

Nonsmooth optimization models and algorithms for data clustering and visualization

Ehsan Mohebi

Principal Supervisor: Assoc. Prof. Adil Bagirov

Associate Supervisor: Dr. Musa Mammadov



Thesis is submitted in total fulfilment of the requirement for the degree of Doctor of Philosophy

School of Applied and Biomedical Sciences
Faculty of Science and Technology
Federation University Australia
PO Box 663
University Drive, Mount Helen
Ballarat, VIC 3353, Australia.

Abstract

Cluster analysis deals with the problem of organization of a collection of patterns into clusters based on a similarity measure. Various distance functions can be used to define this measure. Clustering problems with the similarity measure defined by the squared Euclidean distance have been studied extensively over the last five decades. However, problems with other Minkowski norms have attracted significantly less attention. The use of different similarity measures may help to identify different cluster structures of a data set. This in turn may help to significantly improve the decision making process. High dimensional data visualization is another important task in the field of data mining and pattern recognition. To date, the principal component analysis and the self-organizing maps techniques have been used to solve such problems.

In this thesis we develop algorithms for solving clustering problems in large data sets using various similarity measures. Such similarity measures are based on the squared L_2 as well as L_1 and L_∞ norms. In all cases the clustering problem is a global optimization problem with nonsmooth nonconvex objective functions. In many datasets these problems are large scale and the conventional global optimization algorithms are not efficient for solving such problems. Therefore we propose to apply local search methods for solving clustering problems, however the success of these methods strongly depends on the choice of starting cluster centers. To deal with the nonconvexity of the clustering problems we propose incremental algorithms for their solution which helps us to design a special procedure to generate starting points for cluster centers. Such an approach allows one to find global or near global solutions to the clustering problem. In order to solve nonsmooth clustering problems we apply both efficient nonsmooth optimization algorithms as well as smoothing techniques. To test the proposed algorithms we apply them to solve clustering problems in small, medium size and

large data sets. Furthermore, these algorithms are compared with many other clustering algorithms using results of numerical experiments.

The Self Organizing Maps (SOM) is one of the topology visualizing tool that contains a set of neurons that gradually adapt to input data space by competitive learning and form clusters. The topology preservation of the SOM strongly depends on the learning process. Due to this limitation one cannot guarantee the convergence of the SOM in data sets with clusters of arbitrary shape. Therefore it is important to develop more accurate data visualization and clustering algorithms. In this thesis, Constrained SOM (CSOM) is proposed as the new version of the SOM by modifying the learning algorithm. The idea is to introduce an adaptive constraint parameter to the learning process to improve the topology preservation and mapping quality of the basic SOM. The computational complexity of the CSOM is less than that of the SOM. Mapping quality of the SOM is sensitive to the map topology and initialization of neurons. Thus in this research, a modified version of the SOM (MSOM) is proposed to improve the convergence of the SOM. An initialization algorithm based on split and merge of clusters is introduced to initialize neurons of the SOM. The initialization algorithm speeds up the learning process in large high dimensional data sets. A topology based on this initialization is developed to minimize the vector quantization error and topology preservation of the self organizing maps. The CSOM and MSOM algorithms are tested on small to large size real-world datasets.

Finally, a convolutional structure of the Recursive Modified SOM is proposed to cope with the diversity of styles and shapes in digits recognition. The proposed recursive structure can learn various behaviors of incoming images. The numerical results on the well-known MNIST dataset demonstrate the superiority of the proposed algorithm over existing SOM-based approaches.

Statement of authorship

This thesis contains no work extracted in whole or in part from a thesis, dissertation or research paper previously presented for another degree or diploma except where explicit reference is made. No other persons work has been relied upon or used without due acknowledgment in the main text and bibliography of the thesis.

Ehsan Mohebi
Federation University Australia
March 11, 2015

Acknowledgement

I would like to express my special appreciation and thanks to my principal supervisor, Assoc. Prof. Adil Bagirov, for his strong support, kind consideration, helpful guidance and valuable time throughout my study. His scientific expertise, continuous encouragement and support of my research have been essential for my success. I also thank my associate supervisors, Dr. Musa Mammadov and Dr. Nadezda Sukhorukova for their encouragement and help throughout my PhD study.

I am grateful to the staff at the School of Applied and Biomedical Sciences, Faculty of Science and Technology, Federation University Australia especially, Mrs. Helen Wade, Mrs. Louise Drohan, Mrs. Rebecca Davis and the staff at Research Services, especially Mrs. Lauren Peldys at Federation University Australia for their help and support.

Special thank to my dear Sara for her lovely support and inspiration. I know without her support I could not achieve my ambitions.

Dedication

To my lovely Sara

List of publications

1. Bagirov, A.M. and Mohebi E., **Nonsmooth optimization based algorithms for cluster analysis**, A book chapter in Partitional Clustering Algorithms (Springer, 2014), (To appear).
2. Mohebi E. and Bagirov, A.M., **A convolutional recursive modified self organizing map for handwritten digits recognition**, Neural Networks, 60, pp. 108-114, Elsevier, 2014.
3. Mohebi E. and Bagirov, A.M., **Modified self organizing maps with a new topology and initialization algorithm**, Journal of Experimental & theoretical Artificial Intelligence, Taylor & Francis, June, 2014. (DOI: 10.1080/0952813X.2014.954278).
4. Bagirov, A.M. and Mohebi E., **An algorithm for clustering using L_1 norm based on hyperbolic smoothing Technique**, Computational Intelligence, Wiley, July 2014. (Accepted).
5. Mohebi E. and Marquez, L., **Intelligent packaging in meat industry: an overview of existing solutions**, Journal of Food Science and Technology, Springer, (DOI: 10.1007/s13197-014-1588-z).
6. Bagirov, A.M., Ordin B. and Mohebi E., **An incremental nonsmooth optimization algorithm for clustering using L_1 and L_∞ norms**, Computers & Operations Research, (Submitted).
7. Mohebi E. and Bagirov, A.M., **Constrained self organizing maps for data clusters visualization**, Neural Processing Letters, Springer, (Submitted).

8. Mohebi E. and Marquez, L., **A review: sensors in smart packaging of meat products**, Journal of Food Engineering, Elsevier, (Submitted).
9. Mohebi E. and Marquez, L., **Application of machine learning techniques and RFID in stability optimization of perishable foods in supply chain**, Expert Systems with Applications, Elsevier, (Submitted).
10. Mohebi E. and Bagirov, A.M., **CR-Modified SOM to the problem of handwritten digits recognition**, AI-2014 Thirty-fourth SGAI International Conference on Artificial Intelligence. Cambridge, England, 9-11 Dec. 2014, (Accepted).
11. Mohebi E. and Bagirov, A.M., **A new modification of Kohonen neural network for VQ and clustering problems**, Proceedings of AusDM2013, Canberra, 2013.
12. Mohebi E. and Marquez, L., **Simulation of RFID-induced efficiencies in meat packaging**, The Third International Workshop on Food Supply Chain, the San Francisco State University College of Business, November 4th -7th, 2014. (Accepted for presentation).

Contents

Abstract	i
Statement of authorship	iii
Acknowledgement	iv
Dedication	v
Introduction	1
1 Literature review	6
1.1 Introduction	6
1.2 Hierarchical clustering	7
1.3 Partitional clustering	12
1.4 Clustering using similarity measure based on the L_1 and L_∞ norms	17
1.5 High dimensional data visualization and clustering	19
1.5.1 Self organizing map	21
1.5.2 Modifications of the SOM	23
1.6 Handwritten digits recognition using SOM	26
1.7 Summary	27
2 Optimization formulation of clustering problem	28
2.1 Introduction	28
2.2 Combinatorial formulation of the clustering problem	29
2.3 Mixed integer nonlinear programming formulation of the clustering problem .	30

2.4	Nonsmooth nonconvex optimization formulation of the clustering problem . .	31
2.5	Comparison of different formulations	31
2.6	Summary	32
3	Incremental clustering algorithm	33
3.1	Introduction	33
3.2	The auxiliary cluster problem	34
3.3	An incremental clustering algorithm	34
3.4	Computation of starting points for cluster centers	36
3.5	The modified incremental clustering algorithm	40
3.6	Summary	41
4	Solving optimization clustering problems	42
4.1	Introduction	42
4.2	k -means type heuristic algorithm	42
4.3	Nonsmooth optimization methods	43
4.3.1	An algorithm for solving Problem (2.9)	43
4.3.2	An algorithm for solving Problem (3.2)	47
4.3.3	Solving optimization problems to find center of one cluster	48
4.4	An algorithm based on smoothing techniques	51
4.4.1	Hyperbolic smoothing of the cluster function	53
4.4.2	Hyperbolic smoothing of the auxiliary cluster function	53
4.4.3	Hyperbolic smoothing of L_1 -norm	54
4.4.4	Hyperbolic smoothing of L_∞ -norm	55
4.4.5	Smooth clustering problems	56
4.5	Summary	57
5	Implementation and numerical results	58
5.1	Introduction	58
5.2	Computational results: evaluation of the incremental algorithm	59
5.2.1	Results for the similarity measure based on the squared L_2 -norm . . .	61
5.2.2	Results for the similarity measure based on the L_1 -norm	61

5.2.3	Results for the similarity measure based on the L_∞ -norm	68
5.2.4	Results for purity in data sets with class labels	71
5.2.5	Visualization of results	71
5.3	Computational results: comparison with other clustering algorithms	73
5.3.1	Comparison of algorithms using the similarity measure based on the squared L_2 -norm	73
5.3.2	Comparison of algorithms using the similarity measure based on the L_1 -norm	74
5.3.3	Comparison of algorithms using the similarity measure based on the L_∞ -norm	80
5.4	Summary	81
6	Constrained self organizing maps for high dimensional data visualization	83
6.1	Introduction	83
6.2	CSOM learning algorithm	84
6.2.1	Modified learning algorithm	85
6.2.2	Adaptive selection of parameter γ	87
6.3	The CSOM algorithm and its implementation	89
6.3.1	Implementation of CSOM algorithm	90
6.4	Numerical Results	91
6.4.1	Validation of parameter ρ in adaptive constraints	91
6.4.2	Comparison with the SOM	92
6.4.3	Complexity comparison with SOM	95
6.4.4	Distortion error	96
6.4.5	Topology preservation	98
6.4.6	Comparison with other algorithms	99
6.5	Summary	101
7	Modified self organizing maps for vector quantization and clustering problems	102
7.1	Introduction	102

7.2	Splitting and merging algorithms	102
7.2.1	Splitting	103
7.2.2	Merging	106
7.3	SOM initialization algorithm	108
7.4	SOM with modified topology	109
7.5	The modified SOM algorithm and its implementation	111
7.5.1	Implementation of algorithm 13	114
7.6	Results of the evaluation and discussion	114
7.6.1	Comparison with other algorithms	118
7.6.2	Topology preservation	121
7.7	Summary	121
8	Convolutional recursive modified SOM for handwritten digits recognition	124
8.1	Introduction	124
8.2	Selection of parameters in modified SOM algorithm	124
8.2.1	Comparison with SOM using numerical results	125
8.2.2	Complexity comparison with SOM	126
8.3	Recursive MSOM	127
8.3.1	Training	128
8.3.2	Testing	129
8.4	The convolutional structure	131
8.4.1	Implementation of Algorithms 15 and 16	134
8.4.2	Application to handwritten digit data set	135
8.4.3	Reliability of the proposed method	140
8.5	Summary	142
	Bibliography	146
	Appendix	162

List of Tables

5.1	Results with the similarity measure based on L_2 -norm	62
5.2	Results with similarity measure based on L_2 -norm (cont.)	63
5.3	Results with similarity measure based on L_2 -norm (cont.)	64
5.4	Results with the similarity measure based on the L_1 -norm	65
5.5	Results with the similarity measure based on the L_1 -norm (cont.)	66
5.6	Results with the similarity measure based on the L_1 -norm (cont.)	67
5.7	Results with the similarity measure based on the L_∞ -norm	68
5.8	Results with the similarity measure based on the L_∞ -norm (cont.)	69
5.9	Results with the similarity measure based on the L_∞ -norm (cont.)	69
5.10	Cluster purity for different similarity measures	72
5.11	Comparison of algorithms using the similarity measure based on the L_2 -norm	75
5.12	Comparison of algorithms using the similarity measure based on the L_2 -norm (cont.)	76
5.13	Comparison of algorithms using the similarity measure based on the L_2 -norm (cont.)	77
5.14	Comparison of algorithms using the similarity measure based on the L_1 -norm	78
5.15	Comparison of algorithms using the similarity measure based on the L_1 -norm (cont.)	79
5.16	Comparison of algorithms using the similarity measure based on the L_1 -norm (cont.)	79
5.17	Comparison of algorithms using the similarity measure based on the L_∞ -norm	80
5.18	Comparison of algorithms using the similarity measure based on the L_∞ -norm (cont.)	81

5.19	Comparison of algorithms using the similarity measure based on the L_∞ -norm (cont.)	81
6.1	Initialization of SOM parameters in Algorithm 13.	90
6.2	The quantization error, E , using different settings of parameter ρ in CSOM (Linear, Hyperbolic and Sigmoid functions).	92
6.3	Results for small and medium size data sets.	93
6.4	Results for small and medium size data sets.	94
6.5	Total number of calculations of (1.5), N , for all data sets.	97
6.6	Results of distortion measure on all data sets	98
6.7	Results of quantization error on all data sets	100
6.8	The CPU time required by all algorithms	100
7.1	Initialization of SOM parameters in Algorithm 1.	114
7.2	The results of the Split and Merge algorithm	115
7.3	Results for small and medium size data sets	116
7.4	Results for large and very large data sets	117
7.5	Results of distortion measure on all data sets	118
7.6	Comparison of different algorithms.	119
7.7	Results for the distinctness error	120
7.8	CPU time required by algorithms	121
8.1	The result of E values in A1 dataset using different configurations of parameters ε and γ in Algorithm 13 (s_ε is step length).	125
8.2	Initialization of \bar{n}^y in Algorithm 15.	134
8.3	Initialization of parameters in Algorithms 8 to 13.	134
8.4	Initialization of parameters in Algorithms 16.	134
8.5	Comparison of accuracy obtained by SOM-based handwritten digits recognition methods on MNIST dataset.	137
8.6	The most challenging samples that reported in [108].	138
8.7	The most challenging samples that reported in [108].	139
8.8	The most challenging samples that reported in [108].	140

8.9	Confusion matrix.	142
10	The brief description of data sets	163

List of Figures

1.1	BIRCH concept, a leaf node is not a single data point but, a sub-cluster (which absorbs many data points with diameter (or radius) under a specific threshold T	7
1.2	The images of the points on the one dimensional space are shown as the projections onto the line E. The clusters generated in the reduced dimensional space are those enclosed in the diagonal rectangular boxes A, B and C. Observe that when these clusters are mapped back to the original 2-dimensional feature space, the points a and b are misclassified. The refinement process using the hierarchical algorithm for each cluster independently identifies the true clusters encircled as U, V and W. Using the clusters U, V, W, the significant attribute chosen for clustering is Y since it has a higher discriminant value than X.	10
1.3	Progress of the decomposition process using GDA in 2-D data space. (a) Data space before start of the partitioning process. (b) Initial partitioning of the data space. (c) Repartition of the non-empty blocks in the intermediate stage of GDA. (d) Complete partitioning after checking all non-empty blocks. . . .	11
1.4	Partial/merge k -means, partitioning scheme.	15
1.5	Sketch tables of hash functions.	15
1.6	Patch clustering algorithm.	17
4.1	Smooth approximation of L_1 -norm	55
4.2	Smooth approximation of L_∞ -norm	56
5.1	Results for the similarity measure based on the L_1 -norm	70
5.2	Results for the similarity measure based on the squared L_2 -norm	70

5.3	Results for the similarity measure based on the L_∞ -norm	70
5.4	Visualization of clusters for German towns data set	72
5.5	Visualization of clusters for TSPLIB1060 data set	73
5.6	Visualization of clusters for TSPLIB3038 data set	73
6.1	Contraction of neighbor neurons of the BMU.	84
6.2	The updating procedures in the SOM algorithm.	86
6.3	Performance of constraint γ using equation (6.10) on 20×20 SOM where $r = 3$, $\rho = 5$, $T = 5$ and $ A = 20$	88
6.4	The sensitivity of choosing different values of parameter ρ on Iris and Wine data sets. E is the value of quantization error.	92
6.5	Convergence of CSOM vs SOM.	95
6.6	Topology preservation of CSOM vs SOM.	98
6.7	Topology preservation of CSOM vs SOM.	99
7.1	The splitting procedure.	106
7.2	Initial neurons after split and merge	108
7.3	Topology of the modified SOM.	110
7.4	Initial neurons generated by Algorithms 11 and 12 with $\lambda = 1$	111
7.5	Initial neurons generated by Algorithms 11 and 12 with $\lambda = 1.5$	111
7.6	The summary of steps in Algorithm 13.	112
7.7	The Modified SOM on Chain Link dataset.	113
7.8	SOM vs Modified SOM using CPU time.	117
7.9	Comparison of algorithms using E values.	119
7.10	Visualization of the data set D15112 and clusters.	122
7.11	Topology preservation of algorithms in TSPLIB1060 data set (data points are in blue and neurons are in red color).	123
8.1	The comparison of the SOM and the MSOM.	126
8.2	Comparison of E values obtained by the SOM and the MSOM.	127
8.3	Topology of modified SOM.	127

8.4	A set of images, \aleph^0 , with label 0, which are selected randomly from the training set.	129
8.5	The set of modified SOM networks, $\Psi_h \in \mathcal{M}_0$, after training with samples in Figure 8.4. This set is using for recognition of images with 0 label.	130
8.6	The Modified SOM network, Ψ , in the left and the network $\bar{\Psi}$ in the right after applying filtering on Ψ	131
8.7	The proposed convolutional structure.	131
8.8	Different settings of parameters ε and γ for $\Upsilon_\varepsilon^\gamma$ while learning an input image with label 0.	132
8.9	The networks, $\tilde{\Upsilon}_\varepsilon^\gamma$, with high resolution neurons only. These networks are the results of equation (8.2) on the networks presented in Figure 8.8.	133
8.10	The first 45 digits of the test samples.	135
8.11	100 training samples from the 400 training samples, which have been used to train the sets \mathcal{M}_y , $y = 0, \dots, 9$	136
8.12	The networks $\Psi_1^y \in \mathcal{M}_y$, $y = 0, \dots, 9$ after running Algorithm 15 on training samples.	136
8.13	The 97 misclassified samples from the 10000 test samples.	141
8.14	The set \mathcal{M}_7 used to classify images with label 7.	142
8.15	The reliability of CR-MSOM vs CNN-SOM [33] on 10000 test samples.	142

Introduction

In recent years, the problem of mining very large datasets has become more and more pronounced in diverse areas such as document and web mining, geo information, remote sensing, bioinformatics, and medicine, to name just a few. In these areas, massive volumes of data arise on a daily base which have to be preprocessed and mined to allow further processing and human inspection. *Clustering* and *data visualization* are two most useful approaches for pattern recognition, image processing, decision making, data mining or knowledge discovery in databases. Clustering is among most important tasks in data mining and it is the process of learning concept of raw data by dividing the data into groups of similar objects [8, 26]. As a tool, clustering has a wide range of applications in many applied fields like biomedical, signal analysis, life science taxonomy, remote sensing, demography and social sciences, geology and anthropology, economics and planning [40, 48, 50, 146].

Clustering algorithms can be broadly divided into two groups: *hierarchical* and *partitional*. Hierarchical clustering algorithms recursively find nested clusters either in agglomerative mode that is starting with each data point in its own cluster and merging the most similar pair of clusters successively to form a cluster hierarchy or in divisive (top-down) mode that is starting with all the data points in one cluster and recursively dividing each cluster into smaller clusters. Partitional clustering algorithms decompose a dataset into a set of disjoint clusters. Given a data set of m points, a partitioning method constructs k ($k \leq m$) partitions of the data, with each partition representing a cluster. That is, it classifies the data into k groups by satisfying the following requirements: (1) each group contains at least one data point, and (2) each data point belongs to exactly one group. Usually, the number of clusters should be specified in advance. Many clustering algorithms have been proposed based on statistical, machine learning, neural networks and optimization techniques [21, 19, 26, 84].

The clustering problem can be formulated as an optimization problem and its optimization models include combinatorial, mixed integer nonlinear programming and nonconvex non-smooth optimization formulations [19, 27, 15]. In order to define similarity between points one needs to introduce the so-called *similarity measure*. This measure can be defined by a distance function. Clustering problems with the similarity measure defined by the squared Euclidean distance have been studied extensively over the last five decades. However, problems with other Minkowski norms have attracted significantly less attention.

One should notice that most of clustering algorithms in the literature are able to find only local solutions to clustering problems and such solutions might be significantly different from global solutions. Moreover, these algorithms are sensitive to the choice of starting points. The clustering is a global optimization problem, it has many solutions and only global solutions provide the best cluster structure of a data set. General purpose global optimization algorithms are not efficient for solving such problems in even relatively large data sets.

In this research, we develop three different algorithms for solving clustering problems, where the similarity measure is defined using the L_1 and L_∞ norms. The first algorithm is based on an incremental approach and applies heuristics like the k -means algorithm for finding cluster centers. This algorithm computes clusters gradually starting from one cluster which is the whole data set. Using the incremental approach we introduce an auxiliary clustering problem to find starting points for cluster centers. Such an approach allows one to find either global or near global solutions to clustering problems.

In the second algorithm, we apply the discrete gradient method as a nonsmooth optimization algorithm to solve the clustering problem. This method is a derivative-free method and uses discrete gradients which are approximations to subgradients. In the third algorithm, to deal with nonsmoothness, smoothing techniques are applied to approximate the clustering functions based on the L_1 and L_∞ norms by smooth functions. This allows us to apply powerful smooth optimization algorithms to solve cluster analysis problems. To test the proposed algorithms we apply them to solve clustering problems in small, medium size and large data sets. Furthermore, these algorithm are compared with many other clustering algorithms using results of numerical experiments.

We introduce two approaches to the problems in high dimensional data visualization. High dimensional data visualization is used to analyze the hidden patterns and data relationships, which are hard to illustrate. The self organizing map is one of the well known data mining tools where the aim is to visualize a high dimensional data space into usually a 2-Dim grid [37, 73, 81, 106, 129], which provides a better insight to the structure of the input data set. The SOM contains a set of neurons that gradually adapts to input data space by competitive learning and creates ordered prototypes. The ordered prototypes preserve the topology of the mapped data and make the SOM to be very suitable for cluster analysis [151]. This adaption is based on a similarity measure, which is usually Euclidean distance, and repositioning of neurons in a 2-Dim space using a learning algorithm. The performance of the SOM strongly depends on the learning algorithm and the quality of the self organizing map is measured based on the quantization error and topology preservation of the map [61, 62, 71, 75].

We develop a modified learning algorithm for the Self Organizing Maps. The aim is to propose a learning algorithm which restricts the neighborhood adaptations to only those neurons that are not far from the best matching unit in the n -dimensional space. We introduce an adaptive constraint parameter that is a decreasing function with respect to iterations to be applicable to the SOM learning process (CSOM). The adaptive constraint parameter selected as linear, hyperbolic and sigmoid functions. The results show that the CSOM converges much faster than SOM, requires less computational effort, improves the topology preservation and presents promising clustering results. The proposed algorithm outperforms similar topology preservation algorithms especially in very large data sets in the sense of accuracy and computational time.

The SOM and many of its modifications are sensitive to the initialization of neurons and the topology of the map. All existing modifications of the SOM do not include any specific procedure to initialize neurons of the map. Furthermore, the most of these algorithms, including the SOM, are not efficient in large data sets. We develop a modified version of the SOM (MSOM) to address these drawbacks. The proposed version includes an algorithm for initialization of neurons based on the split and merge procedure. The high dense areas in input data space are detected by this procedure. Then neurons are generated in those detected areas, therefore, the number of neurons in the map is not predefined in advance.

Initialization of neurons in such areas accelerates the convergence of the algorithm and makes it applicable to large data sets. Based on the initial neurons, a new topology is presented to restrict the adaptation of the neurons to those neurons, which are located in the same high density area. Such a topology reduces the contraction of neurons which are far from each other in n -dimensional space, consequently, leads to a better quantization error and topology preservation than that of by the SOM. The CSOM and MSOM algorithms are tested on small to large size real-world datasets.

Finally, we developed a Convolutional Recursive Modified SOM to solve the handwritten digits recognition problem. A recursive form of the Modified SOM is proposed for training process, in order to learn the diverse shapes and styles of incoming images. Then, a convolutional structure is introduced to label the unknown handwritten digits images of the MNIST dataset. The results using this dataset demonstrate the superiority of the proposed algorithm over the existing SOM-based methods in the sense of accuracy.

Structure of the thesis

The thesis is structured as follows.

- Chapter 1 presents a literature review on hierarchical and partitional clustering algorithms, clustering algorithms, which are defined using different distance functions and high dimensional data visualization algorithms including modifications of the SOM.
- Chapter 2 discusses different optimization formulation of the clustering problems and their comparison in the sense of computational complexity.
- In Chapter 3, we present a modified incremental algorithm to the clustering problem.
- Chapter 4 is devoted to solving the optimization problems from clustering problems. The nonsmooth nonconvex formulation of the clustering problem is presented, where the similarity measures is defined using the L_1 and L_∞ norms. Furthermore, hyperbolic smoothing of cluster functions are presented.
- The implementation of algorithms and numerical results are presented in Chapter 5.

- In Chapter 6, we present a new learning algorithm based on a constraint parameter to improve the quantization error of the SOM for data visualization.
- The initialization algorithm of modified SOM and its modified topology are introduced in Chapter 7.
- In Chapter 8, the convolutional structure of the recursive modified SOM is introduced to the problem of handwritten digits recognition.
- We conclude the thesis by giving a short overview of the results obtained in this thesis and discuss the directions for future work.

Chapter 1

Literature review

1.1 Introduction

Advances in sensing and storage technology and dramatic growth in applications such as internet search, digital imaging, and video surveillance have created many high-volume, high dimensional data sets. Most of the data are stored digitally in electronic media, thus providing huge potential for the development of automatic data analysis, classification, and retrieval techniques. In addition to the growth in the amount of data, the variety of available data (text, image, and video) has also increased. Inexpensive digital and video cameras have made available huge archives of images and videos. The prevalence of RFID tags or transponders due to their low cost and small size has resulted in the deployment of millions of sensors that transmit data regularly. E-mails, blogs, transaction data, and billions of Web pages create terabytes of new data every day. Many of these datasets are unstructured, thus difficult to be analyzed.

Clustering and data visualization are two most useful approaches for pattern recognition, image processing, decision making, data mining or knowledge discovery in databases. Clustering is an unsupervised learning problem, which deals with finding structure in a collection of unlabeled data. Clustering algorithms mainly fall into two groups: *hierarchical* and *partitional*.

In this chapter we present a brief overview of hierarchical and partitional clustering algorithms as well as algorithms for data visualization. A comprehensive survey on clustering

algorithms can be found in [26, 84]. We consider only hard clustering problems. Algorithms for the fuzzy clustering problems can be found, for example, in [45, 114].

1.2 Hierarchical clustering

Hierarchical clustering is a widely used data analysis tool. The idea is to build a binary tree of the data that successively merges similar groups of points and visualizing this tree provides a useful information about the data.

One of the first methods is BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [157] that adopts the notion of *clustering feature* to summary description of clustering properties, which is demonstrated that it is especially suitable for very large databases. BIRCH incrementally and dynamically clusters incoming multi dimensional metric data points to try to produce the best quality clustering with the available resources (i.e., available memory and I/O time constraints). It utilizes measurements that capture the natural closeness of data. These measurements can be stored and updated incrementally in a height balanced tree (see Figure 1.1).

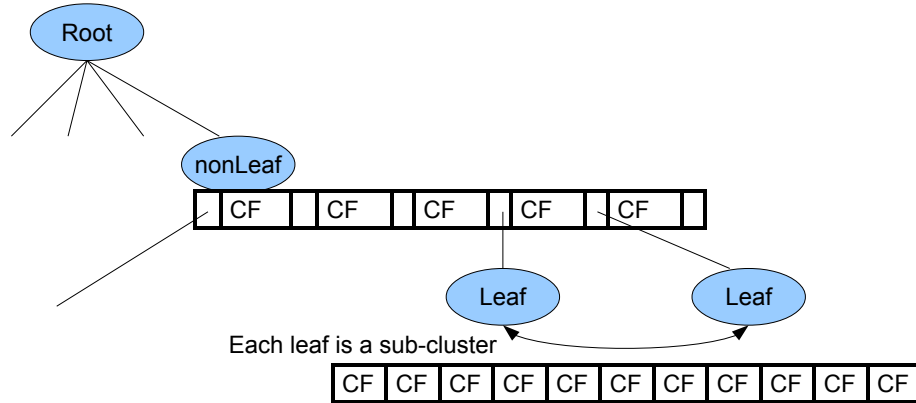


Figure 1.1: BIRCH concept, a leaf node is not a single data point but, a sub-cluster (which absorbs many data points with diameter (or radius) under a specific threshold T)

A Clustering Feature is a triple $CF = (N, \vec{LS}, SS)$ summarising the information that BIRCH maintains about a cluster. Here N is number of data points in the subcluster, $\vec{LS} = \sum_{i=1}^n \vec{X}_i$ and $SS = \sum_{i=1}^n \vec{X}_i^2$. Some shortcomings involved in BIRCH are to have efficient result the initial parameters should be properly assigned, like threshold. Furthermore

it tries to cluster the whole dataset at once, which is not affordable for massive dataset due to memory restriction.

A clustering algorithm based on density of data points, the so called adaptive density-reachable (CADD) [103], is introduced according to the notion and enlightenment of BIRCH. The authors proposed an incremental clustering algorithm based on definitions of subcluster similarity for very large spatial databases. The incremental clustering algorithm is simple and efficient, and has good performance especially for very large spatial databases. Similar to the Clustering Feature (CF), a SubCluster Feature (SCF) is also presented by a triple of numbers, which gives the statistic description of a subcluster. Given N d -dimensional data points \vec{X}_i , ($i = 1, \dots, N$) in a subcluster, a SubCluster Feature $SCF = (N, \vec{X}_0, R)$ is defined, where $\vec{X}_0 = \frac{1}{N} \sum_{i=1}^N \vec{X}_i$ and $R = \sqrt{\frac{1}{N} \sum_{i=1}^N (\vec{X}_i - \vec{X}_0)^2}$.

The algorithm compares the distribution of initial data points D with the distribution of data points in the incremental data space D' based on three similarity measures.

1. Similarity of spatial position:

$$S = e^{-\frac{\|\vec{C} - \vec{C}'\|^2}{2\sigma^2}} \quad (1.1)$$

where \vec{C} , \vec{C}' are subcluster centroids in initial and incremental data space respectively and $\sigma^2 = R \times R'$. R is coherence of a subcluster.

2. Similarity of coherence:

$$\begin{aligned} D &= \frac{R}{R'} && \text{if } R' \geq R \\ D &= \frac{R'}{R} && \text{if } R \geq R' \end{aligned} \quad (1.2)$$

3. Similarity norm: If $S \geq 0.7$ and $D \geq 0.7$ simultaneously, the two subclusters are similar and merged, otherwise, they are not similar and maintained independently.

Although the CADD is an extension of BIRCH it is argued that unlike BIRCH it can detect clusters with arbitrary shape and size, however the same shortcomings of BIRCH still remain unsolved.

A hierarchal clustering algorithm, called CURE, explores sophisticated cluster shapes [74]. The crucial feature of CURE lies in the usage of a set of well-scattered points to represent

each cluster, which makes it possible to find cluster shapes other than hyperspheres and avoids the tendency to find clusters with similar sizes. CURE utilizes random sample (and partition) strategy to reduce computational complexity.

One of the clustering algorithms for massive data sets is Hybrid Cell Density Clustering method (HyCeltyc) [111], which combines a *cell-density based* algorithm with a hierarchical agglomerative method to identify clusters in linear time. The main steps of the algorithm involve sampling, dimensionality reduction, selection of significant features on which to cluster the data and a grid-based clustering algorithm that is linear in the data size. The author of [111] has mentioned some drawbacks of most grid-based algorithms:

1. For large M (bin sizes), the image points in S^M (feature space) become very sparse.
2. The algorithm is sensitive to the choice of the bin sizes, which not only determines the quality of the clusters but also determines the space and time complexity of the algorithm.
3. The quality of algorithm closely depends on the choice of a data structure for managing the non-empty cells since the number of grid cells could be very much larger than even the number of points.

To address the first problem, i.e. the effect of large dimensionality, the author in [59] applied a linear dimensionality reduction method called FastMap, on a sample of the data to reduce the dimensionality from M to K , where K is a user selected dimensionality for clustering. In addition to reducing the dimensionality, a method is developed to order the dimensions according to their ability to discriminate the clusters. To address the second problem of selecting the bin sizes, a threshold τ_1 is specified. The value of τ_1 defines the minimum number of points in a cell that allows it to be considered as dense. This value of τ_1 , expressed either as an absolute value or as a percentage of the number of points, is also utilized to determine the bin sizes of each attribute. The third problem is resolved by hashing the addresses of non-empty cells to a linear address space [59].

A cluster in HyCeltyc is defined as a union of neighbouring dense cells. The determination of clusters in the rectilinearly partitioned cells is done by invoking a simple Cell-Density Clustering algorithm which is abbreviated simply as Celtyc. It takes as input, the set of

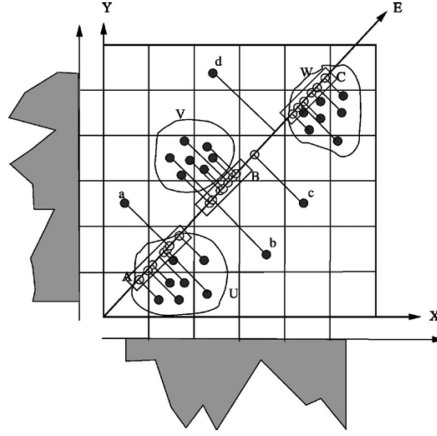


Figure 1.2: The images of the points on the one dimensional space are shown as the projections onto the line E. The clusters generated in the reduced dimensional space are those enclosed in the diagonal rectangular boxes A, B and C. Observe that when these clusters are mapped back to the original 2-dimensional feature space, the points a and b are misclassified. The refinement process using the hierarchical algorithm for each cluster independently identifies the true clusters encircled as U, V and W. Using the clusters U, V, W, the significant attribute chosen for clustering is Y since it has a higher discriminant value than X.

objects R , the number of objects N , the attributes $A = A_0, A_1, \dots, A_{K-1}$, the dimensionality K and an array $M = m_0, m_1, \dots, m_{K-1}$, that specifies the number of bins that each attribute has to be split (see Figure 1.2). However there are many input parameters that should be properly defined prior to clustering as well as using six phases with multiple algorithm execution which may not always guarantee the best performance on massive data sets.

There are various principles and techniques that are not sufficient for clustering data of diverse shape, density and size [36]. Therefore in [36] the author proposed a hybrid approach with the aim to identify clusters of irregular shape and size (which contain concavity and nested shapes), cluster data with non-uniform density, be independent of data order and handle high dimensional data and be insensitive to noise or outlier data. The proposed algorithm is basically a *split-and-merge* based grid-clustering approach and has been named as Genetically Guided Grid Clustering (GGGC) [36].

The hybridization was attained by a combination of *genetic algorithm* and *Tabu Search* (TS) method. Both algorithms involve a population based search technique where the population is represented by a set of individuals. Each individual is a string of binary values chosen from $\{0, 1\}$. The search process advances to a solution depending on the individuals selected using the fitness score. The fitness score of an individual determines its survival

strength in a population.

The grid-clustering approach quantizes the space into a finite number of cells that form a grid structure on which all operations for clustering are performed. The GGC algorithm proceeds as follows:

1. The entire feature space is initially partitioned hierarchically by the multi-dimensional grid structure into a number of cells by Grid based Decomposition Algorithm (GDA). The cells containing data points are finally considered as sub-clusters (see Figure 1.3).

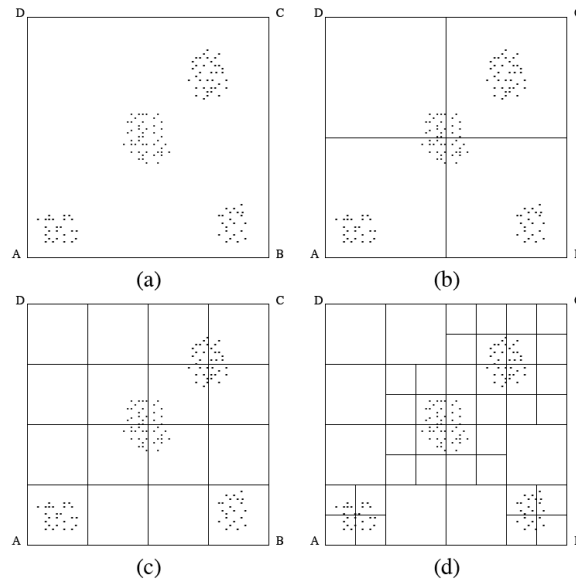


Figure 1.3: Progress of the decomposition process using GDA in 2-D data space. (a) Data space before start of the partitioning process. (b) Initial partitioning of the data space. (c) Repartitioning of the non-empty blocks in the intermediate stage of GDA. (d) Complete partitioning after checking all non-empty blocks.

2. In the second stage, the sub-clusters are merged hierarchically using the hybrid method namely, Cluster Merging with Hybrid Algorithm (CMHA). CMHA is based on the combination of a Modified GA (MGA) and a tabu search method. Both the MGA and TS process are invoked sequentially to run iteratively and in each run, some sub-clusters are merged.
3. The MGA is started first. When the MGA is terminated with an optimal solution after G_{max} iterations or the solution is not changed for a long time, TS is actually invoked to run iteratively.

4. With the completion of the TS procedure another round of MGA followed by TS is executed. The completion of one MGA and TS cycle is referred to as a single epoch.
5. This sequence is continued several times until the expected number of k clusters are found.

The use of TS is effective at this juncture since it can facilitate the genetic method to emerge out from the local optima due to its inherent local search capability. Thus, the TS enhances the performance of the genetic method by reducing the number of overall iterations.

1.3 Partitional clustering

Partitional clustering algorithms divide data into several subsets. Since checking all possible subsets of a given dataset is computationally infeasible, certain greedy heuristics are used in the form of iterative optimization. Specifically, this means different relocation schemes that iteratively reassign points between the k clusters. Unlike traditional hierarchical methods, in which clusters are not revisited after being constructed, relocation algorithms gradually improve clusters.

A k-means clustering algorithm introduced in [97] proceeds as follows: Let have a set of k centers C , for each center c in C , let have a set, $N(c)$, which is the set of data points where c is the nearest neighbor. In each stage of Lloyd’s algorithm every center point c moves to the centroid of the set $N(c)$ and then the algorithm updates $N(c)$ by recomputing the distance from each data point to its nearest center. These steps are repeated until no data point changes its cluster. Note that Lloyd’s algorithm can get stuck in locally minimal solutions that are far from the global ones.

Forgy’s algorithm [63] is a simple alternative of least-squares algorithm. The algorithm, first initialize the center of clusters randomly. Then read the data points and assign each data to the nearest center point using Euclidean distance. In the next stage the algorithm updates each center point with the mean of the set of data points that were assigned to it. The last two stages are repeated until no data point changes its cluster.

The clustering algorithm introduced by MacQueen [99] is similar to the Forgy’s algorithm. The difference is in the last stage where the proposed algorithm by MacQueen moves the

centers points to the mean of their Voronoi set.

The authors in [79] introduced a clustering algorithm that requires a matrix of M points in N dimensions and a matrix of K initial cluster centers in N dimensions as an input. The general procedure is to search for a K -partition with locally optimal within-cluster sum of squares by moving points from one cluster to another. The authors proposed a novel initial seeding of cluster centers based on the distance of each data point to the overall mean of data set. First, the data points are sorted based on their distance to the mean of the data set. Then the initial center for cluster $i, i = 1, \dots, k$ is the data point with index j , where $j = 1 + (i - 1)(M/K)$, in the sorted list.

The k -means++ algorithm is an alternative algorithm to the k -means, which is introduced in [9]. The authors proposed a variant that chooses centers at random from the data points, but weighs the data points according to their squared distance from the closest center already chosen. Then next center is chosen from the weighted data points with the maximum weight. These procedure is repeated until k centers are initialized. The rest of the algorithms is the same as the simple k -means algorithm.

The authors in [113] proposed a clustering algorithm, the so called X -means, based on k -means algorithm. This algorithm is claimed to be an improvement to some shortcomings of the original k -means. The number of clusters, k , is estimated by this algorithm using a scoring system which runs k -means several times. Then the algorithm decides whether to split the current clusters into two new clusters by computing the Bayesian Information Criterion.

Some attempts have been made to parallelize the k -means algorithm to make it applicable to very large datasets. In [88] the authors use a network of homogeneous workstations with Ethernet network and use message-passing for communication between processors. In an Ethernet network, all communications consist of packets transmitted on a shared serial bus available to all processors. The following is a description of the parallel k -means clustering algorithm. A master-slave single program multiple data approach (SPMD) is used. The parallel k -means algorithm can be summarized as follows:

Master Process:

1. Randomly form K equal subsets of the set S

2. Send each subset to each of the K slaves
3. Receive K resulting subsets from K slaves

Slave Process:

1. Receive a vector subset P from master process
2. While Error E is not stable:
3. Compute a mean \bar{X}_{myrank} of the subset P
4. Broadcast the mean \bar{X}_{myrank} to every other slaves
5. Compute distance $d(i, j)$, $1 \leq i \leq K$, $1 \leq j \leq |P|$ of each vector in P such that $d(i, j) = \|\bar{X}_i - v_j\|$
6. Choose vector members of the new K subsets according to their closest distance to \bar{X}_i , $1 \leq i \leq K$
7. Broadcast K subsets computed in Step 6 to every other slaves
8. Form the new subset P by collecting vectors that belong to \bar{X}_{myrank} that were sent from other slaves in Step 7
9. End While
10. Send the subset P to master process

The concept is to distribute processing of k -means on k machines which result in a satisfactory time complexity. On the other hand for k clusters we have to configure exactly k machines and every time rerun the k -means from the start point. Due to the memory limitation this version of the k -means may not be efficient for massive dataset.

The authors in [107] introduce the *partial/merge k-means* algorithm which processes the overall set of points in cells, and merges the results of the partial k -means steps into an overall cluster representation. The partial k -means and the merge k -means are implemented as data stream operators that are adaptable to available computing resources such as volatile memory and processors by parallelizing and cloning operators, and by computing k -means on partitions of data that can be fit into memory.

Instead of storing all data points v_1, \dots, v_n of a grid cell C_s in memory, the data is divided of C_s (see Figure 1.4) into p partitions P_1, \dots, P_p with the condition that all data points v_1, \dots, v_m of partition P_j can be stored into available volatile memory. The stream operator partial k -means selects a set of random k seeds for a partition P_j , and performs a k -means on the subset of data points of the overall grid cell until the convergence criteria is met. This

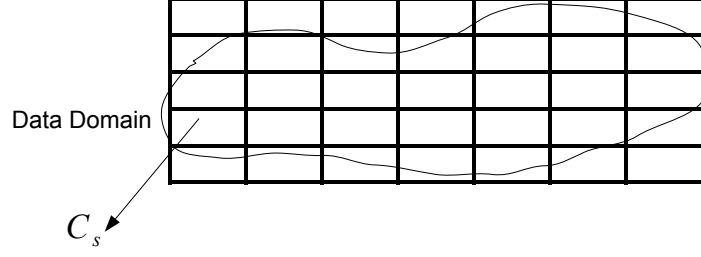


Figure 1.4: Partial/merge k -means, partitioning scheme.

step is repeated for several sets of random k seeds, and the representation with the minimal mean square error is selected to represent the clustering of partition P_j . The partial k -means operator produces a set of weighted centroids $c_{ij} \in P_j\{(c_{1j}, w_{1j}), (c_{2j}, w_{2j}), \dots, (c_{kj}, w_{kj})\}$. The weight w_{ij} is defined as the number of points that are assigned to the centroid c_{ij} . The sum $\sum_{i=1}^k w_{ij}$ is the number of points N_j in the partition P_j . Each partition is clustered independently. The last step is merging process where the set of weighted centroids are clustered and merged according to their minimum distances and minimization of an error function.

Partial/merge k -means re-runs the k -means several times to get better result in each partition. This algorithm is sensitive to the size of partitioning in order to get the best performance in massive datasets.

Sketch based techniques [49] are a natural method for compressing the counting information in the underlying data so that the broad characteristics of the dominant counts can be maintained in a space-efficient way. In [1] the authors applied the count-min sketch [49] to the problem of clustering massive-domain data streams.

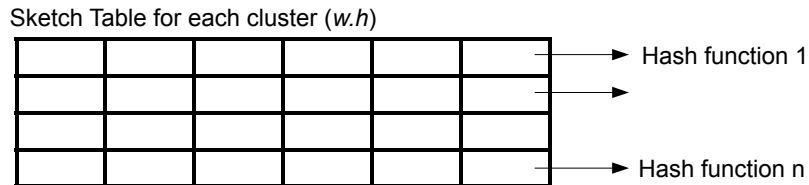


Figure 1.5: Sketch tables of hash functions.

