BETA HEBBIAN LEARNING: DEFINITION AND ANALYSIS OF A NEW FAMILY OF LEARNING RULES FOR EXPLORATORY PROJECTION PURSUIT

by Héctor Quintián

Supervisor:
Dr. Emilio Corchado

Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

VNiVERSiDAD D SALAMANCA

CAMPUS OF INTERNATIONAL EXCELLENCE

Department of Computer Science and Automation
University of Salamanca
(Spain)

April 2017

Abstract

This thesis comprises an investigation into the derivation of learning rules in artificial neural networks from probabilistic criteria.

- **Beta Hebbian Learning (BHL).**
  First of all, it is derived a new family of learning rules which are based on maximising the likelihood of the residual from a negative feedback network when such residual is deemed to come from the Beta Distribution, obtaining an algorithm called Beta Hebbian Learning, which outperforms current neural algorithms in Exploratory Projection Pursuit.

- **Beta-Scale Invariant Map (Beta-SIM).**
  Secondly, Beta Hebbian Learning is applied to a well-known Topology Preserving Map algorithm called Scale Invariant Map (SIM) to design a new of its version called Beta-Scale Invariant Map (Beta-SIM). It is developed to facilitate the clustering and visualization of the internal structure of high dimensional complex datasets effectively and efficiently, specially those characterized by having internal radial distribution. The Beta-SIM behaviour is thoroughly analysed comparing its results, in terms performance quality measures with other well-known topology preserving models.

- **Weighted Voting Superposition Beta-Scale Invariant Map (WeVoS-Beta-SIM).**
  Finally, the use of ensembles such as the Weighted Voting Superposition (WeVoS) is tested over the previous novel Beta-SIM algorithm, in order to improve its stability and to generate accurate topology maps when using complex datasets. Therefore, the WeVoS-Beta-Scale Invariant Map (WeVoS-Beta-SIM), is presented, analysed and compared with other well-known topology preserving models.

All algorithms have been successfully tested using different artificial datasets to corroborate their properties and also with high-complex real datasets.

Resumen

Esta tesis abarca la investigación sobre la derivación de reglas de aprendizaje en redes neuronales artificiales a partir de criterios probabilísticos.

- **Beta Hebbian Learning (BHL).**
  En primer lugar, se deriva una nueva familia de reglas de aprendizaje basadas en maximizar la probabilidad del residuo de una red con retroalimentación negativa cuando se considera que dicho residuo proviene de la Distribución Beta, obteniendo un algoritmo llamado *Beta Hebbian Learning*, que mejora a algoritmos neuronales actuales de búsqueda de proyecciones exploratorias.
- **Beta-Scale Invariant Map (Beta-SIM).**
  En Segundo lugar, *Beta Hebbian Learning* se aplica a un conocido algoritmo de Mapa de Preservación de la Topología llamado Scale Invariant Map (SIM) para diseñar una nueva versión llamada *Beta-Scale Invariant Map (Beta-SIM)*. Este nuevo algoritmo ha sido desarrollado para facilitar el agrupamiento y visualización de la estructura interna de conjuntos de datos complejos de alta dimensionalidad de manera eficaz y eficiente, especialmente aquellos caracterizados por tener una distribución radial interna. El comportamiento de *Beta-SIM* es analizado en profundidad comparando sus resultados, en términos de medidas de calidad de rendimiento con otros modelos bien conocidos de preservación de topología.
- **Weighted Voting Superposition Beta-Scale Invariant Map (WeVoS-Beta-SIM).**
  Finalmente, el uso de ensembles como el *Weighted Voting Superposition (WeVoS)* sobre el algoritmo *Beta-SIM* es probado, con objeto de mejorar su estabilidad y generar mapas topológicos precisos cuando se utilizan conjuntos de datos complejos. Por lo tanto, se presenta, analiza y compara el *WeVoS-Beta-Scale Invariant Map (WeVoS-Beta-SIM)* con otros modelos bien conocidos de preservación de topología.

Todos los algoritmos han sido probados con éxito sobre conjuntos de datos artificiales para corroborar sus propiedades, así como con conjuntos de datos reales de gran complejidad.

TABLE OF CONTENTS

Chapter 1. Introduction 11

Chapter 2. Modelling Human Learning: Artificial Neural Networks 13

2.1. The Human Learning Process 13

2.1.1. The Neuron 14

2.2. Artificial Neural Networks 16

2.2.1. An Artificial Neuron 16

2.2.2. Learning Algorithms in Neural Networks 18

2.2.3. Reinforcement Learning 18

2.2.4. Supervised Learning 19

2.2.5. Unsupervised Learning 19

2.3. Hebbian Learning 20

2.3.1. Hebbian Learning and Statistics: Principal Component Analysis 21

2.3.2. Oja's Models 24

2.3.3. Negative Feedback Network 26

2.4. Competitive Learning 28

2.4.1. The Self-Organizing Map 28

2.4.2. The Visually Induced SOM 31

2.4.3. The Scale Invariant Map 32

2.4.4. Growing Neural Gas 34

2.4.5. Quality Measures for Topology Preserving Maps 35

2.5. Ensembles for Topology Preserving Maps 36

2.5.1. Previously Proposed Models for SOM Ensemble Summarization 37

2.6. Exploratory Projection Pursuit 39

2.6.1. Nonlinear Principal Component Analysis 40

2.6.2. The Exploratory Projection Pursuit Neural Network 41

2.6.3. Maximum Likelihood Hebbian Learning 41

2.6.4. Cooperative Maximum Likelihood Hebbian Learning 42

2.7. Maximum Likelihood Scale Invariant Maps (MLHL-SIM) 43

Chapter 3. A novel Family of Learning Rules: Beta Hebbian Learning 45

3.1. Introduction 45

3.2. Beta Distribution 45

3.3. Beta Hebbian Learning Algorithm 46

3.4. Stability of the Beta Hebbian Learning 48

3.5. Experiments for BHL ... 49

3.5.1. Artificial Datasets for BHL ... 49

3.5.2. Real Datasets for BHL ... 53

3.6. Conclusions for Beta Hebbian Learning ... 69

Chapter 4. Beta Scale Invariant Map ... 71

4.1. Introduction ... 71

4.2. Beta SIM Learning Rule ... 71

4.3. Influence of the Choice of α and β Parameters on the Beta-SIM Learning Rule ... 72

4.3.1. Case 1: $\alpha=\beta$ ... 72

4.3.2. Case 2: $\alpha>\beta$ ... 74

4.3.3. Case 3: $\alpha<\beta$ ... 76

4.4. Experiments for Beta-SIM ... 77

4.4.1. Artificial Datasets for Beta-SIM ... 78

4.4.2. Real Datasets for Beta-SIM ... 83

4.4.3. Validation over 14 Real Benchmark Datasets for Beta-SIM ... 91

4.5. Conclusions for Beta Scale Invariant Map ... 94

Chapter 5. Weighted Voting Superposition Beta Scale Invariant Map ... 95

5.1. Introduction ... 95

5.2. WeVoS-Beta-SIM ... 95

5.3. Experiments and results for WeVoS-Beta-SIM ... 97

5.3.1. Benchmark datasets description for WeVos-Beta-SIM ... 97

5.3.2. Visualization results ... 98

5.3.3. Analytical results ... 105

5.4. Conclusions for WeVoS-Beta-SIM ... 116

Chapter 6. Conclusions & Future Work ... 117

Chapter 7. References ... 121

Chapter 8. Annex ... 129

LIST OF FIGURES

Fig. 2.1 Depiction of the human brain (left) and map of nerve fibers in the human brain (right). Figure taken from [29]. 13

Fig. 2.2 Representation of a biological neuron. Figure taken from [31]. 15

Fig. 2.3 Artificial neuron model. 17

Fig. 2.4 Sample ANN architecture. 17

Fig. 2.5 Basic architecture of a feed-forward ANN. 20

Fig. 2.6 Graphical representation of rotation of the axes for PCA. 22

Fig. 2.7 Basic architecture of a negative feedback network. 27

Fig. 2.8 Conceptual mapping of 3D input data onto a SOM grid with topology preservation. 29

Fig. 2.9 Neurons vectors updating in a SOM. 30

Fig. 2.10 Contraction (Fig. 2.10a) or expansion (Fig. 2.10b) force for the updating of the ViSOM neurons. Figure taken form [49]. 32

Fig. 2.11 Scale invariant map mapping, where each neuron captures a "pie slice" of the data according to the angular distribution of the input data. 33

Fig. 2.12 The SIM results when the learning rate is increased. 34

Fig. 3.1 Probability density function of the beta distribution for different values of α and β. 46

Fig. 3.2 Stability of the BHL learning rule (it becomes unstable for values of $|e|>1$). 48

Fig. 3.3 The Gaussian direction was the second among 8 leptokurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=2$. 50

Fig. 3.4 The Gaussian direction was the eighth among 8 platykurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=4$. 51

Fig. 3.5 The leptokurtic direction was the seventh among 8 Gaussian dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=10$. 51

Fig. 3.6 The platykurtic direction was the fourth among 8 Gaussian dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=1.8$. 51

Fig. 3.7 The platykurtic direction was the third among 8 leptokurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=2$. 52

Fig. 3.8 The beta direction was the fifth among 8 platykurtic dimensions. It has clearly been identified in this Hinton map of the weights when *α=β=10.* 52

Fig. 3.9 The directions with positive skewness was the second (skewness=1) among 8 Gaussian dimensions with skewness=0. The direction with positive skewness has been identified in these Hinton maps of the weights when *α=4 and β=10.* 52

Fig. 3.10 Projection of the algae dataset onto the first two principal components using Neural PCA. 54

Fig. 3.11 Projection of the algae dataset onto the first two filters found using MLHL with *p = 1.* 54

Fig. 3.12 2D projection of the algae dataset using LLE algorithm with *k=60.* 55

Fig. 3.13 2D projection of the algae dataset using CCA algorithm with 1,000 epochs. 55

Fig. 3.14 2D projection of the algae dataset using Isomap algorithm with *k=12.* 56

Fig. 3.15 Projection of the algae dataset onto the first two filters found using BHL with *α=4 and β=20.* 56

Fig. 3.16 Projection of the astronomical dataset onto the first two principal components using Neural PCA. 58

Fig. 3.17 Projection of the astronomical dataset onto the first two filters found using MLHL with *p = 0.5.* 58

Fig. 3.18 2D projection of the astronomical dataset using LLE algorithm with *k=15.* 59

Fig. 3.19 2D projection of the astronomical dataset using CCA algorithm with 5,000 epochs. 59

Fig. 3.20 2D projection of the astronomical dataset using Isomap algorithm with *k=12.* 60

Fig. 3.21 Projection of the astronomical dataset onto the first two filters found using BHL with *α=3.1* and *β=10.* 60

Fig. 3.22 Projection of the bank dataset onto the first two principal components using Neural PCA. 61

Fig. 3.23 Projection of the bank dataset onto the first two filters found using MLHL with *p = 1*. 62

Fig. 3.24 2D projection of the bank dataset using LLE algorithm with *k=40.* 62

Fig. 3.25 2D projection of the bank dataset using CCA algorithm with 1,000 epochs. 63

Fig. 3.26 2D projection of the bank dataset using Isomap algorithm with *k=40.* 63

Fig. 3.27 Projection of the bank dataset onto the first two filters found using BHL with *α=β=2.* 64

Fig. 3.28 Projection of the dental milling dataset onto the first two principal components using Neural PCA. 65

Fig. 3.29 Projection of the dental milling dataset onto the first two filters found using MLHL with *p = 1*. 66

Fig. 3.30 2D projection of the dental milling dataset using LLE algorithm with *k=30*. 66

Fig. 3.31 2D projection of the dental milling dataset using CCA algorithm with 1,000 epochs. 67

Fig. 3.32 2D projection of the dental milling dataset using Isomap algorithm with *k=30*. 67

Fig. 3.33 Projection of the dental milling dataset onto the first two filters found using BHL with *α=β=2*. 68

Fig. 4.1 Beta distribution of residuals for values of *α=β:2,2; 5,5; 10,10*. 73

Fig. 4.2 Beta-SIM learning rule of the residuals for values of *α=β*. 74

Fig. 4.3 Beta distribution for values of *α>β (6,5; 8,4; 10,3)*. 75

Fig. 4.4 Beta-SIM learning rule for values of *α>β*. 75

Fig. 4.5 Beta distribution for values of *α<β (5,6; 4,8; 3,10)*. 76

Fig. 4.6 Beta-SIM learning rule for values of *α<β*. 77

Fig. 4.7 SOM (a), SIM (b), MLHL-SIM (c) and Beta-SIM (d) final network grid (with 20 neurons) for artificial dataset 1. 79

Fig. 4.8 Beta-SIM (h) results for values *α=β*, and its comparison with SOM (a), SIM (b) and MLHL-SIM (c,d,e,f,g) algorithms; 20 neurons; 50,000 iterations; η=0.02; neighborhood function=Gaussian; MLHL-SIM: p=10, 5, 2, 1, 0.8; Beta-SIM *α=β=4*. 81

Fig. 4.9 Beta-SIM results for values *α>β*; *α=5, β=2*; η=0.5; iterations=50,000; neighborhood function=Gaussian; neurons grid=[20,1] 82

Fig. 4.10 Beta-SIM results for values *α<β*: *α=2, β=8*; η=0.01; iterations=50,000; neighborhood function= Gaussian; neurons grid=[20,1]. 83

Fig. 4.11 SOM, SIM, MLHL-SIM and Beta-SIM final network grids adaptation over the E. Coli dataset. 86

Fig. 4.12 Beta-SIM final network grid adaptation over the E. Coli dataset, with *α=2 and β=4*. 87

Fig. 4.13 SOM, SIM, MLHL-SIM and Beta-SIM final maps for E. Coli dataset. 88

Fig. 4.14 Beta-SIM final map when *α=2* and *β=4* for E. Coli dataset. 88

Fig. 4.15 SOM and Beta-SIM final network grids adaptation over the high precision machine dataset. .. 90

Fig. 5.1 Schematic diagram of the weight voting in WeVoS in a 2-D map. 96

Fig. 5.2 Algorithms' final network grids adaptation over the Iris dataset. 100

Fig. 5.3 Algorithms' final unit maps for Iris dataset... 101

Fig. 5.4 Algorithms' final network grid adaptations over the Landsat Satellite dataset.......................... 103

Fig. 5.5 Algorithms final network grid adaptations over the Landsat Satellite dataset. 105

Fig. 5.6 CE measures vs number of samples of each algorithm for each of the 14 datasets. 109

Fig. 5.7 MQE measures vs number of samples of each algorithm for each of the 14 datasets. 112

Fig. 5.8 TE measures vs number of samples of each algorithm for each of the 14 datasets. 115

LIST OF TABLES

Table 3.1 Four real datasets characterized by different number of samples, features and classes belonging to different fields of application for BHL. 53

Table 4.1 SOM, SIM, MLHL-SIM and Beta-SIM parameters and its MQE and TE for the artificial dataset 1. 79

Table 4.2 SOM, SIM, MLHL-SIM and Beta-SIM parameters and their associated measures: CE, MQE and TE for E. Coli dataset. 84

Table 4.3 SOM, SIM, MLHL-SIM and Beta-SIM parameters and their MQE and TE for the high precision machine dataset. 89

Table 4.4 Benchmark datasets for Beta-SIM experiments. 91

Table 4.5 p-values for CE (Beta-SIM against all algorithms). 92

Table 4.6 p-values for MQE (Beta-SIM against all algorithms). 92

Table 4.7 p-values for TE (Beta-SIM against all algorithms). 93

Table 8.1 Average testing CE/MQE/TE ± STD over the 14 benchmark datasets for Beta-SIM experiments. 129

Table 8.2 Algorithms parameters for each benchmark dataset for Beta-SIM and WeVoS-Beta-SIM experiments. 130

Table 8.3 Summary of the results in terms of the three well-known quality measures for each benchmark dataset for Beta-SIM experiments. 132

Table 8.4 p-values for CE (WeVoS-Beta-SIM against all algorithms) 134

Table 8.5 p-values for MQE (WeVoS-Beta-SIM against all algorithms) 136

Table 8.6 p-values for TE (WeVoS-Beta-SIM against all algorithms) 138

Table 8.7 Average testing CE ± STD over the 14 benchmark datasets 140

Table 8.8 Average testing MQE ± STD over the 14 benchmark datasets 142

Table 8.9 Average testing TE ± STD over the 14 benchmark datasets 144

Chapter 1. Introduction

Data mining [1], [2] and the extraction of information from enormous datasets that are generated by modern experimental and observational methods is increasingly necessary in almost all industrial and scientific fields and business operations nowadays. This “information extraction” [3], [4] is defined as the nontrivial data mining of implicit, previously unknown, and potentially useful information. Among several fields where “information extraction” is not an easy task, Big data [5], [6] is one of the most recent and important topics where the use of intelligent techniques becomes crucial to be able to extract knowledge from the enormous amounts of information. One of the many techniques used to extract relevant information is data visualization [7]–[11].

