 2000.
- [164] M.-A. D. Storey and H. A. Müller. Graph layout adjustment strategies. In *Proceedings of Graph Drawing 1995*, (Passau, Germany, September 20 - 22, 1995). Springer Verlag, 1995.
- [165] M. Zakrzewicz T. Morzy, M. Wojciechowski. Scalable hierarchical clustering method for sequences of categorical values. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01)*, pages 282–293, Kowloon, Hong Kong, 2001.
- [166] M. A. Tanner. *Tools for Statistical Inference*. Springer-Verlag, NY, US., 1993.
- [167] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.
- [168] M. B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16:955–961, 1968.
- [169] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, San Diego, 1998.
- [170] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & sons, UK, 1985.
- [171] R. Torres-Velázquez. *Design of Memetic Algorithms for Permutation Problems via Hybridization with Sorting*. PhD thesis, School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan, NSW 2308, Australia, July 2002.

- [172] R. Torres-Velázquez and V. Estivill-Castro. Local search for hamiltonian path with applications to clustering visitation paths. In *Proceedings of Local Search Workshop*, London, UK, April 2002. OR Society (UK): Local Search Study Group, City University, London.
- [173] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.
- [174] C. C. Vogt and G. W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.
- [175] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11:185–195, 1968.
- [176] C. S. Wallace and D. L. Dowe. Minimum message length and Kolmogorov complexity. *Computer Journal*, 42(4):270–283, 1999.
- [177] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B*, 49(3):240–265, 1987.
- [178] W. Wang, J. Yang, and R. R. Muntz. STING: A statistical information grid approach to spatial data mining. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *Twenty-Third International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece, 1997. Morgan Kaufmann Publishers.
- [179] G. Wesolowsky. The Weber problem: history and perspectives. *Location Science*, 1:5–23, 1993.
- [180] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.
- [181] R. Winter. Formal validation of schema clustering for large information systems. In *Proceeding of the First American Conference on Information Systems*, 1995.

- [182] D. Wishart. *Supplement, CLUSTAN user manual*. Prigram Library Unit, Edinburgh University, UK, third edition, 1982.
- [183] J. Xiao, Y. Zhang, X. Jia, and T. Li. Measuring similarity of interests for clustering Web-users. In *Proceedings of the 12th Australian Database Conference ADC 2001*, Gold Coast, Queensland, Australia, 2001.
- [184] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.
- [185] C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.
- [186] A. M. Zarkesh, J. Adibi, C. Shahabi, R. Sadri, and V. Shah. Analysis and design of server informative WWW-sites. In *Proceedings 6th International Conference on Information and Knowledge Management(CIKM)*, pages 254–261, Las Vegas, November 1997. ACM, ACMPress.
- [187] B. Zhang, M. Hsu, and U. Dayal. K -harmonic means — a spatial clustering algorithm with boosting. In J. Roddick and K. Hornsby, editors, *Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining - TSDM2000, in conjunction with the 4th European Conference on Principles and Practices of Knowledge Discovery and Databases*, pages 31–42, Lyon, France, 2000. Springer-Verlag Lecture Notes in Artificial Intelligence 2007.
- [188] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.