The proposed data structure itself consists of a two dimensional array with $w \times h$ cells with a length of h and width of w (see Figure 1.5). Each hash function corresponds to

one of w 1-dimensional arrays with h cells each. In standard applications of the count-min sketch, the hash functions are used in order to update the counts of the different cells in this 2-dimensional data structure. For example, consider a 1-dimensional data stream with elements drawn from a massive set of domain values. When a new element of the data stream is received, each of the w hash functions are applied to map onto a number in $[0, \dots, h - 1]$. The count of each of the set of w cells is incremented by 1. In order to estimate the count of an item, the set of w cells to which each of the w hash-functions map is determined, and the minimum value among all these cells is computed.

Let $q_r^j(x_i^r)$ represents the frequency of the value x_i^r in the j -th cluster. Let m_i be the number of data points assigned to the j -th cluster. Then, the d -dimensional statistics of the record (x_i^1, \dots, x_i^d) for the j -th cluster is given by $(q_1^j(x_i^1), \dots, q_d^j(x_i^d))$. Then, the frequency-based dot-product $D_j(\overline{X}_i)$ of the incoming record with statistics of cluster j is given by the dot product of the fractional frequencies $(q_1^j(x_i^1)/m_j, \dots, q_d^j(x_i^d)/m_j)$ of the attribute values (x_i^1, \dots, x_i^d) with the frequencies of these same attribute values within record \overline{X}_i . Therefore, the corresponding dot product is the following:

$$D_j(\overline{X}_i) = \sum_{r=1}^d q_r^j(x_i^r)/m_j \quad (1.3)$$

The incoming record is assigned to the cluster for which the estimated dot product is the largest.

Batch clustering (like k -means) requires all training data to be stored in the main memory which becomes infeasible for very large or massive datasets. The article [60] proposes a simple and efficient strategy for k -means clustering with restricted buffer where data are processed consecutively in patches of predefined size (Patch Clustering). In [5] the same strategy is transferred to NG (Neural Gas Network) (see Figure 1.6).

Assume a fixed patch size P is chosen such that a number of P examples fits into the buffer. The main idea is to subsequently process patches of size P by batch optimization, thereby enlarging the dataset by patterns which stem from a sufficient statistic of the clusters obtained in the previous patch. A clustering is represented by the cluster centers which are weighted according to the number of data points assigned to it. Note that, with respect to the quantization error, an optimum cluster center is represented by the mean of data points

```

choose the number of clusters  $k$ ;
init  $S = (\text{Sum}^{(A_i)} = \vec{0}, n^{(A_i)} = 0)_{i=1}^k$  for all clusters  $A_i$ 
repeat until all data are processed
    read the next  $P$  data points  $X = \{(\vec{x}_1, 1), \dots, (\vec{x}_p, 1)\}$ 
    extend  $X$  by the points represented in  $S$ :
         $X_{\text{ext}} = X \cup \{(\text{Sum}^{(A_i)} / n^{(A_i)}, n^{(A_i)}) | (\text{Sum}^{(A_i)}, n^{(A_i)}) \in S\}$ 
    cluster on the points in  $X_{\text{ext}}$  counted with multiplicities using
        batch clustering, this gives new clusters  $A_1, \dots, A_k$ 
    update the statistics  $(\text{Sum}^{(A)}, n^{(A)})$  by the cluster centers, i.e.
        
$$S = (\sum_{(\vec{x}^j, n^j) \in X_{\text{ext}} \cap A_i} n^j \cdot \vec{x}^j, \sum_{(\vec{x}^j, n^j) \in X_{\text{ext}} \cap A_i} n^j)_{i=1}^k$$


```

Figure 1.6: Patch clustering algorithm.

assigned to it. To compute the cluster centers, it is sufficient to keep track of the sum of data points assigned to a cluster and the number, i.e. it is sufficient to store $(\text{Sum}^{(A)}, n^{(A)})$ to represent cluster A , where $\text{Sum}^{(A)}$ is the sum of points assigned to the cluster, and $n^{(A)}$ its number. Note that the merging of two clusters A and B is represented by $(\text{Sum}^{(A)} + \text{Sum}^{(B)}, n^{(A)} + n^{(B)})$ which can easily be computed iteratively.

Apart from memory reduction, patch clustering allows a reduction of time because of the faster convergence of the separate patch clustering. This can be explored even further by introducing parallelization into the procedure.

1.4 Clustering using similarity measure based on the L_1 and L_∞ norms

The similarity measure is an important notion in the cluster analysis. It can be defined using various distance functions. The widely used similarity measure is based on the squared Euclidean distance. Such a clustering problem is also known as the minimum sum-of-squares clustering problem. The k -means algorithms are widely used to solve such problems [83, 150]. Optimization techniques such as the branch and bound [39], the variable neighborhood search algorithm [78] and metaheuristics like simulated annealing, tabu search, genetic algorithms [3, 29, 115, 125, 133] have been applied to solve it.

Let $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ be a distance function. Here $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$. The distance

function d can be defined using the L_p -norm:

$$d(x, y) \equiv d_p(u, v) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}, \quad p \geq 1.$$

In particular, the distance function d using the L_1 norm is defined as:

$$d_1(x, y) = \sum_{i=1}^n |x_i - y_i|,$$

and using the L_∞ norm it is:

$$d_\infty(x, y) = \max_{i=1, \dots, n} |x_i - y_i|.$$

Note that both distance functions are nonsmooth.

The similarity measure can also be defined using other Minkowski norms. Among them the d_1 (also known as the Manhattan or City-blocks distance) and d_∞ (also known as the Chebyshev distance) have been frequently used.

There are many applications where clustering algorithms with the distance functions d_1 and d_∞ produce significantly better results than that based on the squared Euclidean norm. In high dimensional data mining applications the distance function d_1 is consistently preferable than the function d_2 [2]. Functions d_1 and d_∞ are more robust to outliers than d_2 [156]. These functions produce the highest identification and best verification rates in closed-set text-independent speaker recognition systems [77]. The use of different distance functions allows one to find different cluster structures in a data set and improve the decision making process.

The paper [31] (see, also [89]) seems to be the first paper where the distance function d_1 was used to define the similarity measure. The paper [130] presents an algorithm for calculating a given number of clusters in a data set by minimizing the total sum of the sums of absolute deviations from the cluster members to the cluster centers (medians).

The ISODATA clustering algorithm [139] using the function d_1 was introduced in [85] where the comparison with the squared Euclidean distance demonstrates that the algorithm with d_1 function is superior when the correlation coefficient is high and negative. In [58],

the authors examine the use of a range of Minkowski norms for clustering. The X -means algorithm introduced in [113] can use various distance functions as similarity measures. In the paper [53], the authors introduce adaptive and non-adaptive partitioning cluster methods for interval data using the d_1 distance function. A one dimensional center-based L_1 -clustering algorithm is proposed in [121] and a local method for solving clustering problems using the d_1 distance function is developed in [120].

A clustering method, called Hyperbox clustering with Ant Colony Optimization, is proposed in [116] where the function d_∞ is used to define the similarity measure. Results show a significant improvement in accuracy and faster processing time when compared to the usual ant colony optimization approach.

1.5 High dimensional data visualization and clustering

High dimensional data visualization is used to analyze the hidden patterns and data relationships, which are hard to illustrate. One of the most popular and classical methods for dimension reduction and data visualization is Principal Component Analysis, PCA [87]. Linear PCA is susceptible to loose useful information in highly nonlinear data sets. Several nonlinear PCA methods have been proposed such as the work presented in [123]. Linear and nonlinear PCA are computationally inexpensive in large data sets, however the memory required by these algorithms is not affordable in very large data sets. The Sammon's mapping [122] is a nonlinear mapping algorithm for visualization of multivariate data, which is shown to be superior to PCA. The Self Organizing Maps [91] are able to learn complex nonlinear relationships of variables in a sample of data points. An overview of methods for data visualization can be found in [142]. The authors in [142] show that the *vector quantization and projection* methods such as SOM give more visual insights to the properties of the data than methods based on mapping or projection of the data points.

Multidimensional (MDS) is known as a collection of visualization techniques for proximity data which yield a set of representative data points in a suitable embedding space. These points are selected in such a way that their mutual distances match the respective proximity values as faithfully as possible [102, 144, 140]. In the more familiar case of data represented by feature vectors, MDS can be used as a visualization tool. It establishes a mapping of these

points to an informative low-dimensional plane or manifold on the basis of pairwise Euclidean distances in the original feature space [90]. The authors in [46] proposed a dynamic learning for metric representations of asymmetric proximity data to improve data visualization. The proposed learning generates two representations (maps) with the row vectors (sending or exporting) and column vectors (receiving or importing) of the matrix, respectively. In [46] the authors supplement the maps with two analysis tools: cluster analysis and distance analysis, which connect and compare the different patterns from the different maps.

Different versions of the SOM for visualization of high dimensional data have been introduced in [42, 86, 98, 145, 147, 152, 153]. An alternative to the SOM algorithm, ViSOM, is introduced by [153] to improve the topological and quantization errors by restricting contractions of neurons. The authors proposed different updating rules for the winner neurons and their neighborhood neurons, which is computationally expensive in very large data sets. The analysis and experimental results show that the ViSOM may offer attractive advantages over the commonly used SOM, PCA, and Sammon’s mapping. The PRSOM [147] is an extension to ViSOM, where the sequential updating rules are extended to optimize a cost function. Unlike the hard assignment in SOM and ViSOM, the assignment of PRSOM is soft such that an input data point belongs to a neuron with certain probability. However, the computational complexity of both ViSOM and PRSOM is shown to be more than the classical SOM and depends quadratically on the number of neurons [147].

In [67], a new self organizing model, the so called Growing Grid (GG), is proposed to the problem of data visualization. The network automatically chooses a height/width ratio suitable for the data distribution. Moreover, locally accumulated statistical values are used to determine where to insert new units (neurons). The Growing Neural Gas (GNG), introduced in [66], is an improvement to the Neural Gas (NG) algorithm [101]. More specifically, it is an incremental version of the NG algorithm which does not require the pre-setting of the network size. The GNG algorithm is able to make explicit topological relations of input data.

The growing hierarchical SOM (GHSOM) generates multiple independent layers of the maps on the top of the first layer to cluster input data points in a hierarchical manner and allows for adaptation of the network architecture simultaneously [117]. The GHSOM is presented to be suitable for visualization of high dimensional document data sets rather than

the SOM, although manipulating multiple layers of the GHSOM is liable to high computational effort in very large data sets. Following the training phase of the GHSOM, a novel visualization method, namely, the ranked centroid projection (RCP), is introduced in [152] for projecting the document vectors to a hierarchy of output maps. The results of the results for the cluster validity Davies-Bouldin index show that RCP produces a better result, which denotes that for the purpose of classification and categorization, the RCP is to be preferred over the SOM, PCA and Sammon’s mapping [152].

The expanding SOM, ESOM, for data visualization is proposed in [86]. The ESOM utilizes the SOM by employing an additional factor, the expanding coefficient, which is used to push neurons away from the center of all data points during the learning process. The authors show that the ESOM has advantages over the SOM and the growing models of the SOM, where the growth criteria in the latter increases the computational time in the learning process. To cope with the problems of selecting learning rate in conventional SOM, the authors in [42] proposed RPSOM in which, for each input data point, the RPSOM adaptively chooses several rivals of the best matching unit (BMU) and penalizes their associated models a little far away from the input data point [42]. The RPSOM converges much faster than the SOM and the two-phase SOM [143] moreover, outperforms them in the term of quantization error.

1.5.1 Self organizing map

A self organizing map (SOM) is a well known data mining tool where the aim is to visualize a high dimensional data into usually a two dimensional grid [91]. The SOM consists of a set of neurons that gradually adapt to input data by competitive learning. It creates ordered prototypes which preserve the topology of the mapped data and make the SOM suitable for cluster analysis [151]. The adaptation of neurons is based on a similarity measure, which is usually Euclidean distance, and their repositioning in a two dimensional space using a learning algorithm. The performance of the SOM strongly depends on a learning algorithm [62, 61, 71, 75]. Furthermore, the SOM and many its modifications are sensitive to the initialization of neurons and the topology of the map.

The SOM is an unsupervised neural network [91] that usually contains a two dimensional array of neurons $\Psi = \{w_1, \dots, w_q\}$. Assume that we are given the set of m input data vectors

$A = \{x_1, \dots, x_m\}$ where $x_i \in \mathbb{R}^n$, $i = 1, \dots, m$. In the SOM a weight $w_j \in \mathbb{R}^n$ is associated with the neuron j , $j = 1, \dots, q$. For given $j \in \{1, \dots, q\}$ define the following set:

$$S_j = \{x_k : d(x_k, w_j) < d(x_k, w_l), l \neq j, l = 1, \dots, q\} \quad (1.4)$$

where

$$d(x, y) = \|x - y\| = \left(\sum_{t=1}^n (x^t - y^t)^2 \right)^{1/2}, \quad x, y \in \mathbb{R}^n$$

is the Euclidean distance.

One data point x_i , $i \in \{1, \dots, m\}$ at a time is presented to the network and is compared with all weight vectors. The nearest w_j , $j = 1, \dots, q$ is selected as the *best matching unit* (BMU) for the i -th data point. This data point is mapped to the best matching neuron. Therefore,

$$S_j = S_j \cup x_i.$$

The set of neighborhood weights $N_c = \{w_l : p(c, l) \leq r, l \neq c\}$ around the BMU are updated as follows:

$$w_j := w_j + \alpha(\tau)h(\tau)(x_i - w_j). \quad (1.5)$$

where $p(c, l)$ is the distance between the BMU and the neighborhood neuron l in 2-Dim coordinates of the network topology and r is the predefined radius. Furthermore, $p(c, l) \in \mathbb{N}$ and $0 < p(c, l) \leq r$. The ordered neurons of the SOM preserve the topology of the mapped data and make the SOM to be very suitable for cluster analysis [151]. The aim is to solve the vector quantization problem:

$$\text{minimize } E = \frac{1}{m} \sum_{i=1}^m \|x_i - w_c\|^2, \quad (1.6)$$

where w_c is the weight of the BMU of x_i , $i = 1, \dots, m$. Note that the vector quantization problem (1.6) is used to measure the quality of the self organizing map.

A general description of the SOM algorithm is as follows.

Algorithm 1. SOM algorithm

Step 1. Initialize the dimension of the network, the maximum number of iterations (T), a radius (r) of the network and weight vectors w_j , $j = 1, \dots, q$. Set stopping criterion ϵ_0 and iteration counter $\tau := 0$.

Step 2. Select data $x_i, i = 1, \dots, m$ and find its closest neuron c , that is

$$c := \underset{j=1, \dots, q}{\operatorname{argmin}} \|x_i - w_j\|. \quad (1.7)$$

Step 3. Update the set of neighborhood neurons $w_j \in N_c$ using (1.5) (Here h is a neighborhood function and $\alpha(\tau)$ is a learning rate at the iteration τ .)

Step 4. If all input data are presented to the network go to Step 5, otherwise go to Step 2.

Step 5. Calculate E using (1.6). If $E < \epsilon_0$ or $\tau > T$ terminate, otherwise set $\tau := \tau + 1$ and go to Step 2.

The neighborhood function $h(\tau)$ and learning rate $\alpha(\tau)$ in Step 3 of Algorithm 1 are defined as

$$h(\tau) = \exp\left(-\frac{r_0^2}{2\sigma(\tau)^2}\right), \quad \alpha(\tau) = \eta \frac{T - \tau}{\tau}, \quad \eta \geq 1. \quad (1.8)$$

Here

$$\sigma(\tau) = p_0 - \frac{\tau}{T}, \quad p_0 \geq 1.$$

The neighborhood function (1.8) plays an important role in the SOM. Usually h is a decreasing exponential function of τ . The learning rate α is a decreasing linear function of τ that reduces the effect of the neighborhood function h as $\tau \rightarrow T$. The learning procedure of SOM is explored in details in the next section.

1.5.2 Modifications of the SOM

The performance of the SOM depends on the initialization of neurons, the choice of the topology and a learning algorithm (Steps 2 and 3 in Algorithm 1). Different versions of the SOM have been proposed to improve its performance (see, for example, [4, 7, 8, 13, 30, 41, 43, 44, 50, 68, 72, 94, 95, 126, 134, 141, 146, 149, 151, 152, 162]). The paper [146] presents an automated detection algorithm based on the SOM assuming that the training data is an adequate representation of the sample distribution. Therefore, the SOM is trained using a small proportion of the sample data set and the algorithm defines a region around prototypes

by employing a parameter r_j that represents the distance of the farthest projected sample into the neuron j , $j = 1, \dots, q$ (where q is the number of neurons). The upcoming samples are distributed into the network and novelties are those samples which cannot fit into these regions.

A combinatorial two-stage clustering algorithm based on the SOM is introduced in [44]. The numerical results using the Ant Colony Optimization technique and the k -means demonstrate the superiority of the proposed algorithm in comparison with the SOM and the k -means. In [30], an enhanced Clusot algorithm [28] is applied in the SOM for automatic cluster detection.

In [134] the SOM's prototypes are clustered hierarchically based on the density instead of the distance dissimilarity. A two-stage algorithm is proposed in [151] that applies the graph cut algorithm (see [128]) to the SOM output. Results demonstrate that this algorithm requires less computational time than direct clustering methods.

A dynamic SOM is a version of the SOM where its structure is not fixed during the learning phase. In [4], a growing self organizing map (GSOM) is presented which defines a spread factor to measure and control the growth of the network. Similarly in [13], a multi level interior growing SOM is introduced. Unlike the GSOM, which allows the growth only from border sides, this algorithm allows neurons to grow also from an interior node of the map.

The Growing Neural Gas (GNG), introduced in [66], is an improvement to the Neural Gas (NG) algorithm [101]. More specifically, it is an incremental version of the NG algorithm which does not require the pre-setting of the network size. The GNG algorithm is able to make explicit topological relations of input signals.

In [67], a new self organizing model, the so called Growing Grid (GG), is proposed to overcome some drawbacks of existing models. The network automatically chooses a height/width ratio suitable for the data distribution. Moreover, locally accumulated statistical values are used to determine where to insert new units.

A novel artificial neural-network architecture, called the growing hierarchical SOM (GH-SOM), is proposed in [117] to resolve two limitations of the SOM due to its static architecture as well as the limited capabilities for the representation of hierarchical relations of the data.

A parameter-less self-organizing map algorithm (PLSOM), proposed in [24], eliminates SOM parameters such as the learning rate and neighborhood size and calculates values of these parameters using the local quadratic fitting error of the map. This allows the map to make large adjustments in response to unfamiliar inputs, i.e., inputs that are not well mapped, while not making large changes in response to inputs it is already well adjusted to. Unfortunately, the PLSOM has the property that it overreacts to extreme outliers, even after long periods of training [25].

Another extension of the SOM algorithm is presented in [75]. This extension automatically calculates the learning parameters during the training. The algorithm is based on the Kalman filter estimation technique and the idea of the topographic product. The Fast Learning SOM (FLSOM) algorithm is presented in [62], which is based on the application of the simulated annealing (SA) metaheuristics to the SOM learning. The SA is used to modify the learning rate factor in an adaptive way. The FLSOM shows a better convergence than the original SOM algorithm.

A two-level clustering algorithm is proposed in [104] to improve clustering output. At the first level of the algorithm the data is trained by the SOM and at the second level the incremental clustering approach is applied to the output of SOM. The optimal number of clusters is found by applying the rough set theory to the output of the SOM. SA algorithm is adopted to minimize the uncertainty due to the overlapping between clusters, which is detected using the rough set theory. Similarly in [105], the overlapping, caused by the cluster structures, is removed by using a genetic algorithm instead of using SA.

All modifications of the SOM, described above, do not include any specific procedure to find initial weights of neurons. Therefore, the most of these algorithms are still sensitive to the initialization of neurons. Furthermore, the most of these algorithms, including the SOM, are not efficient in large data sets. In this research, a new version of the SOM is proposed to address these drawbacks. The proposed version includes an algorithm for initialization of neurons based on the split and merge procedure. The high dense areas in input data space are detected by this procedure. Then neurons are generated in those detected areas. Initialization of neurons in such areas accelerates the convergence of the algorithm and makes it applicable to large data sets. A new topology is presented to restrict the adaptation of the

neurons to a neighborhood which is located in the same high density area. Such an approach leads to a better local minimum of the quantization error than that of by the SOM.

1.6 Handwritten digits recognition using SOM

Handwritten digits recognition has many real world applications such as postal mail sorting or form data processing. Several algorithms based on Neural Network [154, 112, 6, 155, 161, 70, 82], Machine Learning [108, 100, 55], Statistics [158, 52, 137], hybrid and mixture models techniques [132, 38, 137, 76] have been proposed to solve this problem. Among these algorithms, the Self Organizing Maps (SOM) [91] has been shown to be a promising tool to solve such problems [33, 110, 51, 80, 159, 127].

Different versions of the SOM have been applied to solve the handwritten digits recognition problem. A Hierarchically Growing Hyperbolic SOM (H2SOM) [109] is proposed for incremental training with an automated adaptation of lattice size to improve the quantization error. Furthermore, the H2SOM is utilized by a best match search to speed up the training phase in large data sets. The experimental results of H2SOM on MNIST data set demonstrate the dominance of this method over the conventional SOM in the sense of accuracy. In [33], the authors introduce a hybrid method CNN¹-SOM for classification of handwritten digits data. Moreover, a rejection strategy is presented to change the topology of the map during training for improving the reject quality. An Adaptive-Subspace SOM (AOSSOM) algorithm is introduced by [159] and this algorithm is applied to the MNIST data set. The authors in [161] propose a gradient-decent method, Linear Manifold SOM (LMSOM), to find multiple linear manifolds by using the self organizing maps to minimize the projection error function in the learning process. The LMSOM is tested on the MNIST data set and the authors claim that this method overcomes several shortcomings of the Adaptive-Subspace-SOM [92].

A neural model, referred to Locally Linear Online Mapping (LLOM), is introduced by [161] to solve the handwritten digits recognition problem. The idea is to model nonlinear manifolds with mixtures of local linear manifolds via online learning. The mixture of local models is constructed using the Self Organizing framework and the online learning is used for time reduction in large data sets. The classification accuracy of the LLOM presented by the

¹Convolutional Neural Network

authors, is better than that of the ASSOM and AMSOM. A modification of the SOM, Binary Tree Time Adaptive SOM (BTASOM), is introduced in [127] and tested on the MNIST data set. The BTASOM is a dynamic binary tree in such a way that each of the tree's node contains a fixed predetermined number of neurons. But the number of levels of the tree and the children of each node are determined dynamically and via the learning process. The comparative results reported by [127] outline the efficiency of the BTASOM in comparison with the Growing Hierarchical SOM [23]. [80] propose an Elastic Matching [136] concept for the problem of handwritten character recognition, where a character is transformed to match it with the template image with keeping its topology (GP+ASSOM+TM). To implement such transformation the SOM is employed. Furthermore, the authors in [80] show that using a hybrid technique based on the SOM and template matching improves the performance of the template matching technique.

1.7 Summary

In this chapter, we presented a literature review on recent hierarchal and partitional clustering algorithms as well as the data visualization approaches including the modifications of the SOM. Furthermore, the shortcomings of these approaches in large data mining problems are discussed.

Chapter 2

Optimization formulation of clustering problem

2.1 Introduction

In this chapter we present three different optimization formulations of the clustering problem. Assume that a finite set A of points in the n -dimensional space \mathbb{R}^n is given, that is

$$A = \{a^1, \dots, a^m\}, \text{ where } a^i \in \mathbb{R}^n, \ i = 1, \dots, m.$$

The hard unconstrained clustering problem is the distribution of the points of the set A into a given number k of disjoint subsets A^j , $j = 1, \dots, k$, which are called clusters. Each cluster A^j can be identified by its center $x^j \in \mathbb{R}^n$, $j = 1, \dots, k$. Data points from the same cluster are similar and data points from different clusters are dissimilar to each other. The similarity between points can be measured using different distance functions. Let $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ be a distance function. Here $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$.

In this chapter, we define the distance function d using the squared Euclidean norm, L_1 -norm and L_∞ -norm:

1. The distance function using the squared Euclidean norm:

$$d(x, y) = \sum_{i=1}^n (x_i - y_i)^2,$$

2. The distance function using the L_1 -norm:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|,$$

3. The distance function using the L_∞ -norm:

$$d(x, y) = \max_{i=1, \dots, n} |x_i - y_i|.$$

Note that the distance function d defined using the L_1 and L_∞ norms is nonsmooth. The use of different distance functions can lead to finding of different cluster structures in the data set.

2.2 Combinatorial formulation of the clustering problem

Denote the set of k clusters in the set A by $\bar{A} = (A^1, \dots, A^k)$ and a set of all possible k -partitions of the set A by \tilde{A} . Then *the combinatorial formulation* can be given as:

$$\text{minimize } \Psi_k(\bar{A}) = \frac{1}{m} \sum_{j=1}^k \sum_{a \in A^j} d(x^j, a) \quad (2.1)$$

subject to

$$\bar{A} = (A^1, \dots, A^k) \in \tilde{A}. \quad (2.2)$$

Here x^i is the center of the cluster A^i , $i = 1, \dots, k$ which can be found by solving the following optimization problem:

$$\text{minimize } \frac{1}{|A^j|} \sum_{a \in A^j} d(x, a) \quad \text{subject to } x \in \mathbb{R}^n. \quad (2.3)$$

Here $|\cdot|$ stands for the cardinality of a set. If the squared Euclidean distance is used for the similarity measure then the center x^j can be found explicitly as follows:

$$x^j = \frac{1}{|A^j|} \sum_{a \in A^j} a, \quad j = 1, \dots, k. \quad (2.4)$$

Note that in this formulation decision variables are nonempty subsets of the set A and therefore optimization algorithms cannot be directly applied to solve the problem (2.1)-(2.2). The combinatorial formulation can be used to solve clustering problems only in very small data sets.

2.3 Mixed integer nonlinear programming formulation of the clustering problem

Alternatively, the problem of the finding k clusters in the set A can be reduced to the following optimization problem:

$$\min \psi_k(x, w) = \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k w_{ij} d(x^j, a^i) \quad (2.5)$$

subject to

$$x = (x^1, \dots, x^k) \in \mathbb{R}^{k \times n}, \quad (2.6)$$

$$\sum_{j=1}^k w_{ij} = 1, \quad i = 1, \dots, m, \quad (2.7)$$

$$w_{ij} \in \{0, 1\}, \quad i = 1, \dots, m, \quad j = 1, \dots, k. \quad (2.8)$$

Here w_{ij} is the association weight of the pattern a^i with the cluster j , given by

$$w_{ij} = \begin{cases} 1, & \text{if } a^i \text{ is allocated to the cluster } j; \\ 0, & \text{otherwise.} \end{cases}$$

w is an $m \times k$ matrix.

The problem (2.5)-(2.8) is called *the mixed integer nonlinear programming formulation* of the clustering problem. It contains mn integer variables $w_{ij}, i = 1, \dots, m, j = 1, \dots, k$ and kn continuous variables $x^j \in \mathbb{R}^n, j = 1, \dots, k$.

Cluster centers $x^j, j = 1, \dots, k$ can be found by solving the problem (2.3). If the similarity measure is defined using the squared Euclidean norm then x^j is a centroid of the cluster $A^j, j = 1, \dots, k$ and it can be found applying the formula (2.3). In this case the problem (2.5)-(2.8) becomes integer programming problem as cluster centers $x^j, j = 1, \dots, k$ are not

decision variables.

2.4 Nonsmooth nonconvex optimization formulation of the clustering problem

Nonsmooth nonconvex optimization formulation of the clustering problem is as follows [15, 22, 27]:

$$\min f_k(x) \text{ subject to } x = (x^1, \dots, x^k) \in \mathbb{R}^{k \times n}, \quad (2.9)$$

where

$$f_k(x^1, \dots, x^k) = \frac{1}{m} \sum_{i=1}^m \min_{j=1, \dots, k} d(x^j, a^i). \quad (2.10)$$

If $k = 1$ then the function f_k is convex and it is nonconvex if $k > 1$ due to the minimum operation. If the similarity measure is defined using the squared Euclidean distance then for $k = 1$ the function f_k is smooth that is it is continuously differentiable for any $x \in \mathbb{R}^n$. However, for $k \geq 2$ this function is nonsmooth due to the minimum operation that is its gradient does not exist at all $x \in \mathbb{R}^{k \times n}$. If the similarity measure is defined using the L_1 or L_∞ norms then the function f_k is nonsmooth for all $k \geq 1$. This is due to the minimum operation and the fact that both L_1 and L_∞ norm based distance functions are nonsmooth.

2.5 Comparison of different formulations

In this subsection we compare objective functions in different formulations of the clustering problems. We call objective functions Ψ_k, ψ_k and f_k *cluster functions*. Note that the objective function Ψ_k explicitly depends, in particular, on clusters (or subsets of the set A). Optimization methods cannot be applied to minimize such functions and the combinatorial formulation cannot be used to solve clustering problems considered in this chapter. Therefore, we compare only objective functions ψ_k and f_k .

Comparing these two functions (also two different formulations of the clustering problem) one can note that:

1. The objective function ψ_k depends on variables w_{ij} , $i = 1, \dots, m$, $j = 1, \dots, k$ (coefficients, which are integers) and x^1, \dots, x^k , $x^j \in \mathbb{R}^n$, $j = 1, \dots, k$ (cluster centers,

which are continuous variables). However, the function f_k depends only on continuous variables x^1, \dots, x^k .

2. The number of variables in Problem (2.5)-(2.8) is $(m+n) \times k$ whereas in Problem (2.9) this number is only $n \times k$. Notice that the number of variables in Problem (2.9) does not depend on the number m of instances. In many real world data sets the number of instances m is substantially greater than the number of features n .
3. Since the function f_k is represented as a sum of minima functions it is nonsmooth for $k > 1$, that is it is not differentiable everywhere. Both functions ψ_k and f_k are nonconvex for $k > 1$.
4. Problem (2.5)-(2.8) is mixed integer nonlinear programming problem and Problem (2.9) is nonsmooth global optimization problem. However, they are equivalent in the sense that their global minimizers coincide [27].

Items 1 and 2 can be considered as advantages of the nonsmooth optimization formulation (2.9) of the clustering problem. Nonsmooth optimization models of unsupervised and supervised data classification problems are also discussed in [11, 10, 12, 32, 54].

2.6 Summary

Three different optimization formulations of the clustering problem: combinatorial, mixed integer nonlinear programming and nonconvex nonsmooth are discussed in this chapter. The comparative analysis of these formulations is also presented.

Chapter 3

Incremental clustering algorithm

3.1 Introduction

The objective function (2.10) in Problem (2.9) is nonconvex and the problem itself is global optimization problem. However, the most of existing global optimization techniques cannot be applied to solve this problem because its size becomes large as the number of clusters increases. These techniques are extremely time consuming for solving such clustering problems. Therefore, solving clustering problems in large data sets is out of reach of most of global optimization algorithms. Any local optimization algorithms starting from a given point will end up at the closest local minimizer which can be significantly different from the global minimizer. Global minimizers provide the best cluster structure of a data set with the smallest number of clusters.

We propose to apply local search methods, including heuristic methods such as the k -means and deterministic methods of optimization, to solve clustering problems. Such methods are very fast even in large data sets however their success in finding global or near global solutions to the clustering problems highly depends on the choice of starting points. In order to generate such points we introduce the auxiliary cluster function. Starting points are found by minimizing this function.

3.2 The auxiliary cluster problem

Assume that the solution x^1, \dots, x^{l-1} , $l > 1$ to the $(l-1)$ -clustering problem is known. Define by

$$r_{l-1}^i = \min_{j=1, \dots, l-1} d(x^j, a^i)$$

the distance between the data point a^i , $i = 1, \dots, m$ and its cluster center. We will also use the notation r_{l-1}^a for the data point $a \in A$. Consider the following function:

$$\bar{f}_l(y) = \frac{1}{m} \sum_{i=1}^m \min \{r_{l-1}^i, d(y, a^i)\}, \quad y \in \mathbb{R}^n. \quad (3.1)$$

We call this function *the l -th auxiliary cluster function*. One can see that the function \bar{f}_l is represented as a sum of minima of constants r_{l-1}^i and the distance function $d(y, a^i)$, $i = 1, \dots, m$ which is convex. Therefore the function \bar{f}_l is nonsmooth and in general, nonconvex. *The l -th auxiliary clustering problem* is formulated as follows:

$$\min \bar{f}_l(y) \text{ subject to } y \in \mathbb{R}^n. \quad (3.2)$$

This is a nonsmooth global optimization problem and it may have many local minimizers. The success of any local method for solving this problem heavily depends on the choice of starting points. Therefore it is important to develop an algorithm to generate good starting points for its solution. One such algorithm will be designed in Section 3.4.

3.3 An incremental clustering algorithm

In this section we describe a general scheme for an incremental clustering algorithm. Incremental algorithms are becoming popular in data mining and in particular, in the cluster analysis. There are two types of incremental algorithms. In algorithms of the first type new data points are dynamically added at each iteration of the algorithm to the data set and clusters are updated accordingly. In algorithms of the second type a data set is static and an algorithm computes clusters incrementally. Such incremental clustering algorithms build clusters dynamically adding one cluster center at a time. Therefore this type of incremental

algorithms can also be called *sequential clustering algorithms*. In this chapter we consider only the second type of incremental clustering algorithms. The general scheme of such algorithms for finding the k -partition of the set A is as follows.

Algorithm 2. An incremental clustering algorithm.

Input: The data set A and the number k of clusters to be computed.

Output: The l -partition of the set A with $l = 1, \dots, k$.

Step 1. (Initialization). Compute the center $x^1 \in \mathbb{R}^n$ of the set A . Set $l := 1$.

Step 2. (Stopping criterion) Set $l := l + 1$. If $l > k$ then stop. The k -partition problem has been solved.

Step 3. (Computation of the next cluster center). Find a starting point $\bar{y} \in \mathbb{R}^n$ for the l -th cluster center by solving Problem (3.2).

Step 4. (Refinement of all cluster centers). Select $(x^1, \dots, x^{l-1}, \bar{y})$ as a new starting point to solve the l -partition problem (2.9) (or the problem (2.5)-(2.8)) (when $k = l$). Let y^1, \dots, y^l be a solution to this problem.

Step 5. (Solution to the l -partition problem). Set $x^j := y^j$, $j = 1, \dots, l$ as a solution to the l -th partition problem and go to Step 2.

Remark 3.3.1. To date incremental algorithms were designed for solving the minimum sum-of-squares clustering problems [19, 17, 96]. In this chapter, we develop incremental algorithms also for solving clustering problems where the similarity measure is defined using L_1 and L_∞ norms.

Remark 3.3.2. One can see that Algorithm 2 in addition to the k -partition problem solves also all intermediate l -partition problems where $l = 1, \dots, k-1$. Step 3, where one finds a starting point for the l -th cluster center, and Step 4 are the most important steps of Algorithm 2. In the next section we design an algorithm for finding starting points for the l -th cluster center.

3.4 Computation of starting points for cluster centers

Various initial seeding procedures were introduced to improve the efficiency of the k -means algorithms. Such procedures were considered to design different versions of the k -means algorithm such as the X -means [113] and k -means++ [9] algorithms. In the paper [34], a comparative study of various initialization methods for the k -means algorithm is presented. In this section we discuss an initial seeding procedure based on nonsmooth optimization formulation of the clustering problem. This procedure is designed for the incremental algorithm described in the previous section. This means that in order to find initial points for the l -th cluster center, $l > 1$, it is assumed that cluster centers for the $(l - 1)$ -partition problem are known.

Given the solution x^1, \dots, x^{l-1} , $l > 1$ to the $(l - 1)$ -clustering problem one can divide the whole space \mathbb{R}^n into two subsets as follows:

$$S_1 = \{y \in \mathbb{R}^n : d(y, a) \geq r_{l-1}^a, \forall a \in A\},$$

$$S_2 = \{y \in \mathbb{R}^n : \exists a \in A \text{ such that } d(y, a) < r_{l-1}^a\}.$$

All cluster centers $x^1, \dots, x^{l-1} \in S_1$. It is clear that $S_1 \cup S_2 = \mathbb{R}^n$ and $S_1 \cap S_2 = \emptyset$. Moreover,

$$\bar{f}_l(y) = f_{l-1}(x^1, \dots, x^{l-1}) = \frac{1}{m} \sum_{a \in A} r_{l-1}^a, \quad \forall y \in S_1, \quad (3.3)$$

that is the l -th auxiliary cluster function is constant on the set S_1 and any point from this set is a global maximizer of the function \bar{f}_l . In general, a local search method will terminate at any of these points. Hence, points from the set S_1 cannot be considered as starting points to solve the problem (3.2). Therefore, starting points should be chosen from the set S_2 .

For any $y \in S_2$ one can divide the set A into two subsets as follows:

$$B_1(y) = \{a \in A : d(y, a) \geq r_{l-1}^a\},$$

$$B_2(y) = \{a \in A : d(y, a) < r_{l-1}^a\}.$$

It is obvious that the set $B_2(y)$ contains all data points which are closer to the point y than their cluster centers. Since $y \in S_2$ the set $B_2(y) \neq \emptyset$. Furthermore, $B_1(y) \cap B_2(y) = \emptyset$ and $A = B_1(y) \cup B_2(y)$. Then

$$\bar{f}_l(y) = \frac{1}{m} \left(\sum_{a \in B_1(y)} r_{l-1}^a + \sum_{a \in B_2(y)} d(y, a) \right).$$

The difference $v_l(y)$ between the value $f_{l-1}(x^1, \dots, x^{l-1})$ (see (3.3)) and the value of the l -th auxiliary cluster function at y is:

$$v_l(y) = \frac{1}{m} \sum_{a \in B_2(y)} (r_{l-1}^a - d(y, a))$$

which can be rewritten as

$$v_l(y) = \frac{1}{m} \sum_{a \in A} \max \{0, r_{l-1}^a - d(y, a)\}. \quad (3.4)$$

It is obvious that data points $a \in A$, which are not among cluster centers x^1, \dots, x^{l-1} , belong to the set S_2 , because these points attract at least themselves. Therefore in order to compute starting points for solving the auxiliary clustering problem (3.2) we use data points $a \in A \setminus S_1$. We introduce the following number:

$$v_{max}^1 = \max_{a \in A \setminus S_1} v_l(a). \quad (3.5)$$

The number v_{max}^1 is the largest decrease of the auxiliary cluster function $\bar{f}_k(y)$ comparing to the value $f_{l-1}(x^1, \dots, x^{l-1})$ for all $y \in A \setminus S_1$. Among all data points $a \in A$, a point $\bar{a} \in A \setminus S_1$, satisfying the condition $v_l(\bar{a}) = v_{max}^1$, provides the largest decrease of the cluster function f_l comparing with the value $f_{l-1}(x^1, \dots, x^{l-1})$ if \bar{a} is chosen as the l -th cluster center.

Let $\gamma_1 \in [0, 1]$ be a given number. Define the following subset of A :

$$\bar{A}_1 = \{a \in A \setminus S_1 : v_l(a) \geq \gamma_1 v_{max}^1\}. \quad (3.6)$$

The set \bar{A}_1 contains points $a \in A \setminus S_1$ which provide sufficient decrease of the cluster function

f_l comparing with the value $f_{l-1}(x^1, \dots, x^{l-1})$ if these points are chosen as the l -th cluster center. If $\gamma_1 = 1$ then we choose only points with largest decrease and if $\gamma_1 = 0$ then $\bar{A}_1 = A \setminus S_1$.

For each $a \in \bar{A}_1$ we compute the set $B_2(a)$ and its center $c(a)$. In this stage we replace points $a \in \bar{A}_1$ by points $c(a)$, because the center $c(a)$ is the better representative of the set $B_2(a)$ than the point a . If the similarity measure is defined using the squared Euclidean norm then $c(a)$ is the centroid of the set $B_2(a)$. In all other cases $c(a)$ is defined as the solution to the following convex minimization problem:

$$\text{minimize } \frac{1}{|B_2(a)|} \sum_{b \in B_2(a)} d(x, b) \quad \text{subject to } x \in \mathbb{R}^n.$$

As a result we get the following set:

$$\bar{C}_1 = \{c \in \mathbb{R}^n : \exists a \in \bar{A}_1 \text{ such that } c = c(a)\}.$$

Then using (3.4) we compute $v_l^2(c) = v_l(c)$ for each $c \in \bar{C}_1$. Finally, we compute the maximum of all numbers $v_l^2(c)$, $c \in \bar{C}_1$:

$$v_{max}^2 = \max_{c \in \bar{C}_1} v_l^2(c). \quad (3.7)$$

Let $\gamma_2 \in [0, 1]$ be a given number. We define the following subset of \bar{C}_1 :

$$\bar{A}_2 = \{c : c \in \bar{C}_1 \text{ and } v_l^2(c) \geq \gamma_2 v_{max}^2\}. \quad (3.8)$$

The set \bar{A}_2 contains all points $c \in \bar{C}_1$ which provide sufficient decrease of the cluster function f_l comparing with the value $f_{l-1}(x^1, \dots, x^{l-1})$ if these points are chosen as the l -th cluster center. If $\gamma_2 = 1$ then we choose points with the largest decrease and if $\gamma_2 = 0$ then $\bar{A}_2 = \bar{C}_1$.

All points from the set \bar{A}_2 are considered as starting points for solving the auxiliary clustering problem (3.2).

Applying a local search algorithm, Problem (3.2) is solved starting from each point of the set \bar{A}_2 . Such local search algorithms will be discussed in Chapter 4. A local search algorithm generates the same number of solutions as the number of starting points. The set of these solutions is denoted by \bar{A}_3 . Since the local method can arrive to the same solution

starting from different points we remove from the set \bar{A}_3 solutions which are sufficiently close to each other keeping only one of them. Sufficiently close solutions can be defined using some predefined tolerance. It is clear that $\bar{A}_3 \neq \emptyset$.

Next we define

$$\bar{f}_l^{min} = \min_{y \in \bar{A}_3} \bar{f}_l(y). \quad (3.9)$$

Let $\gamma_3 \in [1, \infty)$ be a given number. Compute the following set:

$$\bar{A}_4 = \{y \in \bar{A}_3 : \bar{f}_l(y) \leq \gamma_3 \bar{f}_l^{min}\}. \quad (3.10)$$

If $\gamma_3 = 1$ then \bar{A}_4 contains stationary points of the problem (3.2) with the lowest value \bar{f}_l^{min} .

If γ_3 is sufficiently large then $\bar{A}_4 = \bar{A}_3$.

Points from the set \bar{A}_4 are considered as a starting point for the l -th cluster center in Step 4 of Algorithm 2. Summarizing all steps for computing the set \bar{A}_4 we can design the following algorithm for finding starting points to solve the clustering problem (2.9).

Algorithm 3. Computation of starting points.

Input: The data set A and the solution x^1, \dots, x^{l-1} , $l > 1$ to the $(l-1)$ -clustering problem.

Output: The set of starting points for the l -th cluster center.

Step 0. (Initialization). Select numbers $\gamma_1, \gamma_2 \in [0, 1]$ and $\gamma_3 \in [1, \infty)$.

Step 1. Compute v_{max}^1 using (3.5) and the set \bar{A}_1 using (3.6).

Step 2. Compute v_{max}^2 using (3.7) and the set \bar{A}_2 using (3.8).

Step 3. Apply a local search algorithm to compute a set \bar{A}_3 of solutions to the auxiliary clustering problem (3.2) using points from the set \bar{A}_2 as starting points.

Step 4. Compute \bar{f}_l^{min} using (3.9) and the set \bar{A}_4 using (3.10). \bar{A}_4 is the set of starting points to solve the clustering problem (2.9).

Thus, we use more than one starting point to solve the clustering problem (2.9) in Step 4 of Algorithm 2. Moreover, these points always guarantee decrease of the clustering function at each iteration of the incremental algorithm and they are away from each other in the

search space. Such an approach allows us to apply local search methods to solve the global optimization problem (2.9).

3.5 The modified incremental clustering algorithm

In this section the incremental clustering Algorithm 2 is modified by applying Algorithm 3 in Step 3. Then Algorithm 2 for solving the k -partition Problem (2.9) can be rewritten as follows:

Algorithm 4. A multistart incremental algorithm.

Input: The data set A and the number k of clusters to be computed.

Output: The l -partition of the set A with $l = 1, \dots, k$.

Step 1. (Initialization). Compute the center $x^1 \in \mathbb{R}^n$ of the set A . Set $l := 1$.

Step 2. (Stopping criterion) Set $l := l + 1$. If $l > k$ then stop. The k -partition problem has been solved.

Step 3. (Computation of the set of starting points for the next cluster center). Apply Algorithm 3 to find a set \bar{A}_l of starting points for the l -th cluster center.

Step 4. (Refinement of all cluster centers). For each $\bar{y} \in \bar{A}_l$ select $(x^1, \dots, x^{l-1}, \bar{y})$ as a starting point to solve the l -partition problem (2.9) (when $k = l$). Let $(y^1(\bar{y}), \dots, y^l(\bar{y}))$ be a solution to this problem and $f_{l\bar{y}} = f_l(y^1(\bar{y}), \dots, y^l(\bar{y}))$ be a value of the cluster function at this solution.

Step 5. (Computation of the best solution). Compute

$$f_l^{min} = \min_{\bar{y} \in \bar{A}_l} f_{l\bar{y}}$$

and the set of best solutions

$$C = \left\{ (y^1(\bar{y}), \dots, y^l(\bar{y})) : f_{l\bar{y}} = f_l^{min} \right\}.$$

Step 6. (Solution to the l -partition problem). Take any $(y^1(\bar{y}), \dots, y^l(\bar{y})) \in C$, set $x^j := y^j(\bar{y})$, $j = 1, \dots, l$ as a solution to the l -th partition problem and go to Step 2.