Artificial Neural Networks (ANN) [12], [13] are typically software simulations which emulate some of the features of real neural networks found in animal brains. ANN are a branch of Artificial Intelligence (AI) which consist of connectionist systems that have different applications based on their neural architecture. ANN are usually used to model complex relationships between inputs and outputs, to find patterns in data [14], and also for extracting information from high dimensional datasets by projecting the datasets onto low dimensional (typically 2 dimensional) subspaces.

Among the applications of unsupervised artificial neural networks one of them is the data projection or visualization, which facilitates the analysis of the internal structure of a dataset to the human expert. This can be achieved by data projection over more informative axes or by generating maps representing the inner structure of datasets. For the first kind of data visualization usually techniques such as Exploratory Projection Pursuit (EPP) [15]–[18] can be used by projecting the data onto a low dimensional subspace in which we search for structures by visual inspection. For the second one, the topology preserving maps [19]–[23] may be applied, being probably the best known among these algorithms the Self-Organizing Map (SOM) [19], [21], [24], [25].

This thesis compromises and investigation into the derivation of unsupervised learning rules in artificial neural networks from probabilistic criteria:

- Beta Hebbian Learning (BHL): A novel EPP algorithm based on a novel family of learning rules derived from the PDF of the residual based on Beta distribution to extract information from high dimensional datasets by projecting the data onto low dimensional (typically 2 dimensional) subspaces, improving the existing exploratory methods by providing a clear representation of data’s internal structure.
- Beta-Scale Invariant Map (Beta-SIM): A novel Topology Preserving Map algorithm as a new version of the Scale Invariant Map (SIM), developed to facilitate the clustering and visualization of the internal structure of complex datasets effectively and efficiently. It is based

on the application of a family of learning rules derived from the Probability Density Function (PDF) of the residual based on the beta distribution (BHL), when applied to the Scale Invariant Map.

- Weighted Voting Superposition Beta-Scale Invariant Map (WeVoS-Beta-SIM): A novel ensemble topology preserving map called WeVoS-Beta-Scale Invariant Map (WeVoS-Beta-SIM), based on the application of the Weighted Voting Superposition (WeVoS) meta-algorithm and Beta-Scale Invariant Map (Beta-SIM), to generate effectively and efficiently accurate topology maps over complex data

All algorithms have been tested using different artificial datasets high dimensional real datasets in order to corroborate their theoretical properties.

This thesis is structured as follows: in Chapter 2, the main ideas of the human learning process are introduced, presenting the different types of learning processes and discussing some elementary artificial neural networks, with special attention on the topology preserving maps. Chapter 3, 4 and 5 contain the rest of this thesis original work. In Chapter 3 it is derived a novel family of learning rules, called Beta Hebbian Learning (BHL), which are shown to have sophisticated in information processing capabilities. In Chapter 4, the BHL is applied to improve a topology preserving map called Scale Invariant Map, to provide better visualization of the internal structure of high-dimensional datasets. The novel model called Beta-SIM generate grid maps which adapt better their structure to datasets and at the same time, improve the topology of the grid maps. In Chapter 5, it is investigated the use of Weight Voting Superposition ensemble with the neural architecture presented in Chapter 4. Finally, in Chapter 6 we review the thesis and present directions for future work. Appendix includes several tables with detailed information of statistical results from relevant experiments performed in this thesis.

Chapter 2. Modelling Human Learning: Artificial Neural Networks

2.1. *The Human Learning Process*

The Nervous System conducts stimuli from sensory receptors to the brain and spinal cord and that conducts impulses back to other parts of the body. As with other higher vertebrates, the human nervous system has two main parts: the central nervous system (the brain and spinal cord), depicted in Fig. 2.1 and the peripheral nervous system (the nerves that carry impulses to and from the central nervous system) [26].

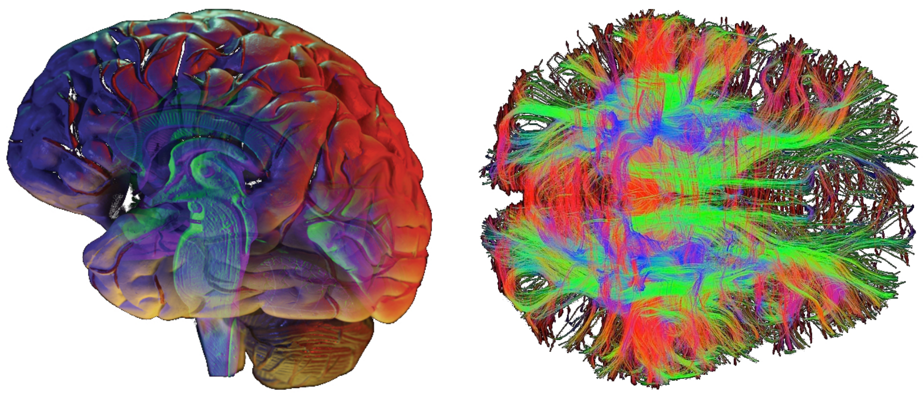


Fig. 2.1 Depiction of the human brain (left) and map of nerve fibers in the human brain (right). Figure taken from [27].

In the following, it is intended to provide a background of the biological foundations of how modern studies in information treatment try to emulate its functioning of the brain to obtain results that are difficult (or impossible) to obtain with other type of computing approaches. In relation with the performance of the human brain, four main functions can be considered:

- *Sense.* The brain has a set of sensory organs (eyes, ears, nose, taste, touch) to tell us some of what is going on in the outside world. It can integrate the information from these different senses and create an internal representation of the external world. All experience is filtered by the senses; and these sensory signals (e.g., sound, sight, taste, touch), in turn, initiate a cascade of cellular and molecular processes in the brain that alter neuronal neurochemistry, cytoarchitecture and, ultimately, brain structure and function. This process of creating some internal representation of the external world (i.e., information) depends upon the pattern, intensity and frequency of neuronal activity produced by sensing, processing and storing signals.

- *Process.* Once the human sensory apparatus has translated physical or chemical information from the outside (or inside) world into neuronal activity, this set of signals travels up into the brain to be processed. Sensory information from the external environment and the internal environment enters the central nervous system at the level of the brain stem and midbrain. As this primary sensory input comes into the brain stem and mid brain, it is matched against previously stored patterns of activation and if unknown, or if associated with previous threat, the brain will activate a set of responses that are designed to help promote survival. Because the brain cannot possibly create a unique neural imprint or pattern of change to store every element of every experience, the brain stores 'template' patterns based upon the first set of organizing experiences. All future incoming input is matched against these stored templates and, if sufficiently different from the original pattern, the brain will make neural changes (i.e., create a memory) that reflects that tiny difference.
- *Store.* Inherent in the processing of information coming into the brain is the capacity to store elements of these incoming signals. The ability of the brain to create memories is due to the capacity of neurons and neural systems to change from one 'homeostatic' state to another. In response to a set of stimuli-induced (e.g., sensations) alterations in activity, neurons undergo molecular changes that reflect this activity.
- *Actions.* Finally, the brain mediates and controls the actions of the human body. By regulating and directing the actions of the neuromuscular, autonomic, endocrine and immune systems the brain controls the actions of the human being.

Now this simple (and somewhat misleading) description of the human learning process is only a crude approximation of the key actions of the brain. Indeed, there are hundreds if not thousands of local and regional feedback loops in an open and interactive dynamic system (well beyond any mathematical models of complex systems yet developed).

Although there are certainly interesting advances in artificially emulating all the four described functions, the scope of this thesis will be centered on the second and third functions of the human cognitive process; which are more directly related with the process commonly known as 'learning'.

2.1.1. The Neuron

[28] was the first who introduced the idea of neurons as structural constituents of the brain. The neuron is the main functional component of the brain. It can be defined as a specialized kind of cell that integrates the (input) activity of other neurons that are connected to it and propagates that integrated (output) activity to other neurons. This process is accomplished by a complex series of biochemical events within the neuron. The parts of a neuron in which we are interested are the following: the dendrites, the cell body, the axon, the terminals and the synapse (see Fig. 2.2).

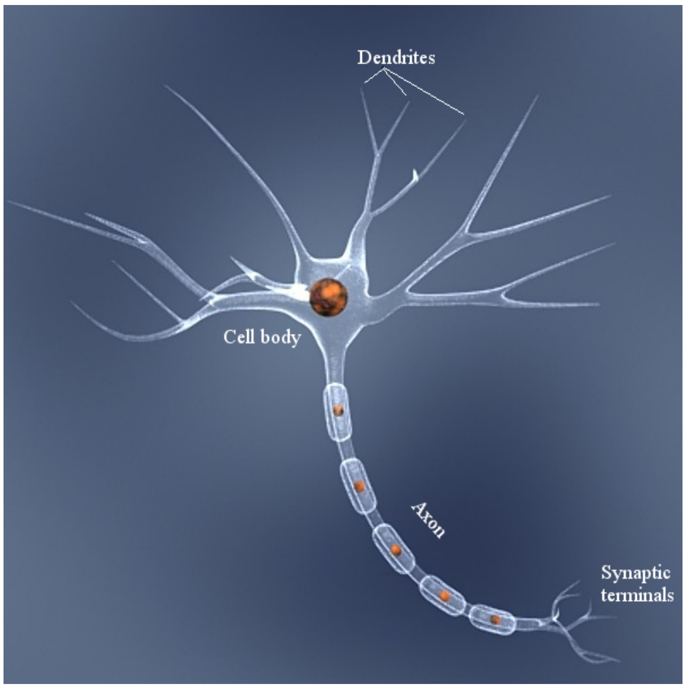


Fig. 2.2 Representation of a biological neuron. Figure taken from [29].

- The dendrites are thin protrusions from the cell body that collect chemical signals from other neurons and convert them into electrical activity along the thin membrane that encloses the cell.
- The cell body contains the nucleus and cellular machinery. The membrane around the cell body integrates the electrical signals arriving from all the dendrites, again coded in terms of a graded potential, and converts it into a series of all-or-none electrical potentials that propagate along the axon.
- The axon is a long, thin projection of the neuron along which action potentials are propagated to other neurons, often over a considerable distance. Most of the axon of most neurons is covered by a myelin sheath, which speeds the conduction of action potentials. The strength of the integrated signal that the axon transmits is encoded primarily in its firing rate, the number of electrical impulses it generates in a given amount of time (e.g., spikes per second).
- The terminals are the branching ends of the axon at which the electrical activity of the axon is converted back into a chemical signal with which it can stimulate another neuron. This is accomplished by releasing a neurotransmitter into the small gap between the terminal and the dendrite of the next neuron. Neurotransmitters are chemical substances that are capable of exciting the dendrites of other neurons. The signal strength transmitted at the terminal is determined by the amount of neurotransmitter released.

- The synapse is the small gap that exists between the terminals of one neuron and the dendrites of another. The neurotransmitter that is released into the synapse rapidly crosses the gap and affects the next neuron's dendrite by occupying specialized sites on its membrane. This is where the chemical signal from one neuron is converted to an electrical signal in the next one.

2.2. *Artificial Neural Networks*

Studies on artificial neural networks [12], [13], [30], [31] have been motivated from its inception by the recognition that the brain computes in an entirely different way from the conventional digital computer.

The brain makes up for the relatively slow rate of operation of a neuron by having a truly staggering number of neurons (nerve cells) with massive interconnections between them. Some studies have estimated that there are around 10 billion neurons in the human cortex, and 60 trillion synapses or connections. So, the brain is an enormously complex structure.

The brain is a highly complex, non-linear, and parallel computer (information-processing system). The brain has the capability of organizing neurons so as to perform certain computations (e.g. pattern recognition, perception and motor control) much faster that than the fastest digital computer in existence today. A clear example of these is that the brain routinely accomplishes perceptual recognition tasks (e.g. recognizing a familiar face embedded in an unfamiliar scene) in something of the order of 100-200 ms while a conventional computer will take much longer to perform the same task.

An artificial neural network is a system that is designed to model the way in which the brain performs a particular task or function of interest. The network is usually implemented using electronic components or simulated in software on a digital computer. To achieve good performance, neural networks use a massive interconnection of simple computing cells referred to as "neurons" or "processing units".

In a network, knowledge is obtained through a learning process and interneuron connection strengths, known as synaptic weights, are employed to store this knowledge.

The procedure for modifying the synaptic weights of the network in an orderly way to reach a desired objective is called a learning algorithm.

2.2.1. An Artificial Neuron

As previously stated, a neuron can be defined as the constituent element of an ANN: the neuron is the smallest processing unit within the ANN [12], [13]. They were initially designed to mimic the human neurons as the constituent element of the neural system. A model of an artificial (or synthetic) neuron was initially proposed in [32] and is depicted in Fig. 2.3.

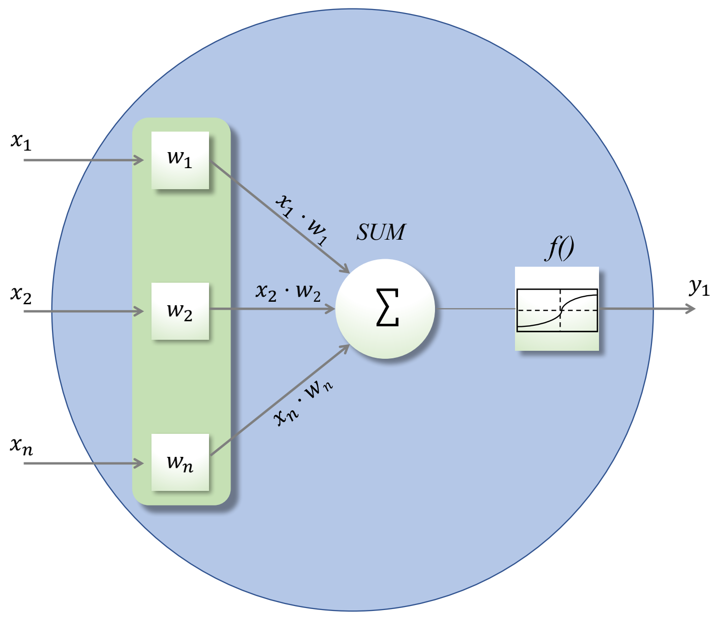


Fig. 2.3 Artificial neuron model.

The network inputs (*x*) are passed through the synaptic weights (*w*) and are then summed. Subsequently, the activation function (*f()*) is applied to calculate the output of the neuron (*y*).

An ANN usually consists of several neurons organized according to a certain network topology. An example of a network topology consisting of an input layer, two hidden layers and an output layer is shown in Fig. 2.4.