3.6 Summary

In this chapter an incremental approach for finding cluster centers is proposed. This algorithm computes clusters gradually starting from one cluster which is the whole data set. Furthermore, we introduced an auxiliary clustering problem to find starting points for cluster centers using the incremental approach.

Chapter 4

Solving optimization clustering problems

4.1 Introduction

In this chapter we discuss local search algorithms for solving both the auxiliary clustering problem (3.2) and the clustering problem (2.9). We will consider three different algorithms for this purpose: (i) k -means type heuristic algorithm; (ii) the nonsmooth optimization algorithms and (iii) an algorithm based on smoothing techniques.

4.2 k -means type heuristic algorithm

This algorithm was discussed in [19] (see, also [17]). In order to solve the auxiliary clustering problem (3.2) this algorithm fixes the first $(l - 1)$ cluster centers and updates only the l -th center. The algorithm proceeds as follows.

Algorithm 5. Heuristic algorithm for minimizing the auxiliary cluster function.

Input: The data set A and the starting point $y \in \bar{A}_4$.

Output: Local minimizer of Problem (3.2).

Step 1. Compute the set $B_2(y)$ and its center c .

Step 2. Compute the set $B_2(c)$ and its center.

Step 3. Recompute the set $B_2(c)$ and its center until no more data points escape or return to this set. The last center c is the solution to Problem (3.2).

It is proved in [19] that Algorithm 5 converges to the local minimizer of Problem (3.2) after finite number of iterations.

We apply the k -means algorithm for solving Problem (2.9). This algorithm converges to the local solutions of the problem (2.9).

4.3 Nonsmooth optimization methods

There are three different optimization problems to be solved to find cluster centers: (i) the optimization problem (2.9) to find all cluster centers; (ii) the optimization problem (3.2) to find starting points for cluster centers; and (iii) an optimization problem for finding centers of each cluster. In this section we describe algorithms for solving each of these problems. We start with the description of the algorithm for solving the problem (2.9).

4.3.1 An algorithm for solving Problem (2.9)

The objective function (2.10) in Problem (2.9) is nonsmooth and nonconvex. Most of algorithms for solving Problem (2.9) are based on the Clarke subdifferential. Let $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ be a locally Lipschitz function. The generalized directional derivative $\varphi^0(x, u)$ of φ at x in the direction u is defined as [47]:

$$\varphi^0(x, u) := \limsup_{y \rightarrow x, \alpha \downarrow 0} \alpha^{-1} [\varphi(y + \alpha u) - \varphi(y)].$$

For locally Lipschitz functions the generalized directional derivative exists. The subdifferential $\partial\varphi(x)$ of the function φ at x is defined as follows [47]:

$$\partial\varphi(x) := \{v \in \mathbb{R}^n : \varphi^0(x, u) \geq \langle v, u \rangle \ \forall u \in \mathbb{R}^n\}.$$

According to Rademacher's theorem the locally Lipschitz function φ is differentiable almost everywhere and its subdifferential $\partial\varphi(x)$ at a point $x \in \mathbb{R}^n$ can also be defined as:

$$\partial\varphi(x) := \text{conv} \left\{ \lim_{i \rightarrow \infty} \nabla\varphi(x^i) : x^i \rightarrow x \text{ and } \nabla\varphi(x^i) \text{ exists} \right\}, \quad (4.1)$$

where “conv” denotes the convex hull of a set. Each vector $v \in \partial\varphi(x)$ is called a subgradient. The subdifferential $\partial f(x)$ is a compact and convex set at any $x \in \mathbb{R}^n$.

A function φ is called regular at $x \in \mathbb{R}^n$, if it is differentiable with respect to any direction $u \in \mathbb{R}^n$ at x and $\varphi'(x, u) = \varphi^0(x, u)$ for all $u \in \mathbb{R}^n$ where $\varphi'(x, u)$ is a derivative of the function φ at the point x in the direction u :

$$\varphi'(x, u) = \lim_{\alpha \downarrow 0} \alpha^{-1} [\varphi(x + \alpha u) - \varphi(x)].$$

Next we will describe some properties of the objective function (2.10). For each $a \in A$ consider the following function:

$$\psi_a(x) = \min_{j=1, \dots, k} d(x^j, a)$$

and let

$$R_a(x) = \{j \in \{1, \dots, k\} : d(x^j, a) = \psi_a(x)\}.$$

The function ψ_a is directionally differentiable and

$$\psi'_a(x, u) = \min_{j \in R_a(x)} d'[(x^j, a), u^j] \quad u = (u^1, \dots, u^k) \in \mathbb{R}^{n \times k}. \quad (4.2)$$

Here $d'[(x^j, a), u^j]$ is the directional derivative of the function d at the point $x^j \in \mathbb{R}^n$ in the direction u^j , $j \in R_a(x)$.

Proposition 1. The function f_k given by (2.10) is directionally differentiable at any $x = (x^1, \dots, x^k) \in \mathbb{R}^{n \times k}$ and

$$f'_k(x, u) = \frac{1}{m} \sum_{a \in A} \min_{j \in R_a(x)} d'[(x^j, a), u^j], \quad u = (u^1, \dots, u^k) \in \mathbb{R}^{n \times k}. \quad (4.3)$$

Proof: Since both distance functions d_1 and d_∞ are convex they are directionally differentiable at any x with respect to any direction $u \in \mathbb{R}^n$. Then the directional differentiability of the function f_k follows from its representation as a sum of minima functions. The expression (4.3) follows from the expression (4.2) for the directional derivative of the function ψ_a . \square

Corollary 4.3.1. *Assume that at a point $x = (x^1, \dots, x^j) \in \mathbb{R}^{n \times k}$ there exist $a \in A$ such that $|R_a(x)| \geq 2$ and $d'[(x^{j_1}, a), u] \neq d'[(x^{j_2}, a), u]$ for some $j_1, j_2 \in R_a(x)$ and $u \in \mathbb{R}^n$. Then the function f_k is not regular at x .*

Proof: The generalized directional derivative of the function f_k at the point $x \in \mathbb{R}^{n \times k}$ in the direction $u = (u^1, \dots, u^k) \in \mathbb{R}^{n \times k}$ is given by

$$f_k^0(x, u) = \frac{1}{m} \sum_{a \in A} \max_{j \in R_a(x)} d'[(x^j, a), u^j].$$

This obviously implies that if the conditions of the corollary are satisfied then $f'_k(x, u) < f_k^0(x, u)$ for some $u \in \mathbb{R}^{n \times k}$. This completes the proof. \square

It is well-known that the Clarke subdifferential calculus with most widely used operations (summation, maximum) exists in the form of equalities only for regular functions. Furthermore, for functions defined using complex compositions more restrictive conditions on component functions are required to have equalities. Calculus exists in the form of inclusions if these conditions are not satisfied which makes such calculus not applicable in numerical algorithms since it cannot be applied to calculate the subgradients. It follows from Corollary 4.3.1 that the Clarke subdifferential calculus for the function f_k in some points $x \in \mathbb{R}^{n \times k}$ can be expressed only as inclusions and therefore, this calculus cannot be applied to compute its subgradients. In this situation derivative free algorithms or algorithms based on the approximation of subgradients using values of a function are only choice.

The Discrete Gradient Method introduced in [16] is one such method. It uses discrete gradients to approximate subgradients and discrete gradients are computed using only values of a function; $n + 1$ function evaluations are required to compute one discrete gradient where n is the number of variables. In [20] the version of the discrete gradient method is introduced where the number of function evaluations can be reduced significantly exploiting a special structure of the objective function such as piecewise partial separability. Next we recall

definitions of partial and piecewise partial separabilities (see [20], for details) and show that the function f_k is piecewise separable for distance functions d_1, d_2^2 and d_∞ .

Definition 1. The function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ is called partially separable iff there exists a family of $n \times n$ diagonal matrices $U_i, i = 1, \dots, M, M \geq 1$ such that the function φ can be represented as follows:

$$\varphi(x) = \sum_{i=1}^M \varphi_i(U_i x).$$

In other terms, the function φ is called partially separable if it can be represented as the sum of functions of a much smaller number of variables. If $M = n$ and $\text{diag}(U_i) = e^i$ where e^i is the i -th unit vector, $i = 1, \dots, n$ then the function φ is separable.

Definition 2. The function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be piecewise partially separable iff there exists a finite family of closed sets D_1, \dots, D_m such that $\bigcup_{i=1}^m D_i = \mathbb{R}^n$ and the function φ is partially separable on each set $D_i, i = 1, \dots, m$.

Next we will show that the function f_k is piecewise partially separable.

Proposition 2. The function f_k defined by (2.10) is piecewise separable for distance functions d_1, d_2^2 and d_∞ .

Proof: It is clear that distance functions d_1, d_2^2 and d_∞ are separable. Since the function $\psi_a(x) = \min_{j=1, \dots, k} d(x^j, a), a \in A$ is represented as a minimum of functions each depending on a subset of variables, it is piecewise separable. Finally the function f_k as a sum of piecewise separable functions $\psi_a, a \in A$ is piecewise separable itself by Proposition 7 in [20]. \square

It is shown in [20] that by exploiting piecewise partial separability it is possible to reduce the number of function evaluations by the discrete gradient method. The number of variables in Problem (2.9) is nk . This means that to compute one discrete gradient of the function f_k one needs $nk + 1$ evaluations of this function. However, since the function f_k is piecewise separable applying the scheme from [20] we need only 2 evaluations of this function to compute one discrete gradient that is using it we can reduce the number of the objective function evaluations $(nk + 1)/2$ times.

4.3.2 An algorithm for solving Problem (3.2)

The objective function (3.1) in Problem (3.2) is nonsmooth and nonconvex. One can use the Clarke subdifferential of this function to design algorithms for solving Problem (3.2). However, it will be shown that, similar to the function (2.10), the computation of the subdifferential of the function (3.1) is not an easy task.

Introducing the sets

$$\bar{A}_1 = \{a \in A : r_{k-1}^a > d(y, a)\}, \quad \bar{A}_2 = \{a \in A : r_{k-1}^a = d(y, a)\},$$

$$\bar{A}_3 = \{a \in A : r_{k-1}^a < d(y, a)\}$$

one can rewrite the function \bar{f}_k as follows:

$$\bar{f}_k(y) = \frac{1}{m} \left(\sum_{a \in \bar{A}_1} d(y, a) + \sum_{a \in \bar{A}_2} \min\{r_{k-1}^a, d(y, a)\} + \sum_{a \in \bar{A}_3} r_{k-1}^a \right). \quad (4.4)$$

Proposition 3. The function \bar{f}_k given by (3.1) is directionally differentiable at any $y \in \mathbb{R}^n$ and

$$\bar{f}'_k(y, u) = \frac{1}{m} \left(\sum_{a \in \bar{A}_1} d'[(y, a), u] + \sum_{a \in \bar{A}_2} \min\{0, d'[(y, a), u]\} \right), \quad u \in \mathbb{R}^n. \quad (4.5)$$

Proof: Distance functions d_1 and d_∞ are convex and therefore they are directionally differentiable at any y in any direction $u \in \mathbb{R}^n$. Then the directional differentiability of the function \bar{f}_k follows from its representation as a sum of minima functions. The expression (4.5) follows from the expression (4.4) for the function \bar{f}_k at the point y . \square

Corollary 4.3.2. *If at a point $y \in \mathbb{R}^n$ there exist at least one $a \in A$ such that $d'[(y, a), u] \neq 0$ for some $u \in \mathbb{R}^n$. Then the function \bar{f}_k is not regular at y .*

Proof: The generalized directional derivative of the function \bar{f}_k at the point $y \in \mathbb{R}^n$ in the direction $u \in \mathbb{R}^n$ is:

$$\bar{f}_k^0(y, u) = \frac{1}{m} \left(\sum_{a \in \bar{A}_1} d'[(y, a), u] + \sum_{a \in \bar{A}_2} \max\{0, d'[(y, a), u]\} \right).$$

Then it follows from the conditions of the corollary and (4.5) that $\bar{f}'_k(y, u) < f_k^0(y, u)$ for some $u \in \mathbb{R}^n$. This completes the proof. \square

Since the function \bar{f}_k is not regular the Clarke subdifferential calculus for this function exists in the form of inclusions at some points and this calculus cannot be used to calculate subgradients of \bar{f}_k at such points. Therefore we apply the discrete gradient method to minimize this function. Next we prove that the function \bar{f}_k is piecewise partially separable.

Proposition 4. The function \bar{f}_k defined in formula (3.1) is piecewise separable for distance functions d_1, d_2^2 and d_∞ .

Proof: Since the distance functions d_1, d_2^2 and d_∞ are separable the first term in the presentation (4.4) is separable and the third term is constant. It is obvious that the function $\min\{r_{k-1}^a, d(y, a)\}$ is piecewise separable. Then the second term in (4.4) is piecewise separable and consequently the function \bar{f}_k is piecewise separable. \square

Since the function \bar{f}_k is piecewise separable one needs only 2 evaluations of this function to compute one discrete gradient. If we do not use this scheme then the number of function evaluations required to compute one discrete gradient will be $n + 1$. This means that using the piecewise separability of the function \bar{f}_k and the scheme from [20] we can reduce the number of objective function evaluations used by the discrete gradient method to solve the problem (3.2) $(n + 1)/2$ times.

4.3.3 Solving optimization problems to find center of one cluster

In this subsection we discuss algorithms for finding a center of one cluster using different distance functions. A center x for a cluster B is found as a solution to the following optimization problem:

$$\text{minimize } \varphi(x) = \frac{1}{|B|} \sum_{b \in B} d(x, b) \text{ subject to } x \in \mathbb{R}^n. \quad (4.6)$$

If the function d is defined using the squared Euclidean norm then the center x is the centroid of the cluster B which can be easily computed without solving the optimization problem (4.6). If the distance function is defined using the L_1 -norm then the center is the median of the cluster B , which first used the k -median algorithm in [130]. In this paper we use the

nonsmooth optimization formulation to prove that the median of the cluster is the solution to the problem (4.6).

Proposition 5. If the distance function d is defined by the L_1 -norm then the median of the set B is the solution to Problem (4.6).

Proof: In this case the function φ is as follows:

$$\varphi(x) = \frac{1}{|B|} \sum_{b \in B} \sum_{i=1}^n |x_i - b_i| = \frac{1}{|B|} \sum_{b \in B} \sum_{i=1}^n |x_i - b_i|$$

Introduce the function

$$\psi_i(x_i) = \sum_{b \in B} |x_i - b_i| = \sum_{b \in B} \max\{x_i - b_i, b_i - x_i\}.$$

Then the function φ is separable and can be represented as

$$\varphi(x) = \sum_{i=1}^n \psi_i(x_i).$$

This means that the minimization of φ is equivalent to the minimization of the functions ψ_i for each $i \in \{1, \dots, n\}$. We assume that all numbers $b_i, b \in B$ are different. To write the subdifferential of ψ_i consider the following sets:

$$B^- = \{b \in B : b_i < x_i\}, \quad B^+ = \{b \in B : b_i > x_i\}, \quad B^0 = \{b \in B : b_i = x_i\}.$$

Since all numbers $b_i, b \in B$ are different it is obvious that for a given $x \in \mathbb{R}^n$ the cardinality $|B^0|$ of the set B^0 is either 0 or 1. Then the subdifferential of the function ψ_i at $x_i, i = 1, \dots, n$ is:

$$\begin{aligned} \partial\psi_i(x_i) &= |B^-| - |B^+| + [-|B^0|, |B^0|] \\ &= [|B^-| - |B^+| - |B^0|, |B^-| - |B^+| + |B^0|]. \end{aligned}$$

The point x_i to be a global minimizer of the function ψ_i it is necessary and sufficient that $0 \in \partial\psi_i(x_i)$. This means that

$$|B^-| - |B^+| - |B^0| \leq 0, \quad |B^-| - |B^+| + |B^0| \geq 0.$$

Now we can consider two cases: (i) $|A^0| = 0$ and (ii) $|A^0| = 1$. In Case (i) we have $|B^-| - |B^+| = 0$ that is $|B^-| = |B^+|$ and the total number of points $b \in B$ with different i -th coordinate is $2|B^-|$ that is this number should be even. In this case x_i is a median. In Case (ii) we have $-1 \leq |B^-| - |B^+| \leq 1$. This means that there can be three different cases: (a) $|B^-| = |B^+| - 1$. This means that the number of points $b \in B$ with different i -th coordinate is even and x_i is a median coinciding with one of $b_i, b \in B$. (b) $|B^-| = |B^+|$. Then the number of points $b \in B$ with different i -th coordinate is odd and x_i is a median coinciding with the point which is exactly in the middle. (c) $|B^-| = |B^+| + 1$. In this case the number of points $b \in B$ with different i -th coordinate is even and again x_i is a median coinciding with one of $b_i, b \in B$. \square

It follows from Proposition 5 that in order to solve Problem (4.6) with the distance function d_1 one needs to find the median of the cluster B that is in this we do not need to solve the optimization problem.

Finally, consider Problem (4.6) with the distance function d_∞ . In this case the objective function φ can be rewritten as:

$$\varphi(x) = \frac{1}{|B|} \sum_{b \in B} \max_{i=1, \dots, n} |x_i - b_i|. \quad (4.7)$$

For each data point $b \in B$ consider the following function:

$$\theta_b(x) = \max_{i=1, \dots, n} |x_i - b_i| = \max_{i=1, \dots, n} \max\{x_i - b_i, b_i - x_i\}$$

and define the set:

$$R_b(x) = \{i \in \{1, \dots, n\} : |x_i - b_i| = \theta_b(x)\}.$$

Consider

$$I_-(b, x) = \{i \in \{1, \dots, n\} : x_i < b_i\}, \quad I_+(b, x) = \{i \in \{1, \dots, n\} : x_i > b_i\},$$

$$I_0(b, x) = \{i \in \{1, \dots, n\} : x_i = b_i\}.$$

The condition $I_0(b, x) \cap R_b(x) \neq \emptyset$ means that $R_b(x) = I_0(b, x) = \{1, \dots, n\}$ and $\theta_b(x) = 0$.

This implies that in this case

$$\partial\theta_b(x) = \text{conv} \left\{ (0, \dots, -e^i, 0, \dots, 0), (0, \dots, e^i, 0, \dots, 0), i \in \{1, \dots, n\} \right\}.$$

If $I_0(b, x) \cap R_b(x) = \emptyset$ then $R_b(x) \subset I_-(b, x) \cup I_+(b, x)$. In this case

$$\partial\theta_b(x) = \text{conv} \left\{ (0, \dots, 0, \text{sign}(x_i - b_i), 0, \dots, 0) : i \in R_b(x) \right\},$$

where

$$\text{sign}(z) = \begin{cases} 0, & \text{if } z = 0; \\ -1, & \text{if } z < 0; \\ 1, & \text{if } z > 0. \end{cases}$$

Then the subdifferential of the function φ at x is:

$$\partial\varphi(x) = \frac{1}{|B|} \sum_{b \in B} \partial\theta_b(x).$$

and the necessary and sufficient condition for a minimum is: $0 \in \partial\varphi(x)$. It is easy to see that even for moderately large number of points in the set B one may have a huge number of extreme points in the subdifferential $\partial\varphi(x)$ and therefore the computation of the whole subdifferential is not an easy task. For this reason we propose to apply smoothing techniques to minimize the function (4.7). More specifically, we apply the hyperbolic smoothing to approximate this function. Details of this approach will be discussed in the next subsection.

4.4 An algorithm based on smoothing techniques

In this subsection we will demonstrate how smoothing techniques can be applied to approximate the objective functions in Problems (2.9) and (3.2) by smooth functions. This will allow us to apply powerful smooth optimization algorithms such as the conjugate gradient and quasi-Newton methods to solve cluster analysis problems. We will use the hyperbolic smoothing for this purpose. Other smoothing techniques can be applied in a similar way.

The hyperbolic smoothing functions were originally introduced to approximate the fol-

lowing nonsmooth function (see, for example, [148]):

$$\varphi(x) = \max\{0, x\}, \quad x \in \mathbb{R}. \quad (4.8)$$

The hyperbolic function smoothing the function (4.8) is as follows:

$$\phi_\tau(x) = \frac{x + \sqrt{x^2 + \tau^2}}{2}. \quad (4.9)$$

Here $\tau > 0$ is called the *precision or smoothing parameter*.

Proposition 6. The function $\phi_\tau(x)$ has the following properties:

1. $\phi_\tau(\cdot)$ is an increasing convex C^∞ function;
2. $\varphi(x) < \phi_\tau(x) \leq \varphi(x) + \frac{\tau}{2}, \quad \forall \tau > 0.$

The hyperbolic smoothing functions for maximum functions

$$\psi(x) = \max_{i=1, \dots, p} \psi_i(x)$$

were studied in [18]. Here $p \geq 1$ and functions $\psi_i, \quad i = 1, \dots, p$ are continuously differentiable.

Using an additional variable $t \in \mathbb{R}$ we introduce the following function:

$$\Psi(x, t) = t + \sum_{i=1}^p \max(0, \psi_i(x) - t).$$

It is clear that $\psi(x) = \Psi(x, \psi(x))$. Then the hyperbolic smoothing function $H(x, \tau)$ for ψ can be written as

$$H(x, \tau) = t + \sum_{i=1}^p \frac{\psi_i(x) - t + \sqrt{(\psi_i(x) - t)^2 + \tau^2}}{2}, \quad t = \psi(x). \quad (4.10)$$

See [148] and [18] for details.

4.4.1 Hyperbolic smoothing of the cluster function

In this subsection we define hyperbolic smoothing of the clustering function f_k defined by (2.10). It is clear that

$$\min_{j=1,\dots,k} d(x^j, a^i) = - \max_{j=1,\dots,k} -d(x^j, a^i), \quad i = 1, \dots, m.$$

Then

$$f_k(x^1, \dots, x^k) = -\frac{1}{m} \sum_{i=1}^m \max_{j=1,\dots,k} -d(x^j, a^i).$$

Consider the function:

$$\Psi_k(x^1, \dots, x^k) = -\frac{1}{m} \sum_{i=1}^m \left(t_i + \sum_{j=1}^k \max(0, -d(x^j, a^i) - t_i) \right).$$

Here

$$t_i = \max_{j=1,\dots,k} -d(x^j, a^i) = -\min_{j=1,\dots,k} d(x^j, a^i) \leq 0, \quad i = 1, \dots, m.$$

It follows from (4.10) that the hyperbolic smoothing $U_k(x^1, \dots, x^k, \tau)$ of the function f_k is as follows:

$$\begin{aligned} U_k(x^1, \dots, x^k, \tau) &= -\frac{1}{m} \sum_{i=1}^m \left(t_i + \sum_{j=1}^k \frac{-d(x^j, a^i) - t_i + \sqrt{(d(x^j, a^i) + t_i)^2 + \tau^2}}{2} \right) \\ &= \frac{1}{m} \sum_{i=1}^m \left(-t_i + \sum_{j=1}^k \frac{t_i + d(x^j, a^i) - \sqrt{(d(x^j, a^i) + t_i)^2 + \tau^2}}{2} \right). \end{aligned} \quad (4.11)$$

4.4.2 Hyperbolic smoothing of the auxiliary cluster function

In this subsection we define the hyperbolic smoothing of the the auxiliary cluster function \bar{f}_k defined by (3.1) (here we take $l = k$). This function can be rewritten as:

$$\bar{f}_k(y) = \frac{1}{m} \sum_{i=1}^m r_{k-1}^i + \frac{1}{m} \sum_{i=1}^m \min(0, d(y, a^i) - r_{k-1}^i).$$

Then we have

$$\bar{f}_k(y) = \frac{1}{m} \sum_{i=1}^m r_{k-1}^i - \frac{1}{m} \sum_{i=1}^m \max(0, r_{k-1}^i - d(y, a^i)).$$

Applying (4.9) we can approximate the auxiliary cluster function \bar{f}_k by the following function:

$$\begin{aligned} \bar{U}_k(y, \tau) &= \frac{1}{m} \sum_{i=1}^m r_{k-1}^i - \frac{1}{m} \sum_{i=1}^m \frac{r_{k-1}^i - d(y, a^i) + \sqrt{(r_{k-1}^i - d(y, a^i))^2 + \tau^2}}{2} \\ &= \frac{1}{m} \sum_{i=1}^m \frac{r_{k-1}^i + d(y, a^i) - \sqrt{(r_{k-1}^i - d(y, a^i))^2 + \tau^2}}{2}, \quad y \in \mathbb{R}^n. \end{aligned} \quad (4.12)$$

Since the squared Euclidean distance is differentiable there is no need to smooth it. However, two other distance functions using L_1 and L_∞ norms are nondifferentiable. Therefore we use the hyperbolic smoothing technique to approximate them with the smooth functions.

4.4.3 Hyperbolic smoothing of L_1 -norm

Here we describe the hyperbolic smoothing of the L_1 -norm. It is obvious that for $x, a \in \mathbb{R}^n$

$$d(x, a) = \|x - a\|_1 = \sum_{q=1}^n \max\{x_q - a_q, a_q - x_q\} = \sum_{q=1}^n [(x_q - a_q) + 2 \max\{0, a_q - x_q\}].$$

Then applying (4.9) we can approximate $\|x - a\|_1$ by the following smooth function:

$$\eta_1(x, a, \tau) = \sum_{q=1}^n [(x_q - a_q)^2 + \tau^2]^{1/2}. \quad (4.13)$$

Figure 4.1 illustrates the shape of the L_1 -norm (blue) and its hyperbolic smooth approximation (red).

Thus, replacing the distance function d in (4.11) and (4.12) by its approximation η defined by (4.13) we get the following smooth approximations for the cluster and the auxiliary cluster functions, respectively:

$$H_k^1(x^1, \dots, x^k, \tau) = \frac{1}{m} \sum_{i=1}^m \left(-t_i + \sum_{j=1}^k \frac{t_i + \eta_1(x^j, a^i, \tau) - \sqrt{(\eta_1(x^j, a^i, \tau) + t_i)^2 + \tau^2}}{2} \right), \quad (4.14)$$

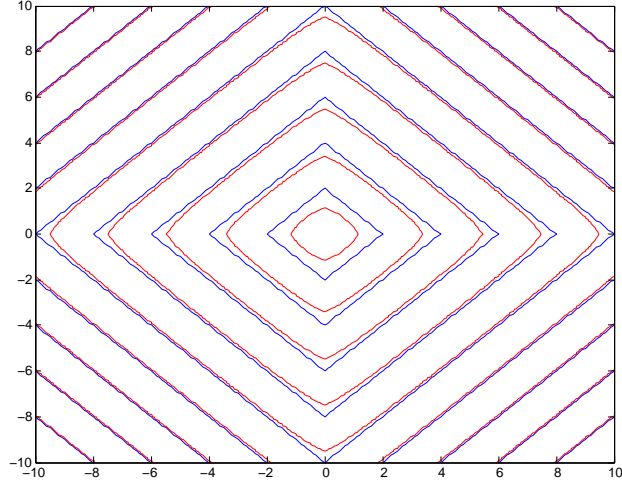


Figure 4.1: Smooth approximation of L_1 -norm

$$\bar{H}_k^1(y, \tau) = \frac{1}{m} \sum_{i=1}^m \frac{r_{k-1}^i + \eta_1(y, a^i, \tau) - \sqrt{(r_{k-1}^i - \eta_1(y, a^i, \tau))^2 + \tau^2}}{2}, \quad y \in \mathbb{R}^n. \quad (4.15)$$

4.4.4 Hyperbolic smoothing of L_∞ -norm

Now consider the smoothing of the distance function using L_∞ -norm. In this case

$$d(x, a) = \|x - a\|_\infty = \max_{q=1, \dots, n} |x_q - a_q|. \quad (4.16)$$

Then

$$\|x - a\|_\infty = \max_{q=1, \dots, n} \max\{\theta_q(x), -\theta_q(x)\}. \quad (4.17)$$

where $\theta_q(x) = x_q - a_q$.

The hyperbolic smoothing $\Omega_q(x, \tau)$ of the function $\beta_q(x) = \max\{\psi_q(x), -\psi_q(x)\}$ is

$$\begin{aligned} \Omega_q(x, \tau) = \beta_q(x) + & \frac{\theta_q(x) - \beta_q(x) + \sqrt{(\theta_q(x) - \beta_q(x))^2 - \tau^2}}{2} + \\ & \frac{-\theta_q(x) - \beta_q(x) + \sqrt{(-\theta_q(x) - \beta_q(x))^2 - \tau^2}}{2}. \end{aligned} \quad (4.18)$$

Then the hyperbolic smoothing of the distance function $d(x, a)$ can be expressed as

$$\eta_2(x, a, \tau) = d(x, a) + \sum_{q=1}^n \frac{\Omega_q(x, \tau) - d(x, a) + \sqrt{(\Omega_q(x, \tau) - d(x, a))^2 - \tau^2}}{2}. \quad (4.19)$$

Figure 4.2 illustrates the shape of the L_∞ -norm (blue) and its hyperbolic smooth approximation (red).

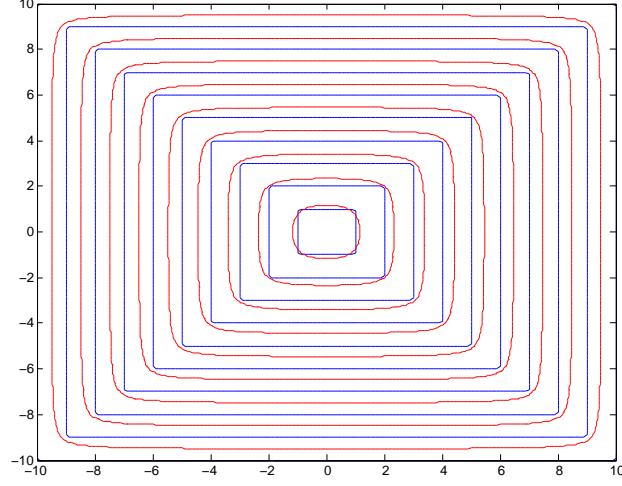


Figure 4.2: Smooth approximation of L_∞ -norm

Replacing the distance function d in (4.11) and (4.12) by its approximation η_2 defined by (4.19) we get the following smooth approximations for the cluster and the auxiliary cluster functions, respectively:

$$H_k^2(x^1, \dots, x^k, \tau) = \frac{1}{m} \sum_{i=1}^m \left(-t_i + \sum_{j=1}^k \frac{t_i + \eta_2(x^j, a^i, \tau) - \sqrt{(\eta_2(x^j, a^i, \tau) + t_i)^2 + \tau^2}}{2} \right), \quad (4.20)$$

$$\bar{H}_k^2(y, \tau) = \frac{1}{m} \sum_{i=1}^m \frac{r_{k-1}^i + \eta_2(y, a^i, \tau) - \sqrt{(r_{k-1}^i - \eta_2(y, a^i, \tau))^2 + \tau^2}}{2}, \quad y \in \mathbb{R}^n. \quad (4.21)$$

4.4.5 Smooth clustering problems

Now we can replace nonsmooth optimization problems (2.9) and (3.2) by their smooth approximations as follows. Smooth auxiliary clustering and clustering problems with the similarity measure defined by the squared Euclidean distance are, respectively:

$$\text{minimize } \bar{U}_k(y, \tau) \quad \text{subject to } y \in \mathbb{R}^n. \quad (4.22)$$

$$\text{minimize } U_k(x^1, \dots, x^k, \tau) \quad \text{subject to } x^1, \dots, x^k \in \mathbb{R}^n. \quad (4.23)$$

Smooth auxiliary clustering and clustering problems with the similarity measure defined

by the L_1 norm are, respectively:

$$\text{minimize } \bar{H}_k^1(y, \tau) \text{ subject to } y \in \mathbb{R}^n. \quad (4.24)$$

$$\text{minimize } H_k^1(x^1, \dots, x^k, \tau) \text{ subject to } x^1, \dots, x^k \in \mathbb{R}^n. \quad (4.25)$$

Finally, smooth auxiliary clustering and clustering problems with the similarity measure defined by the L_1 norm are, respectively:

$$\text{minimize } \bar{H}_k^2(y, \tau) \text{ subject to } y \in \mathbb{R}^n. \quad (4.26)$$

$$\text{minimize } H_k^2(x^1, \dots, x^k, \tau) \text{ subject to } x^1, \dots, x^k \in \mathbb{R}^n. \quad (4.27)$$

In order to solve these problems we take any sequence $\{\tau_l\}$ such that $\tau_l > 0$ and $\tau_l \rightarrow 0$ as $l \rightarrow \infty$. Then applying any smooth optimization algorithms we can get sequences of solutions which converge to the solutions of problems (2.9) and (3.2).

4.5 Summary

In this chapter, we developed algorithms for solving clustering problems. The similarity measure is defined using the L_1, L_2 and L_∞ norms. The first algorithm is based on an incremental approach and applies heuristics like the k -means algorithm for finding cluster centers. In the second algorithm, we proposed to apply the nonsmooth optimization algorithms to solve the clustering problem. In the third algorithm, to deal with nonsmoothness, the clustering functions based on the L_1 and L_∞ norms are approximated by smoothing techniques.

Chapter 5

Implementation and numerical results

5.1 Introduction

In this Chapter we discuss the implementation of Algorithm 4. The most important steps in Algorithm 4 are Steps 3 and 4. Other steps of this algorithm are easy to implement. In Step 3 we apply Algorithm 3 to find the set of starting points for the l -th cluster center. Algorithm 3 contains three parameters, namely, γ_1 , γ_2 and γ_3 . The choice of these parameters depends on the size of a data set.

One can see from (3.6) that if γ_1 is close to 0 then the set \bar{A}_1 will contain most of data points which may lead to a large number of starting points and make the algorithm very time-consuming. Therefore in order to avoid having a large number of starting points the choice of γ_1 and γ_2 should depend on the number of points in a data set. Since the difference between the values of the auxiliary clustering problem (3.2) and the clustering problem (2.9) is not expected to be large one can choose the parameter γ_3 from $[1, 2]$. These parameters are chosen as follows:

1. $\gamma_1 = 0.3, \gamma_2 = 0.5$ and $\gamma_3 = 2$ for data sets with the number of data points $m \leq 200$;
2. $\gamma_1 = 0.6, \gamma_2 = 0.8$ and $\gamma_3 = 1.25$ for data sets with the number of data points $200 < m \leq 2500$;

3. $\gamma_1 = 0.8, \gamma_2 = 0.99$ and $\gamma_3 = 1.05$ for data sets with the number of data points $m > 2500$.

In order to determine the neighboring points from the set \bar{A}_3 we use the following tolerance:

$$\varepsilon = \frac{10^{-4} f_1^{min}}{l}$$

where f_1^{min} is the optimal value of the cluster function f_k when $k = 1$ and l is the number of clusters. If the distance between two points in \bar{A}_3 is less than ε then we remove one of them (any) and keep another one.

Since the proposed algorithm is an incremental one we do not consider data points which are very close to previous cluster centers as candidates to be starting points. In order to do so we apply a scheme already discussed in [17].

We apply the Quasi-Newton method with BFGS update to solve smooth clustering problems (4.22), (4.23), (4.24), (4.25), (4.26) and (4.27).

In general, Algorithm 4 can find only stationary points of the clustering problem (2.9). However, the use of the special procedure to generate good starting points allows us to find either global or near global solutions to this problem which will be confirmed by the computational results presented in the next section.

5.2 Computational results: evaluation of the incremental algorithm

We tested the proposed algorithm using a number of real-world data sets. Numerical experiments were carried out on a PC with Processor Intel(R) Core(TM) i5-3470S CPU 2.90 GHz and RAM 8GB. Algorithm 4 was implemented in Lahey Fortran 95. 12 data sets were used in numerical experiments. The brief description of these data sets is given in Appendix. The more detailed description can be found in [14, 119].

We consider three different versions of Algorithm 4. These versions differ from each other on optimization algorithms used to solve both the auxiliary clustering and clustering problems.

1. Incremental Nonsmooth k -means algorithm: INKA - in this algorithm both the auxil-

iary clustering and clustering problems are solved using the k -means algorithm.

2. Incremental Nonsmooth Optimization Clustering algorithm: INCA - in this algorithm both the auxiliary clustering and clustering problems are solved using the discrete gradient method of nonsmooth optimization.
3. Incremental Smooth Optimization Clustering algorithm: ISCA - in this algorithm both the auxiliary clustering and clustering problems are solved using their smooth approximations.

We compute 10 clusters in small data sets (German towns, Bavaria postal 1, Bavaria postal 2 and Iris Plant), 20 clusters in medium size data sets (Heart Disease, Breast Cancer, TSPLIB1060 and Image segmentation) and 25 clusters in large data sets (TSPLIB3038, Page Blocks, D15112 and Pla85900). Results with different similarity measures are presented separately. In all tables we use the following notation:

- k is the number of clusters;
- f_{best} is the best known value of the cluster function (2.10) (multiplied by m) with the corresponding number of clusters;
- E is the error in %;
- N is the number of the distance function evaluations for the computation of the corresponding number of clusters;
- t is the CPU time.

The error E is computed as

$$E = \frac{(\bar{f} - f_{best}^k)}{f_{best}^k} \cdot 100$$

where \bar{f} is the optimal value of the function f_k given by (2.10) obtained by an algorithm and f_{best}^k is the best known value of f_k for given k .

5.2.1 Results for the similarity measure based on the squared L_2 -norm

Results for small data sets are given in Table 5.1. These results demonstrate that all three algorithms are efficient for finding global or near global solutions to the clustering problem in small data sets. Furthermore, optimization based clustering algorithms: INCA and ISCA are more accurate than the INKA algorithm. On the other hand, the INKA requires significantly less computational effort (both distance function evaluations and CPU time) than other two algorithms. Although the INCA algorithm is slightly more accurate than the ISCA algorithm however the former requires more computational effort than the latter algorithm.

Table 5.2 contains results for medium size data sets. In these results all three algorithms are efficient for finding global or near global solutions to the clustering problem in medium size data sets. The INCA algorithm is most accurate and overall the INKA is more accurate than the ISCA algorithm. The INKA algorithm requires less distance function evaluations and CPU time than other two algorithms. The INCA algorithm uses more distance function evaluations and CPU time than the ISCA algorithm in Heart Disease and Breast Cancer data sets, however the former algorithm requires less computational effort than the latter algorithm in other two data sets.

Table 5.3 presents results for large data sets. One can see that the accuracy of all three algorithms is similar for large data sets used in this chapter. The INKA algorithm requires least number of the distance function evaluations and CPU time among three algorithms. The INCA requires more computational effort than the ISCA algorithm in all data sets except Pla85900 data set where results are mixed.

5.2.2 Results for the similarity measure based on the L_1 -norm

In this subsection we present clustering results with the similarity measure defined by the L_1 -norm. Here we use only the INCA and ISCA algorithms since the INKA algorithm is not applicable with the L_1 -norm.

Results for small data sets are given in Table 5.4. These results clearly demonstrate that the INCA algorithm is more accurate than the ISCA algorithm in all data sets, however in Iris Plant data set results are similar. On the other side the INCA algorithm requires significantly more distance function evaluations and CPU time than the ISCA algorithm.

Table 5.1: Results with the similarity measure based on L_2 -norm

k	f_{best}	INKA			INCA			ISCA		
		E	N	t	E	N	t	E	N	t
German town										
	$\times 10^3$		$\times 10^4$			$\times 10^6$			$\times 10^5$	
2	121.4257	0.00	1.84	0.00	0.00	0.43	0.00	0.00	0.26	0.00
3	77.0086	0.00	4.21	0.02	0.00	0.94	0.00	0.00	0.53	0.02
4	49.6006	0.24	5.72	0.02	0.00	1.18	0.00	0.00	0.97	0.02
5	38.7160	0.00	8.75	0.02	0.00	2.34	0.01	0.00	1.35	0.03
6	30.5354	0.00	11.02	0.02	0.00	3.38	0.01	0.00	1.79	0.03
7	24.4326	0.08	12.51	0.02	0.00	4.13	0.03	0.00	2.23	0.03
8	21.4830	0.00	15.76	0.02	0.00	7.39	0.04	0.00	3.00	0.05
9	18.5504	2.13	18.99	0.02	0.00	10.54	0.06	0.00	3.66	0.06
10	16.3080	1.81	23.37	0.02	0.00	13.44	0.09	0.00	4.36	0.08
Bavaria postal 1										
	$\times 10^6$		$\times 10^4$			$\times 10^6$			$\times 10^6$	
2	602547.2132	0.00	1.20	0.00	0.00	0.28	0.00	0.00	0.09	0.00
3	294506.5545	0.00	3.34	0.00	0.00	0.76	0.01	0.00	0.21	0.00
4	104474.6551	0.00	5.57	0.00	0.00	1.27	0.01	0.00	0.51	0.00
5	59761.5150	0.00	8.36	0.00	0.00	2.27	0.01	0.00	0.88	0.02
6	35908.5267	0.00	11.53	0.00	0.00	3.96	0.03	0.00	1.21	0.03
7	21983.1969	0.61	16.45	0.00	0.00	5.24	0.04	0.00	1.62	0.03
8	13385.4046	0.00	20.67	0.02	0.00	6.95	0.06	0.00	2.12	0.03
9	8423.7401	0.00	22.62	0.02	0.00	9.40	0.07	0.00	2.53	0.03
10	6446.4738	0.00	28.21	0.02	0.00	13.26	0.10	0.00	3.36	0.06
Bavaria postal 2										
	$\times 10^6$		$\times 10^4$			$\times 10^6$			$\times 10^6$	
2	48631.2853	0.00	1.38	0.00	0.00	0.45	0.00	7.32	0.04	0.00
3	17398.7515	0.00	3.19	0.00	0.00	1.70	0.03	0.00	0.11	0.02
4	7559.0849	0.00	6.05	0.00	0.00	2.86	0.05	0.00	0.35	0.02
5	5342.8659	0.00	9.65	0.00	0.00	5.16	0.06	0.00	0.61	0.02
6	3187.5747	0.00	13.41	0.00	0.00	6.68	0.08	0.00	0.80	0.02
7	2215.9024	0.00	17.99	0.00	0.00	11.36	0.12	0.00	1.61	0.03
8	1704.5235	0.18	23.28	0.02	0.00	19.30	0.19	0.00	2.35	0.06
9	1401.0666	1.07	27.70	0.02	0.00	32.97	0.37	0.00	3.32	0.09
10	1181.0406	0.00	31.91	0.02	0.00	54.10	0.56	0.00	3.96	0.12
Iris Plant										
	$\times 10^0$		$\times 10^4$			$\times 10^6$			$\times 10^6$	
2	152.3480	0.00	0.45	0.02	0.00	0.45	0.01	0.00	0.22	0.00
3	78.8510	0.00	1.29	0.02	0.00	1.36	0.01	0.00	0.55	0.00
4	57.2280	0.05	2.69	0.02	0.00	3.83	0.04	0.00	2.09	0.03
5	46.4460	0.06	3.89	0.03	0.00	6.93	0.07	0.00	3.96	0.06
6	39.0400	0.07	5.36	0.03	0.00	12.77	0.15	0.00	5.30	0.08
7	34.2980	0.01	7.74	0.05	0.00	19.24	0.21	0.00	6.88	0.11
8	29.9890	0.25	8.92	0.06	0.00	22.88	0.26	0.00	8.76	0.16
9	27.7860	0.28	12.25	0.08	0.00	52.04	0.62	0.90	15.64	0.31
10	25.8340	0.07	16.30	0.09	0.00	84.94	1.12	0.91	21.86	0.47

Table 5.2: Results with similarity measure based on L_2 -norm (cont.)

k	f_{best}	INKA			INCA			ISCA		
		E	N	t	E	N	t	E	N	t
Heart disease										
	$\times 10^4$		$\times 10^6$			$\times 10^8$			$\times 10^7$	
2	59.8899	0.00	0.41	0.05	0.00	0.04	0.08	0.00	0.08	0.03
5	32.7543	0.00	3.01	0.27	0.00	0.14	0.34	0.13	0.70	0.23
10	20.0521	0.00	15.26	0.94	0.00	1.86	3.62	0.01	5.87	8.46
15	14.7085	0.00	26.51	1.47	0.00	4.13	7.47	0.03	9.47	24.23
20	11.6834	1.02	37.16	1.94	0.39	7.50	14.63	0.00	12.22	45.80
Breast cancer										
	$\times 10^4$		$\times 10^7$			$\times 10^8$			$\times 10^7$	
2	1.9323	0.00	0.10	0.09	0.00	0.03	0.04	0.00	0.23	0.05
5	1.3705	0.00	0.86	0.58	0.00	0.86	1.27	0.00	1.73	0.61
10	1.0194	0.00	2.00	1.06	0.00	2.55	3.51	0.00	8.01	2.14
15	0.8710	0.00	3.71	1.72	0.00	7.14	9.17	0.00	19.78	9.08
20	0.7677	0.00	5.82	2.48	0.00	15.07	18.29	0.00	37.10	27.36
TSPLIB1060										
	$\times 10^6$		$\times 10^7$			$\times 10^8$			$\times 10^8$	
2	9831.9499	0.00	0.14	0.06	0.00	0.05	0.04	0.00	0.04	0.05
5	3791.0203	0.01	1.00	0.34	0.00	0.25	0.18	0.00	0.16	0.17
10	1754.8752	0.22	4.04	1.14	0.00	1.17	0.76	0.00	0.63	0.70
15	1121.9175	0.00	8.22	2.08	0.00	3.04	1.84	0.02	1.08	1.25
20	792.5267	0.14	13.79	3.19	0.00	7.48	4.46	0.03	2.27	2.70
Image segmentation										
	$\times 10^5$		$\times 10^7$			$\times 10^8$			$\times 10^8$	
2	356.0580	0.00	0.64	0.52	0.00	0.16	0.44	0.00	0.08	0.27
5	171.4291	0.00	3.24	2.30	0.00	1.65	4.37	0.00	0.55	1.84
10	97.9546	0.64	9.22	6.00	0.00	12.41	28.42	1.15	1.93	10.26
15	65.5542	0.47	15.76	9.48	0.00	26.61	58.53	1.88	3.83	32.34
20	51.3621	0.89	25.55	14.41	0.12	45.71	98.84	0.00	6.55	89.31

Table 5.3: Results with similarity measure based on L_2 -norm (cont.)

k	f_{best}	INKA			INCA			ISCA		
		E	N	t	E	N	t	E	N	t
TSPLIB3038										
	$\times 10^6$		$\times 10^8$			$\times 10^8$			$\times 10^8$	
2	3168.8047	0.00	0.11	0.56	0.00	0.20	0.20	0.00	0.26	0.25
5	1198.1959	0.00	0.57	2.50	0.00	1.00	0.87	0.00	0.95	0.98
10	560.2569	0.57	1.53	4.92	0.00	2.81	2.23	0.00	2.47	2.64
15	356.0483	0.00	3.09	8.55	0.00	5.42	4.07	0.00	4.71	5.16
20	267.1626	0.20	4.69	11.89	0.00	11.70	7.87	0.11	7.99	8.86
25	214.4985	0.23	8.86	20.95	0.00	26.11	16.33	0.01	12.49	14.01
Page blocks										
	$\times 10^6$		$\times 10^9$			$\times 10^9$			$\times 10^9$	
2	57936.8499	0.00	0.01	1.88	0.00	0.05	0.93	0.00	0.04	0.89
5	13218.3776	0.00	0.07	8.09	0.00	0.22	4.05	0.19	0.17	3.76
10	4532.9960	0.00	0.12	12.08	0.00	0.76	12.27	1.04	0.46	10.44
15	2493.6468	0.00	0.19	16.45	0.06	1.72	25.20	0.31	0.89	21.01
20	1720.3970	0.00	0.25	20.75	0.00	2.98	41.58	0.00	1.45	37.52
25	1210.2370	0.00	0.32	25.28	2.23	4.28	58.29	0.00	2.17	63.31
D15112										
	$\times 10^8$		$\times 10^9$			$\times 10^9$			$\times 10^9$	
2	3684.0305	0.00	0.23	3.23	0.00	0.47	5.52	0.00	0.42	5.15
5	1327.0655	0.00	0.92	10.12	0.00	1.89	20.47	0.00	1.52	18.41
10	644.9025	1.41	2.09	17.38	0.00	4.73	47.64	0.00	3.44	41.50
15	431.3720	0.26	3.28	24.34	0.00	10.99	99.05	0.00	5.74	68.23
20	321.7737	0.01	4.52	31.11	0.00	19.72	165.97	0.00	8.65	101.82
25	253.0012	0.51	5.76	37.53	0.00	41.29	317.07	0.00	11.96	139.53
Pla85900										
	$\times 10^{11}$					$\times 10^{11}$			$\times 10^{11}$	
2	374908.4065	1.44	0.07	143.70	0.00	0.13	132.21	0.00	0.12	123.96
5	133971.8844	2.78	0.30	609.91	0.00	0.39	471.00	0.00	0.37	452.96
10	68294.1490	0.00	0.67	1358.81	0.00	0.83	1028.34	0.00	0.79	1011.17
15	46029.3761	0.17	1.04	1938.13	0.00	1.29	1608.17	0.00	1.24	1596.67
20	34985.9729	0.03	1.42	2382.82	0.00	1.78	2198.69	0.00	1.71	2210.91
25	28275.5162	0.77	1.80	2788.37	0.00	2.30	2808.12	0.00	2.21	2863.18

Table 5.4: Results with the similarity measure based on the L_1 -norm

k	f_{best}	INCA			ISCA		
		E	N	t	E	N	t
German town							
	$\times 10^3$		$\times 10^6$			$\times 10^6$	
2	3.0740	0.00	0.82	0.00	0.00	0.31	0.01
3	2.4450	0.00	1.46	0.00	0.00	0.24	0.01
4	1.8870	0.00	2.93	0.01	0.00	0.19	0.01
5	1.6460	0.00	5.40	0.03	0.30	0.17	0.03
6	1.5060	0.00	8.34	0.04	0.60	0.15	0.03
7	1.3780	0.00	12.22	0.07	0.00	0.14	0.08
8	1.2500	0.00	15.15	0.09	1.44	0.13	0.08
9	1.1390	0.00	17.83	0.10	0.09	0.11	0.09
10	1.0440	0.00	21.60	0.14	0.38	0.10	0.11
Bavaria postal 1							
	$\times 10^6$		$\times 10^7$			$\times 10^7$	
2	4.0249	0.00	0.42	0.03	3.21	0.32	0.09
3	2.8284	0.00	0.59	0.04	2.51	0.39	0.11
4	2.1982	0.00	1.41	0.10	0.22	0.52	0.14
5	1.7208	0.00	2.67	0.20	3.54	0.67	0.19
6	1.3003	0.00	3.29	0.24	8.72	1.13	0.31
7	1.0704	0.00	3.55	0.26	3.25	1.48	0.42
8	0.8510	0.00	3.58	0.28	2.21	1.53	0.44
9	0.7220	0.00	5.11	0.39	1.02	2.38	0.75
10	0.6037	0.00	6.02	0.45	0.33	3.40	1.09
Bavaria postal 2							
	$\times 10^6$		$\times 10^6$			$\times 10^6$	
2	1.8600	0.00	2.36	0.03	2.38	0.52	0.03
3	1.2607	0.00	3.06	0.03	1.82	0.63	0.03
4	0.9633	0.00	5.96	0.06	1.29	0.86	0.03
5	0.7872	0.00	16.44	0.14	6.91	1.35	0.05
6	0.6736	0.00	22.76	0.20	3.64	1.79	0.08
7	0.5659	0.00	27.38	0.24	3.34	1.92	0.08
8	0.5097	0.00	36.97	0.32	1.90	2.68	0.11
9	0.4688	0.00	46.44	0.40	3.67	3.20	0.12
10	0.4340	0.00	55.42	0.48	4.63	4.75	0.23
Iris Plant							
	$\times 10^2$		$\times 10^7$			$\times 10^7$	
2	2.1670	0.00	0.08	0.01	0.00	0.41	0.08
3	1.5920	0.00	0.93	0.09	0.00	1.64	0.28
4	1.3650	0.00	1.91	0.20	0.07	1.93	0.33
5	1.2460	0.00	3.65	0.40	0.40	2.69	0.44
6	1.1530	0.00	5.10	0.54	0.10	3.77	0.61
7	1.0620	0.00	6.38	0.70	0.01	4.06	0.65
8	1.0010	0.00	8.91	1.01	0.05	4.41	0.72
9	0.9540	0.00	13.62	1.65	0.00	5.91	0.98
10	0.9070	0.00	20.36	2.66	0.00	6.78	1.14

Table 5.5: Results with the similarity measure based on the L_1 -norm (cont.)

k	f_{best}	INCA			ISCA		
		E	N	t	E	N	t
Heart disease							
	$\times 10^4$		$\times 10^8$			$\times 10^8$	
2	2.0435	0.00	1.17	4.68	0.02	0.05	0.25
5	1.5760	0.00	5.97	27.78	0.06	1.22	6.21
10	1.3006	0.00	11.19	45.50	2.11	2.23	14.09
15	1.1158	0.00	17.17	61.65	0.00	9.97	114.00
20	1.0190	0.00	19.44	66.87	0.10	12.92	176.48
Breast cancer							
	$\times 10^4$		$\times 10^8$			$\times 10^8$	
2	0.6401	0.00	0.57	1.04	0.02	0.13	0.50
5	0.5032	0.00	4.45	8.84	0.56	1.27	3.90
10	0.4244	0.00	10.57	21.16	1.96	1.60	5.04
15	0.3858	0.00	25.44	46.28	0.08	5.64	21.25
20	0.3583	0.00	35.96	62.29	0.11	6.28	23.52
TSPLIB1060							
	$\times 10^6$		$\times 10^8$			$\times 10^8$	
2	3.8645	0.00	0.13	0.09	0.74	0.23	0.31
5	2.3096	0.00	1.83	1.02	0.62	0.90	1.08
10	1.5628	0.00	10.46	5.75	0.40	2.75	3.48
15	1.1983	0.00	28.89	16.33	0.73	5.21	6.75
20	1.0157	0.00	70.94	40.76	0.42	16.97	19.48
Image segmentation							
	$\times 10^5$		$\times 10^9$			$\times 10^9$	
2	5.1916	0.00	0.04	1.35	0.01	0.14	12.28
5	3.3996	0.00	0.55	16.62	0.00	0.52	39.64
10	2.5663	0.00	8.39	221.88	0.00	3.72	289.71
15	2.1795	0.00	17.68	440.93	1.40	8.84	801.91
20	1.9411	0.00	40.85	968.79	0.05	21.48	2566.79

Results for medium size data sets presented in Table 5.5 demonstrate that the INCA algorithm is more accurate than the ISCA algorithm in all data sets. The ISCA algorithm requires less distance function evaluations than the INKA algorithm however the former uses more CPU time than the latter algorithm.

Table 5.6 contains results with large data sets. These results show that the INCA algorithm is more accurate than the ISCA algorithm in TSPLIB3038 and Pla85900 data sets. In Page Blocks and D15112 data sets both algorithms are equally successful. Again one can see that the ISCA algorithm requires less distance function evaluations than the INCA algorithm. However results for CPU time are mixed. Overall, results presented in this subsection demonstrate that the nonsmooth optimization based INCA algorithm is more accurate, however it requires more computational effort than the smoothing technique based ISCA algorithm.

Table 5.6: Results with the similarity measure based on the L_1 -norm (cont.)

k	f_{best}	INCA			ISCA		
		E	N	t	E	N	t
TSPLIB3038							
	$\times 10^6$		$\times 10^9$			$\times 10^9$	
2	3.7308	0.00	0.04	0.28	0.02	0.02	0.23
3	3.0056	0.00	0.10	0.68	0.00	0.03	0.44
5	2.2551	0.00	0.24	1.59	0.06	0.08	0.97
10	1.5502	0.00	1.70	9.95	0.14	0.99	10.33
15	1.2298	0.00	4.71	27.11	0.27	2.14	22.03
20	1.0597	0.00	9.97	56.89	0.64	4.19	42.37
25	0.9441	0.00	14.38	81.97	0.18	7.60	75.88
Page blocks							
	$\times 10^6$		$\times 10^9$			$\times 10^9$	
2	8.4141	0.00	0.08	1.73	0.00	0.10	3.37
5	4.8821	0.00	1.18	21.43	0.00	0.45	14.88
10	3.1704	0.00	9.17	149.35	0.00	1.83	63.09
15	2.5682	0.00	12.60	197.94	0.00	6.77	241.91
20	2.2127	0.00	27.10	391.28	0.00	10.16	366.84
25	1.9737	0.00	49.83	689.86	0.00	16.43	623.74
D15112							
	$\times 10^8$		$\times 10^9$			$\times 10^9$	
2	0.886	0.00	0.40	4.66	0.00	0.27	3.87
5	0.4998	0.00	1.40	16.08	0.00	1.15	16.08
10	0.3617	0.00	3.50	37.17	0.00	2.78	38.13
15	0.293	0.00	6.84	66.64	0.03	4.74	64.38
20	0.2501	0.00	11.86	106.82	0.00	8.62	112.29
25	0.2243	0.00	16.49	143.66	0.00	11.37	147.12
Pla85900							
	$\times 10^{10}$		$\times 10^{10}$			$\times 10^{10}$	
2	2.0656	0.00	1.33	123.67	0.21	0.82	105.75
5	1.2569	0.00	4.38	455.53	1.84	3.12	412.28
10	0.898	0.00	9.59	1006.40	0.95	7.01	930.83
15	0.7333	0.00	14.98	1576.03	1.38	11.00	1451.20
20	0.6374	0.00	20.89	2171.28	0.91	15.10	1991.88
25	0.5693	0.00	26.67	2748.98	1.69	19.50	2557.04

Table 5.7: Results with the similarity measure based on the L_∞ -norm

L_∞ -norm	$N(\times 10^6)$					
k	INCA					
	f_{best}	N	t	f_{best}	N	t
	German town			Bavaria postal 1		
	$\times 10^3$			$\times 10^6$		
2	2.5573	0.64	0.00	3.9981	1.35	0.01
3	1.8719	1.22	0.00	2.7916	2.02	0.03
4	1.5069	1.78	0.01	2.1713	3.24	0.04
5	1.2476	2.15	0.01	1.7496	9.46	0.09
6	1.1316	5.03	0.03	1.3178	17.30	0.17
7	1.0415	8.81	0.06	1.0997	23.23	0.23
8	0.9595	11.54	0.09	0.9081	31.94	0.31
9	0.8875	14.62	0.10	0.7703	41.43	0.40
10	0.8196	18.32	0.14	0.6486	49.63	0.46
	Bavaria postal 2			Iris Plant		
	$\times 10^6$			$\times 10^2$		
2	0.9646	3.73	0.06	1.3651	3.87	0.06
3	0.6765	4.30	0.06	0.9656	4.53	0.06
4	0.5373	6.52	0.09	0.7922	16.52	0.23
5	0.4592	20.63	0.26	0.6750	18.70	0.24
6	0.3877	35.69	0.45	0.6197	47.87	0.65
7	0.3454	58.20	0.73	0.5650	67.40	0.92
8	0.3132	80.16	1.02	0.5187	106.01	1.46
9	0.2870	89.40	1.15	0.4953	177.23	2.51
10	0.2619	101.53	1.32	0.4729	251.74	3.60

5.2.3 Results for the similarity measure based on the L_∞ -norm

In this subsection we present clustering results with the INCA algorithm using the similarity measure defined by the L_∞ -norm. Results with small data sets are presented in Table 5.7, with medium size data sets in Table 5.8 and with large data sets in Table 5.9. These results show that computational effort required by the INCA algorithm in these data sets is reasonable.

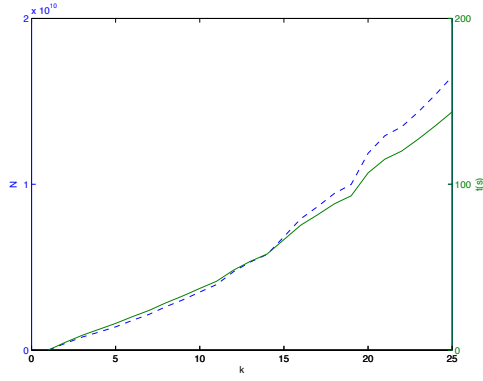
In Figures 5.1, 5.2 and 5.3 we present dependence of both the number of distance calculations and CPU time on the number of clusters for the INCA algorithm using the similarity measures based on the L_1 , L_2 and L_∞ norms, respectively. We use two data sets with the largest number of data points, namely, D15112 and Pla85900. These graphs demonstrate that both the number of distance calculations and CPU time increases almost linearly for Pla85900 data set and for all similarity measures. This dependence is also quite close to linear one for D15112 data set. Since the proposed algorithm includes an optimization solver, it is not possible to estimate its complexity. However, results show that the number of distance calculations depends at most polynomially on the number of clusters and data points.

Table 5.8: Results with the similarity measure based on the L_∞ -norm (cont.)

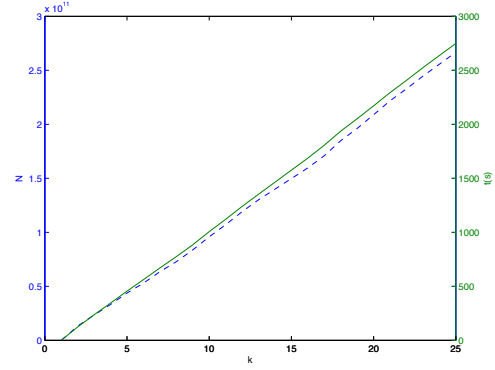
L_∞ -norm	$N(\times 10^8)$					
k	INCA					
	f_{best}	N	t	f_{best}	N	t
	Heart disease			Breast cancer		
	$\times 10^4$			$\times 10^4$		
2	1.0265	0.99	3.15	0.1927	0.49	1.46
3	0.8763	2.13	6.84	0.1737	0.65	1.95
5	0.7267	3.95	12.63	0.1521	1.74	5.36
7	0.6455	6.41	20.20	0.1402	3.15	9.76
10	0.5631	9.93	30.98	0.1294	8.91	27.08
12	0.5198	12.10	37.56	0.1247	15.20	45.83
15	0.4753	14.81	45.83	0.1186	21.31	64.13
18	0.4400	19.12	58.98	0.1138	29.16	87.46
20	0.4207	22.25	68.42	0.1108	34.43	103.08
	TSPLIB1060			Image segmentation		
	$\times 10^6$			$\times 10^5$		
2	2.9215	0.14	0.09	1.5901	5.14	21.68
3	2.3911	0.42	0.29	1.3954	14.03	57.34
5	1.7843	0.93	0.65	1.1563	41.75	164.11
7	1.4847	1.69	1.23	1.0269	70.93	272.14
10	1.1779	3.44	2.52	0.8852	261.00	959.59
12	1.0559	5.63	4.07	0.7998	366.10	1334.30
15	0.9299	11.21	8.19	0.7380	586.25	2110.36
18	0.8409	19.97	14.72	0.6861	831.64	2975.59
20	0.7957	43.37	32.47	0.6586	933.84	3334.88

Table 5.9: Results with the similarity measure based on the L_∞ -norm (cont.)

L_∞ -norm	$N(\times 10^9)$					
k	INCA					
	f_{best}	N	t	f_{best}	N	t
	TSPLIB3038			Page blocks		
	$\times 10^6$			$\times 10^6$		
2	2.8326	0.03	0.26	5.1353	0.06	1.56
3	2.1424	0.06	0.56	4.1871	0.13	3.33
5	1.6562	0.13	1.17	2.9973	0.60	14.05
10	1.1544	0.56	4.50	1.6485	1.92	42.44
15	0.9114	1.60	12.41	1.2580	4.82	101.83
20	0.7750	3.51	26.94	1.0943	10.29	212.56
25	0.6924	7.74	58.70	0.9667	13.98	286.55
	D15112			Pla85900		
	$\times 10^8$			$\times 10^{10}$		
2	0.7018	0.35	5.42	1.5803	11.53	124.16
3	0.5178	0.64	10.12	1.2250	20.74	235.26
5	0.3952	1.27	19.76	0.9184	37.25	442.32
10	0.2663	3.01	45.05	0.6523	78.26	971.83
15	0.2148	5.17	74.30	0.5455	122.83	1507.79
20	0.1859	7.93	109.57	0.4653	172.67	2087.35
25	0.1669	14.18	179.71	0.4155	224.15	2714.76

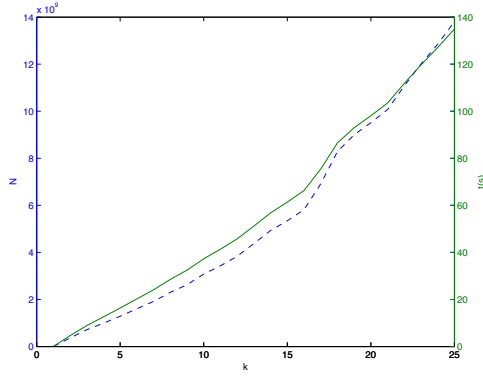


(a) D15112

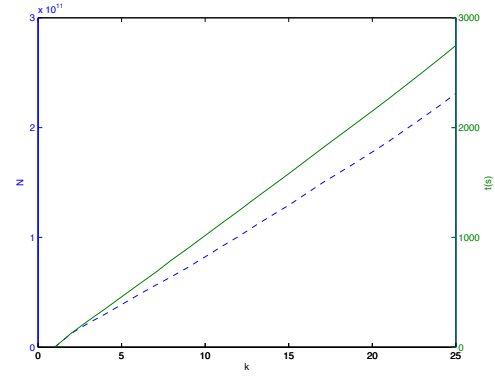


(b) Pla85900

Figure 5.1: Results for the similarity measure based on the L_1 -norm

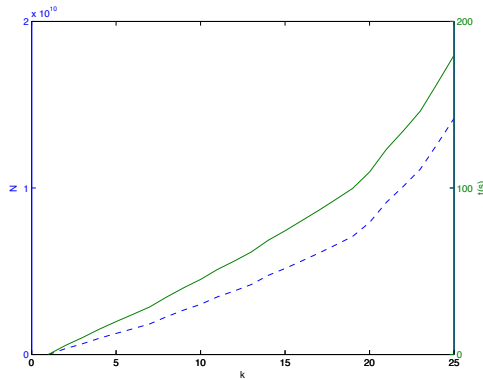


(a) D15112

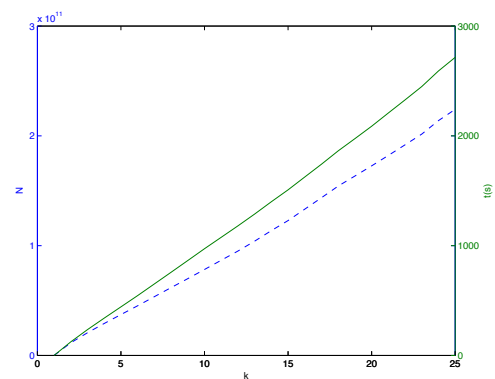


(b) Pla85900

Figure 5.2: Results for the similarity measure based on the squared L_2 -norm



(a) D15112



(b) Pla85900

Figure 5.3: Results for the similarity measure based on the L_∞ -norm

5.2.4 Results for purity in data sets with class labels

In order to determine which similarity measure provides better approximation of a data set one can use the notion of the *cluster purity* in situations where instances are already labeled. In this case we can compare the clusters with the “true” class labels. The purity P_i of the cluster i , $i = 1, \dots, k$ is defined as follows [57]:

$$P_i = \frac{1}{n_i} \max_{j=1, \dots, l} n_i^j.$$

In this expression n_i is the number of points in the cluster i , $i = 1, \dots, k$, n_i^j is the number of points in the cluster i that belong to the true class j and l is the number of true classes. The *total purity* P_t for the whole data set A is computed as:

$$P_t = \frac{1}{m} \sum_{i=1}^k \max_{j=1, \dots, l} n_i^j.$$

Among all data sets used in our experiments only four contain class label: Heart Disease, Breast Cancer, Image Segmentation and Page Blocks. Results of numerical experiments are presented in Table 5.10. In this table k stands for the number of clusters and P_t for the total purity. The number of clusters where the purity achieves its maximum is presented in bold. Results show that the use of the similarity measure based on the L_1 -norm provides better approximation than the use of similarity measures based on other two norms. Results for L_2 and L_∞ norms are similar in the sense of both the number of clusters and accuracy. The exception is the Image Segmentation data sets where the similarity measure based on the squared L_2 norm produced significantly better accuracy.

5.2.5 Visualization of results

We use Voronoi diagrams to visualize results obtained by the INCA algorithm for different similarity measures. In order to draw these diagrams we used the software available from [124]. Figures 5.4-5.6 present Voronoi diagrams for three data sets: German town, TSPLIB1060 and TSPLIB3038 data sets. In all data sets these diagrams are illustrated for five clusters. One can see that cluster structures for different similarity measures are different in all data sets,

Table 5.10: Cluster purity for different similarity measures

L_1		L_2		L_∞	
k	P_t	k	P_t	k	P_t
Heart Disease					
2	59.2593	2	57.5758	2	57.5758
5	65.6566	5	62.6263	5	61.6162
10	68.0135	10	63.2997	10	65.6566
15	69.3603	14	68.0135	13	68.6869
18	71.3805	15	67.0034	15	68.3502
20	70.7071	20	68.0135	20	67.6768
Breast Cancer					
2	94.4363	2	96.0469	2	97.0717
5	96.3397	3	97.0717	3	96.7789
6	97.6574	5	96.3397	5	96.9253
10	96.4861	10	96.9253	10	96.6325
15	97.0717	15	96.9253	15	96.3397
20	96.9253	20	96.9253	20	96.3397
Image Segmentation					
2	28.5714	2	28.5714	2	20.8225
5	60.9091	5	43.8095	5	40.0866
10	74.2857	10	61.342	10	46.1472
15	77.4026	15	71.6883	15	50.4329
19	77.7489	19	74.7619	19	51.6883
20	78.0087	20	75.671	20	54.2424
Page Blocks					
2	89.768	2	89.8593	2	89.768
5	89.9872	5	89.9141	5	89.8776
10	90.0603	10	90.0968	10	90.0238
15	90.1517	15	90.1517	15	90.0786
20	90.1517	20	90.1882	20	90.1151
23	90.8642	23	90.2065	24	90.1334
25	90.8642	25	90.2065	25	90.1334

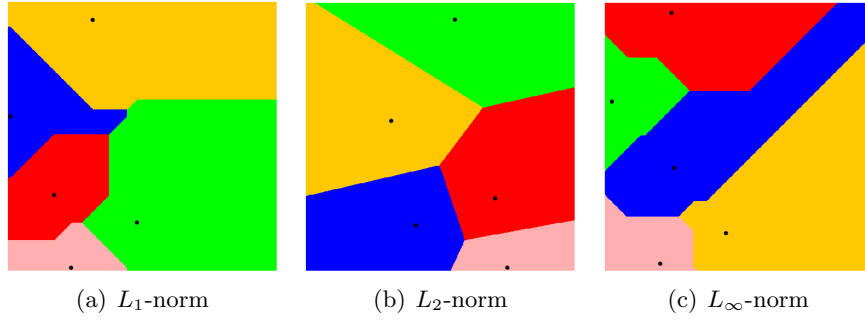


Figure 5.4: Visualization of clusters for German towns data set

however the distributions of cluster centers for different norms are similar in TSPLIB1060 data set. These results confirm that the use of different similarity measures allows one to explore different cluster structures in data sets.

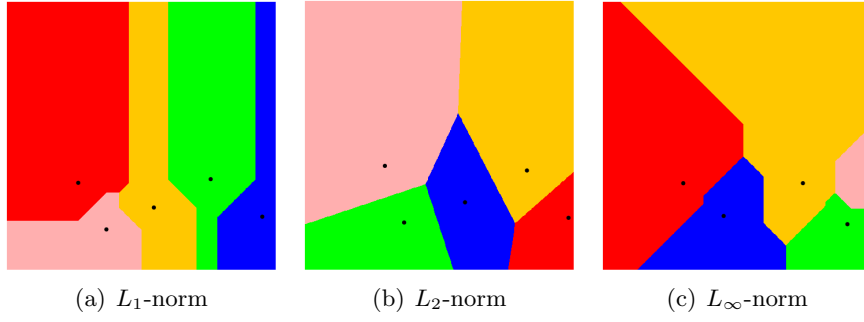


Figure 5.5: Visualization of clusters for TSPLIB1060 data set

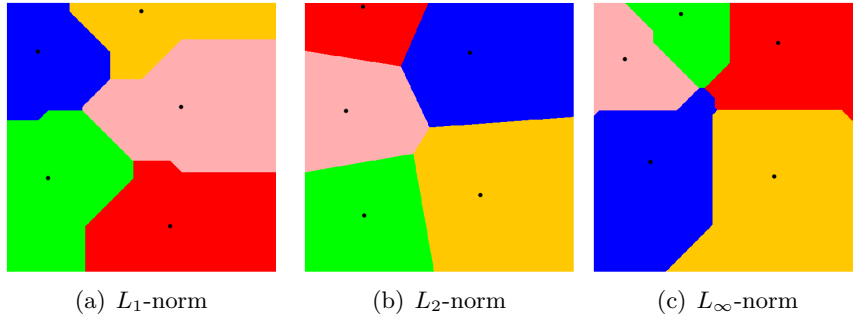


Figure 5.6: Visualization of clusters for TSPLIB3038 data set

5.3 Computational results: comparison with other clustering algorithms

In this section we compare the proposed three algorithms with other clustering algorithms. Comparison of algorithms with different similarity measures are presented separately.

5.3.1 Comparison of algorithms using the similarity measure based on the squared L_2 -norm

In subsection we use the following algorithms for comparison with the INCA algorithm:

1. *Lloyd algorithm*. This clustering algorithm is the version of the k -means algorithm. It was introduced in [97].
2. *Forgy algorithm*. This algorithm is a simple alternative of least-squares algorithm [63].
3. *MacQueen algorithm*. This clustering algorithm introduced in [99] is similar to the Forgy's algorithm. The difference is in the last stage where the MacQueen algorithm

moves the center points to the mean of their Voronoi set.

4. *Hartigan algorithm*. This algorithm was introduced in [79].
5. *K-means++ algorithm*. This algorithm is the version of the k -means algorithm and was introduced in [9]. It uses a special procedure for initialization of cluster centers.
6. *X-means algorithm*. This algorithm is an improvement of the original k -means algorithm [113]. It uses a special procedure for initialization of cluster centers.
7. *Global k-means algorithm (GKM)*. This is an incremental algorithm [96].
8. *Modified Global k-means algorithm (MGKM)*. This is the modified version of the global k -means algorithm [19].

All algorithms listed above, except the GKM and MGKM algorithms, use randomly generated starting points for cluster centers. In order to have a fair comparison with the INCA algorithm we use large number of starting points in these algorithms so that the CPU time used by them is almost the same that used by the INCA algorithm. More specifically, we used 500 starting points in all algorithms and we chose the best solution and compared it to that of found by the INCA algorithm. Results are presented in Tables 5.11-5.13. In these tables we include the error of a solution found by an algorithm.

Results for small data sets are presented in Table 5.11. These results show that the Hartigan, k -means++ and INCA algorithms are most accurate among all algorithms. The error by the GKM and MGKM algorithms are not large in comparison with other algorithms.

Table 5.12 contains results with medium size data sets. Again we can see that the Hartigan, k -means++ and INCA algorithms are among most accurate in these data sets. The GKM and MGKM algorithms also show good performance in most data sets.

5.3.2 Comparison of algorithms using the similarity measure based on the L_1 -norm

In this subsection we present the comparison of the INCA algorithm with two clustering algorithms: the k -means and the X -means algorithms. All algorithms use the similarity

Table 5.11: Comparison of algorithms using the similarity measure based on the L_2 -norm

k	Forgy	Lloyd	McQueen	Hartigan	X-means	K-means++	GKM	MGKM	INCA
German town									
2	0.00	0.00	0.00	0.00	8.76	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	11.63	0.00	1.45	1.45	0.00
4	0.00	0.00	0.00	0.00	15.35	0.00	0.72	0.72	0.00
5	0.00	0.00	0.00	0.00	11.44	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	27.26	0.00	0.00	0.27	0.00
7	0.00	0.00	0.00	0.00	31.00	0.00	0.09	0.00	0.00
8	0.00	0.00	0.00	0.00	46.65	0.00	1.33	1.23	0.00
9	0.00	0.00	0.00	0.00	51.81	0.68	2.13	4.41	0.00
10	0.00	0.00	0.28	0.00	80.32	0.00	1.79	1.51	0.00
Bavaria postal 1									
2	0.00	0.00	0.00	0.00	242.83	0.00	7.75	0.00	0.00
3	0.00	0.00	0.00	0.00	579.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	312.15	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	763.62	0.00	0.00	0.00	0.00
6	27.65	0.00	0.00	27.65	1223.59	0.00	0.00	0.00	0.00
7	0.61	0.61	1.32	0.00	2010.27	0.00	1.50	1.50	0.00
8	0.00	0.00	0.00	0.00	3296.56	0.00	0.00	0.00	0.00
9	0.00	15.43	0.00	0.00	5312.35	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	6811.38	0.00	0.00	0.00	0.00
Bavaria postal 2									
2	0.00	0.00	0.00	0.00	0.00	0.00	7.32	7.32	7.32
3	0.00	0.00	0.00	0.00	110.55	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	317.87	0.00	0.00	0.00	0.00
5	1.14	1.14	0.00	0.00	482.46	0.00	1.86	1.86	0.00
6	39.04	39.02	26.72	0.00	868.32	0.00	1.21	1.21	0.00
7	0.04	0.04	0.04	0.00	1260.32	0.00	0.55	0.55	0.04
8	4.68	12.57	6.06	0.00	1668.73	0.00	0.73	0.73	0.00
9	11.90	1.34	1.34	0.00	2046.36	0.14	0.14	0.14	0.00
10	3.91	11.64	5.54	0.00	2422.10	0.22	1.00	1.00	0.00
Iris Plant									
2	0.00	0.00	0.00	0.00	1.71	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	8.92	0.00	0.01	0.01	0.00
4	0.00	0.00	0.00	0.00	37.30	0.00	0.05	0.05	0.05
5	0.00	0.00	0.00	0.00	11.76	0.00	0.54	0.54	0.00
6	0.00	0.00	0.00	0.00	19.81	0.00	1.44	1.44	0.00
7	0.00	0.00	0.00	0.00	30.49	0.00	3.17	3.17	0.00
8	0.00	0.00	0.00	0.00	47.80	0.00	1.71	1.71	0.00
9	0.00	0.00	0.01	0.00	56.56	0.00	2.85	2.85	0.00
10	0.20	0.41	0.00	0.00	60.86	0.06	3.55	3.55	0.00

Table 5.12: Comparison of algorithms using the similarity measure based on the L_2 -norm (cont.)

k	Forgy	Lloyd	McQueen	Hartigan	X-means	K -means++	GKM	MGKM	INCA
Heart disease									
2	0.00	0.00	0.00	0.00	75.01	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	216.02	0.00	0.58	0.58	0.00
10	0.05	0.06	0.14	0.00	389.14	0.09	2.61	1.33	0.03
15	0.28	0.97	0.61	0.00	531.38	1.17	0.96	1.13	0.16
20	2.90	2.65	0.92	0.00	667.67	1.77	2.18	1.35	0.41
25	3.33	5.44	5.56	0.31	781.51	4.33	3.77	0.75	0.00
30	5.45	4.32	5.66	0.03	845.62	4.68	3.76	1.38	0.00
Breast cancer									
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	2.97	0.00	2.28	0.62	0.02
10	0.10	0.03	0.04	0.00	9.96	0.14	0.25	0.20	0.04
15	0.87	0.84	0.65	0.00	11.87	1.07	1.55	1.49	0.73
20	2.88	2.68	2.59	0.00	23.75	1.92	4.08	1.84	0.82
25	2.10	2.28	3.93	0.50	20.23	3.38	5.04	1.19	0.00
30	4.46	3.68	4.07	0.00	20.98	3.23	4.67	1.06	0.46
TSPLIB1060									
2	0.00	0.00	0.00	0.00	149.58	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	24.48	0.00	0.01	0.01	0.01
10	0.00	0.00	0.00	0.00	46.36	0.00	0.23	0.04	0.03
20	0.05	0.04	0.03	0.00	45.94	0.39	0.39	0.39	0.08
25	0.11	0.26	0.00	0.08	68.25	0.09	1.81	1.81	0.00
30	0.13	0.00	0.48	0.23	73.92	1.47	2.82	2.83	0.75
40	1.57	0.93	1.89	0.00	76.85	2.39	4.00	2.82	0.27
50	3.77	3.75	3.69	0.61	88.96	4.49	2.07	1.50	0.00
Image Segmentation									
2	0.00	0.00	0.00	0.00	6.65	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	54.85	0.00	0.00	0.00	0.00
10	1.53	1.16	0.00	0.63	106.25	0.00	1.76	1.76	1.75
20	12.08	0.18	8.87	2.38	126.52	0.77	0.00	1.40	27.72
25	11.48	15.35	17.29	4.09	173.86	4.18	0.05	0.00	23.39
30	14.65	12.37	17.01	9.35	202.64	5.75	0.07	0.06	0.00
40	14.96	12.26	17.65	15.37	260.99	6.99	1.08	1.07	0.00
50	19.96	18.87	22.55	18.82	323.81	8.13	2.22	2.21	0.00

Table 5.13: Comparison of algorithms using the similarity measure based on the L_2 -norm (cont.)

k	Forgy	Lloyd	McQueen	Hartigan	X -means	K -means++	GKM	MGKM	INCA
TSPLIB3038									
2	0.00	0.00	0.00	0.00	30.17	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	1.77	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	12.43	0.00	2.78	0.58	0.57
20	0.02	0.03	0.02	0.00	10.11	0.14	2.00	0.48	0.14
25	0.05	0.02	0.03	0.00	8.20	0.22	0.75	0.23	0.56
30	0.05	0.03	0.08	0.00	13.22	0.45	1.44	1.04	0.82
40	0.00	0.76	0.66	0.00	20.78	1.26	1.49	1.72	1.01
50	0.34	0.41	0.51	0.33	19.00	1.15	0.68	0.20	0.00
Page blocks									
5	38.99	38.99	38.99	38.99	933.67	0.00	0.00	0.00	0.00
10	208.96	46.37	216.19	38.71	2787.71	5.20	1.53	0.73	0.00
20	247.53	248.08	678.67	200.22	2209.30	36.81	0.64	0.00	1.02
30	497.05	482.87	496.96	417.76	4644.91	31.56	2.16	1.40	0.00
50	1118.04	1110.42	1064.06	943.63	7602.65	69.65	0.11	0.07	0.00
70	1883.18	1905.86	1911.44	1708.40	13742.13	97.83	0.45	0.15	0.00
80	1572.73	1794.92	1428.82	1408.72	16659.75	86.06	1.97	0.51	0.00
100	2069.75	2415.00	2074.10	2045.52	20261.88	107.94	1.03	1.14	0.00
D15112									
5	0.00	0.00	0.00	0.00	4.91	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	1.51	0.00	1.41	1.41	0.62
20	0.00	0.00	0.00	0.00	4.44	0.00	0.25	0.25	0.24
30	0.00	0.01	0.00	0.00	8.55	0.00	0.07	0.10	0.06
50	0.19	0.22	0.31	0.23	5.80	0.00	0.00	0.00	0.08
70	0.51	0.34	0.49	0.28	6.23	0.87	0.96	0.95	0.00
80	0.70	0.97	0.55	0.40	10.85	0.81	0.28	0.27	0.00
100	1.52	1.28	1.34	1.08	10.48	0.82	0.13	0.13	0.00
Pla85900									
5	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00
20	0.00	0.00	0.00	0.00	1.06	0.00	0.31	0.31	0.54
30	0.00	0.00	0.00	0.00	3.13	0.00	0.13	0.13	0.01
50	0.14	0.14	0.00	0.07	0.99	0.04	0.35	0.35	0.70
70	0.18	0.32	0.10	0.00	2.83	0.25	0.24	0.72	1.23
80	0.29	0.00	0.04	0.11	2.78	0.02	0.93	0.87	1.15
100	0.29	0.33	0.13	0.00	3.48	0.16	0.30	0.70	0.68

Table 5.14: Comparison of algorithms using the similarity measure based on the L_1 -norm

k	k -means	X -means	INCA	k -means	X -mean	INCA
German Towns				Bavaria postal 1		
2	4.23	2.68	0.00	18.58	44.21	0.00
3	16.11	4.09	0.00	53.31	34.44	0.00
4	1.64	29.36	0.00	98.27	48.73	0.00
5	6.38	14.27	0.00	146.01	102.81	0.00
6	10.23	11.24	0.00	223.08	149.68	0.00
7	14.15	18.49	0.00	284.29	148.85	0.00
8	10.16	19.12	0.00	384.27	188.39	0.00
9	10.80	26.51	0.00	441.99	338.77	0.00
10	8.14	33.78	0.00	491.15	404.42	0.00
Bavaria postal 1				Iris Plant		
2	1.44	13.31	0.00	0.28	2.05	0.00
3	35.65	15.29	0.00	0.06	2.07	0.00
4	57.83	27.51	0.00	0.51	12.42	0.00
5	90.57	14.30	0.00	2.97	5.34	0.00
6	119.23	10.23	0.00	6.76	7.15	0.00
7	151.72	20.40	0.00	8.00	11.85	0.00
8	178.83	32.68	0.00	9.19	17.12	0.00
9	201.37	39.58	0.00	9.12	21.59	0.00
10	128.65	42.21	0.00	12.35	23.72	0.00

measure defined by the L_1 -norm. Results are presented in Tables 5.14-5.16. We include the error of each algorithm for given number of clusters at these tables.

Table 5.14 contains results on small data sets. These results clearly demonstrate the superiority of the INCA algorithm over other two algorithms in these data sets. Moreover, its superiority over the k -means and X -means algorithms in Bavaria Postal 1 data set is significant.

Results for medium size data sets are given in Table 5.15. Again we can see that the INCA algorithm is much more superior than other two algorithms for these data sets. In most cases results obtained by the k -means and X -means algorithms deteriorate as the number of clusters increases.

Table 5.16 presents results for large data sets. One can see that the INCA algorithm achieves significantly better solutions than other two algorithms in most cases except the case $k = 3$ for Pla85900 data set. The algorithms the k -means and X -means algorithms are quite efficient in TSPLIB3038, D15112 and Pla85900 data sets. All these data sets have two attributes. This means that the k -means and X -means algorithms are efficient when the number of attributes is very small (2 or 3).

Table 5.15: Comparison of algorithms using the similarity measure based on the L_1 -norm (cont.)

k	k -means	X -means	INCA	k -means	X -mean	INCA
Heart disease				Breast cancer		
2	18.31	18.63	0.00	0.05	14.46	0.00
3	26.60	30.37	0.00	5.93	15.35	0.00
5	44.06	49.02	0.00	5.03	18.16	0.00
7	52.37	59.50	0.00	6.28	20.73	0.00
10	69.75	73.73	0.00	8.15	18.87	0.00
12	77.73	84.08	0.00	11.61	19.82	0.00
15	92.18	97.24	0.00	9.18	19.20	0.00
18	101.43	108.08	0.00	8.66	21.68	0.00
20	105.50	112.45	0.00	9.24	23.27	0.00
TSPLIB1060				Image segmentation		
2	37.69	0.17	0.00	1.68	8.78	0.00
3	17.61	18.37	0.00	16.66	5.62	0.00
5	7.30	13.17	0.00	0.13	7.59	0.00
7	6.86	9.35	0.00	2.10	17.72	0.00
10	11.56	11.79	0.00	9.42	21.76	0.00
12	12.58	9.13	0.00	12.80	25.18	0.00
15	13.13	14.00	0.00	18.87	30.73	0.00
18	14.14	9.44	0.00	20.21	33.01	0.00
20	9.17	12.54	0.00	25.86	28.25	0.00

Table 5.16: Comparison of algorithms using the similarity measure based on the L_1 -norm (cont.)

k	k -means	X -means	INCA	k -means	X -mean	INCA
TSPLIB3038				Page blocks		
2	4.91	0.27	0.00	27.42	29.10	0.00
3	2.05	0.62	0.00	33.09	58.19	0.00
5	1.86	4.22	0.00	83.22	111.94	0.00
10	3.24	6.99	0.00	142.63	151.02	0.00
15	2.28	1.95	0.00	164.98	158.37	0.00
20	2.93	3.40	0.00	202.92	141.11	0.00
25	2.07	4.45	0.00	238.26	148.37	0.00
D15112				Pla85900		
2	1.16	0.64	0.00	0.51	0.46	0.00
3	8.45	3.52	0.00	0.00	0.13	0.25
5	0.44	1.05	0.00	0.21	0.37	0.00
10	1.31	0.73	0.00	0.15	0.88	0.00
15	1.03	1.91	0.00	0.27	1.28	0.00
20	2.65	3.27	0.00	0.94	0.85	0.00
25	1.61	5.10	0.00	2.01	2.12	0.00

Table 5.17: Comparison of algorithms using the similarity measure based on the L_∞ -norm

k	k -means	X -means	INCA	k -means	X -mean	INCA
German Towns				Bavaria postal 1		
2	0.00	24.00	5.41	21.62	44.43	0.00
3	0.00	4.69	1.57	69.89	111.24	0.00
4	0.47	0.00	3.41	106.20	63.50	0.00
5	7.21	2.72	0.00	137.93	96.54	0.00
6	5.03	10.58	0.00	225.35	158.52	0.00
7	2.11	14.55	0.00	285.78	207.73	0.00
8	15.37	15.67	0.00	356.89	271.70	0.00
9	10.31	17.74	0.00	413.54	327.02	0.00
10	6.09	25.57	0.00	338.41	409.24	0.00
Bavaria postal 1				Iris Plant		
2	404.07	498.62	0.00	1.08	0.00	39.10
3	601.05	771.69	0.00	0.00	2.45	25.24
4	733.30	560.73	0.00	0.00	2.64	12.37
5	806.53	648.82	0.00	4.94	0.00	4.17
6	1005.88	778.72	0.00	2.47	11.94	0.00
7	1128.26	879.76	0.00	2.74	5.85	0.00
8	1224.71	977.71	0.00	9.41	13.52	0.00
9	1278.33	1046.10	0.00	11.75	17.35	0.00
10	985.72	1161.13	0.00	14.61	15.78	0.00

5.3.3 Comparison of algorithms using the similarity measure based on the L_∞ -norm

In this subsection we present the comparison of the INCA algorithm with two clustering algorithms: the k -means and the X -means algorithms. All algorithms use the similarity measure defined by the L_∞ -norm. Results are presented in Tables 5.17-5.19. In these tables the error of each algorithm for given number of clusters is included.

Results for small data sets are given in Table 5.17. These results demonstrate the significant superiority of the INCA algorithm over other two algorithms in all data sets, except the cases when the number of clusters is small.