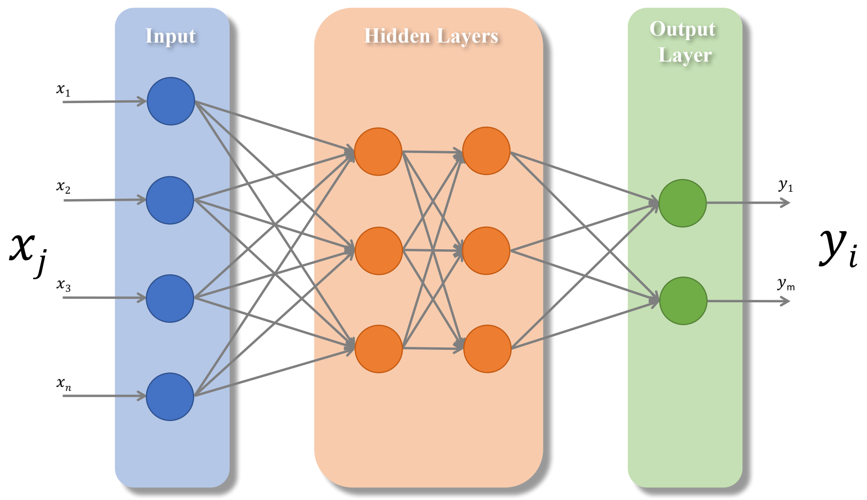


Fig. 2.4 Sample ANN architecture.

The following sections describe the different ways in which the layer connections (weights) of an ANN are modified.

2.2.2. Learning Algorithms in Neural Networks

There are three main types of learning algorithms for automated weight setting in networks:

1. Reinforcement learning.
2. Supervised learning.
3. Unsupervised learning.

Reinforcement learning [33] is defined by characterizing a learning problem, rather than by characterizing learning methods. This type of learning is concerned with how an agent ought to take actions in an environment to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

Supervised learning [34], [35] relies on the existence of the target output that the network should ideally generate. The elements required in supervised learning are:

- Input of the network: numerical values that are introduced in the network according to the problem setting.
- Internal dynamics of the network: determine the output of the network related to the input.
- Evaluation of the target: generates the "measure" to update the weights.

Typically, ANNs based on supervised learning use error descent to modify their weights.

In contrast, for **unsupervised learning** [36], [37] only the first two elements (input and internal dynamics of the network) are available. No external information is used to check on the weight setting process.

2.2.3. Reinforcement Learning

Reinforcement learning [33] involves learning how to map situations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions are the most rewarding by trying them. In the most interesting cases, actions may affect not only the immediate reward but also the next situation and thereby, all subsequent rewards. These two characteristics -trial-and-error search and delayed reward- are the two most important distinguishing features of reinforcement learning.

This learning mechanism is different from supervised learning in the sense that the supervised learning implies learning from examples provided by a knowledgeable external supervisor. This in itself is not enough in order to learn from interaction. In interactive problems, it is often impractical to obtain correct examples of desired behavior that are representative of all the situations in which the agent has to act. In uncharted territory -where one would expect learning to be most beneficial- an agent must be able to learn from its own experience.

2.2.4. Supervised Learning

Associative memory networks are simple one or two layer networks that store patterns for subsequent retrieval. They include the class of networks known as content addressable memories or memory devices [34], [38] that permit the retrieval of data from pattern keys that are based on attributes of the stored data.

Two classes of associative memory can be implemented: autoassociative and heteroassociative. Autoassociative memories recall the same pattern **y** as the input x, that is $x=y$. In heteroassociative memories the recall pattern is different from the input**,** $x \neq y$. Clearly in this case, the association between patterns is being stored.

Autoassociative memories are very useful when a noisy or partially complete pattern is the only available input and the output pattern is the original, complete, non-noisy pattern.

An essential feature of supervised learning is the existence of an external teacher. The network is trained on examples whose target output is known. There must be a training set for which we already know the answer to our question to the network.

For autoassociative supervised networks, we present the input data to the input neurons; propagate the data forward through weights to the hidden neurons and then through the next layer of weights to the output neurons. The target pattern is equal to the input pattern.

2.2.5. Unsupervised Learning

Human beings appear to be able to learn without explicit supervision. One aim of unsupervised learning is to mimic this aspect of human learning; hence this type of learning tends to use more biologically plausible methods than those using error descent methods. A pertinent example is the local processing at each synapse in these algorithms, which involves no global information passing through the network. So, an unsupervised network must self-organize with respect to its internal parameters, without external prompting. To do so, it must react to some aspect of the input data; typically, redundancy, or clusters.

There are, moreover, two major methods of unsupervised learning [36], [37], [39]:

- Hebbian learning.
- Competitive learning.

2.3. ***Hebbian Learning***

Hebbian learning is so-called after [39] who conjectured:

> *"When an axon of a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B, is increased".*

This statement is sometimes expanded [40] into a two-part rule:

1. If two neurons on either side of a synapse (connection) are activated simultaneously (i.e. synchronously), then the strength of that synapse is selectively increased.
2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

If we consider a basic feed forward neural network, this would be interpreted as the weight between an input neuron and an output neuron is very much strengthened when the input neuron's activation, when passed forward to the output neuron, causes the output neuron to fire strongly. It can be seen that the rule favors the strong: if the weights between inputs and outputs are already large (and so an input will have a strong effect on the outputs) the chance of the weights growing is large. The architecture of the feed-forward network is shown in Fig. 2.5.

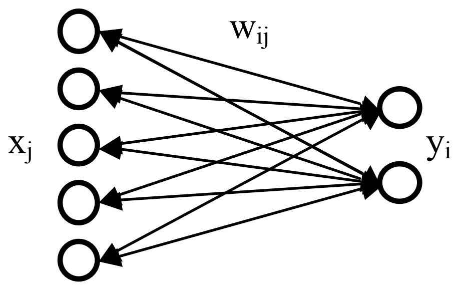


Fig. 2.5 Basic architecture of a feed-forward ANN.

In mathematical terms, if we consider the simplest feed forward neural network which has a set of input neurons with associated input vector, *x*, and a set of output neurons with associated output vector, *y*, we have the expression in Eq. (2.1).

$$y_i = \sum_j W_{ij} \cdot x_j \tag{2.1}$$

where W_{ij} represents the weight vector between input *j* and output *i*.

The Hebbian learning rule is defined by Eq. (2.2).

$$\Delta W_{ij} = \eta \cdot x_j \cdot y_i \tag{2.2}$$

where η is the learning rate parameter.

That is, the weight between each input and output neuron is increased proportional to the magnitude of the simultaneous firing of these neurons.

Now we can substitute into the learning rule the value of y calculated by the feeding forward of the activity to get Eq. (2.3).

$$\Delta W_{ij} = \eta \cdot x_j \cdot \sum_k W_{ik} \cdot x_k = \eta \cdot \sum_k W_{ik} \cdot x_k \cdot x_j \qquad (2.3)$$

The statistical properties of the learning rule are emphasized in the last expression. We now see how the learning rule depends on the correlation between different parts of the inputs data's vector components.

We have a difficulty with the basic rule as it stands which is that we have a positive feedback rule which has an associated difficulty with lack of stability: if the i^{th} input and j^{th} output neurons are tending to fire strongly together, the weight between them will tend to grow strongly; if the weight grows strongly, the j^{th} output will fire more strongly the next time the i^{th} input neuron fires and this will cause an increased value in the rate of change of the weights. So it is necessary to take some preventative measures in order to prevent the weights from growing without bound. These preventative measures can include:

1. Bounding the weights i.e. insisting that there is a range of values $[W_{min}, W_{max}]$ in which the weights must remain.
2. Normalizing the weights after each update, which ensures that the weights into each output neuron have length 1.
3. Having a weight decay term within the learning rule to stop it growing too large.
4. Create a network containing a negative feedback of activation.

2.3.1. Hebbian Learning and Statistics: Principal Component Analysis

In this section the well-known Principal Component Analysis (PCA) technique is outlined to later explain two neural implementations of this method (Oja's Model and Negative Feedback Network).

Principal components analysis originated in work by [41], and independently by [42] to describe the variation in a set of multivariate data in terms of a set of uncorrelated variables each of which is a linear combination of the original variables.

Its goal is to derive new variables, in decreasing order of importance, that are linear combinations of the original variables and are uncorrelated with each other.

From a geometrical point of view, PCA can be defined as a rotation of the axes of the original coordinate system to a new set of orthogonal axes that are ordered in terms of the amount of variation of the original data that they account for. A graphical representation of this characteristic can be found in Fig. 2.6.

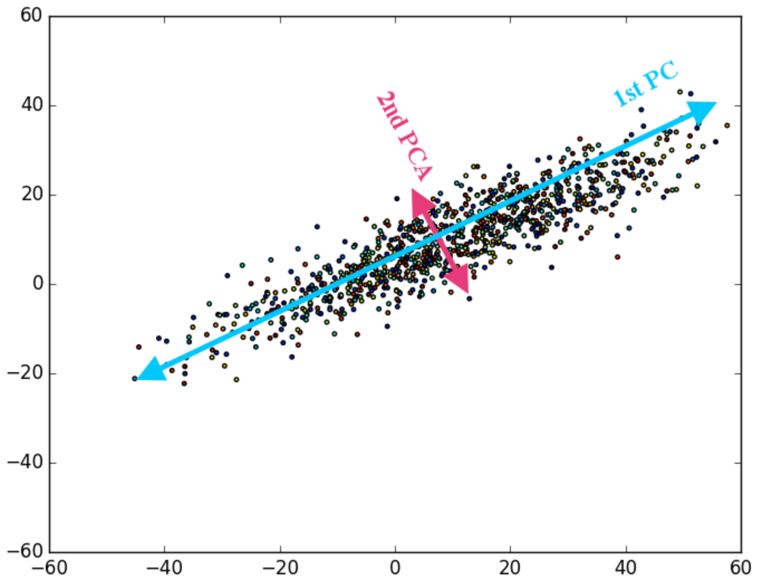


Fig. 2.6 Graphical representation of rotation of the axes for PCA.

Using PCA, it is possible to find a smaller group of underlying variables that describe the data. So, the first few components of this group could be used to explain most of the variation in the original data. Note that, even if we are able to characterize the data with a few variables, it does not follow that we will be able to assign an interpretation to these new variables.

An important problem in analyzing data of high dimensionality is identifying patterns that exist across dimensional boundaries. Such patterns may become visible if a change of basis of the space is made, however an *a priori* decision as to which basis will reveal most patterns requires fore-knowledge of the unknown patterns. PCA is a potential way of solving this problem. PCA aims to find that orthogonal basis which maximizes the data's variance for a given dimensionality of basis. The typical way is to find that direction which accounts for most of the data's variance; this is the first basis vector (the first Principal Component direction). One then finds that direction that accounts for most of the remaining variance- this is the second basis vector and so on. If one then projects data onto the Principal Component directions, we perform a dimensionality reduction that will be accompanied by the retention of as much variance (or information) in the data as possible. The basis vectors of this new co-ordinate system can be shown to be the eigenvectors of the covariance matrix of the dataset and the variance on these co-ordinates are the corresponding eigenvalues. The optimal projection from N to M dimensions given by PCA is the subspace spanned by the M eigenvectors with the largest eigenvalues.

PCA can also be thought of as a data compression technique where there is minimum information loss in the data, in terms of least mean squared error. As a result, it often is used as a pre-processing method in order to simplify further analysis.

Taking an analysis by [14], it is possible to describe this as mapping vectors x^d in an N-dimensional space ($x_1, \ldots, x_n$) onto vectors y^d in an M-dimensional space ($y_1, \ldots, y_m$), where $M \leq N$. x may be represented as a linear combination of a set of N orthonormal vectors W_i (Eq. (2.4)).

$$x = \sum_{i=1}^{N} y_i \cdot W_i \tag{2.4}$$

the vectors W_i satisfy the orthonormality relation presented in Eq. (2.5).

$$W_i^t \cdot W_j = \delta_{ij} \tag{2.5}$$

where δ_{ij} is the Kronecker delta.

Making use of the expression Eq. (2.4), the coefficients y_i may be given by Eq. (2.6).

$$y_{i=} W_i^t \cdot x \tag{2.6}$$

which can be regarded as a simple rotation of the co-ordinate system from the original *x's* to a new set of co-ordinates given by the *y's*. If only a subset $M<N$ of the basis vectors, W_i, are retained so that only M coefficients y_i are used, then replacing the remaining coefficients by constants b_i each vector x may be approximated by the following (Eq. (2.7)).

$$\tilde{x} = \sum_{i=1}^{M} y_i \cdot W_i + \sum_{i=M+1}^{N} b_i \cdot W_i \tag{2.7}$$

Consider the whole dataset of D vectors, x^d where $d=1, \ldots, D$.

It is necessary to make the best choice of W_i and b_i so that y_i values obtained by Eq. (2.6) give the best approximation of Eq. (2.7) over the whole dataset. The vector x^d has an error due to the dimensionality reduction (Eq. (2.8)).

$$x^d - \tilde{x}^d = \sum_{i=M+1}^{N} \left(y_i^d - b_i\right) \cdot W_i \tag{2.8}$$

The best approximation can be defined as that which minimizes the sum of the squares of the errors over the whole dataset (Eq. (2.9)).

$$E_M = \frac{1}{2}\sum_{d=1}^{D} \|x^d - \tilde{x}^d\|^2 = \frac{1}{2}\sum_{d=1}^{D} \sum_{i=M+1}^{N} \left(y_i^d - b_i\right)^2 \tag{2.9}$$

If we calculate the derivative of E_M with respect to b_i and set it to zero then we get Eq. (2.10).

$$b_i = \frac{1}{D}\sum_{d=1}^{D} y_i^d = W_i^T \bar{x} \qquad \forall i \ in \ M+1, \ldots, N \tag{2.10}$$

where the sample mean vector $\bar{x}$ is defined in Eq. (2.11).

$$\bar{x} = \frac{1}{D}\sum_{d=1}^{D} x^d \tag{2.11}$$

Now the sum of squares error can be written as Eq. (2.12).

$$E_M = \frac{1}{2}\sum_{i=M+1}^{N}\sum_{d=1}^{D}\{W_i^T(x^d - \bar{x})\}^2 = \frac{1}{2}\sum_{i=M+1}^{d} W_i^T \Sigma W_i \tag{2.12}$$

where Σ is the sample covariance matrix of the set of vectors $\{x^d\}$ and is given by Eq. (2.13).

$$\Sigma = \frac{1}{D}\sum_{d}(x^d - \bar{x})(x^d - \bar{x})^T \tag{2.13}$$

Then if we minimize E_M with respect to the choice of W_i it can be shown [14] that the minimum occurs when the basis vectors satisfy Eq. (2.14).

$$\Sigma W_i = \lambda_i W_i \tag{2.14}$$

So, those W_i are the eigenvectors of the covariance matrix. The eigenvectors can be proved to be orthogonal as assumed if the covariance matrix is real and symmetric. If we substitute the expression of Eq. (2.14) into Eq. (2.12) and making use of the orthonormality relation (Eq. (2.5)) then the value of the error criterion at the minimum may be represented in the form of Eq. (2.15).

$$E_M = \frac{1}{2}\sum_{i=M+1}^{N} \lambda_i \tag{2.15}$$

Then the minimum error is obtained by choosing the (*N-M*) smallest eigenvalues, and their corresponding eigenvectors, as the ones to discard. We usually call the *y's* the principal components.

2.3.2. Oja's Models

Oja contributed in a very important manner during the resurgence of research into artificial neural networks in the early 80's. It is a well-known fact that Hebbian Learning is inherently unstable, due to a problem with positive feedback causing unconstrained growth. Oja's models deal with this problem in a way that also gives them important information extraction properties.