Table 5.18 contains results for medium size data sets. One can see that the INCA algorithm is much more superior than other two algorithms in these data sets. In most cases results obtained by the k -means and X -means algorithms deteriorate as the number of clusters increases.

Results for large data sets are reported in Table 5.19. The INCA algorithm achieves significantly better solutions than other two algorithms in Page Blocks data set. However in other three data sets the k -means and X -means algorithms are more successful than the INCA algorithm. These results confirm that both the k -means and X -means algorithm is

Table 5.18: Comparison of algorithms using the similarity measure based on the L_∞ -norm (cont.)

k	k -means	X -means	INCA	k -means	X -mean	INCA
Heart disease				Breast cancer		
2	29.51	30.08	0.00	39.34	0.32	0.00
3	50.94	51.55	0.00	12.09	5.70	0.00
5	81.01	80.94	0.00	25.71	4.84	0.00
7	102.73	101.13	0.00	22.18	8.00	0.00
10	129.34	128.72	0.00	26.43	12.78	0.00
12	148.06	145.52	0.00	23.26	11.66	0.00
15	165.77	166.08	0.00	27.23	13.10	0.00
18	184.73	186.18	0.00	28.08	13.18	0.00
20	195.60	197.98	0.00	29.56	12.67	0.00
TSPLIB1060				Image segmentation		
2	30.88	45.57	0.00	12.77	11.62	0.00
3	17.93	23.59	0.00	9.52	13.88	0.00
5	16.21	18.04	0.00	24.66	23.08	0.00
7	12.97	11.35	0.00	25.50	20.02	0.00
10	18.52	17.35	0.00	21.36	29.97	0.00
12	15.58	17.15	0.00	34.01	38.56	0.00
15	16.58	13.69	0.00	38.66	34.67	0.00
18	13.34	13.87	0.00	43.64	45.01	0.00
20	19.86	18.54	0.00	49.17	41.86	0.00

efficient when the number of attributes is very small.

Table 5.19: Comparison of algorithms using the similarity measure based on the L_∞ -norm (cont.)

k	k -means	X -means	INCA	k -means	X -mean	INCA
TSPLIB3038				Page blocks		
2	1.06	4.19	0.00	8.97	7.10	0.00
3	2.74	1.62	0.00	6.21	30.99	0.00
5	0.41	0.00	1.52	47.22	79.15	0.00
10	0.00	2.16	2.82	137.85	205.06	0.00
15	0.00	0.45	2.61	193.57	293.60	0.00
20	3.43	0.00	2.20	235.09	343.56	0.00
25	0.19	0.60	0.00	272.51	329.67	0.00
D15112				Pla85900		
2	1.62	0.00	13.17	0.46	0.00	7.80
3	0.40	0.00	2.96	0.07	0.00	6.74
5	7.17	0.00	5.27	0.29	0.00	5.08
10	0.32	0.55	0.00	0.07	0.00	3.52
15	1.63	0.00	0.38	0.00	0.01	5.86
20	0.00	0.76	0.76	0.03	0.00	2.81
25	0.00	1.67	1.43	1.41	0.00	3.17

5.4 Summary

In this chapter, we discussed the implementation of three clustering algorithms: Incremental Nonsmooth k -means algorithm (INKA), Incremental Nonsmooth Optimization Clustering algorithm (INCA) and Incremental Smooth Optimization Clustering algorithm (ISCA) are

implemented. These algorithms are tested on 12 real world data sets in the sense of clustering accuracy and computational time. Furthermore, the numerical results are compared with the well-known similar clustering algorithms.

Chapter 6

Constrained self organizing maps for high dimensional data visualization

6.1 Introduction

In this chapter, a similar approach to the RPSOM [42] is proposed to improve the performance of the RPSOM and the SOM in the scene of quantization error. Instead of penalizing the rivals of the BMU, which increases the training steps, an adaptive constraint parameter is used in the learning process. The constraint parameter is chosen as a decreasing function with respect to iterations and we consider three such functions: linear, hyperbolic and sigmoid. This parameter restricts the process of updating neighborhoods of the BMU to only those neighbors which are close in the n -dimensional space. Such an approach leads to faster convergence and better local minimum of the quantization error than that of by the RPSOM and the SOM. Furthermore, the distortion error and topology preservation are improved. The proposed algorithm is tested using 8 real-world data sets.

6.2 CSOM learning algorithm

In this section the learning process of the Self Organizing Map is analyzed, furthermore, the quantization performance of the SOM is criticized. As we mentioned in Section 6.1, the set N_c in Step 3 of Algorithm 1 contains all neighborhood neurons which are connected to the BMU. These neurons are selected using the parameter r which is the radius around the BMU in the topological 2-dimensional map. The SOM updates all neighborhood neurons of the BMU in the topological map using (1.5), although relative weights of these neurons may be far from the BMU in the n -dimensional space. In this case, the value of E in (1.6) deviates from its optimal value as the neighborhood neurons in the network topology, which are not in the neighborhood of the BMU, are relocated according to (1.5) (see Figure 6.1). The relocation of the set of neighborhood neurons, M_c , $M_c \subset N_c$, which are far from the winning neuron in the n -dimensional space is illustrated in the Figure 6.1.

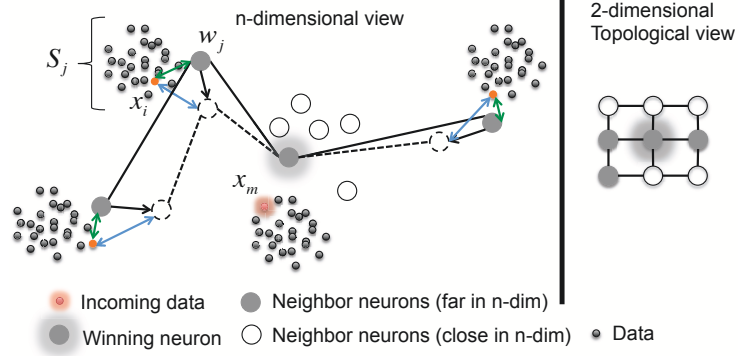


Figure 6.1: Contraction of neighbor neurons of the BMU.

Assume that the SOM algorithm is learning the input data points. In some stage, the relocation of the neighborhood neurons in the set M_c increases the inter-neuron distances (see Figure 6.1). Let have a given data point x_m , which activates the nearest neuron in the map at iteration τ , $\tau > T/\rho$, subject to $\rho \in \mathbb{N}$. The neighborhood neurons of the BMU move towards the data point x_m on the line intersecting both x_m and w_j , $w_j \in N_c$. Based on the updating formula in (1.5) the distance that neuron w_j moves is $\|\Delta w_j\|$ where,

$$\Delta w_j = \beta(x_m - w_j), \quad 0 \leq \beta < 1.$$

Therefore, by applying the updating rule to the neuron w_j , there exist data points $x_i \in S_j$, that the value

$$\Delta d(x_i, w_j) > 0,$$

after updating the neuron w_j , if

$$d(x_m, w_j) \leq \cos(\theta_{m,i})d(x_m, x_i), \quad (6.1)$$

where $\theta_{m,i}$ is the angle between data points x_m and x_i . Let have the subset of data points, $Q_j \subset S_j$, where every data point $x_i \in Q_j$ holds the condition (6.1). Thus, the increment in the value of inter-neuron distance, ΔD_j , is:

$$\Delta D_j = \sum_{i=1}^{|Q_j|} \Delta d(x_i, w_j), \quad (6.2)$$

where, $|\cdot|$ is the cardinality of a set of data. Furthermore, consider a subset of neurons $F_c \subset N_c$ including neurons w_j , which are far from the BMU in the n -dimensional space. Therefore, the value E in (1.6) increases by:

$$\Delta E = \sum_{j=1}^{|F_c|} \Delta D_j, \quad (6.3)$$

after the data point x_m is presented to the network and (1.5) is applied to the set N_c .

6.2.1 Modified learning algorithm

In this section we design an algorithm to minimize ΔE given by (6.3) for w_j , $j = 1, \dots, |N_c|$ at iterations $\tau > T/\rho$. The idea is to modify the update formula (1.5) in Step 3 of Algorithm 1 by adding a constraint parameter. We introduce a constraint parameter $\gamma \in \mathbb{R}$, which defines a subset C_c of the set of neighborhood neurons N_c which are close to the BMU:

$$C_c = \{w_j : d(w_c, w_j) < \gamma d_{min}, w_j \in N_c\}, \quad (6.4)$$

where $\gamma > 1$ and

$$d_{min} = \min \{d(w_c, w_j) : w_j \in N_c\}. \quad (6.5)$$

One can see from (6.4) that $|C_c| < |N_c|$ for values of γ close to 1 and both sets coincide as $\gamma \rightarrow \infty$. The set C_c contains neighborhood neurons in a 2-Dim topological space while preserving the condition of adjacency of $w_j \in C_c$ in the n -dimensional space. In the step 3 of Algorithm 1, if we update neighborhood neurons of the set C_c then the weight vectors, which are far from the best matching unit, will not be considered in equation (1.5). Therefore, following (6.4):

$$d_{min} \leq d(w_c, w_j) < \gamma d_{min} \quad \forall w_j \in C_c, \gamma > 1. \quad (6.6)$$

Since the distance $d(w_c, w_j)$, $w_j \in C_c$ is sufficiently small, it is easy to see that $\Delta d(x_i, w_j)$ in (6.2), where

$$\Delta d(x_i, w_j) < d(w_c, w_j),$$

is depended on the upper bound, $d(w_c, w_j)$ (see Figure 6.2).

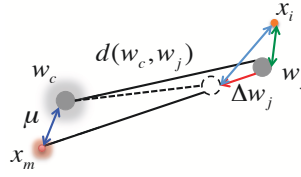


Figure 6.2: The updating procedures in the SOM algorithm.

Therefore, by limiting the value of $d(w_c, w_j)$, then we have:

$$\Delta d(x_i, w_j) < \gamma d_{min}.$$

One can see that the values ΔD_j in (6.3) are minimized.

Thus, the constrained learning algorithm can be summarized as follows.

Algorithm 6. Constrained learning algorithm

Step 1. $C_c = \emptyset$, select γ and find d_{min} from N_c using equation (6.5).

Step 2. Select neuron $w_i \in N_c$, if $\|w_i - w_c\| < \gamma d_{min}$ then

$$C_c = C_c \cup w_i$$

Step 3. If all neurons in N_c have been visited then goto 4, otherwise goto 2.

Step 4. Update the set of neighborhood neurons $w_j \in C_c$ using equation (1.5).

Thus, we minimize the equation (6.3) by applying Algorithm 6 in the step 3 of Algorithm 1 at iteration τ .

6.2.2 Adaptive selection of parameter γ

In this section, we propose linear, hyperbolic and sigmoid formulation of the constraint parameter γ in equation (6.4) with respect to iteration τ . Considering Algorithm 6, one can see that fixed values of γ may prevent spreading the neurons over the whole data space in early iterations. Although, it may take some iterations for d_{min} in equation (6.5) to be a true value as long as the neurons adapt to the input data points entirely.

From equation (6.4) it is easy to see that $C_c \equiv N_c$ for the large values of γ . Therefore the behavior of the algorithm for large values of γ is similar to that of classical SOM. This statement is true for $\gamma > \psi$ where

$$\psi = \frac{d_{max}}{d_{min}}, \quad (6.7)$$

and

$$d_{max} = \max\{d(w_c, w_j) : w_j \in N_c\}. \quad (6.8)$$

In Step 2 of Algorithm 6 we consider γ not as a fixed constant but parameter depending on iteration τ . Moreover, $\gamma(\tau)$ decreases as τ increases. Hence, to ensure the validity of the proposed formulations, the constraint functions allow the algorithm to perform similar to SOM in early stages and gradually converges to the constrained learning algorithm. In order to have such property we require γ to satisfy the following conditions: there exist $\bar{\tau} > 1$ such that:

$$\begin{cases} \gamma(\tau) > \psi & 1 < \tau < \bar{\tau} \\ 1 < \gamma(\tau) \leq \psi & \bar{\tau} \leq \tau < T \end{cases} \quad (6.9)$$

It is clear there are different ways to choose γ satisfying the conditions (6.9). In this research, we will define γ as a linear, hyperbolic and sigmoid constraint functions of τ .

Linear formulation: In the case of linear constraint function one can approximate γ as follows:

$$\gamma(\tau) = \frac{\rho}{(1-\rho)}(\psi - 1)\left(\frac{\tau}{T} - \frac{1}{\rho}\right) + \psi, \quad (6.10)$$

where $\rho \in \mathbb{N}$ defines the $\bar{\tau} = T/\rho$ which is the iteration from which the constrained learning algorithm is applied to Algorithm 6. It is easy to see that the function (6.10) satisfies condition (6.9).

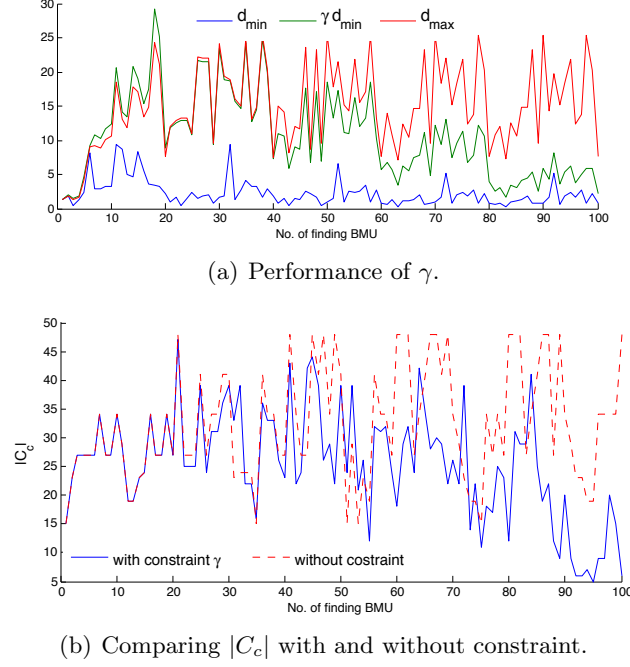


Figure 6.3: Performance of constraint γ using equation (6.10) on 20×20 SOM where $r = 3$, $\rho = 5$, $T = 5$ and $|A| = 20$.

Hyperbolic formulation: The hyperbolic expression for γ is given as

$$\gamma(\tau) = \begin{cases} \infty & 1 < \tau < \bar{\tau} \\ \frac{\psi}{\tau - \varphi} + 1 & \bar{\tau} < \tau < T \end{cases} \quad (6.11)$$

where

$$\varphi = \bar{\tau} - \frac{d_{\max}}{d_{\max} - d_{\min}},$$

and $\bar{\tau} = T/\rho$.

Sigmoid formulation: Finally, we propose the following sigmoid function to define γ :

$$\gamma(\tau) = \varepsilon(\psi - 1) \frac{\exp(\frac{-\tau + \bar{\tau}}{\delta}) - \exp(\frac{\tau - \bar{\tau}}{\delta})}{\exp(\frac{-\tau + \bar{\tau}}{\delta}) + \exp(\frac{\tau - \bar{\tau}}{\delta})} + \frac{1}{2}(\psi + 1), \quad (6.12)$$

where $0 < \varepsilon < 1$ and $1 < \delta < T$ defines the slope of changes depending on iterations. It

should be noted that if δ is chose close to 1 then we have a rapid changes in constraint function (6.12).

To explain the idea, the performance of the proposed algorithm is presented in Figure 6.3. The Figure 6.3(a) shows that from iteration 2 onwards ($\tau > \bar{\tau}$) the algorithm applies the constraint to the learning process and γd_{min} is decreasing and always less than d_{max} as $\tau \rightarrow T$, consequently, the cardinality of C_c in (6.4) decreases (see Figure 6.3(b)). The algorithm prevents further modification to the neighborhood neurons which are far from the BMU in the n -dimensional space.

6.3 The CSOM algorithm and its implementation

In this section, we apply adaptive formulations of constraint γ in Step 2 of Algorithm 6. Then Algorithm 1 for solving vector quantization problem (1.6) can be rewritten as follows.

Algorithm 7. CSOM algorithm

Step 1. Initialize dimension, maximum iteration (T), radius (r) of the network and finally weight vectors $w_j, j = 1, \dots, q$. Set $\tau := 1$.

Step 2. Select data x_i and find closest neuron c , where

$$c = \underset{j}{\operatorname{argmin}} \|x_i - w_j\|. \quad (6.13)$$

Step 3. Compute the set N_c and d_{min} and d_{max} using equations (6.5) and (6.8) respectively.

Step 4. Compute $\gamma(\tau)$ using equations (6.10), (6.11) or (6.12) and set $C_c = \emptyset$.

Step 5. Select neuron $w_i \in N_c$, if $\|w_i - w_c\| < \gamma(\tau)d_{min}$ then

$$C_c = C_c \cup w_i.$$

Step 6. If all neurons in N_c have been visited then goto Step 7, otherwise goto Step 5.

Step 7. Update the set of neighborhood neurons C_c using the following equation:

$$w_j(t+1) = w_j(t) + \alpha(\tau)h(\tau)(x_i - w_j(t)), \quad w_j \in C_c. \quad (6.14)$$

Step 8. If all input data x_i are presented to the network go to Step 9 otherwise, go to Step 2.

Step 9. Calculate E_τ using equation (1.6) and if $\tau > T$ terminate otherwise set $\tau = \tau + 1$ and go to step 2.

Note that the neighborhood function in equation (7.22) of Algorithm 13 is as follows.

$$h(\tau) = \exp\left(-\frac{r^2}{2\sigma(\tau)^2}\right), \quad (6.15)$$

where

$$\sigma(\tau) = \eta \frac{T - \tau}{T}, \quad \eta \in \mathbb{R}, \quad (6.16)$$

and usually $\eta \geq 1$.

6.3.1 Implementation of CSOM algorithm

In Algorithm 13, weight vectors $w_j, j = 1, \dots, q$ are initialized with a uniform pattern in order to be comparable with the basic SOM. The maximum number of iterations T is set between 10 and 50 for small to large dataset respectively. We set T large for large data sets because for such data sets more time is required to obtain stable network over input data. The topology of network is rectangular [91] with the same number of neurons in each column and row (i.e. $n \times n$). Each interior neuron is connected with 8 neighborhood neurons, however this number is less than 5 for border neurons. Furthermore, the radius of the map r is set to 2 for small and 3 for large number of neurons (see Table 7.1).

Table 6.1: Initialization of SOM parameters in Algorithm 13.

Data sets	Input Size	SOM Dimension	r	T
Small	$(A < 10^3)$	10×10	2	200
Medium	$(10^3 < A < 10^4)$	15×15	3	300
Large	$(10^4 < A < 0.5 \times 10^5)$	20×20	2	300
Very large	$(0.5 \times 10^5 < A < 0.8 \times 10^5)$	15×15	3	100
	$(A > 0.8 \times 10^5)$	25×25	3	30

As it is presented in Table 7.1, the number of neurons, maximum iteration number T and r are chose incrementally in order to be applicable on larger input data sets. The exception are very large data sets where r and T are smaller than those for small data sets to decrease the computational complexity.

In step 4 of Algorithm 13, we set values of $\rho \in \{\rho : 2 \leq \rho \leq 0.7 \times T, \rho \in \mathbb{N}\}$ for equations (6.10) and (6.11) which depends on data set size and the maximum iteration number T . If

the number of data points is large then we choose ρ so that to apply the constrained learning in the most first $T/2$ iterations. If the size of data set is small, then ρ is chosen in order to apply the constraint in early iterations ($\rho \geq 2$). In all experiments, the parameters ε and δ in equation (6.12) are set to 0.83 and $T/4$, respectively. The parameter η in equation (7.25) is set to 1 for all data sets.

6.4 Numerical Results

To demonstrate the efficiency of the proposed algorithm, the numerical experiments were carried out using a number of real-world data sets. Algorithm 13 was coded in NetBeans IDE under Java platform and the algorithm is tested on a MAC OSX with 2.7GHz core i7 CPU and 10GB of RAM. Eight data sets: 2 small (Iris and Wine), 2 medium size (TSPLIB1060 and Image Segmentation), 2 large (D15112, Gamma Telescope) and 2 very large (NE¹ and Pla85900) data sets are used in experiments. Iris, Wine, Image Segmentation and Gamma Telescope data sets are data sets with 4, 13, 19 and 10 attributes, respectively, which can be found in UCI Machine Learning Repository [14]. TSPLIB1060, D15112 and Pla85900 are 2 dimensional data sets from TSPLIB library [119] and NE data set from [135]. A brief description of data sets is presented in Table 10.

6.4.1 Validation of parameter ρ in adaptive constraints

The sensitivity of choosing different values of parameter ρ in (6.10), (6.11) and (6.12) on Iris and Wine data sets is presented in Figure 6.4. One can see that this parameter in sigmoid constraint is more sensitive than other two constraint functions in both data sets. Furthermore, in both data sets, the hyperbolic constraint is less sensitive to the parameter ρ and produces high quality maps with less quantization error.

In order to validate the performance of the constraint parameter ρ , the quantization errors using different settings of this parameter in four data sets, Iris, Wine, TSPLIB1060 and Image Segmentation are presented in Table 6.2. Furthermore, the average and standard deviation of quantization errors are calculated to check the robustness of this parameter in each constraints. From these results one can see that the CSOM with hyperbolic constraint

¹North East

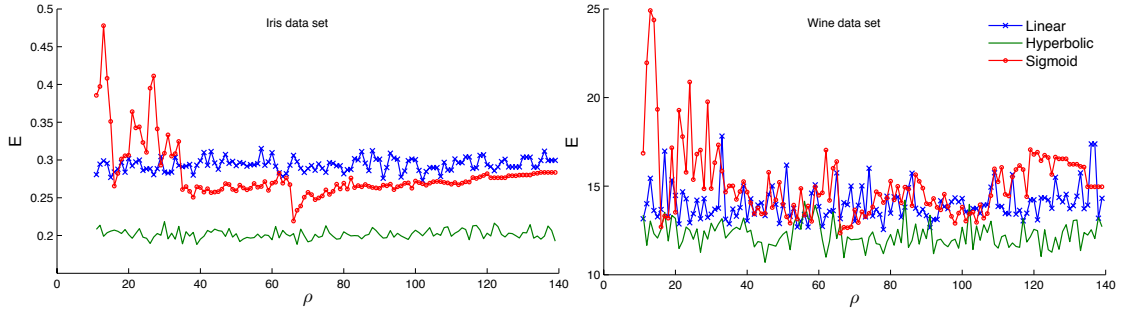


Figure 6.4: The sensitivity of choosing different values of parameter ρ on Iris and Wine data sets. E is the value of quantization error.

obtains less average quantization error and sufficiently low deviation in high dimensional data sets, Iris, Wine and Image segmentation than other two constraints. The sigmoid constraint is the one with low average quantization error and deviation in TSPLIB1060 data set. The linear constraint is the second-best constraint, which obtains less deviations in all four data sets.

Table 6.2: The quantization error, E , using different settings of parameter ρ in CSOM (Linear, Hyperbolic and Sigmoid functions).

Const.	ρ								
	10	20	40	60	80	100	140	Ave.	Std.
Iris									
Lin.	0.2951	0.3019	0.2990	0.3097	0.2817	0.3004	0.2994	0.30	0.01
Hyp.	0.2106	0.2011	0.1952	0.1944	0.2049	0.2069	0.1925	0.20	0.01
Sigm.	0.7730	0.3062	0.2636	0.2692	0.2693	0.2720	0.2834	0.35	0.19
Wine									
Lin.	14.5293	14.4811	13.0406	13.8492	13.4721	14.3073	14.3171	14.00	0.57
Hyp.	12.1234	13.1810	12.4003	13.1966	12.2707	12.6803	12.7122	12.65	0.42
Sigm.	54.1000	13.5297	14.6650	14.4896	15.2834	13.8161	14.9614	20.12	15.00
	ρ								
	10	40	80	100	140	180	200	Ave.	Std.
TSPLIB1060									
Lin.	346.8374	351.0316	345.5140	358.7634	332.5446	333.2026	371.6440	348.51	13.83
Hyp.	468.8084	462.0523	418.3928	401.7283	502.9605	376.1342	371.4968	428.80	50.17
Sigm.	344.4523	347.7434	376.8300	324.9194	328.8889	336.5058	334.9467	342.04	17.30
Image Segmentation									
Lin.	25.3648	25.4697	25.1439	24.8948	25.0708	24.6638	24.7583	25.05	0.30
Hyp.	22.0328	21.3622	21.7399	20.9122	21.4309	21.9969	21.2374	21.53	0.41
Sigm.	39.7772	26.5647	28.4100	22.8137	26.2363	26.5249	26.1327	28.07	5.42

6.4.2 Comparison with the SOM

The error values e of the quantization values E using equation (1.6) for different iterations on all data sets are presented in Tables 6.3-6.4. The error e is calculated using the following formula:

$$e = \frac{E - E_{best}}{E_{best}} \cdot 100\%. \quad (6.17)$$

In these tables t stands for the CPU time used by an algorithm and the numbers in these tables should be multiplied by the number indicated after the name of each data set to get true values of E_{best} . We include results for different iterations of each to demonstrate their performance. As it is presented in these tables, in early iterations the CSOM is performing same as the classical SOM until the constraint is applied to the learning process, then the CSOM performs much better than the SOM. From these results one can see that the CSOM outperforms SOM in all data sets and the hyperbolic constraint finds best solution (providing the lowest value of E) on 5 data sets (Iris, Wine, Image Segmentation, Gamma Telescope and NE), sigmoid on 3 data sets (TSPLIB1060, D15112 and Pla85900) and linear is the second-best constraint in 3 data sets (TSPLIB1060, Gamma Telescope and Pla85900).

Table 6.3: Results for small and medium size data sets.

τ	E_{best}	CSOM							
		SOM		Linear		Hyperbolic		Sigmoid	
		e	t	e	t	e	t	e	t
	$\times 10^0$	Iris							
1	2.4136	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
5	0.8222	0.00	0.08	0.00	0.01	0.00	0.01	0.00	0.01
10	0.4199	80.52	0.10	80.52	0.02	0.00	0.02	80.52	0.02
50	0.2160	119.03	0.21	119.03	0.12	0.00	0.09	119.03	0.12
100	0.1941	51.16	0.34	46.01	0.23	0.00	0.17	57.50	0.24
200	0.1881	53.06	0.53	43.86	0.43	0.00	0.33	16.59	0.40
	$\times 10^1$	Wine							
1	1.4507	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
5	0.3349	144.94	0.10	144.94	0.03	0.00	0.02	144.94	0.03
10	0.2605	231.32	0.13	231.32	0.06	0.00	0.04	231.32	0.06
50	0.1343	318.54	0.36	331.72	0.27	0.00	0.16	318.54	0.28
100	0.1093	111.99	0.60	70.72	0.53	0.00	0.29	123.97	0.54
200	0.1082	59.98	1.10	18.02	0.93	0.00	0.55	9.24	0.78
	$\times 10^2$	TSPLIB1060							
1	5.7347	0.00	0.12	0.00	0.04	0.00	0.04	0.00	0.04
5	3.2195	0.00	0.32	15.95	0.31	0.00	0.30	0.00	0.30
15	3.8098	37.06	0.69	0.00	0.96	37.06	0.98	37.06	0.94
75	2.7782	0.00	2.93	8.64	4.93	0.00	4.98	0.00	4.81
150	0.9196	23.26	5.58	0.00	9.75	23.26	9.87	23.26	9.56
300	0.3124	7.23	10.48	1.66	18.53	16.55	18.75	0.00	17.98
	$\times 10^1$	Image Segmentation							
1	1.1162	0.00	0.32	0.00	0.37	0.00	0.36	2.57	0.34
5	0.8451	16.36	1.21	48.89	1.66	0.00	1.55	14.71	1.57
15	0.6216	71.44	3.19	76.54	4.57	0.00	3.77	50.97	4.31
75	0.2999	173.92	14.87	122.67	21.37	0.00	14.51	137.71	21.14
150	0.2152	100.33	29.34	77.83	41.58	0.00	26.92	1.77	33.43
300	0.2051	27.50	56.53	18.77	76.05	0.00	51.06	0.20	57.09

Table 6.4: Results for small and medium size data sets.

τ	E_{opt}	CSOM							
		SOM		Linear		Hyperbolic		Sigmoid	
		e	t	e	t	e	t	e	t
	$\times 10^2$	D15112							
1	2.3218	0.00	0.38	0.00	0.53	0.00	0.53	0.00	0.44
5	1.3541	0.00	1.47	0.00	2.39	0.00	2.40	0.00	2.26
15	1.3809	0.00	3.86	0.00	6.76	20.75	5.98	0.00	6.78
75	0.6992	46.57	18.02	46.57	32.92	0.00	24.09	46.57	33.52
150	0.4455	17.53	35.81	17.53	65.82	0.00	46.26	17.53	67.26
300	0.3698	0.57	71.02	1.35	121.22	16.68	90.40	0.00	122.56
	$\times 10^2$	Gamma Telescope							
1	1.2449	0.00	0.67	0.00	1.12	0.00	1.20	0.00	1.21
5	1.3468	0.00	3.01	0.00	5.70	0.00	5.64	0.00	6.10
10	1.3494	0.00	5.78	0.00	11.16	3.50	10.26	0.00	11.80
50	0.7721	35.40	27.09	42.07	56.26	0.00	35.50	35.40	59.44
100	0.4555	62.70	55.89	44.35	109.41	0.00	60.30	45.27	115.24
200	0.3510	5.16	109.68	3.28	196.90	0.00	107.47	0.37	159.77
	$\times 10^{-1}$	NE							
1	2.6821	0.00	0.94	0.00	2.48	0.00	2.49	0.00	2.81
5	0.9135	195.44	4.02	195.44	12.88	0.00	11.68	195.44	14.00
10	0.8027	237.05	7.48	237.05	25.89	0.00	20.50	237.05	28.10
25	0.5573	361.44	17.69	270.02	64.85	0.00	41.76	361.44	72.94
50	0.2725	819.82	34.62	231.30	130.26	0.00	71.59	819.82	147.50
100	0.1515	4.16	68.15	6.14	248.12	9.31	124.55	0.00	263.54
	$\times 10^5$	Pla85900							
1	3.9514	0.00	4.51	0.00	9.44	0.00	8.94	0.00	8.53
5	3.6387	0.00	20.99	0.00	52.89	0.00	50.47	0.00	49.78
10	3.8160	0.00	40.62	0.00	106.15	0.00	100.96	0.00	100.30
15	3.0293	0.00	59.94	0.00	166.88	0.00	156.32	0.00	156.20
20	1.2720	31.16	77.79	0.00	218.68	14.34	199.25	31.16	209.46
30	0.1747	6.64	114.26	0.11	298.67	98.17	274.29	0.00	289.62

CSOM finds significantly better solution on 5 data sets (Iris, Wine, TSPLIB1060, Image Segmentation, and Pla85900) than SOM from 6.2% on Pla85900 up to 37.48% on Wine data set. In this case, the minimum improvement gained by the CSOM is on D15112 data set where the obtained solution is only 0.57% better than SOM. In other two data sets (Gamma Telescope and NE) the CSOM reduces the value of the quantization error E about 5% on the rest of the data sets. Note that in the most data sets CPU time required by the Constrained SOM is slightly larger than that of by the SOM. This is due to the fact that the CSOM tries to find nearest neighborhood neurons in n -dimensional space. Since both the NE and Pla85900 are large data sets the SOM algorithm requires more iterations than in other smaller data sets to spread neurons over the whole data set. Therefore, in these data sets constraints are applied at iterations close to $T/2$ to ensure that values calculated by constraint functions (6.10)-(6.12) are true values.

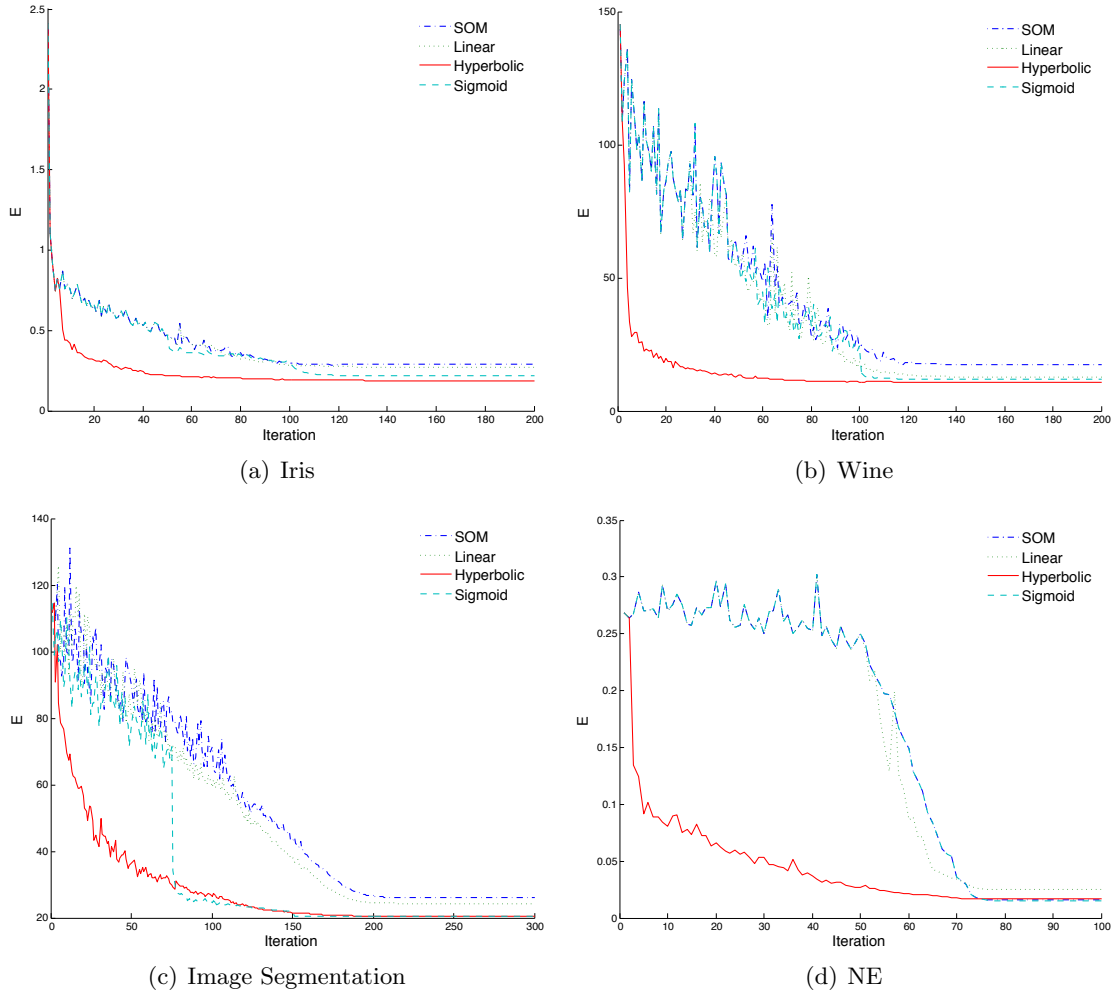


Figure 6.5: Convergence of CSOM vs SOM.

To see the performance of proposed algorithm, In Figures 6.5 we present values of E obtained by the SOM and the CSOM depending on iteration τ in Iris, Wine, Image Segmentation and NE data sets. From these figures, it is obvious that both the SOM and the CSOM converge as $\tau \rightarrow T$. Comparing values of E in all data sets, one can see that the CSOM converges significantly faster than the SOM in iterations less than 20. The CSOM with the hyperbolic constraint produces best results in all 4 data sets.

6.4.3 Complexity comparison with SOM

The time complexity of Self Organizing Map is linear with respect to number of data points but it is dependent quadratically on the number of the neighborhood neurons to be updated by the winner neuron. The most time consuming step in the SOM algorithm is Step

2 and Step 3 where the equation (1.5) is calculated. The total number of calculations, ρ , in Step 3 depends on the parameter r which defines the set of neighborhood neurons to be updated by equation (1.5).

The complexity of SOM can be formulated as $O(T\rho n)$, where

$$\rho = \sum_{r_0=1}^r r_0^2 + 4r_0 + 4,$$

and n is number of data points. The proposed CSOM reduces the total value of ρ significantly by restricting the number of neighborhood neurons to be updated by the algorithm. This restriction is applied by the constraint parameter. In Table 6.5, the total number of calculations of (1.5), $N = T\rho n$, for all data sets is presented. From these results, one can see that the number N obtained by CSOM is considerably less than those with the classical SOM in all data sets. The complexity is improved significantly using the proposed algorithm in 4 data sets: Iris, Image Segmentation, D15112 and NE.

6.4.4 Distortion error

Note that the error E shows the quantization quality of the network. However, there is a distortion measurement which can be used to calculate the overall quality of the map. Unlike the quantization error, the distortion measure ξ considers both vector quantization and topology preservation of the SOM. The distortion measure is defined as follows [8]:

$$\xi = \sum_{x_i \in A} \sum_{w_j \in \Psi} h_{cj} \|x_i - w_j\|^2, \quad j \neq c, \quad (6.18)$$

where c is the BMU of x_i and h_{cj} is the neighborhood function of neurons c and j defined by Equation (7.23).

Table 7.5 presents the numerical results of distortion measure (7.28) on all data sets. From these results, one can see that the proposed algorithm outperforms the SOM in all data sets. The improvement of distortion error, ξ , is significance in eight data sets (Wine, TSPLIB1060, Image Segmentation, TSPLIB3038, D15112, Gamma Telescope, NE and Pla85900). The value n_d in the Table 7.5 presents the number of neurons which never activated (dead neurons) by the input data points. The number of dead neurons, n_d , for CSOM in all data sets is equal

Table 6.5: Total number of calculations of (1.5), N , for all data sets.

CSOM					CSOM				
τ	SOM	Linear	Hyperbolic	Sigmoid	τ	SOM	Linear	Hyperbolic	Sigmoid
Iris ($\times 10^4$)					Wine ($\times 10^4$)				
1	1.05	1.05	1.05	1.05	1	4.72	4.72	4.72	4.72
5	5.34	5.34	5.34	5.34	5	22.4	22.4	18.1	22.4
10	10.5	10.5	9.05	10.5	10	44.4	44.4	28.1	44.4
50	51.6	51.6	18.6	51.6	50	220	212	68.4	220
100	103	99.0	23.1	92.8	100	437	392	93.7	407
200	205	159	31.0	96.2	200	872	611	135	421
TSP.1060 ($\times 10^5$)					Image Seg. ($\times 10^6$)				
1	0.81	0.81	0.81	0.81	1	1.59	1.59	1.59	1.29
5	3.82	3.85	3.82	3.82	5	7.82	7.83	6.59	6.72
15	11.3	10.9	11.3	11.3	15	23.6	23.3	13.5	20.6
75	56.1	50.3	56.1	56.1	75	118	111	26.1	105
150	112	93.0	112	112	150	235	214	33.0	110
300	223	143	123	141	300	462	361	42.9	110
D15112 ($\times 10^6$)					Gamma Tel. ($\times 10^6$)				
1	0.62	0.62	0.62	0.620	1	3.71	3.71	3.71	3.71
5	3.09	3.09	3.09	3.09	5	18.6	18.6	18.6	18.6
15	9.26	9.26	6.24	9.26	10	37.1	37.1	32.3	37.1
75	46.5	46.5	11.7	46.5	50	187	178	70.9	187
150	93.4	93.4	15.1	93.4	100	376	338	89.3	353
300	187	146	21.1	141	200	756	558	120	372
NE ($\times 10^6$)					Pla85900 ($\times 10^6$)				
1	1.94	1.94	1.94	1.94	1	6.90	6.90	6.90	6.90
5	9.68	9.68	7.78	9.68	5	34.5	34.5	34.5	34.5
10	19.4	19.4	12.7	19.4	10	68.9	68.9	68.9	68.9
25	48.6	48.6	21.3	48.6	15	101	101	101	101
50	97.9	97.9	29.4	97.9	20	130	127	120	130
100	196	147	40.6	170	30	200	164	135	175

Table 6.6: Results of distortion measure on all data sets

Dataset		CSOM							
		SOM		Linear		Hyperbolic		Sigmoid	
		ξ	n_d	ξ	n_d	ξ	n_d	ξ	n_d
Iris	$\xi \times 10^{-3}$	1.75	33	0.47	29	1.18	23	1.37	27
Wine	$\xi \times 10^{-3}$	1.85	24	0.29	17	0.25	24	0.13	22
TSP1060	$\xi \times 10^4$	1.32	16	0.82	7	0.99	20	0.57	10
Image Seg.	$\xi \times 10^{-4}$	7.13	9	0.74	6	1.01	13	0.74	4
D15112	$\xi \times 10^3$	4.04	0	0.49	0	20.50	49	0.44	1
Gamma Tel.	$\xi \times 10^{-9}$	4.85	0	0.02	1	0.12	0	0.03	0
NE	$\xi \times 10^{-6}$	1.09	19	0.37	1	0.19	7	0.01	9
Pla85900	$\xi \times 10^5$	16.30	0	0.10	0	0.26	4	0.08	0

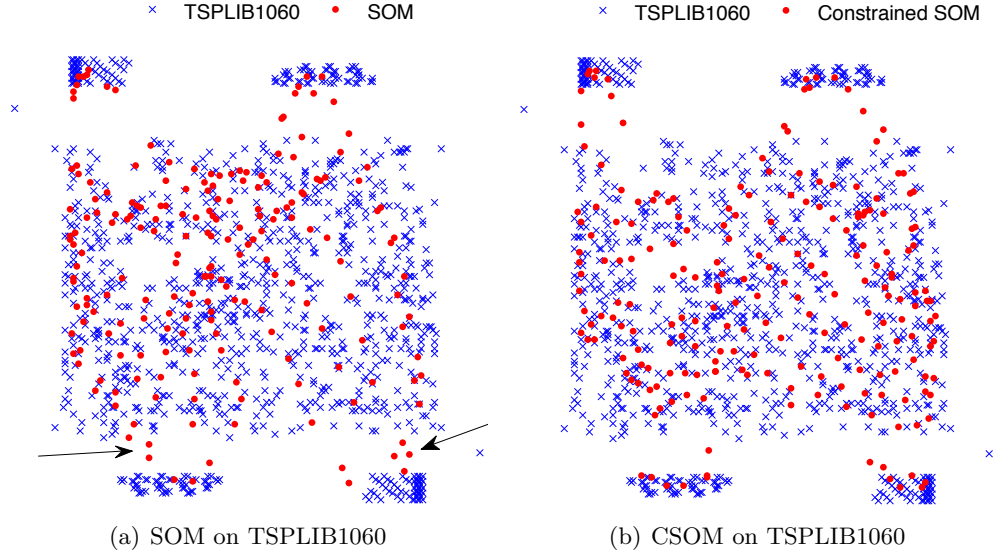


Figure 6.6: Topology preservation of CSOM vs SOM.

or less than that for the SOM. This means that the proposed algorithm activates more neurons and distributes the neurons more efficiently than the classical SOM.

6.4.5 Topology preservation

In this section we present the topology preservation of the proposed algorithm. We choose 2-dimensional data sets which can be easily displayed. Figures 6.6 and 6.7 show the topology of the SOM and CSOM for two data sets: TSPLIB1060, NE. It should be noted that only active neurons are presented in these figures. In Figure 6.6(a) one can see that some of the SOM's neurons for the TSPLIB1060 data set are far from the mapped data points which leads to increments in the quantization error E . These is due to the absorption of these neurons by their neighborhoods which, in fact, are far from these neurons in the n -dimensional space.

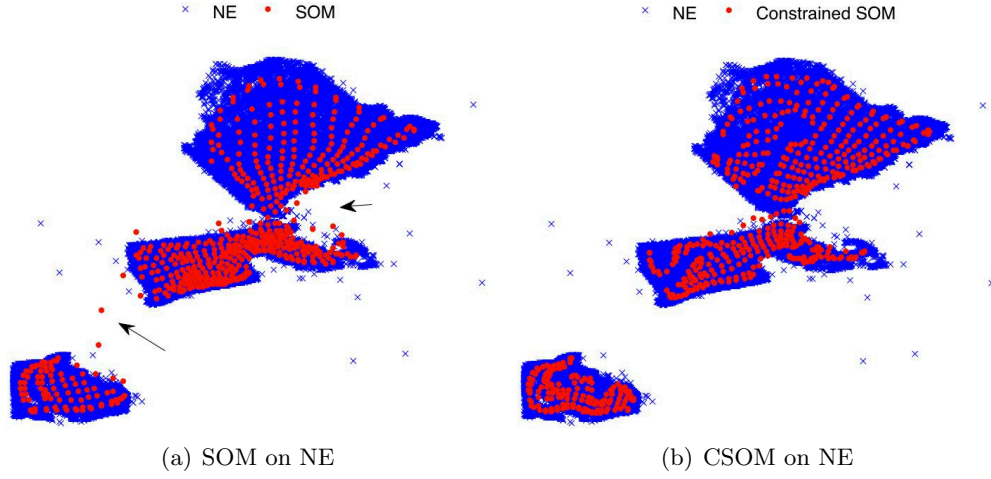


Figure 6.7: Topology preservation of CSOM vs SOM.

On the contrary one can see from Figure 6.6(b) that the CSOM spreads the neurons over the mapped data more accurately by overcoming the above mentioned deficiency in the SOM. These improvements can be obviously seen by comparing Figures 6.6(a) and 6.6(b) at the bottom-left and right of the maps where exists a gap between two groups of data points. Figure 6.7 presents the topology of the SOM and CSOM on NE data set. The CSOM in Figure 6.7(b) is more accurate than the SOM in Figure 6.7(b) where some active neurons are located in the space outside of the region where the data set is located (they are displayed by an arrow).

6.4.6 Comparison with other algorithms

In this section the CSOM is compared to similar topology preservation algorithms in the sense of accuracy and computational time. The CSOM is compared with Growing Grid (GG) [67], Growing Neural Gas (GNG) [66], Growing Hierarchical SOM (GHSOM) [117], Principal Component Analysis (PCA) [123], Sammon's Mapping [122], Fuzzy Sammon Mapping [93] and RPSOM [42] algorithms. The value e using 6.17 and the best value of quantization error, \mathbf{E}_{best} , are presented in Table 6.7. In all data sets, except Gamma Telescope, the CSOM produces less quantization error than other algorithms. The GNG algorithm obtains less error on Gamma Telescope data set in comparison with other algorithms. From the results presented in the Table 6.7, one can see that the PCA is the second best algorithm in Iris, D15112 and Gamma Telescope data sets. Moreover, the RPSOM is the second best

algorithm in Wine, NE and Pla85900 data sets. The results obtained by GG and GNG algorithms are close to those obtained by CSOM in TSPLIB1060 and Image Segmentation data sets, respectively. The RPSOM algorithm generates satisfactory results on NE and Pla85900 data sets in comparison with CSOM. Note that the PCA, Sammon's Mapping and Fuzzy Sammon algorithms failed in two very large data sets, NE and Pla85900, due to high computational time requirement.

Table 6.7: Results of quantization error on all data sets

Alg.	Iris	Wine	TSP.1060	Image Seg.	D15112	Gamma Tel.	NE	Pla85900
E_{best}								
	$\times 10^{-1}$	$\times 10^1$	$\times 10^2$	$\times 10^1$	$\times 10^2$	$\times 10^1$	$\times 10^{-2}$	$\times 10^4$
	1.88	1.08	3.12	2.05	3.70	3.14	1.51	1.75
e								
GG	45.35	66.34	2.73	29.59	32.24	5.93	51.04	1452.03
GNG	35.42	34.75	20.36	2.94	4.50	0.00	13.73	249.67
GHSOM	30.23	45.62	47.37	157.50	123.25	109.72	-	-
PCA	5.15	16.09	14.47	29.09	4.25	3.23	-	-
Sammon	5.79	16.15	14.47	29.10	4.25	3.54	-	-
FuzSam	39.17	23.67	13.88	59.85	3.75	42.24	-	-
RPSOM	12.01	11.91	8.59	15.72	2.69	14.57	3.00	2.12
Linear	43.84	18.03	1.66	18.78	1.37	15.27	6.19	0.10
Hyper.	0.00	0.00	16.56	0.00	16.71	11.61	9.34	98.14
Sigm.	16.58	9.22	0.00	0.21	0.00	12.04	0.00	0.00

Table 6.8: The CPU time required by all algorithms

Alg.	Iris	Wine	TSP.1060	Image Seg.	D15112	Gamma Tel.	NE	Pla85900
t								
GG	4.40	4.31	10.67	30.70	163.63	297.60	166.91	622.70
GNG	3.16	4.48	6.20	13.90	26.85	102.10	97.76	115.82
GHSOM	2.06	2.87	62.88	303.68	10848.36	18363.03	-	-
PCA	0.04	0.05	0.31	2.31	34.07	131.72	-	-
Sammon	3.81	4.36	40.35	194.72	6620.42	9386.41	-	-
FuzSam	2.43	3.01	24.64	25.26	144.93	196.65	-	-
RPSOM	6.29	7.58	55.57	130.11	804.15	726.64	968.34	445.05
Linear	0.42	0.87	17.89	92.19	128.00	211.62	236.54	338.24
Hyper.	0.33	0.55	18.74	73.62	96.05	127.55	156.08	325.50
Sigm.	0.37	0.73	19.78	68.10	123.20	140.72	253.70	315.96

The CPU time required by all algorithms are presented in the Table 6.8. The CSOM is the best algorithm in Iris, Wine, NE and Pla85900 data sets in the sense of both accuracy and the required cpu time. The PCA and GNG algorithms are fast in most of the data sets, but the generated errors by these algorithms are quit far from the satisfactory results. From these results one can see that the CSOM outperforms the RPSOM algorithm in all cases.

6.5 Summary

The Constrained SOM algorithms (CSOM) and its modified learning algorithm are developed in this chapter to overcome the shortcomings that are involved with the SOM algorithm. The proposed algorithm is implemented and tested on 8 real world datasets. The numerical results are compared with the existing data visualization algorithms, including SOM, in the sense of quantization error and the required CPU time.

Chapter 7

Modified self organizing maps for vector quantization and clustering problems

7.1 Introduction

In this chapter, to improve the performance of the SOM, an initialization algorithm based on the split and merge procedure is proposed. The high dense areas in input data space are detected by this procedure. Then neurons are generated in those detected areas. A new topology is introduced to restrict the adaptation to the neurons from the same high density area. Such an approach leads to a better local minimum of the quantization error than that of by the SOM. The proposed algorithm is tested using eight real-world data sets.

7.2 Splitting and merging algorithms

In this section splitting and merging procedures in cluster analysis are introduced. More specifically, first one splitting algorithm and one merging algorithm are described and then an algorithm based on the combination of these two is presented.

Assume that k clusters C_1, \dots, C_k and their corresponding centers c_1, \dots, c_k are given.

These centers are solutions to the following problem:

$$\text{minimize } f = \sum_{i=1}^k \sum_{x \in C_i} \|x - c_i\|^2. \quad (7.1)$$

In some cases data points from the cluster C_i are not dense in some neighborhood of its center c_i . Given a radius $\varepsilon > 0$ consider the following two sets for the cluster C_i :

$$\Phi_c^i(\varepsilon) = \{x_j \in C_i \mid d(x_j, c_i) \leq \varepsilon\} \quad (7.2)$$

and

$$\Phi_s^i(\varepsilon) = \{x_j \in C_i \mid \varepsilon < d(x_j, c_i) \leq r_i\}, \quad (7.3)$$

where

$$r_i = \max\{d(x, c_i) \mid x \in C_i\}, \quad i = 1, \dots, k. \quad (7.4)$$

Two clusters C_i and C_l are said to be well separated if $d(c_i, c_l) \geq (r_i + r_l)$.

It is clear that for any cluster $C_i, i = 1, \dots, k$ there exist $\varepsilon_i \in (0, r_i]$ such that $|\Phi_c^i(\varepsilon)| = \max(|\Phi_c^i(\varepsilon)|, |\Phi_s^i(\varepsilon)|)$ for all $\varepsilon \in (\varepsilon_i, r_i]$. Let $\varepsilon_i = \beta r_i$, where $\beta \in (0, 1)$. If ε_i is sufficiently small then data points from the cluster C_i are dense around its center c_i . ε_i will be used to design a splitting algorithm for clusters whereas the definition of well separated clusters will be used to design a merging algorithm.

7.2.1 Splitting

In this subsection the splitting procedure for clusters is described. This procedure is designed using the parameter β and also the special scheme to identify parts of a cluster where most of points reside.

Assume that a set of k clusters, $\Omega = \{C_1, \dots, C_k\}$ and a number $\beta \in (0, 1)$ are given. The number of points within the radius $\varepsilon_i = \beta r_i$ from the center of the cluster C_i is:

$$L_c^i = |\Phi_c^i(\varepsilon_i)|.$$

Introduce the angle $\theta_{i,j}$ between the cluster center c_j and the data point $x_i \in C_j$ assuming

that $c_j \neq 0$ and $x_i \neq 0$:

$$\theta_{i,j} = \arccos \frac{\langle x_i, c_j \rangle}{\|x_i\| \|c_j\|}. \quad (7.5)$$

Remark 7.2.1. In order to make (7.5) well-defined the cluster C_j is transformed so that the point $v = (\delta, \dots, \delta) \in \mathbb{R}^n$ becomes its center. Here $\delta > 0$ is a sufficiently small number, say $\delta \in (0, 0.1]$. It is clear that points x_i from this cluster will be transformed as follows:

$$\bar{x}_i^t = x_i^t - c_j^t + \delta, \quad t = 1, \dots, n.$$

Moreover, only \bar{x}_i 's satisfying the condition

$$\varepsilon_j < d(\bar{x}_i, v) \leq r_j$$

are considered. Then the angle $\theta_{i,j}$ between v and \bar{x}_i is well defined. □

Now introduce the following two sets:

$$\Phi_u^j(\varepsilon_j) = \left\{ x_i \in C_j \mid \varepsilon_j < d(x_i, c_j) \leq r_j, \quad 0 \leq \theta_{i,j} \leq \frac{\pi}{2} \right\}, \quad (7.6)$$

and

$$\Phi_d^j(\varepsilon_j) = \left\{ x_i \in C_j \mid \varepsilon_j < d(x_i, c_j) \leq r_j, \quad \frac{\pi}{2} < \theta_{i,j} \leq \pi \right\}. \quad (7.7)$$

The cardinalities of these sets are $L_u^j = |\Phi_u^j(\varepsilon_j)|$ and $L_d^j = |\Phi_d^j(\varepsilon_j)|$. The sets $\Phi_c^i(\varepsilon_i)$, $\Phi_u^j(\varepsilon_j)$ and $\Phi_d^j(\varepsilon_j)$ satisfy the following conditions:

1. $L_u^j + L_d^j + L_c^j = |C_j|$;
2. $\Phi_c^j(\varepsilon_i) \cup \Phi_u^j(\varepsilon_j) \cup \Phi_d^j(\varepsilon_j) = C_j$;
3. $\Phi_c^j(\varepsilon_i) \cap \Phi_u^j(\varepsilon_j) = \emptyset$, $\Phi_c^j(\varepsilon_i) \cap \Phi_d^j(\varepsilon_j) = \emptyset$, $\Phi_u^j(\varepsilon_j) \cap \Phi_d^j(\varepsilon_j) = \emptyset$.

The outcome of the splitting procedure on the cluster C_j depends on the values of L_u^j , L_d^j and L_c^j . If

$$L_c^j \geq \max\{L_d^j, L_u^j\} \quad (7.8)$$

then data points are dense around the cluster center and such a cluster is not split. If

$$L_c^j < \max\{L_d^j, L_e^j\} \quad (7.9)$$

then this cluster is divided into two new clusters. In order to do so the following two subsets of $\Phi_c^i(\varepsilon_i)$ are defined:

$$\Phi_{cu}^j(\varepsilon_j) = \left\{ x_i \in \Phi_c^i(\varepsilon_i) \mid d(x_i, c_j) \leq \varepsilon_j, \ 0 \leq \theta_{i,j} \leq \frac{\pi}{2} \right\}, \quad (7.10)$$

and

$$\Phi_{cd}^j(\varepsilon_j) = \left\{ x_i \in \Phi_c^i(\varepsilon_i) \mid d(x_i, c_j) \leq \varepsilon_j, \ \frac{\pi}{2} < \theta_{i,j} \leq \pi \right\}. \quad (7.11)$$

Then the cluster C_j is split into two new clusters C_j^* and $C_{j'}^*$ as follows:

$$C_j^* = \{\Phi_u^j(\varepsilon_j) \cup \Phi_{cu}^j(\varepsilon_j)\}, \quad (7.12)$$

with the center

$$c_j^* = \frac{1}{|C_j^*|} \sum_{x_i \in C_j^*} x_i, \quad (7.13)$$

and

$$C_{j'}^* = \{\Phi_d^j(\varepsilon_j) \cup \Phi_{cd}^j(\varepsilon_j)\}. \quad (7.14)$$

with the center

$$c_{j'}^* = \frac{1}{|C_{j'}^*|} \sum_{x_i \in C_{j'}^*} x_i. \quad (7.15)$$

Thus, the splitting algorithm can be summarized as follows:

Algorithm 8. Splitting algorithm

Step 0. *Input:* A collection of k clusters $\Omega = \{C_1, \dots, C_k\}$, and the ratio $\beta \in (0, 1)$.

Step 1. Select cluster $C_j \in \Omega$ and calculate its center c_j .

Step 2. Calculate $d(x_i, c_j)$ and also $\theta_{i,j}$ using (7.5) for all data point $x_i \in C_j$.

Step 3. For each cluster C_j calculate sets $\Phi_c^j(\varepsilon_i)$, $\Phi_u^j(\varepsilon_j)$, $\Phi_d^j(\varepsilon_j)$ using (7.2), (7.6) and (7.7), respectively.

Step 4. If (7.8) is satisfied then go to Step 6, otherwise go to Step 5.

Step 5. Split the cluster C_j into two new clusters C_j^* and $C_{j'}^*$ using (7.12) and (7.14), respectively. Update Ω and set $k := k + 1$.

Step 6. If all clusters C_j , $j = 1, \dots, k$ are visited terminate, otherwise go to Step 2.

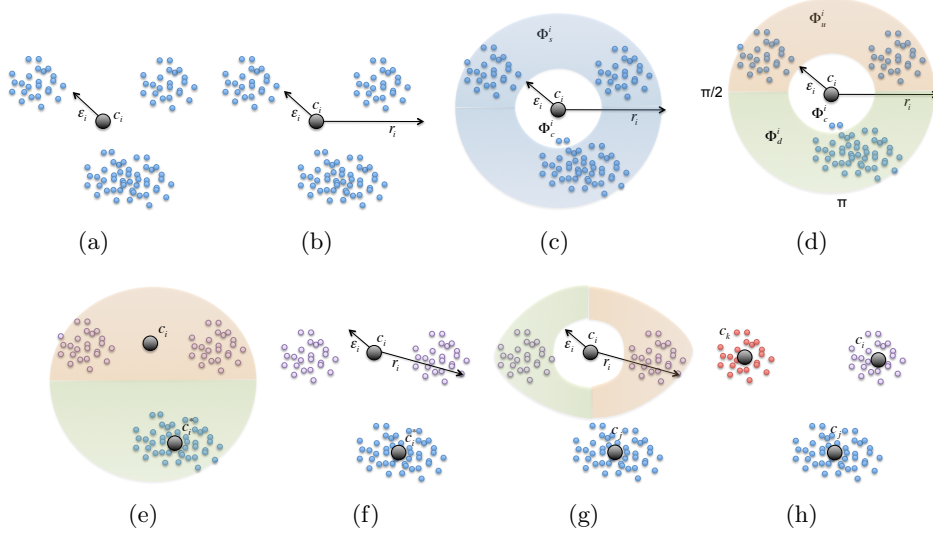


Figure 7.1: The splitting procedure.

Figure 7.1 illustrates the splitting procedure. Figure 7.1(a) presents the center c_i of the dataset and the radius ε_i . The radius r_i given by (7.4) is shown in Figure 7.1(b). Figure 7.1(c) illustrates the sets $\Phi_c^i(\varepsilon_i)$ and $\Phi_s^i(\varepsilon_i)$ defined by (7.2) and (7.3) and colored in white and blue, respectively. In Figure 7.1(d) the set $\Phi_s^i(\varepsilon_i)$ is divided into two disjoint subsets $\Phi_u^i(\varepsilon_i)$ and $\Phi_d^i(\varepsilon_i)$ defined by (7.6) and (7.7). It is obvious that for this set the condition (7.9) is satisfied, therefore, the cluster should be split. In Figure 7.1(e) the sets $\Phi_{cu}^i(\varepsilon_i)$ and $\Phi_{cd}^i(\varepsilon_i)$ are calculated according to (7.10) and (7.11), respectively. Then the set $\Phi_{cu}^i(\varepsilon_i)$ is combined with the set $\Phi_u^i(\varepsilon_i)$ and the set $\Phi_{cd}^i(\varepsilon_i)$ is combined with the set $\Phi_d^i(\varepsilon_i)$ to form two new disjoint clusters. The centers c_i and c_i^* of these clusters computed using (7.13) and (7.15), respectively and are illustrated in Figure 7.1(e). Figures 7.1(f)-7.1(h) illustrate the same procedure for the cluster with the center c_i , where the condition (7.9) is satisfied.

7.2.2 Merging

Assume that the collection of k clusters, $\Omega = \{C_1, \dots, C_k\}$, is given. It may happen that (also after applying the splitting algorithm) some clusters are not well separated. In this subsection an algorithm is designed to merge clusters which are not well separated from each

other.

According to the definition of well separated clusters, two clusters $C_j, C_p \in \Omega$ should be merged if

$$d(c_j, c_p) - (r_j + r_p) < 0. \quad (7.16)$$

These two clusters are merged into one cluster as follows:

$$C_j^* = C_j \cup C_p, \quad (7.17)$$

with the center

$$c_j^* = \frac{1}{|C_j^*|} \sum_{x_i \in C_j^*} x_i. \quad (7.18)$$

For the cluster C_j^* only the index j is used meaning that the cluster C_p joins the cluster C_j . Then the merging algorithm can be summarized as follows:

Algorithm 9. Merging algorithm

Step 0. Input: A collection of k clusters $\Omega = \{C_1, \dots, C_k\}$.

Step 1. Select cluster $C_j \in \Omega$ and calculate its center c_j .

Step 2. Select cluster $C_p \in \Omega$ and calculate its center c_p , where $p \neq j$.

Step 3. If the condition (7.16) is satisfied then go to Step 4, otherwise go to Step 6.

Step 4. Merge clusters C_j and C_p using (7.17). Update the set Ω and set $k := k - 1$.

Step 5. If all clusters C_p , $p = 1, \dots, k$, $p \neq j$ are visited go to Step 6, otherwise go to Step 2.

Step 6. If all clusters $C_j \in \Omega$ are visited terminate, otherwise go to Step 1.

It is obvious that Algorithms 8 and 9 are complementary. In other words, to have stable cluster centers these algorithms should be applied iteratively until the cluster centers become stable. The stability can be checked by monitoring the value of (7.1) for strict decrease. Alternatively, the maximum number of iteration can be predefined in advance. In this research we use the first criterion. Then the split and merge algorithm can be summarized as follows.

Algorithm 10. Split and Merge algorithm

Step 0. Input: A collection of k clusters $\Omega = \{C_1, \dots, C_k\}$ and the ratio $\beta \in (0, 1)$.

Step 1. Apply Algorithm 8 to the collection of clusters Ω . This algorithm will generate a new collection of clusters Ω .

Step 2. Apply Algorithm 9 to the collection of clusters Ω .

Step 3. If the value of the function (7.1) increase then terminate, otherwise go to Step 1.

7.3 SOM initialization algorithm

Usually the set of SOM neurons $\Psi = \{w_1, \dots, w_q\}$ are initialized randomly [91]. This leads the network to converge only to local solutions of Problem (1.6). Furthermore, the SOM suffers from slow convergence. In other words, the number of iterations to learn the input data become large and the neurons may not learn some data points correctly. In this section, a new algorithm is presented where Algorithm 10 is applied to initialize the neurons of the SOM and a modified topology of neurons at the initial points is used.

Algorithm 11. SOM initialization algorithm.

Step 0 (Initialization). A set of m input data vectors $A = \{x_1, \dots, x_m\}$. Set $\Psi := \emptyset$.

Step 1. Calculate the center c^* of the set A , set $w_1 := c^*$ and

$$\Psi := \Psi \cup \{w_1\}.$$

Step 2. Apply Algorithm 10 on Ψ . This algorithm will generate a new set Ψ of neurons.

Step 3. Set the final Ψ as initial neurons of the SOM.

Algorithm 11 ensures that the initial neurons are located in distinct high density area of the input data space which is found by Algorithm 8. Algorithm 9 guarantees that initial neurons are not close to each other. Figure 7.2 illustrates the performance of Algorithm 11.

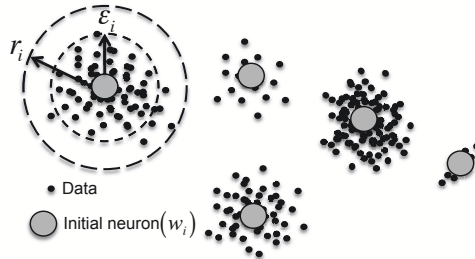


Figure 7.2: Initial neurons after split and merge

7.4 SOM with modified topology

Assume that the set of initial neurons $\Psi = \{w_1, \dots, w_{\hat{q}}\}$ computed by applying Algorithm 11 is given. A set of e number of neurons $u_z \in \mathbb{R}^n$, $z = 1, \dots, e$, using each individual neuron w_i , $i = 1, \dots, \hat{q}$, is generated as follows:

$$g_i = \{u_z | w_i^t - \lambda \varepsilon_i \leq u_z^t \leq w_i^t + \lambda \varepsilon_i\}, \quad (7.19)$$

where $t = 1, \dots, n$, $z = 1, \dots, e$ and $\lambda \geq 1$. It can be observed that all neurons in the set g_i are close to w_i to cover up the dense area represented by neuron w_i . The use of such neurons allows to decrease the value of the quantization error (1.6). In Problem (1.6) the local solution is obtained while none of the activated neurons are far from its mapped data points. Therefore, the set of neurons defined by (7.19) guarantees that the SOM learning process escapes from local solutions of Problem (1.6) and converges to the global solution.

Usually in the SOM topology, all neighborhood neurons are connected to each other in order to spread the adaptation to adjacent neurons. For each pair w_i, w_j , $i, j = 1, \dots, \hat{q}$, $i \neq j$ the following integer number is defined:

$$\hat{r}_{ij} = \left\lceil \frac{d(w_i, w_j)}{\varepsilon_i + \varepsilon_j} \right\rceil. \quad (7.20)$$

Here $\lceil x \rceil$ is a smallest integer greater than or equal to x , called its ceiling. Note that the parameter \hat{r}_{ij} for neurons $u_i, u_j \in g_k$, $i, j = 1, \dots, e$, $i \neq j$, $k = 1, \dots, \hat{q}$ is set to 1. In order to determine the connectivity of neurons the threshold $r_0 \geq 1$ is defined and two neurons w_i, w_j are said to be connected if $\hat{r}_{ij} \leq r_0$. The threshold r_0 is defined for the whole data set. The neurons in the sets g_i , $i = 1, \dots, \hat{q}$ are connected to their parent neuron w_i and to each other. Then the connectivity matrix for the new topology is as follows:

1. $con(i, j) \in \{0, 1\}$, $w_i, w_j \in \Psi$.
2. $con(i, j) \in \{0\}$, $u_i \in g_k$, $u_j \in g_p$, $k \neq p$.
3. $con(i, j) \in \{1\}$, $u_i \in g_k$, $u_j \in g_p$, $k = p$.
4. $con(i, j) \in \{1\}$, $u_i \in g_k$, $w_j \in \Psi$, $k = j$.

$$5. \text{con}(i, j) \in \{0\}, u_i \in g_k, w_j \in \Psi, k \neq j.$$

This new topology guarantees that neurons from one dense area are not connected with those from another dense area and therefore according to the equation (1.5) such neurons do not change each others weight.

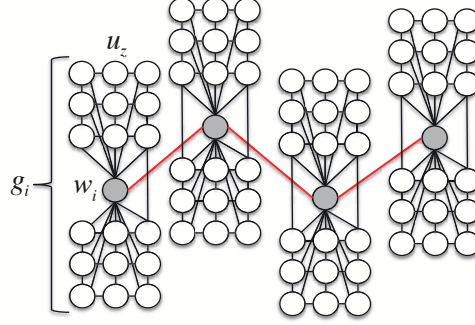


Figure 7.3: Topology of the modified SOM.

The new topology is illustrated in Figure 7.3 where the initial neurons are in gray and the generated neurons around each initial neuron are in white color. There is no any connection between two separate set of generated neurons.

Algorithm 12. SOM topology generation.

Step 0. Given: A set $\Psi = \{w_1, \dots, w_q\}$ of initial neurons and a number $\lambda \geq 1$.

Step 1. Select a w_i and generate g_i using equation (7.19).

Step 2. Connect w_i to all $u_z \in g_i$.

Step 3. If all $w_i \in \Psi$ are visited go to Step 4, otherwise go to Step 1.

Step 4. Select a w_i and connect it to all $w_j \in \Psi$ with $\hat{r}_{ij} < r_0$.

Step 5. If all $w_i \in \Psi$ are visited terminate, otherwise go to Step 4.

The well-known Chain Link data set is used to illustrate the performance of Algorithms 11 and 12. This data set was widely used to demonstrate the performance of the SOM. More information on the Chain Link data set can be found in [118]. The performance of Algorithms 11 and 12 is displayed in Figures 7.4 and 7.5.

In Figure 7.4, the initial neurons are close to the domain of the input data space which helps to avoid any deterioration in convergence of the maps. It should be noted that in this figure λ is set to 1. In Figure 7.5, the results of the neuron initialization is presented with the same setting as in Figure 7.4, except the value of λ which is 1.5. The parameter λ defines

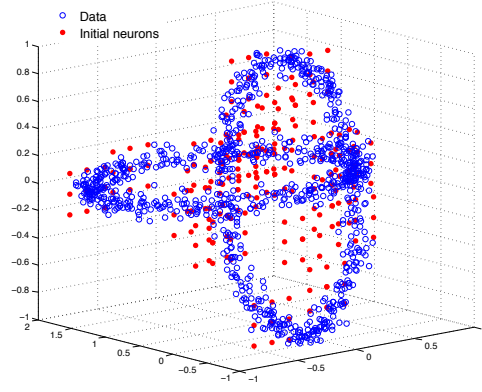


Figure 7.4: Initial neurons generated by Algorithms 11 and 12 with $\lambda = 1$.

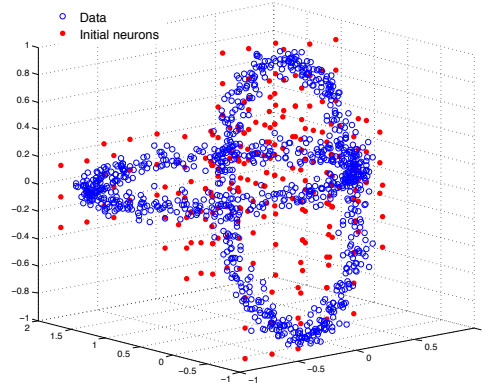


Figure 7.5: Initial neurons generated by Algorithms 11 and 12 with $\lambda = 1.5$.

the proximity of initial neurons with respect to data points. These figures demonstrate that the increase of the values of λ leads to the increase of the distance between neurons and data points.

7.5 The modified SOM algorithm and its implementation

In this section, Algorithm 1 is modified by applying the new initialization algorithm for neurons and the modified SOM topology. The new algorithm can be summarized as follows.

Algorithm 13. Modified SOM algorithm.

Step 0. (Initialization) Initialize the ratio $\beta \in (0, 1)$ in Algorithm 10. Define the maximum number of iterations T , the maximum value β_{max} of the ratio β and the step length β_s . Set the iteration counter $\tau := 0$.

Step 1. (Split and Merge). Apply Algorithm 11 to A for $\beta \rightarrow \beta_{max}$ to generate the set $\Psi = \{w_1, \dots, w_{\hat{q}}\}$ of neurons which minimizes the function f in (7.1). This set contains

initial weights of neurons.

Step 2. (SOM topology). Apply Algorithm 12 to the set Ψ .

Step 3. Select data point $x_i, i = 1, \dots, m$ and find its closest neuron $w_c \in \Psi \cup \{\bigcup_{i=1}^{\hat{q}} g_i\}$, that is

$$c := \underset{j=1, \dots, (\hat{q}(e+1))}{\operatorname{argmin}} \|x_i - w_j\|. \quad (7.21)$$

Step 4. Update the set of neighborhood neurons $w_j \in N_c$ using the following equation:

$$w_j := w_j + \alpha(\tau)h(\tau)(x_i - w_j), \quad (7.22)$$

where

$$N_c = \begin{cases} g_i \cup w_i & \text{if } w_c = u_z \in g_i, \\ g_i \cup w_i \cup \Xi & \text{if } w_c = w_i \in \Psi, \end{cases}$$

subject to

$$\Xi = \{w_j | j \neq i \text{ and } r_{ij} < r_0\}.$$

Step 5. If all input data points are presented to the network go to Step 6, otherwise go to Step 3.

Step 6. Calculate E_τ using (1.6). If $\tau > T$ terminate, otherwise set $\tau := \tau + 1$ and go to Step 3.

Steps of Algorithm 13 are illustrated in Figure 7.6.

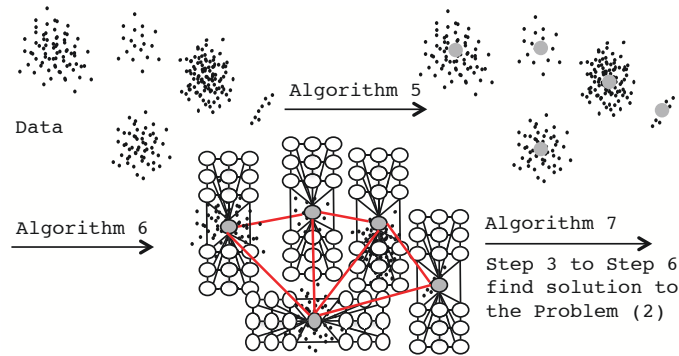


Figure 7.6: The summary of steps in Algorithm 13.

The neighborhood function h in (7.22) of Algorithm 13 is defined differently for sets Ψ

and g_i , $i = 1, \dots, \hat{q}$. More specifically, for neurons $w_i, w_j \in \Psi$

$$h(\tau) = \exp \left(-\frac{r_{ij}^2}{2\sigma(\tau)^2} \right), \quad (7.23)$$

and for neurons $u_z \in g_i$, $i = 1, \dots, \hat{q}$

$$h(\tau) = \exp \left(-\frac{1}{2\sigma(\tau)^2} \right). \quad (7.24)$$

Here

$$\sigma(\tau) = p_0 - \frac{\tau}{T}, \quad p_0 \geq 1. \quad (7.25)$$

The learning rate $\alpha(\tau)$ is defined as

$$\alpha(\tau) = \eta \frac{T - \tau}{\tau}, \quad \eta \geq 1. \quad (7.26)$$

The main difference between Algorithm 13 and the classical SOM is in Steps 1 and 2. The SOM algorithm does not have such steps. Furthermore, in Step 4 the set N_c is modified to improve the approximation of the global solution to the vector quantization problem. Other steps in Algorithm 13 are similar to those in the classical SOM. Note also that unlike the classical SOM, in the proposed algorithm the number of neurons and the radius of the map are not given a priori. They are calculated by the algorithm itself.

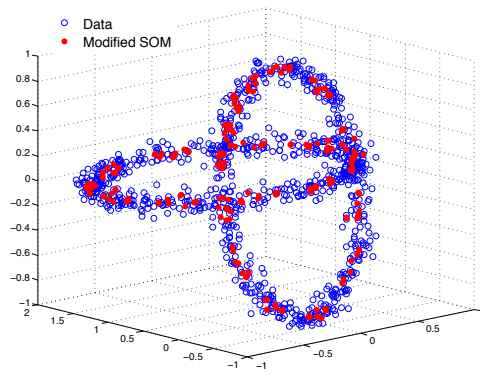


Figure 7.7: The Modified SOM on Chain Link dataset.

The results obtained by Algorithm 13 on Chain Link data set is presented in Figure 7.7. One can notice that the Modified SOM converged to the input data accurately.

7.5.1 Implementation of algorithm 13

In Algorithm 1, weight vectors $w_j, j = 1, \dots, q$ are initialized randomly. The maximum number of iterations T is set between 20 and 40 for small to large data sets, although for large data sets more iterations might be required to obtain stable network over input data. The topology of the SOM network is rectangular [91] with the same number of neurons on each column and row (i.e. $n \times n$). Each interior neuron is connected with 8 neighborhood neurons, however this number is less than 5 for border neurons. Furthermore, the radius of map r is set to 2 for small and 4 for large number of neurons (see Table 7.1).

Table 7.1: Initialization of SOM parameters in Algorithm 1.

Data sets	Input Size	SOM Dim.	r	T
Small	$(A < 10^3)$	10×10	2	20
Medium	$(10^3 \leq A < 10^4)$	15×15	3	30
	$(10^4 \leq A < 0.5 \cdot 10^5)$	20×20	4	40
Large	$(0.5 \cdot 10^5 \leq A < 0.8 \cdot 10^5)$	25×25	3	20
	$(A \geq 0.8 \cdot 10^5)$	20×20	2	20

As it is presented in Table 7.1, the number of neurons, the maximum number of iterations T and the radius r are chosen in increasing order to make the SOM applicable to large data sets. The exception is for the data sets with the the number of points $|A| \geq 0.5 \cdot 10^5$. For these data sets, r and T are smaller comparing to other large data sets. This is done to decrease the computational complexity.

In Step 0 of Algorithm 13, the values of T are set as same as in Table 7.1. The initial value of the parameter β and the values of parameters β_{max} and β_s are defined as follows: $\beta = 0.05, \beta_{max} = 0.6, \beta_s = 0.05$. In Step 2, the parameter λ , which is used in Algorithm 12, is set to 1.5 for small and medium size data sets and to 2.5 for large data sets. In Algorithm 12 the parameter $e = 9$. Therefore the total number of neurons is $|\Psi| \times |g_i| + |\Psi|$. For all data sets the parameter $r_0 = 3$. Finally, the parameters p_0 and η in (7.25) and (7.26) are set to 5 and 1, respectively, for all data sets.

7.6 Results of the evaluation and discussion

To demonstrate the effectiveness of the proposed algorithm, numerical experiments were carried out using a number of real-world data sets. Algorithm 13 was coded in NetBeans IDE

under Java platform and tested on a MAC OSX with 2.7GHz core i7 CPU and 10GB of RAM. 8 data sets, 2 small (Iris and Wine), 2 medium size (TSPLIB1060 and Image Segmentation), 2 large (D15112 and Gamma Telescope) and 2 very large (NE and Pla85900) were used in experiments. A brief description of data sets is presented in Table 10. More details can be found in [14, 119, 135].

The results obtained by the Split and Merge algorithm, which is in Step 1 of Algorithm 13, are presented in Table 7.2. In this table, f_{min} is the value of the problem (7.1), β^* is the value of the parameter β corresponding to f_{min} , $|\Psi|$ is the number of initial neurons and t is the CPU time.

Table 7.2: The results of the Split and Merge algorithm

Data sets	β^*	$ \Psi $	f_{min}	t
Fisher's Iris Plant	0.05	23	4.95×10^1	0.02
Wine	0.05	31	8.62×10^5	0.02
TSPLIB1060	0.05	30	6.43×10^9	0.05
Image Segmentation	0.10	62	3.17×10^7	0.14
D15112	0.10	72	2.53×10^{11}	7.13
Gamma Telescope	0.05	87	2.01×10^8	6.61
NE	0.20	61	3.66×10^2	4.57
Pla85900	0.05	75	3.07×10^{15}	1.09

Let E_{SOM} and E be values of the quantization error obtained by the SOM and the Modified SOM, respectively. Then the improvement P achieved by the Modified SOM in comparison with the result of the SOM is defined as

$$P = \frac{E_{SOM} - E}{E_{SOM}} \cdot 100\%. \quad (7.27)$$

The values of quantization error using equation (1.6) for different iterations and different data sets are presented in Tables 7.3-7.4. From these results one can see that the Modified SOM outperforms SOM in all data sets. The maximum improvement of 42.7% is obtained in Wine data set. The improvement P in Image Segmentation and Iris data sets is 38.3% and 24.8%, respectively. The minimum improvement is obtained in Pla85900 data set which is 4.4%. On other data sets the improvement P is between 6.9% and 14.1%. Note that the Modified SOM starts with a smaller value of E than the SOM algorithm. This is due to the use of the special initialization procedure in the Modified SOM algorithm.

The computational effort used by the Modified SOM is much less than that of the SOM

Table 7.3: Results for small and medium size data sets

<i>iter</i>	<i>E</i>	<i>t</i>	<i>E</i>	<i>t</i>	<i>iter</i>	<i>E</i>	<i>t</i>	<i>E</i>	<i>t</i>
Iris					TSPLIB1060				
	SOM		Modified SOM			SOM		Modified SOM	
2	3.17E+00	0.06	2.96E-01	0.02	2	4.84E+08	0.23	2.47E+03	0.08
4	2.05E+00	0.08	2.76E-01	0.03	4	2.03E+05	0.39	5.60E+03	0.11
6	1.95E+00	0.09	2.29E-01	0.03	6	7.32E+03	0.52	2.03E+03	0.12
8	9.70E-01	0.09	2.23E-01	0.05	10	5.57E+03	0.70	3.66E+02	0.17
10	5.56E-01	0.11	2.22E-01	0.05	14	3.82E+03	0.90	3.19E+02	0.22
12	3.51E-01	0.12	2.22E-01	0.05	18	1.29E+03	1.08	3.17E+02	0.25
14	2.88E-01	0.12	2.22E-01	0.06	22	3.26E+02	1.23	3.17E+02	0.30
16	2.86E-01	0.14	2.22E-01	0.06	25	3.21E+02	1.36	4.67E+02	0.33
20	2.86E-01	0.16	2.15E-01	0.08	30	3.21E+02	1.56	2.99E+02	0.37
Wine					Image Seg.				
	SOM		Modified SOM			SOM		Modified SOM	
2	6.65E+02	0.08	3.45E+01	0.03	2	1.82E+07	0.73	1.01E+02	0.40
4	2.21E+02	0.11	1.33E+01	0.06	4	2.41E+03	1.28	8.01E+01	0.62
6	2.00E+02	0.12	1.19E+01	0.08	6	1.84E+02	1.75	2.90E+01	0.84
8	1.48E+02	0.14	1.11E+01	0.09	10	1.42E+02	2.74	1.89E+01	1.26
10	6.18E+01	0.17	1.12E+01	0.09	14	1.02E+02	3.65	1.75E+01	1.68
12	2.93E+01	0.19	1.12E+01	0.11	18	4.55E+01	4.54	1.74E+01	2.09
14	1.87E+01	0.20	1.12E+01	0.12	22	2.69E+01	5.40	1.74E+01	2.51
16	1.85E+01	0.20	1.54E+01	0.14	25	2.69E+01	6.04	1.75E+01	2.84
20	1.85E+01	0.23	1.06E+01	0.16	30	2.69E+01	7.00	1.66E+01	3.35

in all data sets except Pla85900. The Split and Merge algorithm that initializes the Modified SOM is very efficient and it is not time consuming. The new initialization algorithm which is based on the Split and Merge algorithm speeds up the convergence of the Modified SOM and makes it less time consuming than the SOM. The maximum time reduction by the Modified SOM, comparing with the SOM, was achieved in D15115 and Gamma Telescope data sets. On the other hand the minimum computational time reduction is on two very large data sets: Pla85900 and NE data sets.

The dependence of the CPU time on the number of iterations for the SOM and Modified SOM algorithms using D15112 and Gamma Telescope data sets is given Figure 7.8. Note that the Modified SOM requires more CPU time at the early iterations due to the use of the Split and Merge algorithm for initialization. Once the Modified SOM initialized, it converges much faster than the SOM which is initialized randomly.

Note that the error E shows the quantization quality of the network. However, there is a distortion measurement which can be used to calculate the overall quality of the map. Unlike the quantization error, the distortion measure ξ considers both vector quantization

Table 7.4: Results for large and very large data sets

<i>iter</i>	<i>E</i>	<i>t</i>	<i>E</i>	<i>t</i>	<i>iter</i>	<i>E</i>	<i>t</i>	<i>E</i>	<i>t</i>
D15112					NE				
	SOM		Modified SOM			SOM		Modified SOM	
2	3.65E+04	7.36	1.56E+04	7.77	2	2.02E+07	5.18	3.28E+07	6.47
4	2.52E+08	14.35	5.15E+06	8.32	4	3.32E-01	9.86	8.56E+32	8.18
6	1.60E+04	21.28	1.71E+09	8.86	6	3.01E-01	14.56	3.88E-02	9.89
10	9.59E+03	34.99	1.37E+05	9.97	8	2.70E-01	19.25	2.15E-02	11.54
18	1.57E+03	61.85	4.09E+02	12.20	10	2.56E-01	23.93	1.23E-02	13.20
22	7.26E+02	75.30	3.89E+02	13.29	12	1.94E-01	28.52	1.12E-02	14.88
26	4.28E+02	88.73	3.85E+02	14.37	14	5.41E-02	32.99	1.12E-02	16.55
30	3.80E+02	102.23	3.85E+02	15.46	16	1.16E-02	37.46	1.12E-02	18.24
40	3.80E+02	135.96	3.51E+02	18.14	20	1.12E-02	46.30	1.03E-02	21.51
Gamma Telescope					Pla85900				
	SOM		Modified SOM			SOM		Modified SOM	
2	3.93E+20	10.97	8.28E+01	8.39	2	2.17E+113	4.29	5.81E+112	6.49
4	5.11E+12	21.32	4.08E+01	10.00	4	4.41E+05	8.14	1.43E+05	11.23
6	2.22E+03	31.73	3.46E+01	11.62	6	4.32E+05	11.72	1.99E+88	15.19
10	2.21E+02	53.17	3.15E+01	14.84	8	4.09E+05	16.99	3.09E+04	20.84
18	1.30E+02	95.26	3.02E+01	21.23	10	3.87E+05	18.69	2.74E+04	22.73
22	7.56E+01	116.35	3.00E+01	24.43	12	3.71E+05	22.23	2.63E+04	26.46
26	4.53E+01	137.56	2.99E+01	27.63	14	1.87E+05	25.91	2.61E+04	30.19
30	3.34E+01	158.70	2.99E+01	30.81	16	5.00E+04	29.44	2.61E+04	33.96
40	3.33E+01	210.52	2.86E+01	38.44	20	2.73E+04	36.41	2.61E+04	41.31

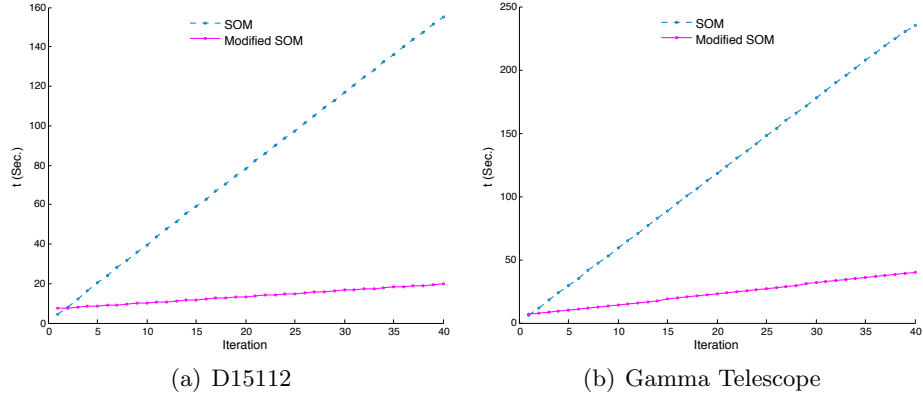


Figure 7.8: SOM vs Modified SOM using CPU time.

and topology preservation of the SOM. The distortion measure is defined as follows [8, 13]:

$$\xi = \sum_{x_i \in A} \sum_{w_j \in \Psi, w_j \neq w_c} h_{cj} \|x_i - w_j\|^2, \quad (7.28)$$

where c is the BMU of x_i and h_{cj} is the value of the neighborhood function h , defined by (7.23), for neurons c and j .

Table 7.5 presents the distortion measure (7.28) and the number of active neurons n_{act} for all data sets. One can see that the distortion error ξ obtained by the Modified SOM is less than that obtained by the SOM in all data sets. This is due to the topology of the

Modified SOM where the neurons from different dense areas are not connected. This prevents deterioration of the network from its optimal value of ξ and E simultaneously.

Table 7.5: Results of distortion measure on all data sets

Dataset	SOM		Modified SOM	
	ξ	n_{act}	ξ	n_{act}
Fisher's Iris Plant	1.25×10^{-6}	69	3.62×10^{-7}	92
Wine	1.68×10^{-2}	72	6.23×10^{-3}	104
TSPLIB1060	1.63×10^{-1}	204	2.11×10^{-2}	258
Image Seg.	3.26×10^{-4}	210	8.73×10^{-5}	490
D15112	6.19×10^{-2}	397	2.69×10^{-2}	710
Gamma Telescope	2.35×10^{-5}	400	1.39×10^{-5}	759
NE	1.66×10^{-1}	375	3.11×10^{-2}	610
Pla85900	7.66×10^{-6}	400	4.97×10^{-6}	636

7.6.1 Comparison with other algorithms

In this subsection the Modified SOM (MSOM) is compared with well-known high dimensional visualization algorithms such as: Growing Grid [67], Growing Neural Gas [66] and Growing Hierarchal SOM [117] using computational results. In the Growing Grid algorithm the number of iterations are set 10 times of that for the Modified SOM. In other algorithms parameters are defined as the same as the similar parameters in the Modified SOM. The CPU time limitation is set to 6 hours. The results for the quantization error E , defined by (1.6), are presented in Table 7.6. In order to compare results by different algorithms the best known value E_{best} of the quantization error and relative errors R_e of algorithms are included in this table. The relative error is computed as

$$R_e = \frac{\bar{E} - E_{best}}{E_{best}} 100,$$

where \bar{E} is the value of the quantization error (1.6) obtained by an algorithm.