Oja's One Neuron Model

[43] developed a model of Hebbian learning with weight decay that not only stopped the weights growing without bound, it also caused the weights to converge to the Principal Components of the input data. A single output neuron sums the weighted inputs as follows (Eq. (2.16)).

$$y = \sum_{i=1}^{N} W_i \cdot x_i \tag{2.16}$$

In [43] it was introduced a modification to the Hebbian rule based on Eq. (2.17).

$$\Delta W_i = \eta \cdot (x_i y - y^2 W_i) \tag{2.17}$$

It has been shown that the weight decay term has the effect of making the length of W_i tend towards 1 [43] and to converge to the principal eigenvector of the input covariance matrix. The weights are therefore normalized and so do not grow without bound. This rule will find only the first eigenvector (that direction corresponding to the largest eigenvalue) of the covariance matrix of the data.

It is not sufficient to simply throw a cluster of neurons at the data since all will find the same (first) Principal Component (PC); in order to find other PCs, there must be some interaction between the neurons.

Oja's Subspace Algorithm

This algorithm [44] provides a major step forward from the previous one. The network has *N* output neurons each of which learns using a Hebb type rule with weight decay. It does not guarantee to find the actual directions of the Principal Components; the weights do, however, converge to an orthonormal basis of the Principal Component Space, i.e. collectively the *M* outputs will capture the *M* filters with greatest variance. The space spanned by this basis is called the Principal Subspace. The learning rule is presented in Eq. (2.18).

$$\Delta W_i = \eta \cdot \left(x_i y_i - y_i \sum_{k} W_{kj} y_k \right) \tag{2.18}$$

As before, the weight growth is constrained and a major advantage of this technique is that it is completely homogeneous, i.e. the operations at each neuron are identical. Of course, the disadvantage is that it only finds the Principal Subspace spanned by the eigenvectors and not the actual eigenvectors themselves.

Oja's Weighted Subspace Algorithm

Oja developed his previous work [44], [45] to now identify the actual principal components using the Weighted Subspace Algorithm. It recognized the importance of introducing asymmetry into the weight decay process in order to force weights to converge to the Principal Components. The model is defined by Eq. **(2.19)**.

$$y_i = \sum_{i=1}^{N} W_{ij} x_j \tag{2.19}$$

where a Hebb-type rule with decay modifies the weights according to Eq. (2.20).

$$\Delta W_i = \eta \cdot y_i \cdot \left(x_i - \theta_i \sum_{k=1}^{M} y_k W_{kj} \right) \tag{2.20}$$

Ensuring that $\theta_1 < \theta_2 < \theta_3 < \cdots$ allows the neuron whose weight decays proportional to θ_1 (i.e. whose weight decays least quickly) to learn the principal values of the correlation in the input data. That means that this neuron will respond maximally to directions parallel to the first principal component. The second output cannot compete with the first but is in a stronger position to identify the second principal component and so on for all outputs in the network.

It can be shown that the weight vectors will converge to the principal eigenvectors in the order of their eigenvalues.

2.3.3. Negative Feedback Network

Now it is introduced the Negative Feedback Network [46], which we will use in the subsequent experiments. Feedback is said to exist in a system whenever the output of an element in the system influences in part the input applied to that particular element. It is used in this case to maintain the equilibrium on the weight vectors.

Consider an N-dimensional input vector, *x*, and a *M*-dimensional output vector, *y*, with W_{ij} being the weight linking input *j* to output i and let η be the learning rate.

The initial situation is that there is no activation at all in the network. The input data is feedforward via weights from the input neurons (the *x*-values) to the output neurons (the *y*-values) where a linear summation is performed to give the activation of the output neuron (see Fig. 2.7). It can express this by Eq. (2.21).

$$y_i = \sum_{j=1}^{N} W_{ij} x_j \quad \forall i \tag{2.21}$$

The activation is feedback through the same weights and subtracted from the inputs. (where the inhibition takes place) [46] uses the notation:

$$x'_i = x_j - \sum_{i=1}^{M} W_{ij} y_j \quad \forall j \tag{2.22}$$

since he is interested in biologically plausible learning. However, it will be used *e* instead of *x'*. Then, Eq. (2.22) can be rewritten as:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_j \quad \forall j \tag{2.23}$$

After that simple Hebbian learning is performed between input and outputs, the weights update is obtained by means of Eq. (2.24).

$$\Delta W_{ij} = \eta \cdot e_j \cdot y_i \tag{2.24}$$

The effect of the negative feedback is to stabilize the learning in the network. Because of that it is not necessary to normalize or clip the weights to get convergence to a stable solution.

Note that this algorithm is clearly equivalent to Oja's Subspace Algorithm [44] since:

$$\Delta W_{ij} = \eta \cdot e_j \cdot y_i = \eta \cdot \left(x_j - \sum_{k} W_{kj} y_k \right) \cdot y_i \tag{2.25}$$

This network is capable of finding the principal components of the input data [47] in a manner that is equivalent to Oja's Subspace algorithm [44], and so the weights will not find the Principal Components but a basis of the Subspace spanned by these components.

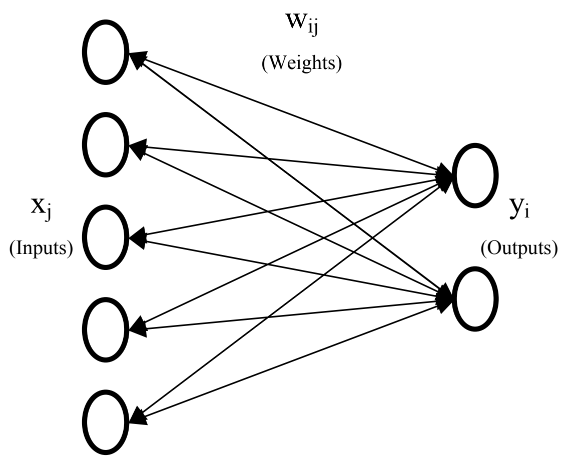


Fig. 2.7 Basic architecture of a negative feedback network.

It may be stated that the network thus uses simple Hebbian learning to enable the weights to converge to extract the maximum information content from the input data.

Since the model is equivalent to Oja's Subspace algorithm, we might legitimately ask what we gain by using the negative feedback in this way.

Writing the algorithm in this way, it gives a model of the process which allows to devise different versions and algorithms like the Maximum Likelihood Hebbian learning rule [15], which is based on an explicit view of the residual $(x - Wy)$ which is never independently calculated using e.g. Oja's learning rule.

Feedback is said to exist in a system whenever the output of an element in the system influences in part, the input applied to that particular element. It is used in this case to maintain the equilibrium of the weight vectors.

2.4. *Competitive Learning*

In this kind of learning [48], the output neurons of a neural network compete among themselves for being the one to be active (firing). This too mirrors the reality of what happens in the brain in that there are finite resources for learning and so one neuron's gain means another's loss. This is the biggest difference with Hebbian learning in which several output neurons may be active simultaneously; in the case of competitive learning only a single output neuron is active at any one time. This characteristic makes competitive learning a highly suitable tool to find those statistically salient features that may be used to classify a set of input patterns. [48] claim that there are three basic elements to a competitive learning rule:

1. **A set of neurons** that are all the same except for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input patterns.
2. **A limit** imposed on the "strength" of each neuron.
3. **A competition mechanism** that permits neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron, or only one neuron per group, is active (i.e., "on") at a time. The neuron that wins the competition is called a winner-takes-all neuron.

By means of an adaptive process, each individual neuron of the neural network gradually becomes sensitive to different input categories, or sets of samples in a specific domain of the input space, thereby becoming a feature detector.

2.4.1. The Self-Organizing Map

The Self-Organizing Map (SOM) [25], [49] is the most widely of a family of a neural network models called Topology Preserving Maps. All these models share the same objective: to produce a low dimensional representation of the training samples while preserving the topological properties of the input space. Because of this characteristic their main use is the visualization and clustering of data. In most cases the data analyzed is high dimensional and any Data Mining techniques that use the data space a bound to be overly complicated. An accurate low dimensional representation of the data will be of great advantage for most Data Mining algorithms.

The basic SOM consists of *m* units located on a regular low-dimensional grid, *U*, usually 1- or 2-dimensional (see Fig. 2.8)

Each unit *j* has an associated d-dimensional characteristic vector $w_j = [w_{j1}, \ldots, w_{jd}]$. The unit positions k_j on the grid are fixed from the beginning. The map adjusts to the data by adapting the prototype vectors. Together the grid and the set of characteristics vectors form a low-dimensional map of the data manifold: a 2-dimensional representation where topologically closely related objects (map units or neurons) are close to each other.

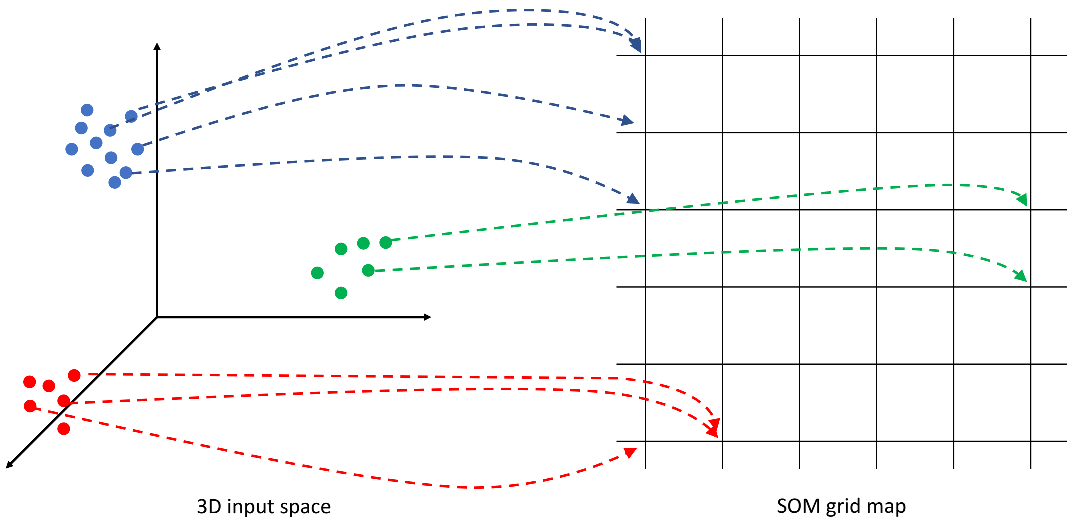


Fig. 2.8 Conceptual mapping of 3D input data onto a SOM grid with topology preservation.

The learning process by which the neurons of the network adapt to the data is an iterative process. At each training step *t* there are several operations to perform. The first operation is to randomly select an entry from the dataset analyzed, consisting on a *d*-dimensional vector x_i. This constitutes the input to the network. Then, the Euclidean distance between the input vector and the characteristics vector of all the neurons in the network is calculated. The neuron with the lowest distance to the input is deemed as the winning neuron (*v*), which will be called best matching unit (BMU). This situation is expressed mathematically in Eq. (2.26).

$$w_v = \arg min_k\{\|x_i - w_k(t)\|\} \tag{2.26}$$

Next, the characteristic vectors for the BMU and its neighborhood are “moved” towards the presented input, to reinforce the similarity between the BMU (and its neighborhood) and the inputs. That way, a neuron specializes in recognizing similar input patterns to the one presented. The most characteristic part of the SOM learning is that a neighborhood is also used. This enables not only the BMU to update its vector to the input, but also neighboring neurons in direct proportion to their distance to the BMU in the map lattice.

The update of the neurons in the SOM is presented graphically in Fig. 2.9.

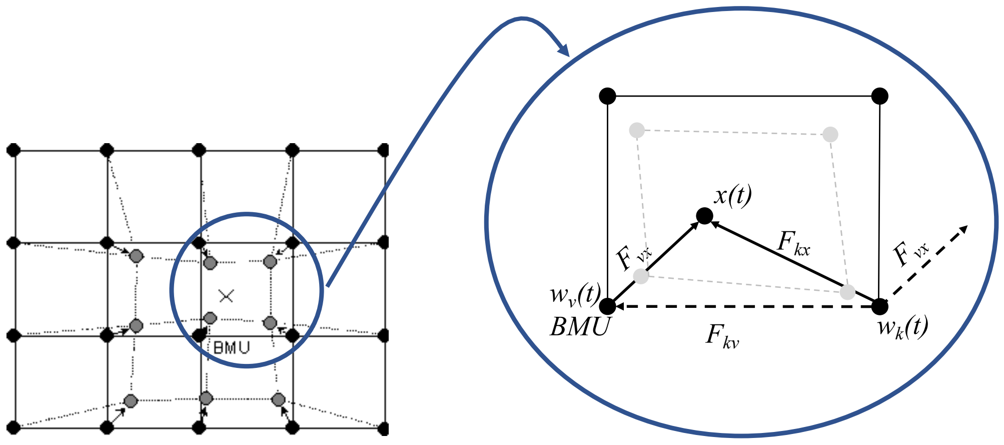


Fig. 2.9 Neurons vectors updating in a SOM.

The update of neurons in the SOM can be represented by means of Eq. (2.27).

$$w_k(t+1) = w_k(t) + \alpha(t) \cdot \eta(v,k,t) \cdot \big(x(t) - w_k(t)\big) \tag{2.27}$$

where x is denoting the input to the network, w_k the characteristics vector of each neuron; is the α learning rate of the algorithm, $\eta(v,k,t)$ is the neighborhood function where v represents the position of the winning neuron (BMU) in the lattice and k the positions of the neurons in the neighborhood of this one. The most common function is a Gaussian function centered on the position of the BMU; although other functions, like a difference of Gaussians, are also widely used. The characteristic Gaussian function would be defined by Eq. (2.28).

$$\eta(v,k,t) = e^{\frac{\|w_v - k\|}{2\sigma^2(t)}} \tag{2.28}$$

where v and k are positions of the BMU and k^{th} unit on the SOM grid and σ is the neighborhood radius. Both the learning rate $\alpha(t)$ and the neighbourhood radius $\sigma(t)$ decrease monotonically during training; learning rate to zero and neighborhood radius to some suitable non-zero value, usually one.

Additionally, an extra step can be performed. By randomly choosing the weights of the neurons as the input at a small percentage of updating times (e.g., 10% iterations), the learning of the neighborhood of the chosen neurons is reinforced; avoiding to have empty spaces in the map because the neurons that have not reacted to any data.

As a result of the learning process, i.e. the presentation of all input vectors and the adaptation of the weight vectors, the SOM generates a mapping from the input space onto the lattice *U*, in which the topological relationships in input space are preserved in *U* to the highest degree as possible. By updating the neurons taking into account not only the BMU, but also their neighboring neurons (Eq. (2.27)); close neurons gradually specialize to represent similar inputs, and the representations become ordered on the map lattice. This topological ordering of data into a 2-D map is one of the main features of the SOM.

2.4.2. The Visually Induced SOM

In the SOM, a neighborhood learning is adopted to form topological ordering among the neurons in the map. Thus, the map can be used to show the relative relationships among data points. However, the SOM does not directly show the inter-neuron distances on the map. For visualization, the SOM requires assistance from some coloring scheme to imprint the inter-neuron distances and therefore the clusters and boundaries can be marked.

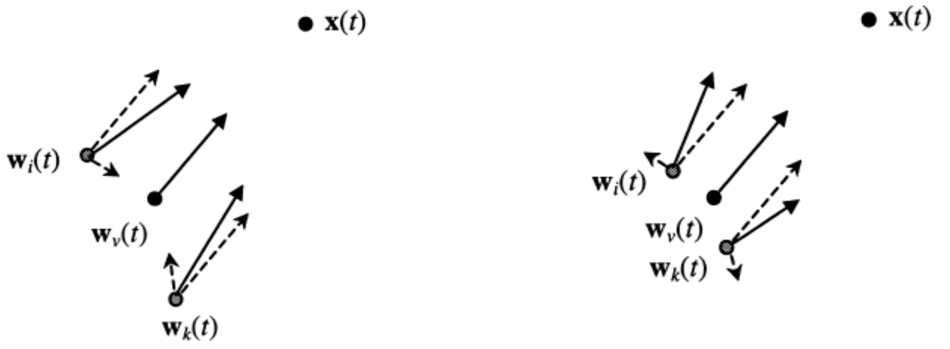
For the map to capture the data structure naturally and directly, the distance quantity must be preserved on the map, along with the topology. Ideally the nodes should be uniformly and smoothly placed in the nonlinear manifold of the data space. The distances of any two nearest neighboring neurons are approximately the same and the distances between a neuron and its further neighboring neurons increase proportionally and regularly according to the structure of the map grid. So that the positions of the neurons can be served as grades for measuring the distance of any mapped points. Then, the map can be seen as a smooth and graded mesh embedded into the data space, onto which the data points are mapped and the inter-point distances are approximately preserved.

The Visually Induced SOM (ViSOM) [50], [51] projects the high-dimensional data in an unsupervised manner as does the SOM, but constrains the lateral contraction force and hence regularizes the inter-neuron distance to a parameter that defines and controls the resolution of the map. It preserves the data structure as well as the topology as faithfully as possible. The ViSOM uses a similar grid structure of neurons as does the SOM. Its training algorithm is basically the same as that of the SOM's. The difference between both algorithm lies in the updating the weights of its composing units. The steps of this algorithm can be summarized as follow:

At time step t, an input *x(t)* is drawn randomly from the dataset or data space. A winning neuron can be found according to its distance to the input, using the same expression as the SOM (Eq. (2.26)). Then, in the SOM algorithm, the weights of the neurons in a neighborhood of the winner are updated by Eq. (2.27) (see also Fig. 2.9). The second term in this equation $\left(x(t) - w_k(t)\right)$ which can be considered as the updating force, can be decomposed into two different forces (see Eq. (2.29)).

$$F_{kx} \equiv x(t) - w_k(t) = [x(t) - w_v(t)] + [w_v(t) - w_k(t)] \equiv F_{vx} + F_{kv} \tag{2.29}$$

The first force, F_{vx}, represents the updating force from the winner v to the input x, which is the same as that used by the winner in Eq. (2.27). It adapts the neurons toward the input in a direction that is orthogonal to the tangent plane of the winner. While the second force, F_{kv}, is a lateral force bringing neuron k to the winner v, i.e., a contraction force. It is this contraction force that brings neurons in the neighborhood toward the winner and thus forms a contraction around the winner on the map at each time step. In the ViSOM, this lateral contraction force is constrained through regularizing the distance between a neighboring neuron to the winner. These forces are represented graphically in Fig. 2.10.



(a) Contraction force between the BMU and 2 neighboring neurons

(b) Expansion force between the BMU and 2 neighboring neurons

Fig. 2.10 Contraction (Fig. 2.10a) or expansion (Fig. 2.10b) force for the updating of the ViSOM neurons. Figure taken form [50].

The scale of the force is controlled by the normalized distance between these two weights until they are in proportion to the distances of their weights in the data space. Therefore, the updating of neurons in the case of the ViSOM can be expressed by Eq. (2.30).

$$w_k(t+1) = w_k(t) + \alpha(t) \cdot \eta(v,k,t) \cdot \left[\left(x(t) - w_v(t)\right) + \left(w_v(t) - w_k(t)\right) \frac{d_{vk} - \Delta_{vk}\lambda}{\Delta_{vk}\lambda} \right] \qquad (2.30)$$

where d_{vk} and Δ_{vk} are the distances between neurons and in the data space v and k on the unit grid or map, respectively, and λ is a positive pre-specified resolution parameter. It represents the desired inter-neuron distance (of two neighboring nodes) reflected in the input space.

It can be seen that if the d_{vk} is larger than $\Delta_{vk}\lambda$; i.e. the w_k is farther away from w_v under the specified resolution, the constraint is positive (Fig. 2.10a), so a contraction force remains. Otherwise, the constraint becomes negative, so an opposite or expansion force applies (Fig. 2.10b).

The ViSOM produces a smooth and regularly graded mesh through the data points and enables a quantitative, direct, and visually appealing measure of inter-point distances on the map.

2.4.3. The Scale Invariant Map

A Scale Invariant Map (SIM) [11], [52] is a regular array of nodes arranged on a lattice, similar to a Self-Organizing Map (SOM), but training uses a method based on the negative feedback network. A neighborhood function and competitive learning are used in the same way as with the SOM. The input data is feedforward to the outputs in the usual way. After selection of a winner, the winner, c, is deemed to be firing (y_c=*1*) and all other outputs are suppressed ($y_i = 0; \forall i \neq c$). The winner's activation is then feedback through its weights and this is subtracted from the inputs, and simple Hebbian learning is used to update the weights of all nodes in the neighborhood of the winner.

Training on a SOM relies on iteratively selecting a winner stimulated by the inputs, and updating the weights. With the scale invariant map, the weights of the winning node are feedback as inhibition at the inputs, and simple Hebbian learning is then used to update the weights of all nodes in the neighborhood of the winner (Eq. (2.31) and (2.32)).

$$e = x - W_c y_c, \quad (y_c = 1) \tag{2.31}$$

$$\Delta W_i = h_{ci} \cdot \eta \cdot e, \ \ \forall i \in N_c \tag{2.32}$$

It has the effect of updating all weight vectors in parallel to the vector $(x - W_c)$. So, Eq. (2.32) can be rewritten as Eq. (2.33).

$$\Delta W_i = h_{ci} \cdot \eta \cdot (x - W_c), \ \ \forall i \in N_c \tag{2.33}$$

In the negatively feedback from a neuron to the input data for which it wins, there is a large number of residuals which are relatively small in magnitude and a much larger number of residuals which are much larger in magnitude. This is more likely to be approximated by an exponential distribution than a Gaussian. The final effect of this type of learning is that the SIM is such that a pie-slice of data is actually won by each neuron. This algorithm is called a scale-invariant feature map since it ignores the magnitude of each input vector and responds solely to the relative proportion of the magnitudes of the elements of the input vectors. The way the SIM matches the data in a "pie-slice" manner is shown in Fig. 2.11.

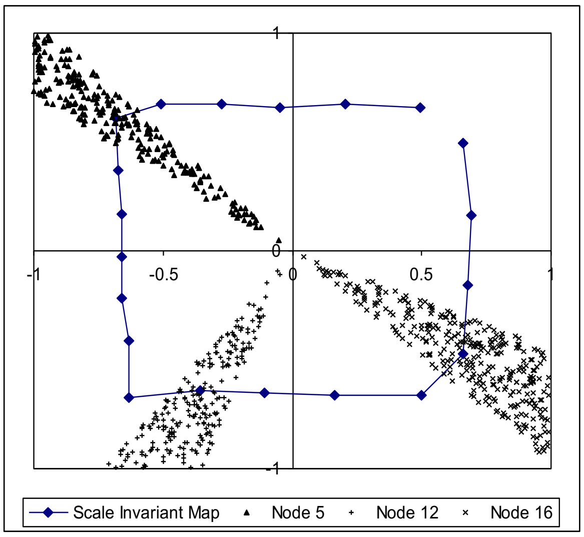


Fig. 2.11 Scale invariant map mapping, where each neuron captures a "pie slice" of the data according to the angular distribution of the input data.

It is well known that, given similar input data to that used above, a one dimensional self-organizing map (SOM) will self-organize to spread itself over the square to minimize the expected distance between the code points and the points of the square. However, if the learning rate is increased with this feature map, an interesting effect comes into play: the mapping winds around upon itself so that each outer neuron (which is currently winning competitions) is backed up by a set of support neurons. The results of such an experiment are shown in Fig. 2.12.

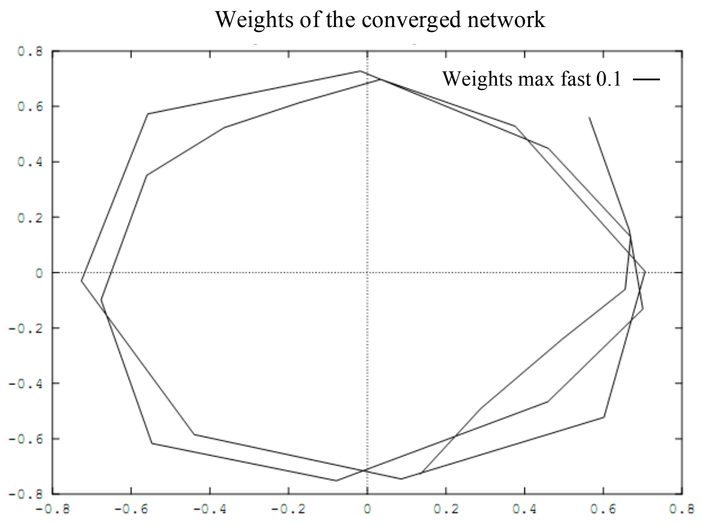


Fig. 2.12 The SIM results when the learning rate is increased.

2.4.4. Growing Neural Gas

The Growing Neural Gas (GNG) [53], [54] it is based on the Neural Gas (NG) algorithm previously proposed by Martinetz et al. [55] for finding optimal data representations based on feature vectors, which is in turn a modification of the widely known SOM. The main characteristic of the NG algorithm is that instead of expanding through the data input space as a fixed grid of units (as done by the SOM algorithm), the NG algorithm allows the neighboring relationships of its units to change, expanding more like a gas over the data space.

The Growing Neural Gas (GNG) method is different from the previous competitive algorithms in that it is an incremental algorithm, so there is no need to determine a priori the number of nodes. Network shape and size are determined during the training, while the SOM and NG are often trained on a fixed network size throughout.

The GNG is a combination of Fritzke's Growing Cell Structures (GCS) [56] and Martinetz's Competitive Hebbian Learning (CHL) [57]. In each step of the algorithm, the error of the units in representing the data is calculated as the Euclidean distance of units and inputs by means of Eq. (2.34).

$$\Delta errors(s_1) = \left\| w_{s_1} - x \right\|^2 \tag{2.34}$$

where s_1 is the closest unit to the input entry x and w_{s1} is the weights vector corresponding to that unit. The network topology of the GNG is generated incrementally by the CHL algorithm, which successively inserts topological connections or edges in the points where the error is higher. The main principle of the CHL is: for each input x connect the two closest centers (measured by Euclidean distance) by an edge. Then, the weights of the units are updated, for the BMU $(x - w_{s_1})$ and for their neighbors (w_n) as it is presented in Eq. (2.35) and (2.36).

$$w_{s_1} = e_b(x - w_{s_1}) \tag{2.35}$$

$$w_n = e_n(x - w_n) \tag{2.36}$$

2.4.5. Quality Measures for Topology Preserving Maps

Usually, when dealing with supervised learning; having the collection of desired outputs for the inputs to the ANN, a certain deviation from the ANN's learning state and the desired output can be calculated quite easily. When dealing when unsupervised learning, the process depends only on the inputs and the dynamics of the learning rule; making much more di-cult to determine the degree of accuracy of the training in relation to the input dataset. Such a measure would be relevant, not only for theoretical analysis of the learning process, but also for practical purposes, as it could be used to determine how well the algorithm is adapting to the characteristics of the data. Unfortunately, there does not exist in general a canonical measure to determine the quality of the training of unsupervised learning algorithms.

Several quality measures have been proposed in literature to study the reliability of the results displayed by topology preserving models in representing the dataset that have been trained with [11], [58]. There is not a global and unified measure, but rather a set of complementary ones, as each measure assesses a specific feature of the performance of the map in different visual representation areas. The three measures used in this thesis are briefly described in the following paragraphs.

Classification Error (CE). Using its inherent pattern matching characteristics, the topology preserving maps in general terms can be used for classification tasks. Intuitively, the samples activating the same neuron of the network are very likely to belong to the same class. When a new sample is presented to the network, the sample can be classified in the same class as the majority of samples activating the same neuron belong to. A consistent behavior when classifying samples points to a correctly trained map. Although this is not the main function of this kind of networks, the measure of how many samples are wrongly classified has been used, to an extent, to assess the quality of the final map in numerous previous studies [11], [58].

Mean Quantization Error (MQE). MQE is related to all forms of vector quantization and clustering algorithms. Thus, this measure completely disregards map topology and alignment. MQE is computed by determining the average distance of the dataset entries to the cluster centroids by which they are represented. In case of SOM, the cluster centroids are the characteristic vectors.

Topographic Error (TE). TE is the simplest of the topology preservation measures. A dataset is also needed to calculate this measure. For all data samples, the respective best and second-BMUs (1st BMU and 2nd BMU) are determined. If these BMUs are not adjacent on the map lattice, it is considered an error. Finally, the total error is normalized to a range from 0 to 1, where 0 means perfect topology preservation.

2.5. *Ensembles for Topology Preserving Maps*

The main concept behind ensemble learning model is the simple intuitive idea of a committee of experts working together to solve a problem. It seems obvious that when dealing with a complicated problem, a group of experts among the same area, has a higher probability of reaching a satisfactory solution than a single expert. Each of them contributes with its own experience and initiative and the rest of the group can sum up to the new ideas or refute them in case one of them is wrong.

In the field of Artificial Intelligence (AI), ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one.

The application of ensembles with ANNs is not something new. When looking at bibliography regarding ensemble use in supervised ANNs is easy to see that a lot of different models and applications have been proposed, tested and analyzed since more than 15 years now. On the other hand, the use of any kind of ensemble meta-algorithms with ANNs making use of unsupervised learning is a much less explored option.

This situation is not strange, as the difficulties of obtaining a meaningful combination of several unsupervised ANNs are evident. Almost all the supervised ANNs ensembles rely on the performance of each of its composing units to calculate in some way the combination of results that will be the final output of the ensemble. Many of the most sophisticated methods rely on the networks' performance measures to try to improve also their training.

The main obstacle in the case of the unsupervised ANNs, is the difficulty of determining to which degree the performance of the network deviates from the expected optimal one. In many of the cases, this is because that expected performance is not even determined.

Despite this, many steps have already been done in the direction of the improvement of the performance of unsupervised ANNs with the aid of ensembles. The use of ensembles is one of the most spread techniques for increasing the stability and performance of an analysis model [59]. Some of the most recent applications of ensembles in unsupervised learning includes new models based on topology preserving maps [11], [37], [51], [60], [61].

2.5.1. Previously Proposed Models for SOM Ensemble Summarization

When using the SOM as an analysis tool, the combination of results is easy, as there is not a real need for human interpretation of the intermediate results, since in the majority of cases the interesting result is the final one. The problem of this type of combinations arises when the desired output is not a numeric prediction or a degree of membership to a class, but the plain 2-D representation of a multi-dimensional dataset for easy inspection by humans.

Representing all the networks in a simple image can only be useful when dealing with 1-D maps but gets too messy when visualizing 2-D maps. In that case, the expected output would not be a value yield by the ANN, but the map generated by the ANN itself. A much more complex fusion algorithm is needed. It is necessary to devise a system that would calculate a final topography preserving map that somehow summarizes the best aspects of the maps composing the ensemble.

To summarize things, the desired outcome of the ensemble combination process is a single map that unites the added performance improvement and stability characteristics of the use of ensembles with the simplicity and clarity of representing multi-dimensional data in a map for its inspection.