Results presented in Table 7.6 demonstrate that the Modified SOM algorithm outperforms all other algorithms in all data sets used in this research. The dash line shows that GHSOM algorithm failed to produce results in large data sets NE and Pla85900. Results also demonstrate that the SOM is quite efficient in data sets with small number of features (2 or 3). In small data sets (Iris Plant and Wine) GHSOM produced good results however it fails as the number of data points increases. Although the GG and GNG algorithms are not

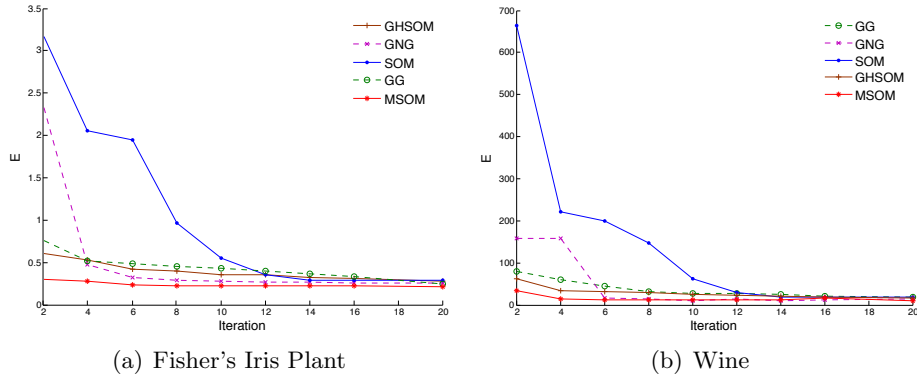
Table 7.6: Comparison of different algorithms.

Data set	E_{best}	R_e				
		GG	GNG	SOM	GHSOM	MSOM
Fishers Iris Plant	2.148×10^{-1}	27.27	18.58	33.33	14.04	0.00
Wine	1.061×10^1	69.65	37.43	74.28	48.51	0.00
TSPLIB1060	2.991×10^2	7.27	25.68	7.36	53.88	0.00
Image Seg.	1.658×10^1	60.34	27.36	62.22	218.59	0.00
D15112	3.510×10^2	39.30	10.09	8.31	135.18	0.00
Gamma Telescope	2.855×10^1	16.67	10.14	16.76	130.99	0.00
NE	1.032×10^{-2}	121.63	66.88	8.23	-	0.00
Pla85900	2.610×10^4	938.81	134.04	4.65	-	0.00

computationally expensive, their results are not satisfactory in comparison with the Modified SOM and in some data sets also in comparison with the classical SOM.

Note that the quantization error, E , is similar to the notion of the *compactness error*, which is used in [56] to express the quality of the clusters obtained.

In Figure 7.9, the values of E obtained by the Modified SOM are compared with those obtained by other algorithms on Iris and Wine data sets. On both data sets the Modified SOM starts with a value of E close to the value of E at the global solution and converges to the optimal value within the given number of iterations. Since the SOM is initialized randomly, it takes more time to converge. The initial grown neurons of the GHSOM are closer to the optimal solution than those generated by GG and GNG algorithms.

Figure 7.9: Comparison of algorithms using E values.

The notion of *distinctness error* is introduced in [56] and can be formulated as:

$$D = \sum_{w_i, w_j \in \Psi, i \neq j} \|w_i - w_j\|. \quad (7.29)$$

The value of D characterizes the distribution of neurons in the input space. Larger value

of D means better distribution of neurons. Results for the distinctness error using all data sets are presented in Table 7.7. In this table the best value D_{best} of D obtained using all five algorithms and also the relative error R_D of results obtained by these algorithms are included. The relative error R_D is computed as follows:

$$R_D = \frac{D_{best} - \bar{D}}{\bar{D}} 100.$$

Here \bar{D} is the value of the distinctness error obtained by an algorithm.

Results from Table 7.7 demonstrate that the Modified SOM outperforms other algorithms in all data sets except the Iris Plant data set, where GHSOM reached the maximum value of D . However the GHSOM algorithm fails in two large data sets NE and Pla85900 and performs poorly in two dimensional data sets TSPLIB1060 and D15112. In other data sets this algorithm performs better than GG, GNG and SOM.

Table 7.7: Results for the distinctness error

Data set	D_{best}	R_D				
		GG	GNG	SOM	GHSOM	MSOM
Fisher's Iris Plant	1.460×10^4	1479.67	857.19	179.23	0.00	43.44
Wine	1.633×10^6	1436.65	120.54	128.91	35.64	0.00
TSPLIB1060	1.989×10^8	533.91	259.51	65.88	1082.92	0.00
Image Seg.	2.085×10^7	2764.31	1132.94	454.36	323.44	0.00
D15112	2.167×10^9	2216.38	384.15	221.13	1341.98	0.00
Gamma Telescope	6.050×10^7	2330.17	368.15	400.61	65.37	0.00
NE	4.924×10^4	7676.32	242.36	113.17	-	0.00
Pla85900	6.144×10^{10}	1499.12	768.24	160.30	-	0.00

In order to demonstrate the time efficiency of the proposed algorithm in comparison with other algorithms the CPU time t required by algorithms is reported in Table 7.8. The GHSOM is not efficient in large data sets. The Modified SOM converges faster than other algorithms on three data sets: Iris, Wine and TSPLIB1060. In all other data sets the GG and GNG are faster than the Modified SOM. Only exception is the Pla85900 data set where the Modified SOM is faster than the GG.

Figure 7.10 displays the visualization of the D15112 data set and clusters in it obtained by algorithms. Clusters are visualized using Voronoi diagrams. One can see that the Modified SOM identifies dense areas and generates more neurons in such areas more efficiently than all other algorithms. The GHSOM algorithm performs better than other algorithms (except the Modified SOM) in identifying dense areas. However it fails to generate more neurons in such

Table 7.8: CPU time required by algorithms

Data set	t				
	GG	GNG	SOM	GHSOM	MSOM
Fisher's Iris Plant	0.44	0.31	0.25	2.06	0.08
Wine	0.43	0.44	0.26	2.87	0.15
TSPLIB1060	1.06	0.62	2.48	62.88	0.48
Image Seg.	3.07	1.39	7.40	303.68	3.57
D15112	16.36	2.68	277.61	10848.36	20.40
Gamma Telescope	29.76	10.21	385.67	18363.03	37.13
NE	16.69	9.77	74.00	-	21.05
Pla85900	62.27	11.58	42.18	-	38.07

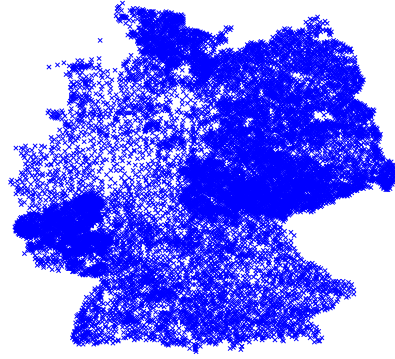
areas in the given number of iterations. Notice that the SOM algorithm tries to distribute neurons uniformly over the data set, which is one of the drawbacks of the SOM [117].

7.6.2 Topology preservation

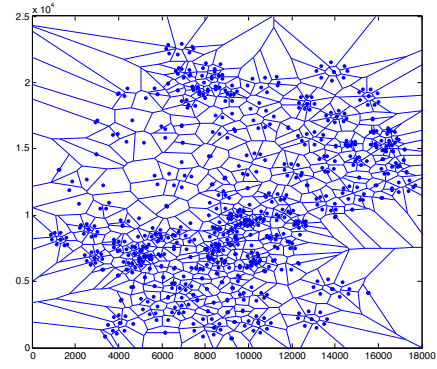
The comparison of topology preservation of the Modified SOM and other algorithms in TSPLIB1060 data set is presented in Figure 7.11. It can be observed that the Modified SOM spreads the neurons more efficiently than other algorithms. This can be proved by the error values, which are reported in Tables 7.6-7.7. If consider white areas (where there is no input data) in Figure 7.11, one can see that the Modified SOM forces neurons to map the data accurately whereas many neurons of other algorithms are located in white areas. The topology of the Modified SOM is defined in order to prevent any attraction of neurons from different dense areas. This decreases the value of the quantization error E .

7.7 Summary

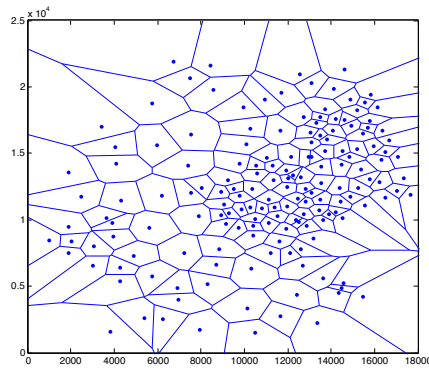
In this chapter, the Modified SOM (MSOM) algorithm is developed to solve large data visualization problems. The MSOM is novel in the sense of initialization algorithm and topology. The proposed algorithm is tested on 8 small to large data sets. Furthermore, the MSOM is compared with SOM-based data visualization algorithms in the sense of computational time and topology preservation.



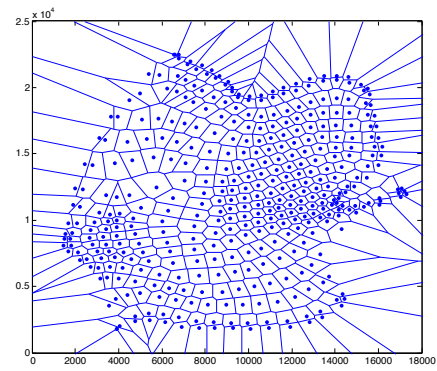
(a) D15112



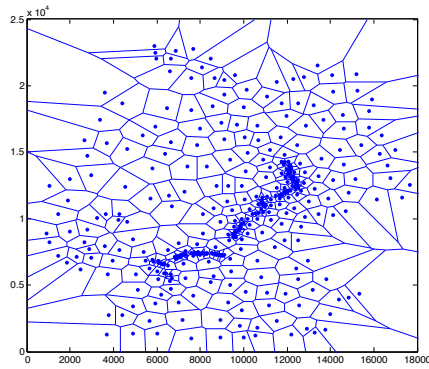
(b) MSOM



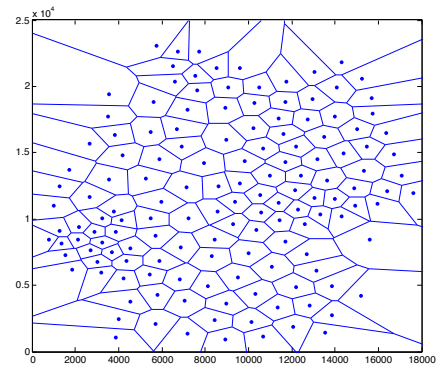
(c) GHSOM



(d) SOM



(e) GNG



(f) GG

Figure 7.10: Visualization of the data set D15112 and clusters.

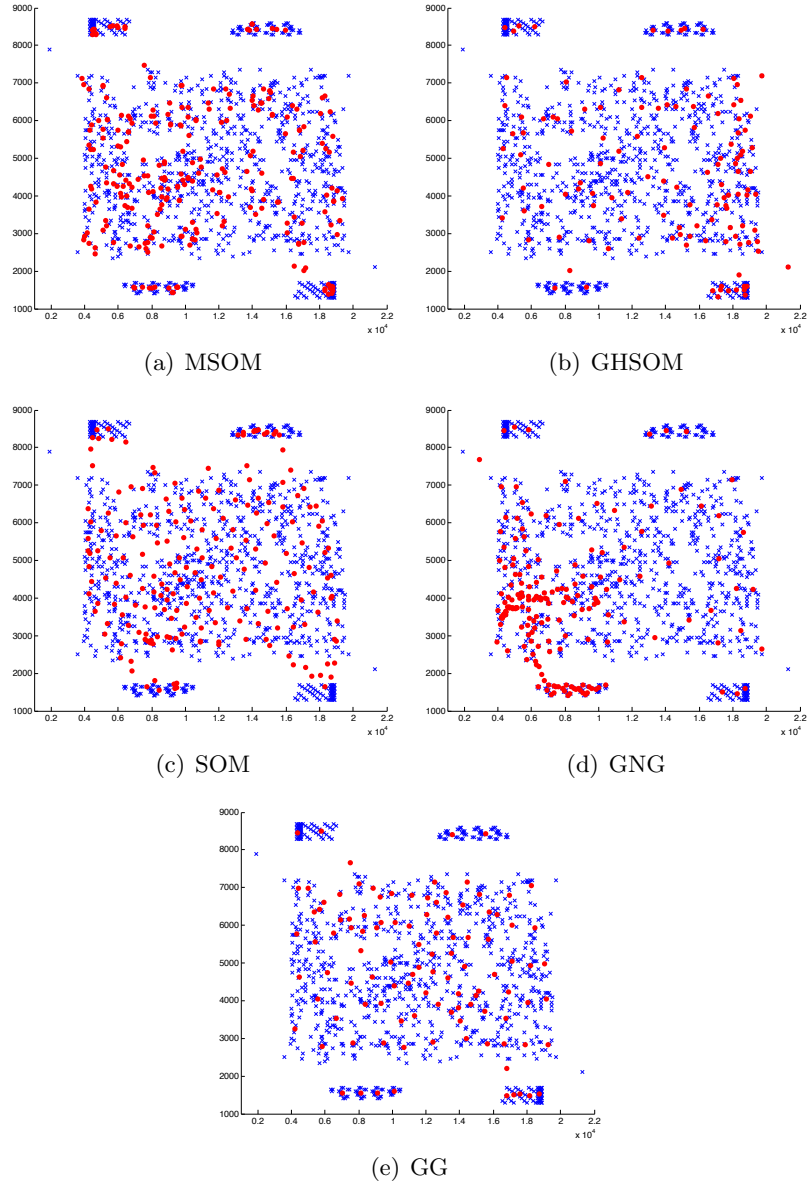


Figure 7.11: Topology preservation of algorithms in TSPLIB1060 data set (data points are in blue and neurons are in red color).

Chapter 8

Convolutional recursive modified SOM for handwritten digits recognition

8.1 Introduction

In this chapter, we present a semi-supervised tool for handwritten digit recognition using a Convolutional Structure of Recursive Modified SOM. The Modified SOM is presented in Chapter 7.

8.2 Selection of parameters in modified SOM algorithm

Algorithm 13 contains the following parameters: the maximum number of iterations T and radius r_0 ; ε and γ for the split and merge, their minimum values ε_0 and γ_0 and maximum values ε_{max} and γ_{max} ; the parameter λ . Among these parameters ε and γ are most important which may significantly affect the convergence of the map. These parameters are used in Algorithm 13 to initialize the neurons of the MSOM.

Using the well-known A1 dataset from [131] we demonstrate how the parameters ε and γ can be chosen. Results are presented in Table 8.1 where the quantization error E is included for different settings of parameters ε and γ . Note that the parameter ε has less

influence on the performance of the MSOM than the parameter γ . However, in all cases these parameters generate a distinct MSOM. Results show that the recommended values for γ are between $\gamma_0 = 5$ and $\gamma_{max} = 6$ and for ε they are between $\varepsilon_0 = 0.1$ and $\varepsilon_{max} = 1.1$. Since the parameter ε has less influence on the performance of the MSOM its values between 0.1 and 0.5 can be chosen to reduce the computational time. However these values can be vary depending on a dataset. Results show that the performance of Algorithm 13 does not strongly depend on the parameter λ .

Table 8.1: The result of E values in A1 dataset using different configurations of parameters ε and γ in Algorithm 13 (s_ε is step length).

γ_{max}	$\varepsilon_0 \rightarrow \varepsilon_{max}, s_\varepsilon = 0.2$							
	0.1	0.3	0.5	0.7	0.9	1.1		
	E						AV.	SD.
5	293	291	302	291	297	295	295	4.07
6	281	287	284	280	285	278	283	3.29
7	325	335	344	328	323	321	329	8.48
AV.	300	304	310	300	302	298		
SD.	22.56	26.66	30.59	25.49	19.53	21.73		

In numerical experiments, we choose $\lambda = 1.5$, the maximum number of iterations $T = 30$ and the coefficient $\eta = 1$. Note that the last two parameters are included also in the classical SOM algorithm.

8.2.1 Comparison with SOM using numerical results

The following data sets are used to compare the performance of the MSOM and the SOM: Path [35], Spiral [35], R15 [138], Aggregation [69], D31 [138], S1 [64], Mopsi Users' locations in Joensuu [65] and A3 [131]. All datasets are 2-dimensional. Results are illustrated in Figure 8.1 where the data points are indicated in blue and the neurons are presented as red circles. One can see that, in all datasets the quality of the map generated by the MSOM is significantly better than that of by the SOM. This is due to the improved initialization process and topology preservation used by the MSOM.

Comparison of E values obtained by the SOM and MSOM on Path, Spiral, and Mopsi datasets is given in Figure 8.2. One can see that in the MSOM the initial position of neurons before training is close to the optimal solution. In Path, Spiral and Mopsi datasets, the SOM obtains optimal values of the problem (1.6) 1.19, 1.50 and 5.69×10^{-3} , respectively. The MSOM improves these results to 0.91, 0.54 and 1.07×10^{-3} .

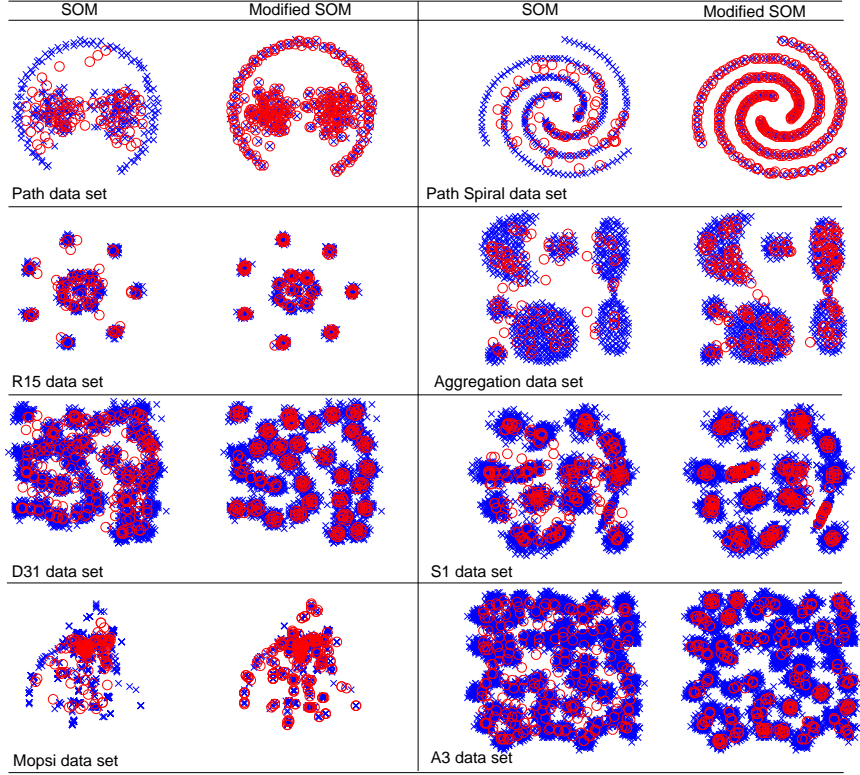


Figure 8.1: The comparison of the SOM and the MSOM.

8.2.2 Complexity comparison with SOM

The time complexity of the SOM is linear with respect to the number of data points. The total number of calculations, ρ of (1.5) in the SOM depends on the parameter r which defines the set of neighborhood neurons to be adapted by (1.5). The complexity of SOM can be formulated as $O(T\rho n)$, where

$$\rho = \sum_{r_0=1}^r r_0^2 + 4r_0 + 4,$$

n is the number of data points and r_0 is defined in Section 7.4. The proposed topology in the MSOM improves the complexity and reduces the number ρ to $\bar{\rho} = e + r_0$ and therefore the complexity of the MSOM is $O((k\gamma + T\bar{\rho})n)$. One can see that the coefficient of n is increasing quadratically in the SOM, whereas it is increasing only linearly in the MSOM.

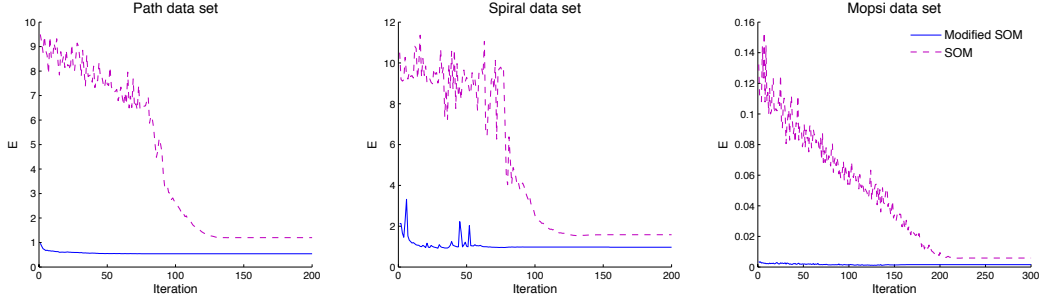


Figure 8.2: Comparison of E values obtained by the SOM and the MSOM.

8.3 Recursive MSOM

The recursive neural network is a combination of neural model and feedback, in order to learn different behaviors of input images by one neural network. First, the idea is to have distinct sets of recursive neural networks, $\mathcal{M}_0, \dots, \mathcal{M}_9$ for training a sequence of images with $0, \dots, 9$ labels, respectively. Therefore,

$$\mathcal{M}_y = \{\Psi_h | h = 1, \dots, \bar{n}^y\}, \quad y = 0, \dots, 9,$$

where Ψ_h is the result of training the set of B input images, P_b , $b = 1, \dots, B$. P_b is a set of vectors indicating image pixels, $P_b = \{x_i | x_i \in \mathbb{R}^3\}$. We propose a recursive form of Algorithm 13 for training process and the scheme of Rec-MSOM is presented in Figure 8.3. As it is presented in this figure, The $P(0)$ is the first incoming image and the feedback of the network at time t is combined with the incoming picture at time $t + 1$ in order to be learned by the network.

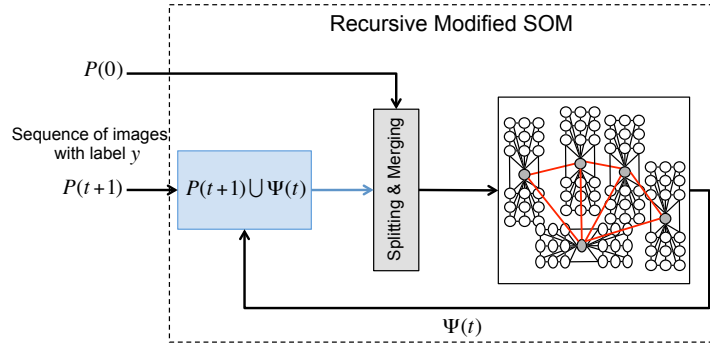


Figure 8.3: Topology of modified SOM.

The Recursive Modified SOM Algorithm is as follows:

Algorithm 14. Recursive Modified SOM algorithm

Step 1. Let have a set of images $\aleph = \{P_1, \dots, P_B\}$. Initialize parameters of the Algorithm

13. Set network $\Psi = \emptyset$.

Step 2. Select a $P_b \in \aleph$.

Step 3. (Training) Apply Algorithm **13** on the set $P_b \cup \Psi$ as input data vectors.

Step 4. (Update Ψ) Set the output neurons of Modified SOM into the parameter Ψ .

Step 5. If all $P_b \in \aleph$ are visited terminate, otherwise go to Step 2.

In Step 3 of Algorithm **14**, the training is done on the union of new input image and the network obtained from the previous sequence. Usually all $P_b \in \aleph$ are from same label, therefore the final set Ψ is a trained network with a label as the input images. Assume that all $P_b \in \aleph$ are from the images of digit with label 0, therefore we add the network Ψ to the set \mathcal{M}_0 ,

$$\mathcal{M}_0 = \mathcal{M}_0 \cup \Psi,$$

and the network Ψ is a recognition tool for images with label 0 in the set \mathcal{M}_0 .

8.3.1 Training

In the training phase, the cardinality of sets $\mathcal{M}_0, \dots, \mathcal{M}_9$ are predefined. Assume that the cardinality of sets \mathcal{M}_y , $y = 0, \dots, 9$ are set as \bar{n}^y , $y = 0, \dots, 9$, therefore, we have \bar{n}^y number of networks, $\Psi_1^y, \dots, \Psi_{\bar{n}^y}^y$, in the set \mathcal{M}_y . The set of training sample images \aleph^y of label y are divided into \bar{n}^y subsets $\aleph_1^y, \dots, \aleph_{\bar{n}^y}^y$ to be learned by the networks $\Psi_1^y, \dots, \Psi_{\bar{n}^y}^y$, in the set \mathcal{M}_y , respectively. It should be noted that

$$\aleph_i^y \cap \aleph_j^y = \emptyset, \quad i \neq j, \quad i, j = 1, \dots, \bar{n}^y.$$

Each subset \aleph_i^y learned by Ψ_i^y , where $i = 1, \dots, \bar{n}^y$, using Algorithm **14**. The training Algorithm is defined as follows:

Algorithm 15. Training algorithm

Step 1. (Initilization) Let have a set of images $\aleph = \{P_1, \dots, P_B\}$. Initialize the values \bar{n}^y and set $\mathcal{M}_y = \emptyset$ for $y = 0, \dots, 9$. Set label $y = 0$.

Step 2. (Selection and devision) Select all images with label y in \aleph and put them in \aleph^y .

Divide the set \aleph^y into \bar{n}^y number of distinct subsets \aleph_j^y , $j = 1, \dots, \bar{n}^y$.

Step 3. (Training) Select a \aleph_j^y and its corresponding network Ψ_j^y then send them as an input to Algorithm 14.

Step 4. The output of Algorithm 14 is a trained network Ψ_j^y , therefore,

$$\mathcal{M}_y = \mathcal{M}_y \cup \Psi_j^y.$$

Step 5. If all \aleph_j^y , $j = 1, \dots, \bar{n}^y$ are visited go to Step 6, otherwise go to Step 3.

Step 6. If $y > 9$ terminate, otherwise $y = y + 1$ and go to Step 2.

After termination of Algorithm 15 the sets \mathcal{M}_y , $y = 0, \dots, 9$ contain trained networks which are used for recognizing unknown images.

Assume that we have selected 13 images with label 0 as is presented in Figure 8.4. We set \bar{n}^0 to 5 and apply Step 2 of Algorithm 15 on this set and divide these images into \bar{n}^0 subsets. The Steps 3 to 5 of Algorithm 15 generate 5 different networks, which are presented in Figure 8.5.

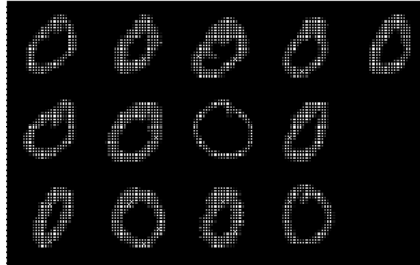


Figure 8.4: A set of images, \aleph^0 , with label 0, which are selected randomly from the training set.

Figure 8.5 presents the set \mathcal{M}_0 , which contains 5 Rec-MSOM networks.

8.3.2 Testing

In this section we propose the testing procedure for handwritten digits recognition. Assume a Modified SOM network Υ as a screen to train the input unknown images using Algorithm 13. Then, the network Υ is compared with all the networks in the sets \mathcal{M}_y , $y = 0, \dots, 9$. The aim is to recognize the label, \bar{y} , of an input image by minimizing the following

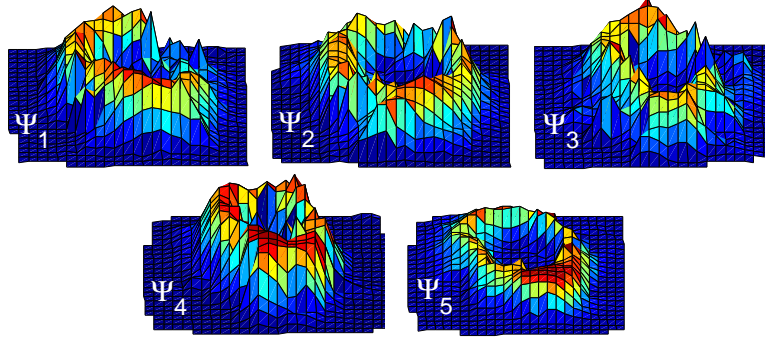


Figure 8.5: The set of modified SOM networks, $\Psi_h \in \mathcal{M}_0$, after training with samples in Figure 8.4. This set is using for recognition of images with 0 label.

equation,

$$\bar{y} = \min_{j=1, \dots, \bar{n}^y} \mathcal{E}(\Psi_j^y, \Upsilon) \quad (8.1)$$

$$\mathcal{E}(\Psi_j^y, \Upsilon) = \sum_{i=1}^{|\Upsilon|} \min_k \|\hat{w}_i - w_k^y\|, \quad w_k^y \in \Psi_j^y,$$

where $y = 0, \dots, 9$. There are many neurons in the the network Υ and also in all the networks in the sets \mathcal{M}_y , $y = 0, \dots, 9$, which are dark and carry no information. Therefore, before using equation (8.1) for image recognition, we introduce a filtering parameter, δ , to remove those neurons in the sets of training networks Ψ_j^y , $j = 1, \dots, \bar{n}^y$, $y = 0, \dots, 9$ and also in the screen network, Υ , which are low in luminance. One can see that, in Figure 8.5 the shape of digit is held by neurons which are in white or light colors. Therefore, we discard the neurons which are dark or low in luminance by setting filtering parameter δ , in advance. The set of neurons that are extracted by filtering from the network Ψ_j^y is calculated as follows:

$$\bar{\Psi}_j^y = \left\{ w_k^y \mid \|w_k^y\| \geq \delta, \quad w_k^y \in \Psi_j^y \right\}, \quad (8.2)$$

where $k = 1, \dots, |\Psi_j^y|$, $j = 1, \dots, \bar{n}^y$, $y = 0, \dots, 9$.

In Figure 8.6 the result of filtering on the network, Ψ , which is on the left with dark background, is presented. The result is a reduced size network, $\bar{\Psi}$, on the right of the Figure 8.6 with a white background.

The set of black dots in Figure 8.6, $\bar{\Psi}$, are those neurons with high luminance in the network Ψ . These neurons should be extracted from the network, Υ , as well as all the sets

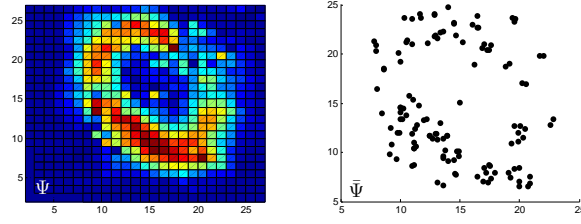


Figure 8.6: The Modified SOM network, Ψ , in the left and the network $\bar{\Psi}$ in the right after applying filtering on Ψ .

of trained networks, \mathcal{M}_y , $y = 0, \dots, 9$, before using equation (8.1) for digit recognition.

8.4 The convolutional structure

In this section a convolutional structure of the the proposed tool for digit recognition is presented. It should be noted that, the screen network, Υ and the sets of training network,

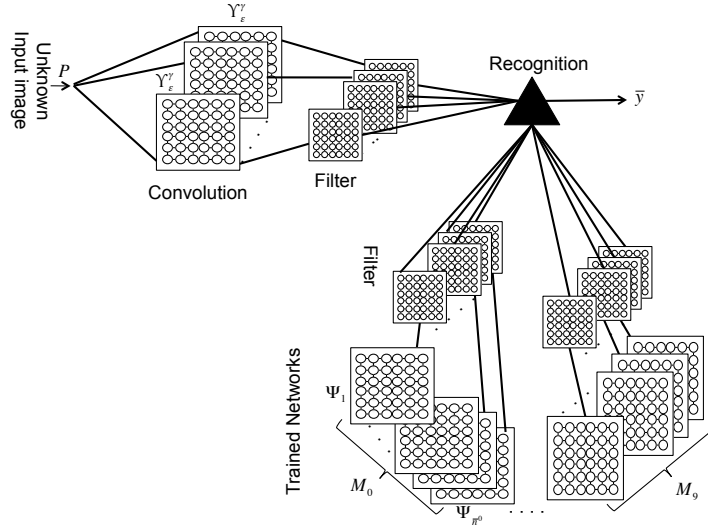


Figure 8.7: The proposed convolutional structure.

\mathcal{M}_y , $y = 0, \dots, 9$, are Modified SOM and follow the topology which is presented in Figure 7.3.

In the convolutional structure , see Figure 8.7, we learn the input image by different settings of the network Υ in the sense of parameters ε and γ in the Algorithm 10. The result is a set of derivative networks of Υ ,

$$\mathcal{I} = \{\Upsilon_\varepsilon^\gamma | 0 < \varepsilon < 1, 1 < \gamma < \gamma_{max}\}.$$

The parameter ε is defined in Section 7.2.1. This parameter is used to split the clusters with a low density center point. One can see that, according to equations (7.2) to (7.7), changing the ε to a high value reduce the number of initial neurons which, consequently, generates a new topology based on Algorithm 12. If we run the Algorithm 10 with a low to a high value of ε , the number of initial neurons will change in a decreasing format in Algorithm 12. In Figure 8.8, we present the results of changing ε and γ of the network $\Upsilon_\varepsilon^\gamma$ from 0.01 to 0.31 and 5 to 6, respectively, while learning an digit image with label 0.

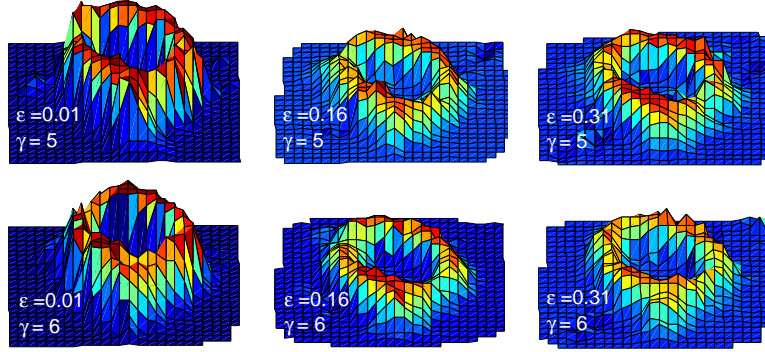


Figure 8.8: Different settings of parameters ε and γ for $\Upsilon_\varepsilon^\gamma$ while learning an input image with label 0.

One can see that, different settings of parameters ε and γ , represent different networks, $\Upsilon_\varepsilon^\gamma$, while learning an input image. In the recognition procedure, the filtering process is applied by equation (8.2) on all networks $\Upsilon_\varepsilon^\gamma \in \mathcal{I}$. The result is a set of filtered networks, $\bar{\mathcal{I}} = \{\bar{\Upsilon}_\varepsilon^\gamma | 0 < \varepsilon < 1, 1 < \gamma < \gamma_{max}\}$, with high luminance neurons. The result of filtered networks, $\bar{\Upsilon}_\varepsilon^\gamma$, obtained from networks in Figure 8.8 is presented in Figure 8.9.

In order to recognize an unknown image with its convolutional networks $\bar{\Upsilon}_\varepsilon^\gamma \in \bar{\mathcal{I}}$, we reformulate equation (8.1) to be applicable for these sets of networks along with the filtered training networks, $\bar{\Psi}_j^y$, $j = 1, \dots, \bar{n}^y$, in the sets $\bar{\mathcal{M}}_y$, $y = 0, \dots, 9$. The reformulation of equation (8.1), for comparing networks in the set $\bar{\mathcal{I}}$ with trained networks in the sets $\bar{\mathcal{M}}_y$, $y = 0, \dots, 9$ to label an unknown input image is as follows:

$$\bar{y} = \min_y \min_{\substack{\varepsilon, \gamma \\ j=1, \dots, \bar{n}^y}} \mathcal{E}(\bar{\Psi}_j^y, \bar{\Upsilon}_\varepsilon^\gamma) \quad (8.3)$$

$$\mathcal{E}(\bar{\Psi}_j^y, \bar{\Upsilon}_\varepsilon^\gamma) = \sum_{i=1}^{|\bar{\Upsilon}_\varepsilon^\gamma|} \min_k \|\hat{w}_i - w_k^y\|, \quad w_k^y \in \bar{\Psi}_j^y, \quad (8.4)$$

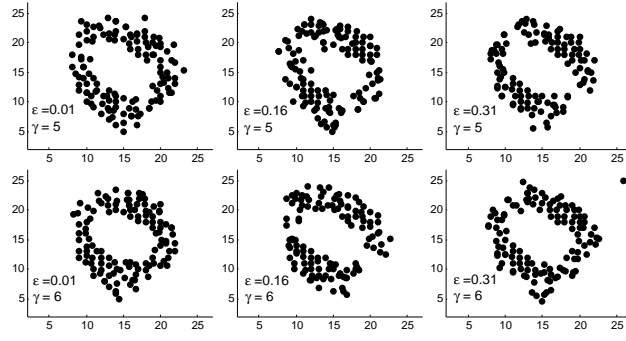


Figure 8.9: The networks, $\bar{\Upsilon}_\varepsilon^\gamma$, with high resolution neurons only. These networks are the results of equation (8.2) on the networks presented in Figure 8.8.

where $y = 0, \dots, 9$.

The handwritten digit recognition algorithm is presented as follows:

Algorithm 16. Handwritten digit recognition algorithm

Step 1. (Initialization) Set of unknown images $\Xi = \{P_1, \dots, P_N\}$. Initialize the parameters ε , γ and get trained networks \mathcal{M}_y for $y = 0, \dots, 9$ by applying Algorithm 15. Set filtering parameter δ in equation (8.2). Set $\varepsilon_t = e_0$ and $\gamma_t = g_0$ as starting values with step lengths s_ε and s_γ , respectively. Set $\min E = \infty$ and $\bar{y} = \text{null}$.

Step 2. Apply filtering equation (8.2) on the sets \mathcal{M}_y to extract high luminance neurons. Set extracted networks to $\bar{\mathcal{M}}_y$ for $y = 0, \dots, 9$, respectively.

Step 3. Select an images P_i in Ξ .

Step 4. (Training) Apply Algorithm 13 on the input image P_i . The result is the network $\Upsilon_{\varepsilon_t}^{\gamma_t}$.

Step 5. Calculate the network $\bar{\Upsilon}_{\varepsilon_t}^{\gamma_t}$ using equation (8.2).

Step 6. (Testing) Select a network $\bar{\Psi}_j^y$ in sets $\bar{\mathcal{M}}_y$ for $y = 0, \dots, 9$ and calculate equation (8.4) with $\bar{\Psi}_j^y$ and $\bar{\Upsilon}_{\varepsilon_t}^{\gamma_t}$ as input networks. If the result of \mathcal{E} is less than $\min E$, then $\min E = \mathcal{E}$ and $\bar{y} = y$.

Step 7. If all networks in trained sets $\bar{\mathcal{M}}_y$ for $y = 0, \dots, 9$ are visited then go to Step 8, otherwise go to Step 6.

Step 8. Set all networks in trained sets $\bar{\mathcal{M}}_y$ for $y = 0, \dots, 9$ as un-visited. If $\varepsilon_t > \varepsilon$ set $\varepsilon_t = e_0$ and go to Step 9, otherwise $\varepsilon_t = \varepsilon_t + s_\varepsilon$ and go to Step 4.

Step 9. If $\gamma_t > \gamma$, set $\gamma_t = g_0$ and go to Step 10, otherwise $\gamma_t = \gamma_t + s_\gamma$ and go to Step 4.

Step 10. Label P_i as \bar{y} . If all $P_i \in \Xi$ are visited terminate, otherwise Set $\varepsilon_t = e_0$, $\gamma_t = g_0$ and go to Step 3.

In Step 1 of Algorithm 16, we initialize the parameters ε_t and γ_t , which are used to produce different networks, $\Upsilon_{\varepsilon_t}^{\gamma_t}$, from the input image. Though, we increment ε_t and γ_t using parameters s_ε and s_γ as step lengths, until reaching ε and γ , respectively. These conditions are holding in Step 8 and 9 of the Algorithm 16. In Step 2, the low luminance neurons are removed from the trained networks, which are the output of Algorithm 15, using equation (8.2). Steps 4 and 5 calculate the network from the input image P_i and make it ready for testing in Step 6. The Step 6 is executed for all trained networks in sets $\bar{\mathcal{M}}_y$ for $y = 0, \dots, 9$.

8.4.1 Implementation of Algorithms 15 and 16

In the training phase, we start by running Algorithm 15. In Step 1 of the Algorithm 15 we initialize the values of \bar{n}^y as in the Table 8.2.

Table 8.2: Initialization of \bar{n}^y in Algorithm 15.

Image label	0	1	2	3	4	5	6	7	8	9
\bar{n}^y	5	7	3	21	1	2	19	18	14	22

In Step 3 of Algorithm 15 we apply Algorithm 14. In step 3 of Algorithm 14 where Algorithm 13 is called, we initialize the parameters of Algorithm 13 as presented in the Table 8.3.

Table 8.3: Initialization of parameters in Algorithms 8 to 13.

Parameter	γ	ε	λ	η	T
Algorithm	10	10	12	13	13
Initial Value	10	0.1	1.5	1	30

The Algorithm 12 generates 9 neurons ($|g_i|$) around all neurons $w_i \in \Psi$. Therefore the total number of neurons is $|\Psi| \times |g_i| + |\Psi|$. Parameter η in equation (7.25) is set to 1 for all trained data.

In the testing phase we initialize the parameters of Algorithm 16 as Table 8.4.

Table 8.4: Initialization of parameters in Algorithms 16.

Parameter	γ	ε	γ_0	ε_0	s_γ	s_ε	δ
Initial Value	6	0.4	5	0.01	1	0.15	0.5

It should be noted that, in the Algorithm 16, the parameters λ , η and T are set to the same values as presented in the Table 8.3.

8.4.2 Application to handwritten digit data set

To demonstrate the effectiveness of the proposed algorithm, the experiments is done on the well known MNIST data set. The Algorithms 1 to 16 have been coded in NetBeans IDE under Java platform. Then the algorithms are tested on a MAC OSX with 2.7GHz core i7 CPU and 10GB of RAM. The MNIST dataset consists of 60000 training samples from approximately 250 writers and 10000 test samples from a disjoint set of 250 other writers. We used the original 784-dimensional dataset which resembles 28x28 pixel grey level images of the handwritten digits. The first 45 digits of the test samples are presented in Figure 8.10.

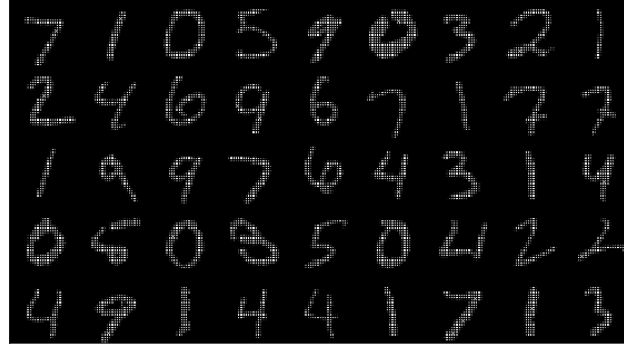


Figure 8.10: The first 45 digits of the test samples.

To validate the proposed method in the environments where there is not many training data available, in the training phase, we select only 40 image samples for each class of 0 to 9, randomly, from the 60000 training samples. Therefore, 400 training samples used to train the sets \mathcal{M}_y , $y = 0, \dots, 9$. 100 training samples from the 400 training samples is presented in the Figure 8.11.

One can see that the total number of networks that are trained for digits recognition is,

$$U = \sum_{y=0}^9 \bar{n}^y.$$

The parameters \bar{n}^y , $y = 0, \dots, 9$ are set according to the values, which are presented in the Table 8.2. The value \bar{n} plays an important role in classification accuracy. Assume that, we have many misclassifications between digit 9 and 4. Then we set the values \bar{n}^9 and \bar{n}^4 in a such way to increase $|\bar{n}^9 - \bar{n}^4|$. Though, in the Table 8.2, the parameter \bar{n}^y for image with label 9 is set to 22 while this parameter for the image 4 is set to 1. This significant difference

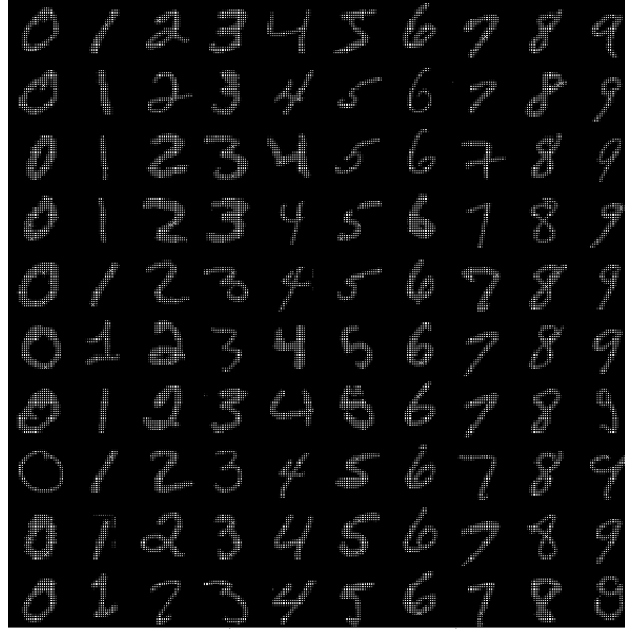


Figure 8.11: 100 training samples from the 400 training samples, which have been used to train the sets \mathcal{M}_y , $y = 0, \dots, 9$.

in \bar{n} leads to improvement in the misclassifications where image 9 misclassified as 4 and vice versa. The same criteria is hold for images with label 7 and 1, 3 and 8 and also for images with label 2 and 7. We have presented the networks Ψ_1^y , $y = 0, \dots, 9$ in Figure 8.12.