There have previously been some works [61]–[63] devoted to the development of a summarization of maps to obtain a final map. Three of them are the following:

- **Map Fusion by Euclidean Distance**: It propose is to obtain a fusion of an ensemble of maps by first "aligning" the neurons and then by calculating the centroid of the vectors corresponding to neurons aligned to the same position. These centroids are calculated as the simple sum of all vectors (see Eq. (2.37)).

  $$w_c = \frac{1}{|W_k|} \sum_{w_i \in W_k} w_i \tag{2.37}$$

  To determine how this alignment can be done, the authors apply the following idea:
  Let $\mathcal{L}_{ij}$ denote the set of feature vectors that are assigned to the neuron w_i^j, which is the set of training inputs to which the neuron, as the BMU in the map, is reacting. Neurons that are close

to each other in the $\Re^{N_w}$ should be similar. That is, let w_1 and w_2 be the neurons whose respective sets $\mathcal{L}_1^1$ and $\mathcal{L}_3^2$ contain more common features than any other neuron couple. Then, the reference vectors corresponding to these neurons will, in all probability, be closer to each other than another possible neuron combination under Euclidean distance.

- **Map Fusion by Voronoi Polygons Similarity**: It proposed a SOM fusion model that is mainly aimed to learn data topologies in a more precise way than the single SOM model. The Fusion-SOM model is an ensemble of Self-Organizing Maps that are combined by fusing prototypes that are modelling similar Voronoi polygons (partitions) and the neighborhood relation are given by the edges that measures the similarity between the fused nodes. The aim of combining the SOM is to improve the quality and robustness of the results. In order to calculate to which degree, the partition of data that two different neurons are representing is overlapping - and therefore, to which degree they can be considered similar - a binary vector of the same size of the dataset used to train the network is associated to each neuron of a map. This vector contains a "1" in the position of data that was recognized by the neuron and "0" in the positions of data that was not recognized. This vector will serve to compute the dissimilarity between two neurons by using Eq. (2.38).

$$ds(b_r, b_q) = \frac{\sum XOR(b_r, b_q)}{\sum OR(b_r, b_q)} \sum_{w_i \in W_k} w_i \tag{2.38}$$

being r and q the neurons to determine their dissimilarity and b_r and b_q the binary vectors relating each of the neurons with the data samples recognized by it. This vector will also serve to calculate the usage of each of the neurons, that will be used in the fusion algorithm. Those neurons with a recognition rate lower than a given threshold, are discarded for the ensemble fusion calculations. After eliminating the poor reacting ones; the remaining neurons of all the map are considered, altogether, for a clustering of neurons process; that groups neurons with similar Voronoi polygons in the same sets. sets become neurons in the final map, by calculating the centroids of its composing neurons (Eq. (2.37)). Finally, to reconstruct the map lattice; the connections between the neurons in the fused map must be recalculated. Neurons obtained from fusing clusters that are similar enough, are considered neighbors in the final map.

- **Weighted Voting Superposition**: The aim is to obtain the final units of the map by a weighted voting among the units in the same position in the different maps, in accordance to a quality measure. This measure can be any measure found in literature, as long as it can be calculated in a unit by unit basis. The voting process used is the one described in Eq. (2.39).

$$V_{p,m} = \frac{\sum b_{p,m}}{\sum_{i=1}^{M} b_{p,i}} \cdot \frac{q_{p,m}}{\sum_{i=1}^{M} q_{p,i}} \tag{2.39}$$

where $V_{p,m}$ is the weight of the vote for the unit included in map m of the ensemble, in its position p, M is the total number of maps in the ensemble, $b_{p,m}$ is the binary vector used for marking the dataset entries recognized by unit in position p of map m, and $q_{p,m}$ is the value of the desired quality measure for unit in position p of map m.

b is a binary vector of the same length as data samples are in the dataset; that is used to store the samples recognized by a single unit. So, the first term of the equation accounts for the recognition rate of a unit in a map in relation with the total units recognized in the same position on all the maps of the ensemble. Following the same reasoning, the second term accounts for the quality of a unit in a map in relation with the overall quality obtained by the units in the same position in all maps.

2.6. *Exploratory Projection Pursuit*

Exploratory Projection Pursuit (EPP) [64] is a statistical method aimed at solving the difficult problem of identifying structure in high dimensional data. It is done by projecting the data onto a low dimensional subspace in which structure may be visually identified. As not all projections will reveal the data's structure equally well, it defines an index that measures the extent to which a given projection is "interesting", and then represents the data in terms of projections that maximize that index.

Then, the first step for EPP is to define which indices represent interesting directions. Concerning projections, "interestingness" is usually defined with respect to the fact that most projections of high-dimensional data give almost Gaussian distributions [65]. Thus, those directions which reveal data-projections that are as far from the Gaussian as possible should be found in order to identify the most "interesting" features of the data. Transforming the data into a zero mean and identity covariance matrix enables a network to respond solely to higher order statistics, which can be used to look for interesting structure in the data.

There are two simple statistical measures of deviation from a Gaussian distribution, Skewness and Kurtosis, which are based on the higher order moments of the distribution. The first, Skewness, is based on the normalized third central moment of the distribution and measures the deviation of the distribution from bilateral symmetry. The second, Kurtosis, is based on the normalized fourth central moment of the distribution and measures the heaviness of the tails of a distribution. A bimodal distribution will often have a negative kurtosis and therefore negative kurtosis can signal that a particular distribution contains clustering.

Because a Gaussian distribution with mean a and variance x is no more interesting than a Gaussian distribution with mean b and variance y —indeed this second order structure can obscure higher order and more interesting structure— this information needs to be removed from the data. This is known as "sphering" or "whitening". That is, the raw data is translated so that its mean is zero, projected onto the principal component directions and multiplied by the inverse of the square root of its eigenvalue to give data that has a mean of zero and is of unit variance in all directions.

Thus, if we wish to find a direction which maximizes the kurtosis of the distribution which is measured by s^4, we will use a function $f(s)\approx s^3$ in the algorithm. If we wish to find the direction with maximum skewness, we use a function $f(s)\approx s^2$ in the algorithm.

2.6.1. Nonlinear Principal Component Analysis

Non-linear PCA is a fairly popular nonlinear method [66]–[68], which has been applied to Independent Component Analysis (ICA) [69] quite successfully by a number of authors [66], [70]–[72]. ICA is a special case in the world of independence seeking networks in which linear mixtures of independent signals at the inputs of a network are to be completely separated at the outputs. The dimensionality of the inputs generally needs to be the same as the dimensionality of the outputs. [66] introduced a nonlinear extension to Oja's Subspace Algorithm [44] by means of Eq. (2.40).

$$\Delta W_{ij} = \eta \cdot \left(x_j f(y_i) - f(y_i) \sum_k W_{kj} f(y_k) \right) \tag{2.40}$$

which can be derived as an approximation to the best non-linear compression of the data as follows.

Starting with the cost function of Eq. (2.41).

$$J(W) = 1^T E\left\{\left(x - Wf(W^T x)\right)^2\right\} \tag{2.41}$$

the objective is to minimize the sum of the squared representation errors for the vector x. By taking the instantaneous gradient of this and implementing a stochastic gradient descent it may be derived a weight update rule (see Eq. (2.42)).

$$\Delta W \propto \frac{-\delta J(W)}{\delta(W)} \propto \eta \cdot \left(xe^T W f'(x^T W) + ef(x^T W)\right) \tag{2.42}$$

where e is the error vector, $e = \left(x - Wf(W^T x)\right)$, and $f'(x^T W)$ is the element-wise derivative of $f(x^T W)$ with respect to W. [66] argue that the term $xe^T W f'(x^T W)$ has a negligible effect on the learning compared to the second term, $ef(x^T W)$, and so can be neglected. So, the weight update rule can be expressed through Eq. (2.43).

$$\Delta W = \eta \cdot \left(\left(x - Wf(W^T x)\right) \cdot f(x^T W)\right) \tag{2.43}$$

This rule has been used to extract higher order structure than the PCA network, for instance the Independent Components of data [66] [70].

2.6.2. The Exploratory Projection Pursuit Neural Network

The EPP neural network [73] is essentially a nonlinear modification of Oja's subspace algorithm [44], which can be described by the following set of equations (Eq. (2.44), (2.45), (2.46) and (2.47)).

$$s_i = \sum_{j=1}^{N} W_{ij} x_j \tag{2.44}$$

$$e_j = x_j - \sum_{k=1}^{M} W_{kj} s_k \tag{2.45}$$

$$r_i = f(s_i) \tag{2.46}$$

$$\Delta W_{ij} = \eta \cdot r_i \cdot e_j \tag{2.47}$$

where x_j is the sphered activation of the *j*th input, s_i is the activation of the *i*th output neuron, W_{ij} are the weights between the former and the latter, and r_i is the value of the function $f(\,)$ on the *i*th output neuron.

Initially there is no activation in the network. The input data is feedforward via the weights to the output neurons where a simple summation is performed. The activations of the output neurons are fed back via the same weights to the input neurons as inhibition and are therefore subtracted. Then a (nonlinear) function of the weights is calculated and used to update the weights by applying the simple Hebbian learning rule.

It was shown in [66] that the use of a nonlinear function $f(\,)$ in the above equations creates an algorithm to find those values of W that maximize that function, subject to the constraint that W is an orthogonal matrix. This idea was applied in [73] to the previously described network in the context of the network performing EPP. Thus, the function $f(s) \approx s^3$ is applied to the algorithm if we wish to find the direction which maximizes the kurtosis of the distribution that is measured by s^4; similarly, the function $f(s) \approx s^2$ is applied to find the direction with maximum skewness.

2.6.3. Maximum Likelihood Hebbian Learning

Maximum Likelihood Hebbian Learning [11], [15], [74], [75] is a EPP neural model of a family of rules based on exponential distributions, which can be derived expressing the Probability Density Function (PDF) of the residual after feedback as Eq. (2.48).

$$p(e) = \frac{1}{Z} exp(-|e|^p) \tag{2.48}$$

where *Z* is a factor that normalizes the integral of *p(y)* to unity, *e* is the residual and *p* is a parameter related to the energy function.

Then, a general cost function associated with the Maximum Likelihood Hebbian Learning network can be denoted as Eq. (2.49).

$$J = E\left(-log\big(p(e)\big)\right) = E(|e|^p + K) \tag{2.49}$$

where *K* is a constant independent of *W* (weights of the MLHL network) and the expectation is taken over the input dataset. Therefore, the gradient descent *J* is presented in Eq. (2.50).

$$\Delta W \propto -\frac{\partial J}{\partial W}\bigg|_{W_{(t-1)}} = -\frac{\partial J}{\partial e}\frac{\partial e}{\partial W}\bigg|_{W_{(t-1)}} \approx E\left\{y\big(p|e|^{p-1}sign(e)\big)^T\Big|_{W_{(t-1)}}\right\} \tag{2.50}$$

where *E{}* is the expected value operator, $W_{(t-1)}$ are the weights at time instant *t-1* and *T* denotes the transpose of a vector. The power of the norm of *e* is taken on an element wise basis as it is derived from a scalar form of the vector.

If the conditions of stochastic approximation [76] are satisfied, the mean can be approximated with a difference equation. The function to be approximated is clearly sufficiently smooth and the learning rate can be designed to approximately $\eta_k \geq 0$, $\sum_k \eta_k = \infty$, $\sum_k \eta_k^2 < \infty$ and so we have the rule (Eq. (2.51)).

$$\Delta W_{ij} = \eta \cdot y_i \cdot sign(e_j) \cdot |e_j|^{p-1} \tag{2.51}$$

where ΔW_{ij} is the weights update, e the residual, η is the learning rate and *y* is the output of the network.

It is expected that for leptokurtic residuals (more kurtotic than a Gaussian distribution), values of *p*<2 would be appropriate, while for platykurtic residuals (less kurtotic than a Gaussian), values of *p*>2 would be appropriate. Finally, the network operation can be expressed by Eq. (2.52), (2.53) and (2.54).

$$\text{Feedforward: } \mathrm{y_i} = \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{W_{ij}x_j}\,,\ \forall i \tag{2.52}$$

$$Feedback\text{: } e_j = x_j - \sum_{i=1}^{M} W_{ij}y_i \tag{2.53}$$

$$Weights\,update\text{: } \Delta W_{ij} = \eta \cdot y_i \cdot sign(e_j)|e_j|^p \tag{2.54}$$

2.6.4. Cooperative Maximum Likelihood Hebbian Learning

Cooperative Maximum Likelihood Hebbian Learning (CMLHL) is based on the EPP neural model Maximum Likelihood Hebbian Learning. The main difference between these two models is that CMLHL includes lateral connections [77], [78] derived from the Rectified Gaussian Distribution (RGD) [79]. The RGD is a modification of the standard Gaussian distribution in which the variables are constrained to be non-negative, enabling the use of non-convex energy functions. In a more precise way, CMLHL includes lateral connections based on the mode of the cooperative distribution that is closely spaced along a nonlinear continuous manifold. By including these lateral connections, the resulting network can find the independent factors of a dataset in a way that captures some type of global ordering in the dataset.

Considering an *N*-dimensional input vector x, an *M*-dimensional output vector y and with W_{ij} being the weight (linking input j^{th} to output i^{th}), CMLHL can be expressed by means of Eq. (2.55), (2.56), (2.57) and (2.58).

$$\text{Feedforward: } y_i = \sum_{j=1}^{N} W_{ij} x_j \,, \;\; \forall i \tag{2.55}$$

$$\text{Lateral activation passing: } y_i(t+1) = \left[y_i(t) + \tau(b - Ay)\right]^{+} \tag{2.56}$$

$$Feedback\text{: } e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{2.57}$$

$$Weights\ update\text{: } \Delta W_{ij} = \eta \cdot y_i \cdot sign(e_j)\left|e_j\right|^{p} \tag{2.58}$$

where η is the learning rate, τ is the "strength" of the lateral connections, b the bias parameter, and p a parameter related to the energy function.

A is a symmetric matrix used to modify the response to the data whose effect is based on the relation between the distances among the output neurons. It is based on the cooperative distribution, but to speed learning up, it can be simplified to Eq. (2.59).

$$A(i,j) = \delta_{ij} - \cos(2\pi(i-j)/M) \tag{2.59}$$

where δ_{ij} is the Kronecker delta.

The application of CMLHL, initially in the field of artificial vision [77], [78], and subsequently to other interesting topics [75], [80]–[83], has proven that this model can successfully perform data visualization.

2.7. *Maximum Likelihood Scale Invariant Maps (MLHL-SIM)*

The Maximum Likelihood Scale Invariant Map (MLHL-SIM) [84] is an extension of the SIM based on the application of the Maximum Likelihood Hebbian Learning (MLHL).

The main difference with regards to the SIM is how the MLHL is used to update the weights of all nodes in the neighborhood of the winner, once the winner has been updated. This can be expressed by means of Eq. (2.60).

$$\Delta w_i = h_{ci} \cdot \eta \cdot sign(e - W_c) \cdot |e - W_c|^{p-1}, \forall i \in N_c, \tag{2.60}$$

By giving different values to p, the learning rule is optimal for different probability density functions of the residuals. h_{ci} is the neighborhood function as in the case of the SOM and N_c is the number of output neurons. Finally, η represents the learning rate.

During the training of the SIM or the MLHL-SIM, the weights of the winning node are feedback as inhibition to the input vector, and then in the case of the MLHL-SIM, MLHL learning is used to update the weights of all nodes in the neighborhood of the winner as explained above.

Chapter 3. A novel Family of Learning Rules: Beta Hebbian Learning

3.1. *Introduction*

In this chapter a novel family of learning rules called Beta Hebbian Learning (BHL) is thoroughly investigated to extract information from high dimensional datasets by projecting the data onto low dimensional (typically 2 dimensional) subspaces, improving the existing exploratory methods by providing a clear representation of data's internal structure. BHL applies a family of learning rules derived from the Probability Density Function (PDF) of the residual based on the beta distribution. This family of rules may be called Hebbian in that all use a simple multiplication of the output of the neural network with some function of the residuals after feedback. The derived learning rules can be linked to an adaptive form of Exploratory Projection Pursuit and with artificial distributions, the networks perform as the theory suggests they should: the use of different learning rules derived from different probability density functions allows the identification of "interesting" dimensions (as far from the Gaussian distribution as possible) in high dimensional datasets. This novel algorithm, BHL, has been tested over 7 artificial datasets to study the behavior of BHL parameters, and was later applied successfully over 4 real datasets, comparing its results, in terms of performance, with other well-known Exploratory and projection models such as Maximum Likelihood Hebbian Learning, Locally-Linear Embedding, Curvilinear Component Analysis, Isomap and Neural Principal Component Analysis.