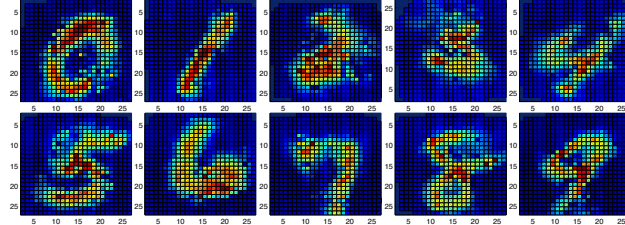


Figure 8.12: The networks $\Psi_1^y \in \mathcal{M}_y$, $y = 0, \dots, 9$ after running Algorithm 15 on training samples.

Recently, a number of handwritten digits recognition methods based on SOM have been proposed. Comparing the existing methods, the best accuracy of %98.73 on MNIST dataset is reported by [33]. In our experiment on MNIST dataset, the accuracy of %99.03 is obtained by using the Convolutional Rec-MSOM on the 10000 test samples. Therefore, only 97 samples were misclassified. The proposed method in this research boosted up the classification results up to %23.62,

$$(1.27 - 0.97)/1.27 \times 100\%,$$

compared with the method proposed by [33]. The comparative results of Convolutional Rec-MSOM vs well known SOM-based handwritten digits recognition methods on MNIST dataset are presented in the Table 8.5.

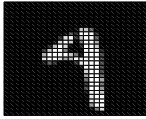
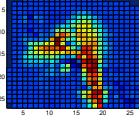
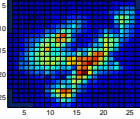

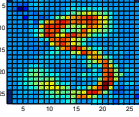
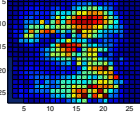
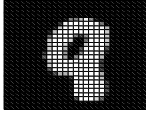
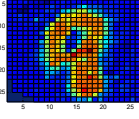
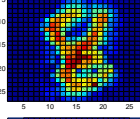
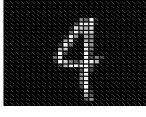
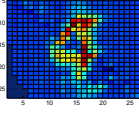
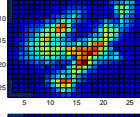
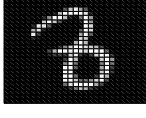
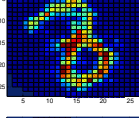
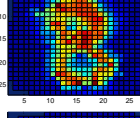

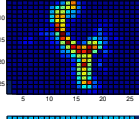
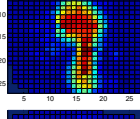

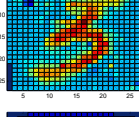
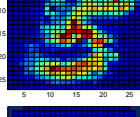
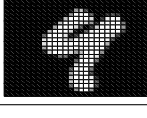
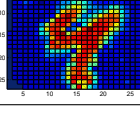
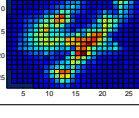
Table 8.5: Comparison of accuracy obtained by SOM-based handwritten digits recognition methods on MNIST dataset.

Reference	Method	Accuracy (%)
[127]	BTASOM	85.24
[80]	GP+ASSOM+TM	93.20
[109]	H ² SOM	94.60
[110]	H ² SOM	95.80
[159]	AOSSOM	97.30
[160]	LMSOM	97.30
[161]	LLOM	97.74
This Chapter	CR-SOM	97.75
[33]	CNN-SOM	98.73
This Chapter	CR-MSOM	99.03

An up to date results on MNIST dataset by a Hybrid CNN-SVM method is presented by [108]. The authors in [108] claim that there are 25 samples included in the test samples which are the most challenging ones. However, their proposed method can predict only 6 out of 25 samples, correctly. We have presented our results on these samples in the Tables 8.6 to Table 8.8. In these tables, the prediction results by the Algorithm 16 is indicated with bold numbers. The screen network, Υ , with its parameters ε_t and γ_t is presented in columns 5, 6 and 7, respectively. This network is the result of training the shape in column 2 using Algorithm 13 in the Step 5 of Algorithm 16. Then the network Υ is compared with all the networks Ψ_j^y in the sets \mathcal{M}_y , $y = 0, \dots, 9$, through the Steps 6 to 9 in the Algorithm 16. The trained network Ψ , which is presented in column 8, is the solution to the minimization problem (8.3). From the results presented in the Tables 8.6 to Table 8.8, one can see that, the proposed method in this research can predict 24 samples correctly, thus outperforms the previous results, significantly. The only misclassification is on the sample with the id-number 8409, which is predicted as $8 \rightarrow 6$ (see Table 8.8).

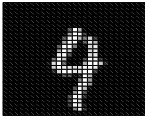
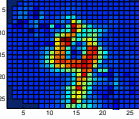
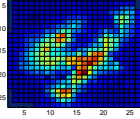

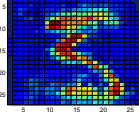
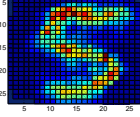
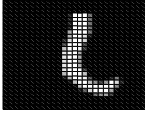
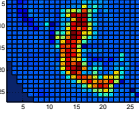
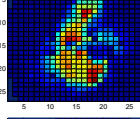
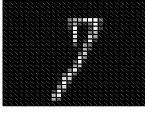
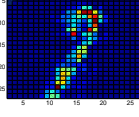
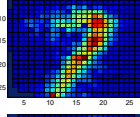
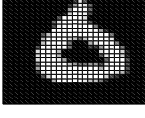
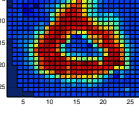
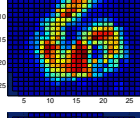

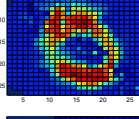
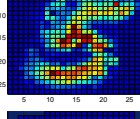

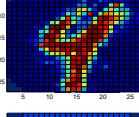
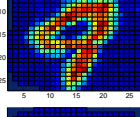
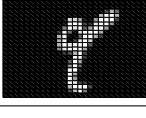
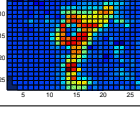
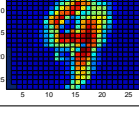
The 97 misclassified samples by using the proposed method in this research are presented in Figure 8.13. In Figure 8.13, the ID of each sample is printed at the top and the prediction is presented as **label**→**prediction** at the bottom. The results in Figure 8.13 show that there

Table 8.6: The most challenging samples that reported in [108].

ID	Shape	Label	Pred.	Υ	ε_t	γ_t	Ψ
741		4	4		0.16	5	
939		3	3		0.31	6	
948		8	8		0.31	5	
1243		4	4		0.16	5	
1879		8	8		0.01	6	
1902		9	9		0.01	6	
2036		5	5		0.31	6	
2131		4	4		0.16	5	

are 6 digits with label 1 which have been predicted as a sample with label 9. These digits are a thick written number with label 1 which may caused the machine misclassifications. There are also 5 misclassification as $3 \rightarrow 8$. By analysis such errors, we found that, most of the errors are due to rotation, misalignment and broken structure in the handwritten numerals. For example, the samples with the ID 342, 4079, 4815 and 6167 are rotated which is caused by the people with bad habits in handwritings. The samples with the ID 620, 1045, 2381, 3845 and 6652 are related to the errors which are even hard for a human to predict without displaying their labels. Some of other errors are introduced by scanning procedure, like samples with ID 3061, 9630 and 9694. The confusion matrix of these misclassifications is presented in the Table 8.9.

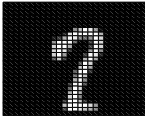
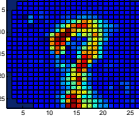
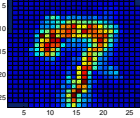

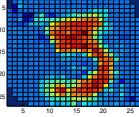
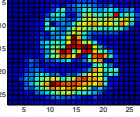

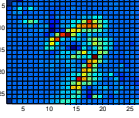
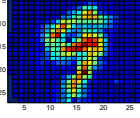

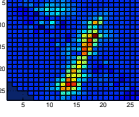
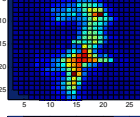

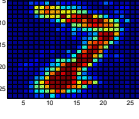
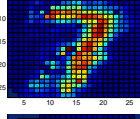

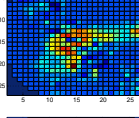
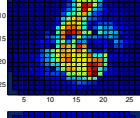

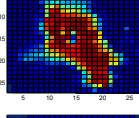
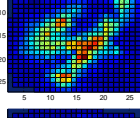
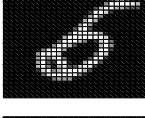
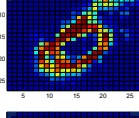
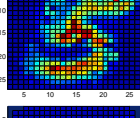
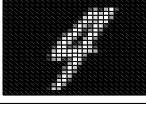
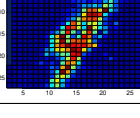
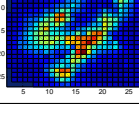
Table 8.7: The most challenging samples that reported in [108].

ID	Shape	Label	Pred.	Υ	ε_t	γ_t	Ψ
2448		4	4		0.31	5	
2598		5	5		0.16	5	
2655		6	6		0.31	5	
3226		7	7		0.01	5	
3423		6	6		0.16	5	
3559		5	5		0.16	6	
3870		9	9		0.01	6	
4762		9	9		0.31	5	

Considering the misclassifications among 10 classes of images, which is presented in the Table 8.9, the max number of misclassification is 25 on the images with label 7. We present the set \mathcal{M}_7 in the Figure 8.14.

The set of networks presented in the Figure 8.14 are used to classify images with label 7. By comparing these networks with the digits with label 7 in the Figure 8.13, one can see that the most errors are on the images which are written in a thick shape, like images with ID 1207, 6742, 7903, 7904 and 7928. The same criteria is true for the images with the label 1 in the Figure 8.13. There are 1028 images with label 7 included in the 10000 test samples. It should be noted that by learning only 40 images with label 7, the set \mathcal{M}_7 misclassified 25 samples which is about %2.4 of the total images with the label 7. This denotes the superiority

Table 8.8: The most challenging samples that reported in [108].

ID	Shape	Label	Pred.	Υ	ε_t	γ_t	Ψ
5655		7	7		0.16	5	
5938		5	5		0.31	5	
6572		9	9		0.31	6	
6577		7	7		0.16	5	
8317		7	7		0.01	5	
8409		8	6		0.31	5	
8528		4	4		0.01	5	
9730		5	5		0.01	5	
9793		4	4		0.01	6	

of the proposed classifier where the number of training samples are limited.

8.4.3 Reliability of the proposed method

To demonstrate the performance of the proposed method in the sense of reliability, we compare our method with the SOM-based method, CNN-SOM, in [33]. We present the reliability based on the same rejection procedure, which is introduced by [33]. If we consider the solution and the second best solution to the problem (8.4) as e_1^{max} and e_2^{max} , respectively,



Figure 8.13: The 97 misclassified samples from the 10000 test samples.

then a test sample is rejected if the value of the following equation,

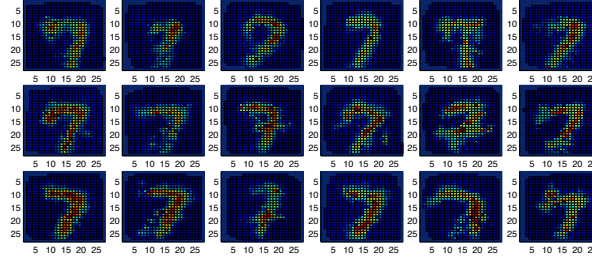
$$\bar{\mathcal{E}} = \frac{|e_1^{max} - e_2^{max}|}{2}, \quad (8.5)$$

is less than a predefined threshold. In the Figure 8.15, the error and rejection rates on the MNIST test dataset is presented, furthermore, the reliability of the proposed method is compared with the results that are reported by [33].

From the results, which are presented in the Figure 8.15, one can see that the reliability of the Convolutional Rec-MSOM reached to a high recognition rate of %99.59 with a low rejection rate of %6.34. These results outperform the previous results, which are obtained by the CNN-SOM [33], where the recognition rate is reported as %99.55 while the rejection rate is %7.45. This denotes the effectiveness of the proposed method and its dominance over the existing SOM-based methods, which are proposed for hand written digits recognition.

Table 8.9: Confusion matrix.

Truth	Prediction									
	0	1	2	3	4	5	6	7	8	9
0			1				3	1	2	
1				4			1	1	1	6
2										
3			2	1				2	6	4
4										
5										
6	3	1	1			3			3	1
7		11	2	1	1				1	9
8	2		1	1			3			4
9	2	1		3			1	3	4	

Figure 8.14: The set \mathcal{M}_7 used to classify images with label 7.

8.5 Summary

The problems of handwritten digits recognition are discussed in this chapter. A convolutional structure based on the MSOM is proposed to cope with diversity of shapes and styles in handwritten digits data set. The proposed model is tested on the well-known MNIST data set and its accuracy is compared with existing SOM-based approaches for handwritten digits recognition.

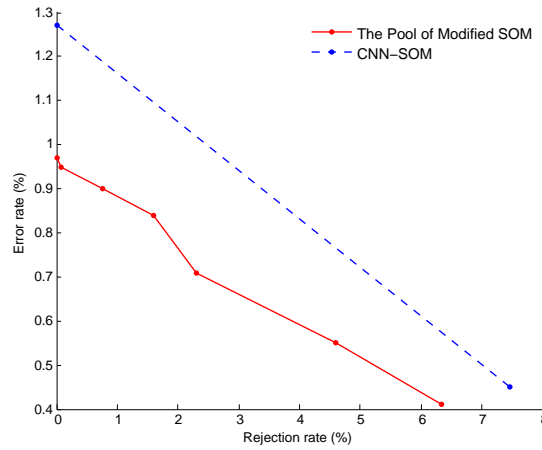


Figure 8.15: The reliability of CR-MSOM vs CNN-SOM [33] on 10000 test samples.

Conclusion and future work

In this thesis, we presented different optimization models for the hard clustering problems. We demonstrated that the use of the nonsmooth nonconvex optimization model has some advantages over other optimization models of the clustering problems. The nonsmooth nonconvex optimization model allows one to easily apply different similarity measures in clustering problems. In particular, one can use similarity measures based on the L_1 , the squared L_2 and L_∞ norms. Applying this approach algorithms for solving clustering problems are designed. The proposed clustering algorithms are based on the incremental approach and involves a special procedure for finding starting points for cluster centers. Starting points are found by minimizing the so-called auxiliary cluster function. Three different algorithms are introduced to minimize both the auxiliary and cluster functions. These algorithms include the k -means type heuristic algorithm, the discrete gradient method of the nonsmooth optimization and the algorithm based on the smoothing of the cluster functions.

The proposed algorithms are tested on twelve real world data sets using different similarity measures. These results demonstrate that these algorithms are efficient in finding global or near global solutions to clustering problems with different similarity measures. The comparison of the algorithm with the use of the discrete gradient method with a number of other clustering algorithms is also presented. This comparison demonstrate that the proposed algorithm is more accurate than other algorithms in many data sets. In some data sets the performance of the proposed algorithm, the Hartigan and the k -means++ algorithms are quite similar.

Data visualization is another crucial task in mining and pattern recognition. Among the existing data visualization algorithms, the Self Organizing Maps has been shown to be a promising tool. However, the quality of data visualization depends on the SOM learning

algorithm. In this thesis, we develop a modified learning algorithm for the SOM. The aim is to propose a learning algorithm which restricts the neighborhood adaptations to only those neurons that are not far from the best matching unit in the n -dimensional space. Therefore, we introduced an adaptive constraint parameter. This parameter is a decreasing function with respect to iterations of the SOM learning process. The adaptive constraint parameter selected as linear, hyperbolic and sigmoid functions. The experiments on eight real-world data sets demonstrate the superiority of the proposed algorithm over the SOM in the sense of accuracy. Moreover, the results demonstrate that the CSOM requires less computational efforts than the SOM. The results show the superiority of CSOM over similar topology preservation algorithms in large data sets.

In this research, a modified version of the Self Organizing Maps is developed to the problems in the large high dimensional data sets visualization. The aim is to design a new more efficient version of the SOM by using a new algorithm for initialization of neurons and a new topology which restricts the neighborhood adaptations to only those neurons that are not in different dense areas. For this purpose the Split and Merge algorithm is introduced to generate neurons. This algorithm is a part of the initialization algorithm in the Modified SOM and the numerical results show that the initialization algorithm generates neurons close to the optimal solution. A topology is introduced to generate neurons in high dense areas of input data space and to prevent attraction of neurons from different such areas. The results show that this restriction reduces the quantization and distortion errors. Numerical results on eight data sets are presented to demonstrate the efficiency of the Modified SOM and compare it with other four algorithms including the classical SOM. These results show that the Modified SOM is more accurate than all other algorithms. Results for the distinctness error confirm that the Modified SOM is able to distribute neurons over the input data space more efficiently than any other algorithm. Moreover visualization of clusters obtained by different algorithms show that the proposed algorithm is more efficient in identifying dense areas and generating neurons in such areas. Although the Modified SOM is faster than the classical SOM on some data sets it requires more computational time than other three algorithms used in comparison.

We developed a Convolutional Recursive Modified SOM for the problem of handwrit-

ten digits recognition. First, we presented a modified version of the Self Organizing Maps. The aim was to propose an initialization algorithm and a new topology which restricts the neighborhood adaptations to only those neurons that are not in different dense areas. We introduced split and merge algorithm to initialize the neurons optimally. The experiments show that the initialization is close to optimal solution. We presented a topology for SOM to generate neurons in high dense areas of input data space and not to connect neurons which are in separate dense areas. The experiments show that this restriction reduces the quantization error, E . Furthermore, demonstrate the improvements in topology preservation of the Modified SOM. The new topology also reduced the complexity of neighborhood adaptations from quadratic to a linear one, in comparison with the conventional SOM. We proposed a recursive form of Modified SOM for training process, in order to learn the diversity of shapes and styles of incoming images. Finally, a convolutional structure introduced to label the unknown handwritten digits images. The experiments on the well known data set, MNIST, demonstrate the superiority of the proposed algorithm over the existing SOM-based methods in the sense of accuracy.

Future work

As the data sets become larger and more diverse, new algorithms and modifications of existing algorithms are required to solve clustering problems in such data sets. With the increasing availability of very large, dynamic, and time-varying data sets and especially massive datasets from various sources, most clustering algorithms are not capable of generating satisfactory results. Development of new and extension of algorithms proposed in this thesis for solving clustering problems in such data sets will be a part of future research.

In many applications, clustering is an off-line task. In such applications the CPU time required by clustering algorithms is not very crucial. However, there some other applications where decisions have to be made in real time. The problem of real time clustering is a challenging problem, which refers to solving clustering problems in a given period of time. Developing real time clustering algorithms based on the clustering algorithms, developed in this thesis, is a part of the future work.

The cluster analysis algorithms such as k -means and fuzzy c -means are known to be

sensitive to noisy data or data sets with outliers. The performance of introduced clustering algorithms in this thesis on noisy data and their modifications to the problems in data sets with outliers are other parts of future work.

Bibliography

- [1] C. Aggarwal. A framework for clustering massive-domain data streams. In *IEEE International Conference on Data Engineering*, pages 102–113. IEEE, 2009.
- [2] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *ICDT '01 Proceedings of the 8th International Conference on Database Theory*, pages 420 – 434, 2001.
- [3] K. S. Al-Sultan. A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451, 1995.
- [4] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE transactions on neural networks*, 11(3):601–14, 2000.
- [5] N. Alex, A. Hasenfuss, and B. Hammer. Patch clustering for massive data sets. *Neurocomputing*, 72(7-9):1455–1469, 2009.
- [6] K. Appiah, A. Hunter, M. Hobden, N. Priestley, P. Hobden, and C. Pettit. A binary self-organizing map and its FPGA implementation. In *2009 International Joint Conference on Neural Networks*, pages 164–171. IEEE, June 2009.
- [7] K. Appiah, A. Hunter, P. Dickinson, and H. Meng. Implementation and applications of tri-state self-organizing maps on FPGA. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(8):1150–1160, August 2012.
- [8] N. Arous. On the search of organization measures for a kohonen map case study: Speech signal recognition. *International Journal of Digital Content Technology and its Applications*, 4(3):75–84, 2010.

- [9] D. Arthur and S. Vassilvitskii. k -means++: the advantages of careful seeding. In Bansal, N. and Pruhs, K. and Stein, C., editor, *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. SIAM, 2007.
- [10] A. Astorino and A. Fuduli. Nonsmooth optimization techniques for semi-supervised classification. *IEEE Trans. Pattern Anal. Machine Intell.*, 29:2135–2142, 2007.
- [11] A. Astorino and M. Gaudioso. Polyhedral separability through successive LP. *Journal of Optimization Theory and Applications*, 112(2):265–293, 2002.
- [12] A. Astorino, A. Fuduli, and E. Gorgone. Non-smoothness in classification problems. *Optimization Methods and Software*, 23:675–688, 2008.
- [13] T. Ayadi, T. M. Hamdani, and A. M. Alimi. MIGSOM: Multilevel interior growing self-organizing maps for high dimensional data clustering. *Neural Processing Letters*, 36:235–256, 2012.
- [14] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL <http://archive.ics.uci.edu/ml>.
- [15] A. M. Bagirov, A. M. Rubinov, N. Soukhoroukova, and J. Yearwood. Unsupervised and supervised data classification via nonsmooth and global optimization. *Top*, 11(1): 1–93, 2003.
- [16] A. M. Bagirov, B. Karasozen, and M. Sezer. Discrete gradient method: Derivative-free method for nonsmooth optimization. *Journal of Optimization Theory and Applications*, 137:317–334, 2008.
- [17] A. M. Bagirov, J. Ugon, and D. Webb. Fast modified global k -means algorithm for incremental cluster construction. *Pattern Recognition*, 44(4):866–876, 2011.
- [18] A. M. Bagirov, A. Al Nuaimat, and N. Sultanova. Hyperbolic smoothing function method for minimax problems. *Optimization*, 62(6):759–782, 2013.
- [19] A. M. Bagirov. Modified global k -means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition*, 41(10):3192–3199, 2008.

- [20] A. M. Bagirov and J. Ugon. Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization. *Journal of Global Optimization*, 35:163–195, 2006.
- [21] A. M. Bagirov and J. Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2): 578–596, 2006.
- [22] A. M. Bagirov, A. M. Rubinov, and J. Yearwood. A global optimization approach to classification. *Optimization and Engineering*, 3(2):129–155, 2002.
- [23] L. B. Batista, H. M. Gomes, and R. F. Herbster. Application of growing hierarchical self-organizing map in handwritten digit recognition. In *Proceedings of 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 1539–1545, 2003.
- [24] E. Berglund and J. Sitte. The parameterless self-organizing map algorithm. *IEEE Transactions on Neural Networks*, 17(2):305–316, 2006.
- [25] E. Berglund. Improved PLSOM algorithm. *Applied Intelligence*, 32(1):122–130, 2010.
- [26] P. Berkhin. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 1–56, 2006.
- [27] H. H. Bock. Clustering and neural networks. In *Advances in Data Science and Classification*, pages 265–277. Springer-Verlag Berlin and Heidelberg, 1998.
- [28] M. Bogdan and W. Rosenstiel. Detection of cluster in self-organizing maps for controlling a prostheses using nerve signals. In *ESANN’01*, pages 131–136, 2001.
- [29] D. E. Brown and C. L. Huntley. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25(4):401–412, 1992.
- [30] D. Brugger, M. Bogdan, and W. Rosenstiel. Automatic cluster detection in kohonen’s som. *IEEE transactions on neural networks*, 19(3):442–59, 2008.
- [31] J. W. Carmichael and P. H. A. Sneath. Taxometric maps. *Systematic Zoology*, 18: 402–415, 1969.

- [32] E. Carrizosa and D. Romero Morales. Supervised classification and mathematical optimization. *Computers and Operations Research*, 40:150–165, 2013.
- [33] H. Cecotti and A. Belaíd. A new rejection strategy for convolutional neural network by adaptive topology. In M. Kurzyński, E. Puchala, M. Woźniak, and A. Zolnierrek, editors, *Computer Recognition Systems*, volume 30 of *Advances in Soft Computing*, pages 129–136. Springer Berlin Heidelberg, 2005.
- [34] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k -means clustering algorithm. *Expert Systems with Applications*, 40:200–210, 2013.
- [35] H. Chang and D.-Y. Yeung. Robust path-based spectral clustering. *Pattern Recogn.*, 41(1):191–203, 2008.
- [36] B. B. Chaudhuri and G. Garai. Grid clustering with genetic algorithm and tabu search process. *Journal of Pattern Recognition Research*, 4(1):152–168, 2009.
- [37] N. Chen, B. Ribeiro, A. Vieira, and A. Chen. Clustering and visualization of bankruptcy trajectory using self-organizing map. *Expert Systems with Applications*, 40(1):385–393, January 2013.
- [38] X. Chen, X. Liu, and Y. Jia. Discriminative structure selection method of gaussian mixture models with its application to handwritten digit recognition. *Neurocomputing*, 74(6):954–961, 2011.
- [39] C.-H. Cheng. A branch and bound clustering algorithm. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(5):895–898, 1995.
- [40] K.-O. Cheng, N.-F. Law, W.-C. Siu, and A. W.-C. Liew. Identification of coherent patterns in gene expression data using an efficient biclustering algorithm and parallel coordinate visualization. *BMC bioinformatics*, 9(1):210, 2008.
- [41] S.-S. Cheng, H.-C. Fu, and H.-M. Wang. Model-based clustering by probabilistic self-organizing maps. *IEEE transactions on neural networks*, 20(5):805–26, 2009.

- [42] Y.-M. Cheung and L.-T. Law. Rival-model penalized self-organizing map. *IEEE transactions on neural networks: a publication of the IEEE Neural Networks Council*, 18(1): 289–95, 2007.
- [43] S.-C. Chi and C. C. Yang. Integration of ant colony SOM and k -means for clustering analysis. In *Proceedings of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part I*, KES’06, pages 1–8, 2006.
- [44] S.-C. Chi and C.-C. Yang. A two-stage clustering method combining ant colony SOM and k -means. *Journal of Information Science and Engineering*, 24(5):1445–1460, 2008.
- [45] Z. Chi, H. Yan, and T. Pham. *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. Advances in fuzzy systems - applications and theory. World Scientific, 1996.
- [46] H. Choi. Data visualization for asymmetric relations. *Neurocomputing*, 124:97 – 104, 2014.
- [47] F. H. Clarke. *Optimization and nonsmooth analysis*. Canadian Mathematical Society series of monographs and advanced texts. Wiley, 1983.
- [48] E. Come, M. Cottrell, M. Verleysen, and J. Lacaille. Aircraft engine fleet monitoring using self-organizing maps and edit distance. In *Proceedings of the 8th international conference on Advances in Self-Organizing Maps*, WSOM’11, pages 298–307, Berlin, Heidelberg, 2011.
- [49] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.
- [50] M. Cottrell, P. Gaubert, C. Eloy, D. Francois, G. Hallaux, J. Lacaille, and M. Verleysen. Fault prediction in aircraft engines using self-organizing maps. *Security*, pages 37–44, 2009.
- [51] M. Cutajar, E. Gatt, J. Micallef, I. Grech, and O. Casha. Digital hardware implemen-

- tation of self-organising maps. In *Melecon 2010, 15th IEEE Mediterranean Electrotechnical Conference*, pages 1123–1128. IEEE, 2010.
- [52] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. A statistical-topological feature combination for recognition of handwritten numerals. *Applied Soft Computing*, 12(8):2486–2495, 2012.
- [53] R. M. C. R. de Souza and F. d. A. T. de Carvalho. Clustering of interval data based on city-block distances. *Pattern Recognition Letters*, 25:353–365, 2004.
- [54] V. F. Demyanov, A. Astorino, and M. Gaudioso. Nonsmooth problems in mathematical diagnostics. In *Advances in Convex Analysis and Global Optimization*, pages 11–30. Springer US, 2001.
- [55] L. Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *Signal Processing Magazine, IEEE*, 29(6):141–142, Nov 2012.
- [56] V. Dey, D. K. Pratihar, and G. L. Datta. Genetic algorithm-tuned entropy-based fuzzy c -means algorithm for obtaining distinct and compact clusters. *Fuzzy Optimization and Decision Making*, 10(2):153–166, 2011.
- [57] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*, pages 357–382. Kluwer Academic Publishers, 2001.
- [58] K. A. J. Doherty, R. G. Adams, and N. Davey. Non-Euclidean norms and data normalization. In *Proceedings of ESANN*, pages 181–186, April 2004.
- [59] C. Faloutsos. FastMap: A fast algorithm for indexing, and data-mining and visualization of traditional and multimedia datasets. *Computer*, pages 163–174, 1995.
- [60] F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *ACM SIGKDD Explorations Newsletter*, 2(1):51–57, 2000.

- [61] A. Fiannaca, G. Fatta, S. Gaglio, R. Rizzo, and A. Urso. Improved SOM learning using simulated annealing. In *Artificial Neural Networks AI ICANN 2007*, volume 4668 of *Lecture Notes in Computer Science*, pages 279–288. Springer Berlin Heidelberg, 2007.
- [62] A. Fiannaca, G. Di Fatta, R. Rizzo, A. Urso, and S. Gaglio. Simulated annealing technique for fast learning of som networks. *Neural Computing and Applications*, 22(5):889–899, 2013.
- [63] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.
- [64] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39:761–765, 2006.
- [65] P. Fränti, J. Kuittinen, A. Tabarcea, and L. Sakala. MOPSI location-based search engine: concept, architecture and prototype. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC’10, pages 872–873. ACM, 2010.
- [66] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.
- [67] B. Fritzke. Growing grid-a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2:9–13, 1995.
- [68] M. H. Ghaseminezhad and A. Karami. A novel self-organizing map (SOM) neural network for discrete groups of data clustering. *Applied Soft Computing*, 11(4):3771–3778, 2011.
- [69] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.
- [70] A. Goltsev and V. Gritsenko. Investigation of efficient features for image recognition by neural networks. *Neural networks : the official journal of the International Neural Network Society*, 28:15–23, 2012.
- [71] M. L. Goncalves, M. L. de Andrade Netto, and J. Zullo. A neural architecture for the classification of remote sensing imagery with advanced learning algorithms. In

Neural Networks for Signal Processing VIII, 1998. Proceedings of the 1998 IEEE Signal Processing Society Workshop, pages 577–586, 1998.

- [72] F. L. Gorgonio and J. A. F. Costa. Combining parallel self-organizing maps and k -means to cluster distributed data. In *2008 11th IEEE International Conference on Computational Science and Engineering - Workshops*, pages 53–58. IEEE, 2008.
- [73] J. Gorricha and V. Lobo. Improvements on the visualization of clusters in geo-referenced data using self-organizing maps. *Computers & Geosciences*, 43:177–186, 2012.
- [74] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. ACM SIGMOD Int’l Conf. Management of Data*, pages 73–84. ACM Press, 1998.
- [75] K. Haese. Self-organizing feature maps with self-adjusting learning parameters. *IEEE transactions on neural networks*, 9(6):1270–8, 1998.
- [76] T. M. Hamdani, J.-M. Won, A. M. Alimi, and F. Karray. Hierarchical genetic algorithm with new evaluation function and bi-coded representation for the selection of features considering their confidence rate. *Applied Soft Computing*, 11(2):2501–2509, 2011.
- [77] C. Hanilci and F. Ertas. Comparison of the impact of some Minkowski metrics on VQ/GMM based speaker recognition. *Computers and Electrical Engineering*, 37:41–56, 2011.
- [78] P. Hansen, N. Mladenovic, and D. Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.
- [79] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k -means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 100–108, 1979.
- [80] K. Horio and T. Yamakawa. Handwritten character recognition based on relative position of local features extracted by self-organizing maps. *International Journal of Innovative Computing Information and Control*, 3(4):789–798, 2007.

- [81] A. Hsu and S. K. Halgamuge. Class structure visualization with semi-supervised growing self-organizing maps. *Neurocomputing*, 71(16-18):3124–3130, 2008.
- [82] D. Huang, Z. Yi, and X. Pu. A new local PCA-SOM algorithm. *Neurocomputing*, 71(16-18):3544–3552, 2008.
- [83] A. K. Jain. Data clustering: 50 years beyond k -means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
- [84] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [85] K. Jajuga. A clustering method based on the l_1 -norm. *Computational Statistics & Data Analysis*, 5(4):357–371, 1987.
- [86] H. Jin, W.-H. Shum, K.-S. Leung, and M.-L. Wong. Expanding self-organizing map for data visualization and cluster analysis. *Information Sciences*, 163(1-3):157–173, 2004.
- [87] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 2002.
- [88] S. Kantabutra and A. L. Couch. Parallel k -means clustering algorithm on NOWs. *NECTEC Technical journal*, 1(6):243–247, 2000.
- [89] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 1990.
- [90] H. Klock and J. M. Buhmann. Data visualization by multidimensional scaling: a deterministic annealing approach. *Pattern Recognition*, 33(4):651 – 669, 2000.
- [91] T. Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer, 2001.
- [92] T. Kohonen, S. Kaski, and H. Lappalainen. Self-organized formation of various invariant-feature filters in the adaptive-subspace som. In *Neural Computation*, pages 1321–1344, 1997.

- [93] A. Kovács and J. Abonyi. Vizualization of fuzzy clustering results by modified Sammon mapping. In *Proceedings of the 3rd International Symposium of Hungarian Researchers on Computational Intelligence*, pages 177–188, 2002.
- [94] I. Lapidot, H. Guterman, and A. Cohen. Unsupervised speaker recognition based on competition between self-organizing maps. *IEEE transactions on neural networks*, 13(4):877–87, 2002.
- [95] C.-H. Lee and H.-C. Yang. A multilingual text mining approach based on self-organizing maps. *Applied Intelligence*, 18(3):295–310, 2003.
- [96] A. Likas, N. Vlassis, and J. J. Verbeek. The global k -means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.
- [97] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.
- [98] E. López-Rubio. Improving the quality of self-organizing maps by self-intersection avoidance. *ieeexplore.ieee.org*, 24(8), 2013.
- [99] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.
- [100] Z. Man, K. Lee, D. Wang, Z. Cao, and S. Khoo. An optimal weight learning machine for handwritten digit image recognition. *Signal Processing*, pages 1–15, 2012.
- [101] T. Martinetz and K. Schulten. A “Neural-Gas” network learns topologies. *Artificial Neural Networks*, I:397–402, 1991.
- [102] M. J. McQuaid, T.-H. Ong, H. Chen, and J. F. N. Jr. Multidimensional scaling for group memory visualization. *Decision Support Systems*, 27(1-2):163 – 176, 1999.
- [103] H.-D. Meng, Y.-C. Song, F.-Y. Song, and S.-L. Wang. Clustering for complex and massive data. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1–4, Dec 2009.

- [104] E. Mohebi and M. N. M. Sap. An optimized hybrid kohonen neural network for ambiguity detection in cluster analysis using simulated annealing. In J. Filipe and J. Cordeiro, editors, *Enterprise Information Systems*, volume 24 of *Lecture Notes in Business Information Processing*, pages 389–401. Springer Berlin Heidelberg, 2009.
- [105] E. Mohebi, A. Bouyer, and M. Karimi. An efficient clustering of the som using rough set and genetic algorithm. In *GCC Conference Exhibition, 2009 5th IEEE*, pages 1–5, 2009.
- [106] J. Nikkilä, P. Törönen, S. Kaski, J. Venna, E. Castrén, and G. Wong. Analysis and visualization of gene expression data using self-organizing maps. *Neural networks : the official journal of the International Neural Network Society*, 15(8-9):953–66, 2002.
- [107] S. Nittel, K. T. Leung, and A. Braverman. Scaling clustering algorithms for massive data sets using data streams. In *Proceedings of the 19th International Conference on Data Engineering, March*, pages 5–8. Citeseer, 2003.
- [108] X.-X. Niu and C. Y. Suen. A novel hybrid CNN-SVM classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4):1318–1325, 2012.
- [109] J. Ontrup and H. Ritter. A hierarchically growing hyperbolic self-organizing map for rapid structuring of large data sets. In *In Proceedings of 5th Workshop On Self-Organizing Maps*, pages 471–478, 2005.
- [110] J. Ontrup and H. Ritter. Large-scale data exploration with the hierarchically growing hyperbolic SOM. *Neural networks : the official journal of the International Neural Network Society*, 19(6-7):751–61, 2006.
- [111] E. J. Otoo, A. Shoshani, and S.-W. Hwang. Clustering high dimensional massive scientific datasets. In *Scientific and Statistical Database Management, 2001. SSDBM 2001. Proceedings. Thirteenth International Conference on*, pages 147–157, 2001.
- [112] L. Pape, F. Gomez, M. Ring, and J. Schmidhuber. Modular deep belief networks that do not forget. *The 2011 International Joint Conference on Neural Networks*, pages 1191–1198, July 2011.

- [113] D. Pelleg and A. W. Moore. X-means: Extending k -means with efficient estimation of the number of clusters. In P. Langley, editor, *ICML'00 Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734. Morgan Kaufmann Publishers Inc., 2000.
- [114] T. D. Pham, M. Brandl, and D. Beck. Fuzzy declustering-based vector quantization. *Pattern Recognition*, 42(11):2570 – 2577, 2009.
- [115] M. A. Rahman and M. Z. Islam. A hybrid clustering technique combining a novel genetic algorithm with k -means. *Knowledge-Based Systems*, 2014.
- [116] G. Ramos, Y. Hatakeyama, F. Dong, and K. Hirota. Hyperbox clustering with ant colony optimization (HACO) method and its application to medical risk profile recognition. *Applied Soft Computing*, 9(2):632–640, 2009.
- [117] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.
- [118] A. Rauber. Data mining with the java som toolbox, 2011. URL <http://www.ifs.tuwien.ac.at/dm/somtoolbox/datasets.html>. Accessed: 2013.
- [119] G. Reinelt. TSPLIB -a traveling salesman problem library. *ORSA Journal of Computing*, 3(4):376–384, 1991.
- [120] K. Sabo. Center-based l_1 clustering method. *International Journal of Applied Mathematics and Computer Science*, 24(1):151–163, 2014.
- [121] K. Sabo, R. Scitovski, and I. Vazler. One-dimensional center-based l_1 -clustering method. *Optimization Letters*, 7(1):5–22, 2013.
- [122] J. W. Sammon. A nonlinear mapping for data structure analysis. *Computers, IEEE Transactions on*, C-18(5):401–409, 1969.
- [123] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig. Non-linear PCA: a missing data approach. *Bioinformatics*, 21(20):3887–3895, 2005.

- [124] R. Sedgewick and K. Wayne. *Introduction to Programming in Java*. Addison-Wesley, 2007. URL <http://introcs.cs.princeton.edu/java/36inheritance/Voronoi.java.html>.
- [125] S. Z. Selim and K. Al-Sultan. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008, 1991.
- [126] H. Shah-Hosseini and R. Safabakhsh. TASOM: a new time adaptive self-organizing map. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 33(2):271–82, 2003.
- [127] H. Shah-Hosseini. Binary tree time adaptive self-organizing map. *Neurocomputing*, 74(11):1823–1839, 2011.
- [128] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.
- [129] M. Sirola and J. Talonen. Self-organizing map in process visualization. In A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6882 of *Lecture Notes in Computer Science*, pages 196–202. Springer Berlin Heidelberg, 2011.
- [130] H. Späth. Algorithm 30 l_1 cluster analysis. *Computing*, 16(4):379–387, 1976.
- [131] Speech and Image Processing Unit. Clustering datasets, February 2012. URL <http://cs.joensuu.fi/sipu/datasets/>.
- [132] C. D. Stefano, F. Fontanella, C. Marrocco, and A. S. di Freca. A GA-based feature selection approach with an application to handwritten character recognition. *Pattern Recognition Letters*, 35:130 – 141, 2014.
- [133] L. X. Sun, Y. L. Xie, X. H. Song, J. H. Wang, and R. Q. Yu. Cluster analysis by simulated annealing. *Computers & Chemistry*, 18(2):103–108, 1994.
- [134] K. Tasdemir, P. Milenov, and B. Tapsall. Topology-based hierarchical clustering of self-organizing maps. *IEEE Transactions on Neural Networks*, 22(3):474–485, 2011.

- [135] Y. Theodoridis. Spatial datasets—an unofficial collection. <http://www.dias.cti.gr/~ytheod/research/datasets/spatial.html>, 1996.
- [136] S. Uchida and H. Sakoe. A survey of elastic matching techniques for handwritten character recognition. *IEICE - Trans. Inf. Syst.*, E88-D(8):1781–1790, 2005.
- [137] G. Vamvakas, B. Gatos, and S. J. Perantonis. Handwritten character recognition through two-stage foreground sub-sampling. *Pattern Recognition*, 43(8):2807–2816, 2010.
- [138] C. J. Veenman, M. J. T. Reinders, and E. Backer. A maximum variance cluster algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1273–1280, 2002.
- [139] N. B. Venkateswarlu and P. S. V. S. K. Raju. Fast ISODATA clustering algorithms. *Pattern Recognition*, 25(3):335–342, 1992.
- [140] J. Venna and S. Kaski. Local multidimensional scaling. *Neural Networks*, 19(6-7):889 – 899, 2006.
- [141] J. Vesanto and R. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):587–600, 2000.
- [142] J. Vesanto. SOM-based data visualization methods. *Intelligent Data Analysis*, 3:111–126, 1999.
- [143] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-organizing map in matlab: the SOM Toolbox. In *In Proceedings of the Matlab DSP Conference*, pages 35–40, 2000.
- [144] J. A. Walter. H-MDS: A new approach for interactive visualization with multidimensional scaling in the hyperbolic space. *Information Systems*, 29(4):273 – 292, 2004.
- [145] H. Wang, H. Zheng, and F. Azuaje. Poisson-based self-organizing feature maps and hierarchical clustering for serial analysis of gene expression data. *IEEE/ACM transactions on computational biology and bioinformatics*, 4(2):163–75, 2007.

- [146] M. L. D. Wong, L. B. Jack, and A. K. Nandi. Modified self-organising map for automated novelty detection applied to vibration signal monitoring. *Mechanical Systems and Signal Processing*, 20(3):593–610, 2006.
- [147] S. Wu and T. W. S. Chow. PRSOM: a new visualization method by hybridizing multidimensional scaling and self-organizing map. *IEEE transactions on neural networks: a publication of the IEEE Neural Networks Council*, 16(6):1362–80, 2005.
- [148] A. E. Xavier and A. A. F. D. Oliveira. Optimal covering of plane domains by circles via hyperbolic smoothing. *Journal of Global Optimization*, 31(3):493–504, 2005.
- [149] P. Xu, C.-H. Chang, and A. Paplinski. Self-organizing topological tree for online vector quantization and data clustering. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 35(3):515–26, 2005.
- [150] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.
- [151] L. Yang, Z. Ouyang, and Y. Shi. A modified clustering method based on self-organizing maps and its applications. *Procedia Computer Science*, 9:1371–1379, 2012.
- [152] G. G. Yen and Z. Wu. Ranked centroid projection: a data visualization approach with self-organizing maps. *IEEE transactions on neural networks*, 19(2):245–59, 2008.
- [153] H. Yin. ViSOM -a novel method for multivariate data projection and structure visualization. *IEEE transactions on neural networks: a publication of the IEEE Neural Networks Council*, 13(1):237–43, 2002.
- [154] H. Yin. On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural networks : the official journal of the International Neural Network Society*, 19(6-7):780–4, 2006.
- [155] D. Yu and L. Deng. Efficient and effective algorithms for training single-hidden-layer neural networks. *Pattern Recognition Letters*, 33(5):554–558, 2012.

- [156] J. Zhang, L. Peng, X. Zhao, and E. E. Kuruoglu. Robust data clustering by learning multi-metric l_q -norm distances. *Expert Systems with Applications*, 39:335–349, 2012.
- [157] T. Zhang and R. Ramakrishnan. BIRCH: An efficient data clustering databases method for very large databases. *ACM SIGMOD Record*, 1:103–114, 1996.
- [158] H. Zhao, A. Robles-Kelly, J. Zhou, J. Lu, and J.-Y. Yang. Graph attribute embedding via Riemannian submersion learning. *Computer Vision and Image Understanding*, 115(7):962–975, 2011.
- [159] H. Zheng, P. Cunningham, and A. Tsymbal. Adaptive offset subspace self-organizing map: an application to handwritten digit recognition. In *In Proc. Seventh Int. Workshop on Multimedia Data Mining*, pages 29–38, 2006.
- [160] H. Zheng, P. Cunningham, and A. Tsymbal. Learning multiple linear manifolds with self-organizing networks. *International Journal of Parallel, Emergent and Distributed Systems*, 22(6):417–426, 2007.
- [161] H. Zheng, W. Shen, Q. Dai, S. Hu, and Z.-M. Lu. Learning nonlinear manifolds based on mixtures of localized linear manifolds under a self-organizing framework. *Neurocomputing*, 72(13-15):3318–3330, 2009.
- [162] Y. Zheng and J. F. Greenleaf. The effect of concave and convex weight adjustments on self-organizing maps. *IEEE transactions on neural networks*, 7(1):87–96, 1996.

Appendix

Data sets

Table 10: The brief description of data sets

Data sets	Number of instances	Number of attributes
German towns	59	2
Bavaria postal 1	89	3
Bavaria postal 2	89	4
Fisher's Iris Plant	150	4
Heart Disease	297	13
Breast Cancer	683	9
TSPLIB1060	1060	2
Image Segmentation	2310	19
TSPLIB3038	3038	2
Page Blocks	5473	10
D15112	15112	2
Gamma Telescope	19020	10
NE	50000	2
Pla85900	85900	2