3.2. *Beta Distribution*

Beta distribution is a family of continuous probability distributions defined in the interval [0, 1] with two positive shape parameters, denoted by α and β. Beta distribution is defined by Eq. (3.1).

$$f(x;\alpha,\beta)=\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{3.1}$$

where *f(x;α,β)* is the PDF, x is the input value to the distribution, α and β are the parameters that determine the shape of the PDF curve, and *B(α,β)* is the beta function, which is a normalization constant to ensure that the total probability integrates to 1. The Beta function is calculated using the gamma function (Eq. (3.2)) and defined by Eq. (3.3).

$$\Gamma(n)=(n-1)! \tag{3.2}$$

$$B(\alpha,\beta)=\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{3.3}$$

Beta distribution is very malleable based on the parameters α and β (see Fig. 3.1). The relation between parameters α and β determines the shape of the PDF, with the capability of generating distributions with positive (α>β) and negative (α<β) skewness, platykurtic (α=β smalls values <3), mesokurtic (α=β ≈3) and leptokurtic (α=β large values >3) distributions, and combinations of these.

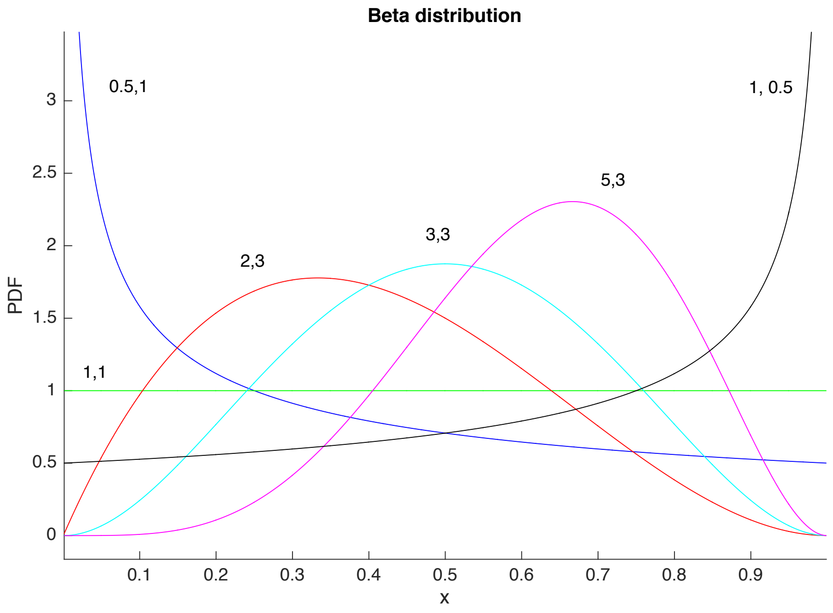


Fig. 3.1 Probability density function of the beta distribution for different values of α and β.

3.3. ***Beta Hebbian Learning Algorithm***

This thesis presents a novel family of learning rules derived from the PDF of the residual based on Beta distribution. This novel family, called Beta Hebbian Learning (BHL), is thoroughly investigated and the stability of its learning rule is also analyzed for first time.

In general, the minimization of the cost function associated with this network, may be thought to make the probability of the residuals more dependent on the PDF of the residuals. Thus, if the probability density function of the residuals is known, this knowledge could be used to determine the optimal cost function. So, the residual ($e = x - Wy$) is draw from the Beta distribution (Eq. (3.1)), with the following probability density function (Eq. (3.4)):

$$p(e) = e^{\alpha-1}(1-e)^{\beta-1} = (x - Wy)^{\alpha-1}(1 - x + Wy)^{\beta-1} \tag{3.4}$$

Where α and β are the parameters that determines the shape of the PDF curve of the Beta distribution, x is the input to the network, W is the weight vector associated with network neurons and y is the output to the network.

Then, to maximize the likelihood of the data with respect to the weights, the gradient descent is performed using Eq. (3.5):

$$\frac{\partial p}{\partial W} = \left(e_j^{\alpha-2}(1-e_j)^{\beta-2}\left(-(\alpha-1)(1-e_j)+e_j(\beta-1)\right)\right) = \left(e_j^{\alpha-2}(1-e_j)^{\beta-2}\left(1-\alpha+e_j(\alpha+\beta-2)\right)\right) \tag{3.5}$$

For instance, in the case in which *α=β=2,* Eq. (3.5) is simplified to Eq. (3.6).

$$\frac{\partial p}{\partial W} = y(-(1-e)+e) = y(2e-1) \tag{3.6}$$

In the case of the BHL, by maximizing the likelihood of the residual with respect to the actual distribution, the learning rule is matched to the PDF of the residual. The BHL may also be linked to the standard statistical method of Exploratory Projection Pursuit, as the nature and quantification of the interestingness is in terms of how likely the residuals are under a particular model of the PDF of the residuals.

Researchers from the community investigating Independent Component Analysis [85] have shown that it is less important to get exactly the correct distribution when searching for a specific source than it is to get an approximately correct distribution i.e. all supergaussian signals can be retrieved using a generic leptokurtic distribution and all subgaussian signals can be retrieved using a generic platykurtic distribution. The experiments in this thesis seem to support this to some extent but accuracy and speed of convergence are generally improved with the appropriate choices of α and β, in the same way as MLHL does in relation with p value.

Therefore, the new neural architecture is defined as follows (Eq. (3.7), (3.8) and (3.9)):

$$\text{Feedforward: } y_i = \sum_{j=1}^{N} W_{ij}x_j\,,\ \forall i \tag{3.7}$$

$$\textit{Feedback: } e_j = x_j - \sum_{i=1}^{M} W_{ij}y_i \tag{3.8}$$

$$\textit{Weights update: } \Delta W_{ij} = \eta\left(e_j^{\alpha-2}(1-e_j)^{\beta-2}\left(1-\alpha+e_j(\alpha+\beta-2)\right)\right)y_i \tag{3.9}$$

Where α and β are the parameters that determines the shape of the PDF curve of the Beta distribution, x is the input to the network, W is the weight vector associated with network neurons, e is the residual and y is the output to the network.

Following, in algorithm 1 it is presented the pseudocode of BHL.

Algorithm 1 BHL algorithm

```
1: function BHL($data$,$\alpha$,$\beta$,$\eta$,$iters$)
2:     $nData := normalize(data)$
3:     $\langle i, j\rangle := getSize(nData)$
4:     $W := rand(i, j)$                                  ▷ Get a random weight matrix
5:     $\eta Step := \eta/iters$
6:     for each $iter \in iters$ do
7:         $x := selectRandomData(nData, 1)$               ▷ Get a random sample data
8:         $y := W \times x$                                ▷ Feedforward
9:         $e := x - (W^T \times y)$                        ▷ Feedback
10:        for each $e_j \in e$ do                          ▷ Apply the learning rule
11:            $lRule_j := e_j^{\alpha-2}(1 - e_j)^{\beta-2}(1 - \alpha + e_j(\alpha + \beta - 2))$
12:        end for
13:        $\Delta W := \eta(y \times lRule^T)$             ▷ Weights update
14:        $W := W + \Delta W$
15:        $\eta := \eta - \eta Step$
16:    end for
17:    $pData := data \times W^T$                          ▷ Data projected onto new subspace
18:    $graphData(pData)$
19:    return $\langle W, pData\rangle$
20: end function
```

3.4. ***Stability of the Beta Hebbian Learning***

The stability of the learning rule was also analyzed in this study, and based on such analysis it can be concluded that the BHL algorithm is stable when the residuals have positive values lower than 1 (see Fig. 3.2). When values of the residuals are beyond this limit, the value of the weights update tends towards infinity. To avoid the possibility of the residuals having values higher than 1, the datasets should be normalized in order to satisfy this limitation and preserve the internal topology between dataset dimensions.

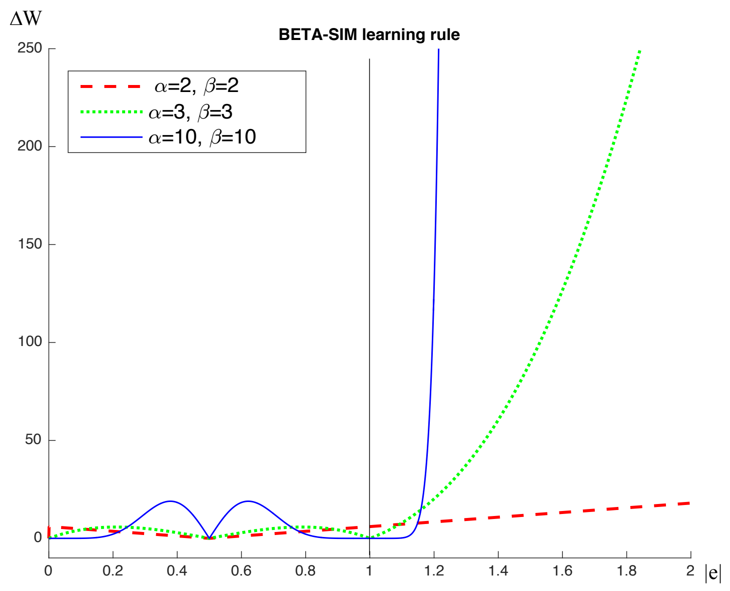


Fig. 3.2 Stability of the BHL learning rule (it becomes unstable for values of |*e*|>*1*).

3.5. *Experiments for BHL*

In this section, BHL is initially tested over 7 artificial datasets to study the behavior of its parameters, α and β. Later, it is successfully applied over 4 real datasets to evaluate its capabilities and to compare it with other projection and EPP methods.

3.5.1. Artificial Datasets for BHL

From a statistical point of view, the most interesting directions are those which are as non-Gaussian as possible, [65] points out that a typical random projection of a dataset is Gaussian. To identify the interesting features in data, we should investigate the interesting directions. Therefore, to illustrate our novel method (BHL), we follow the work of [15] in creating 7 artificial datasets, each containing 9 dimensions. For instance, a dataset containing 8 independent Gaussian dimensions and one leptokurtic dimension, is a typical dataset in which one might search for interestingness: most directions through the 9 dimensional dataset will be approximately Gaussian while there will be a single direction which is positively kurtotic (as shown in dataset 3).

In dataset 1 we have 8 independent leptokurtic dimensions and one Gaussian dimension; this is almost the opposite of the standard EPP datasets described in [15] and is rather far from being a typical dataset in that most projections onto its natural basis are interesting.

As this thesis wishes to thoroughly investigate the possibilities of the new BHL algorithm, it is a good test to investigate as many combinations as possible of different dimensions (Gaussian, leptokurtic and platykurtic). Results of the experiments are represented using Hinton Maps (or Hinton diagrams) [86] of the elements of W, where the area occupied by a square is proportional to each element of W, and the color indicates its sign (positive/negative – black/white).

- In dataset 1, we have 8 leptokurtic dimensions and one Gaussian.
- In dataset 2, we have 8 platykurtic dimensions and one Gaussian.
- In dataset 3, we have 8 Gaussian dimensions and one leptokurtic.
- In dataset 4, we have 8 Gaussian dimensions and one platykurtic.
- In dataset 5, we have 8 leptokurtic dimensions and one beta distribution with $\alpha=\beta=2$.
- In dataset 6, we have 8 platykurtic dimensions and one beta distribution with $\alpha=\beta=10$.
- In dataset 7, we have 7 Gaussian dimensions and one distribution with positive skewness.

In [87], only datasets like 1 to 4 were created but we considered that there is interest in identifying the structure in the other datasets too.

In all experiments, once selected the right combination α and β parameter, all results reported are based on a set of 10 simulations each with different initial conditions, in order to avoid "lucky runs". Finally, in the design of experiments and in the selection of the parameters, it was important to take into account that when $\alpha=\beta$, the fourth central moment (kurtosis) is restricted to values from -2 *($\alpha=\beta=0$)* to 0 ($\alpha=\beta=\infty$), expressed by the general Eq. (3.10), and when $\alpha=\beta$, through the Eq. (3.11).

$$Kurtosis = \frac{6[(\alpha-\beta)^2(\alpha+\beta+1)-\alpha\beta(\alpha+\beta+2)]}{\alpha\beta(\alpha+\beta+2)(\alpha+\beta+3)} - 3 \tag{3.10}$$

$$Kurtosis\ (\alpha=\beta) = -\frac{6}{3+2\alpha} \tag{3.11}$$

3.5.1.1. *Dataset 1: 8 leptokurtic dimensions and one Gaussian*

In this dataset, we have 8 leptokurtic dimensions and one Gaussian dimension; this is almost the opposite of the standard EPP datasets described in [15] and is far from being a typical dataset in that most of the directions in terms of its natural basis are interesting. However, since the aim of this research is to investigate the new family of learning rules (BHL), it was deemed appropriate since we can easily see the results of our method. In this experiment, the objective is to identify the single Gaussian dimension and ignore the leptokurtic dimensions. The leptokurtic dimensions are characterized as having long tails; if a residual can be created by removing the Gaussian direction from the dataset, the residual will automatically be leptokurtic. Thus, the strategy is to maximizing the likelihood of the residual using the updating rule associated with the BHL (see Eq. (3.9)).

After experimentation with different values of $\alpha=\beta$ (as the skewness is 0), the final simulations were performed with *$\alpha=\beta=2$.* Typical results are shown in Fig. 3.3.

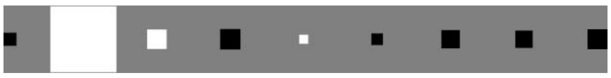


Fig. 3.3 The Gaussian direction was the second among 8 leptokurtic dimensions. It has clearly been identified in this Hinton map of the weights when *$\alpha=\beta=2$*.

By using appropriate values of α and β parameters ($\alpha=\beta=2$), the obtained model maximizes the likelihood of platykurtic residuals, in this case the Gaussian dimension. This is shown in the results of the Hinton map (see Fig. 3.3), where it can be seen that the highest values of the weights are assigned to the Gaussian dimension (dimension with lowest kurtosis) and the leptokurtic dimensions are associated with smaller weights.

3.5.1.2. *Dataset 2: 8 platykurtic dimensions and one Gaussian*

This dataset contains 8 platykurtic dimensions and one Gaussian dimension. As above the objective is to identify the Gaussian dimension and an appropriate combination of values of α and β must use for the learning rule. Since if the Gaussian dimension is removed, it will result in a platykurtic residual, the strategy is to maximize the likelihood of a platykurtic residual under the model. As in the previous case, different values of $\alpha=\beta$ were tested, reporting the results of $\alpha=\beta=4$ to study the behavior of the BHL model. If we select $\alpha=\beta=4$, typical results are shown in Fig. 3.4, where it can be seen that the Gaussian dimension is clearly identified.

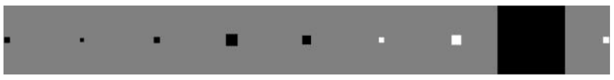


Fig. 3.4 The Gaussian direction was the eighth among 8 platykurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=4$.

3.5.1.3. <u>Dataset 3: 8 Gaussian dimensions and one leptokurtic</u>

This dataset contains 8 Gaussian dimensions and one leptokurtic dimension. This dataset is more typical in datasets in which one might search for interestingness: most directions through the 9 dimensional data will be approximately Gaussian while there will be a single direction with maximum positive kurtosis. Following the successful strategy used on datasets 1 and 2, we might suggest that maximizing the likelihood of the residuals under a leptokurtic model would be optimal. The empirical finding is in line with this criterion, allowing us to identify the leptokurtic dimension. In this case, the objective is to identify the single leptokurtic dimension and ignore the Gaussian dimensions.

After experimenting with a number of values of $\alpha=\beta$, final simulations were made with $\alpha=\beta=10$. The best result is shown in Fig. 3.5; the leptokurtic direction is clearly identified.

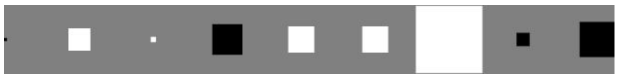


Fig. 3.5 The leptokurtic direction was the seventh among 8 Gaussian dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=10$.

3.5.1.4. <u>Dataset 4: 8 Gaussian dimensions and one platykurtic</u>

This dataset contains 8 Gaussian dimensions and one platykurtic dimension. A platykurtic model was used on this dataset thereby attempting to maximize the likelihood of the data under a negative kurtotic model ($\alpha=\beta=1.8$). The learning rules can therefore attempt to remove the platykurtic dimension, leaving as much kurtosis in the residuals as possible. The best results are shown in Fig. 3.6.

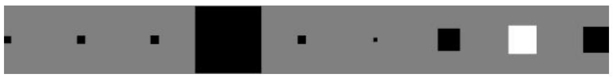


Fig. 3.6 The platykurtic direction was the fourth among 8 Gaussian dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=1.8$.

3.5.1.5. *Dataset 5: 8 leptokurtic dimensions and one Beta distribution with $\alpha=\beta=2$*

In dataset 5, there are 8 leptokurtic dimensions and one Beta distribution with $\alpha=\beta=2$, which has a kurtosis of -0.86, so it can be considered as a platykurtic distribution. Again, a platykurtic model is used to identify the Beta dimension. The best result is shown in Fig. 3.7 for which $\alpha=\beta=2$ is used.

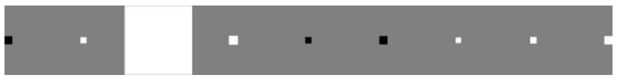


Fig. 3.7 The platykurtic direction was the third among 8 leptokurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=2$.

3.5.1.6. *Dataset 6: 8 platykurtic dimensions and one Beta distribution with $\alpha=\beta=10$*

In dataset 6, we have 8 platykurtic dimensions (kurtosis=-1.9) and one Beta distribution with $\alpha=\beta=10$ with kurtosis of -0.26, so it can be considered as a Gaussian distribution. A Gaussian model is now used to attempt to identify the Beta dimension. The best result is shown in Fig. 3.8 for which $\alpha=\beta=10$ is used.

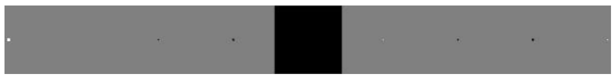


Fig. 3.8 The beta direction was the fifth among 8 platykurtic dimensions. It has clearly been identified in this Hinton map of the weights when $\alpha=\beta=10$.

3.5.1.7. *Dataset 7: 8 Gaussian dimensions and one distribution with positive skewness*

In this dataset, the aim is to investigate the effect of skewness when we apply the BHL algorithm, so 8 Gaussian dimensions (kurtosis=3 and skewness=0), and one distribution with positive skewness (kurtosis=3 and skewness=1) are used. Initially, values of $\alpha=\beta$ were used, with no good results. Then, combinations of $\alpha>\beta$ were tested, in order to see the effect of the distribution with positive skewness on the BHL. The best results were obtained using $\alpha=4$ and $\beta=10$ (see Fig. 3.9), where the dimensions with positive skewness are identified, but with some noise.

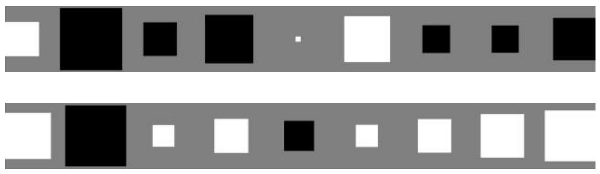


Fig. 3.9 The directions with positive skewness was the second (skewness=1) among 8 Gaussian dimensions with skewness=0. The direction with positive skewness has been identified in these Hinton maps of the weights when $\alpha=4$ and $\beta=10$.

3.5.2. Real Datasets for BHL

To show the power of the BHL family of learning rules, it was applied to different real datasets, with a diverse number of samples, features and classes belonging to different sectors of application. The results were compared with those obtained by other methods such as Neural Principal Component Analysis (Neural PCA) [44], Maximum Likelihood Hebbian Learning (MLHL) [15], [74], Locally-Linear Embedding (LLE) [88], Curvilinear component analysis (CCA) [89] and Isomap [90]. Table 3.1 shows a summary of the real datasets used in the experiments.

Table 3.1 Four real datasets characterized by different number of samples, features and classes belonging to different fields of application for BHL.

Dataset	Name	Number of sample	Number of features	Number of classes
1	Algae	118	18	10
2	Astronomical	65	115	13
3	Bank	800	11	Unknown
4	Dental milling	190	11	Unknown

The analysis of the results presented in the experiments is done following the EPP definition, where the identification of the internal structure of high dimensional datasets is based on the projection of the data onto lower dimensional subspaces, where its internal structure is search by visual inspection (by eye) [64].

3.5.2.1. Algae dataset

This real dataset comes from a research study of various forms of algae, some of which have been manually labeled. The dataset is formed by 118 samples and each sample is recorded as an 18 dimensional vector representing the magnitudes of various pigments [15]. Some algae have been identified as belonging to specific classes which are numbered from 1 to 9. Others remain unclassified and these are labelled as 0.

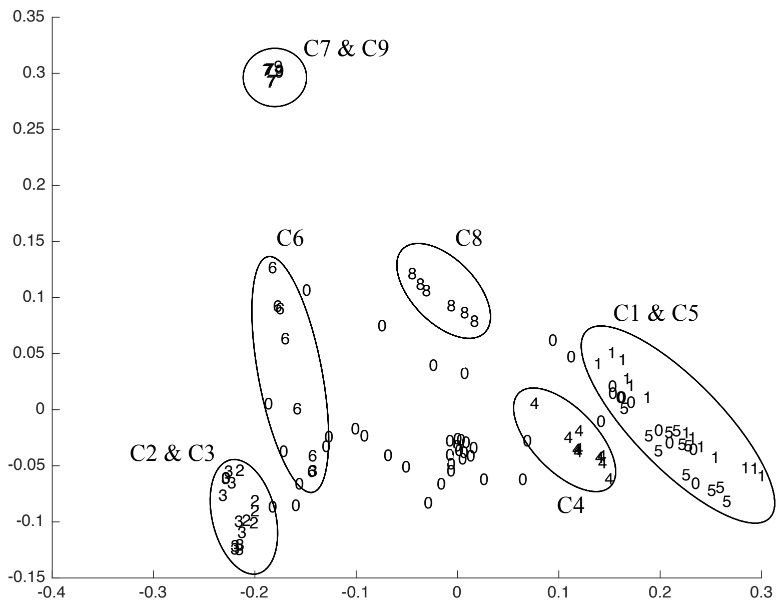


Fig. 3.10 Projection of the algae dataset onto the first two principal components using Neural PCA.

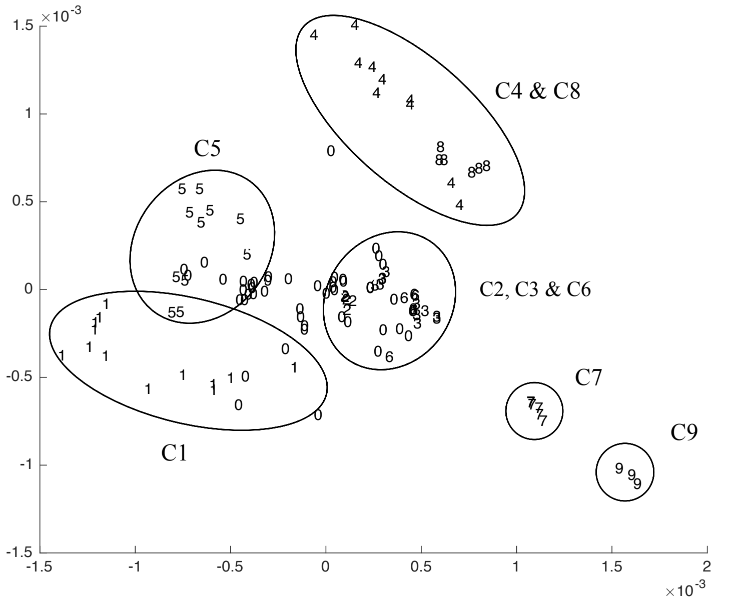


Fig. 3.11 Projection of the algae dataset onto the first two filters found using MLHL with *p = 1*.

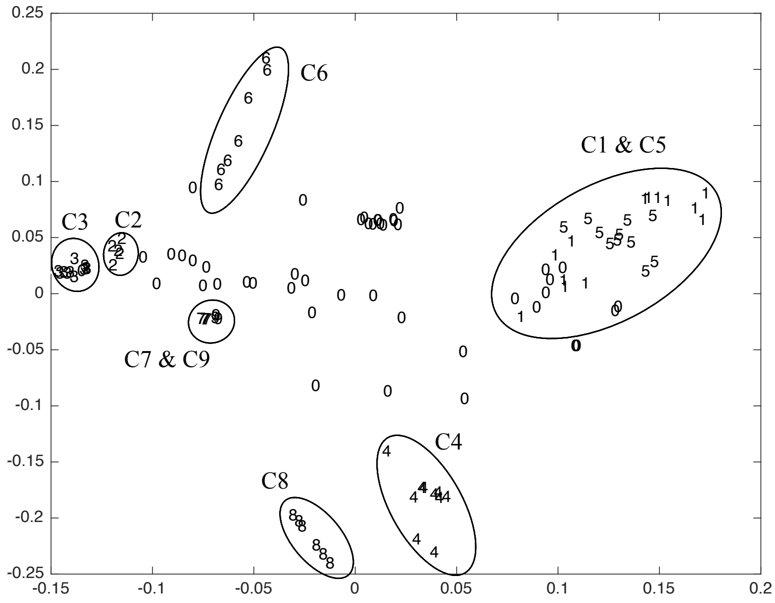


Fig. 3.12 2D projection of the algae dataset using LLE algorithm with *k=60*.

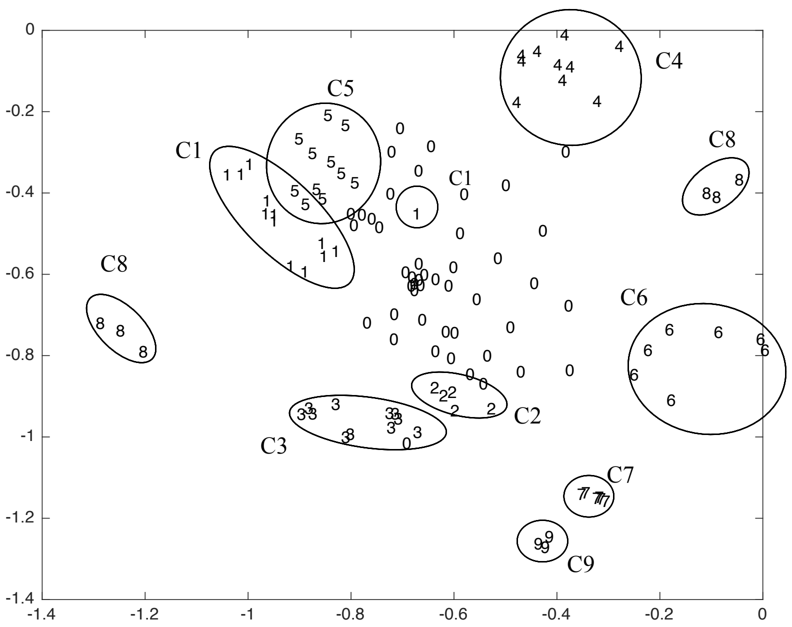


Fig. 3.13 2D projection of the algae dataset using CCA algorithm with 1,000 epochs.

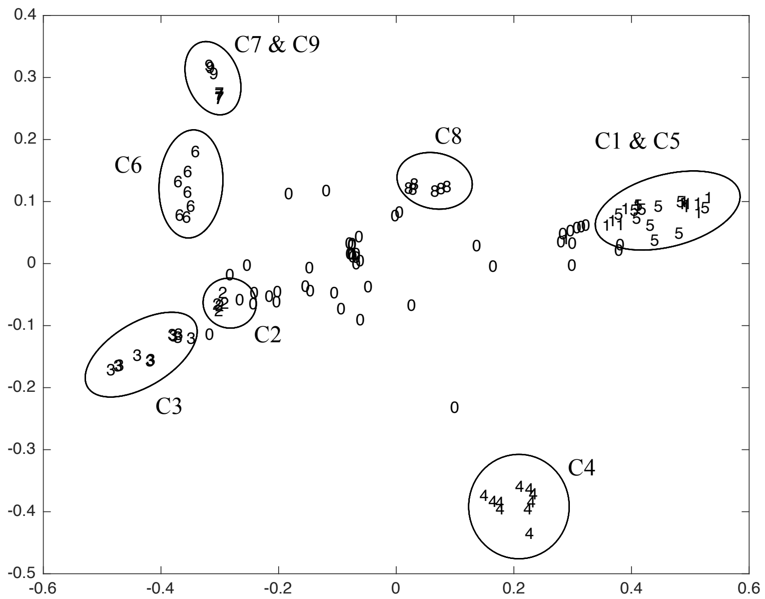


Fig. 3.14 2D projection of the algae dataset using Isomap algorithm with *k=12*.

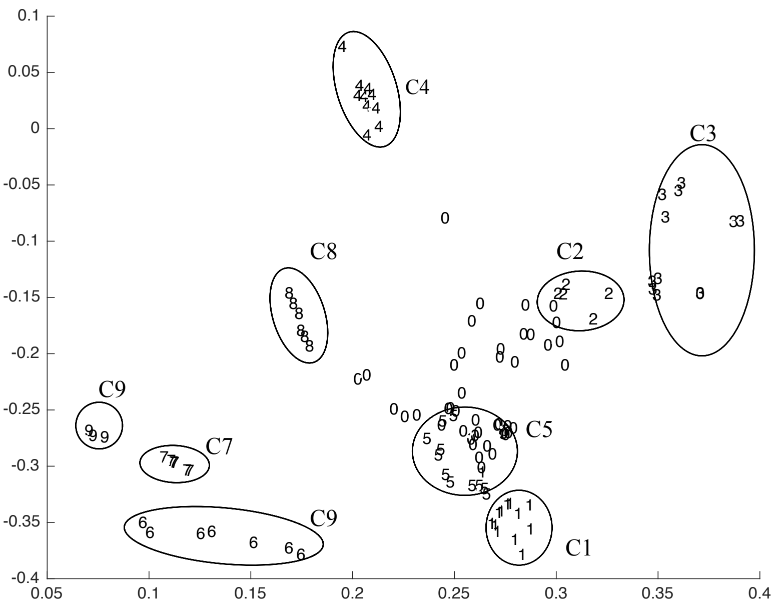


Fig. 3.15 Projection of the algae dataset onto the first two filters found using BHL with *α=4 and β=20*.

Fig. 3.10 shows a projection of this dataset onto the first two Principal Components, and Fig. 3.11 to Fig. 3.15 show the MLHL, LLE, CCA, Isomap and BHL results respectively. We can see by visual inspection that some separation of the classes has been achieved in all cases.

In the case of Neural PCA (Fig. 3.10), it is able to clearly identify three classes: C4, C6 and C8, but other classes seem to be mixed: C1 & C5, C2 & C3 and C7 & C9.

MLHL algorithm (Fig. 3.11) shows a clear identification of four classes: C1, C5, C7 and C9, while classes C2, C3 and C6 are mixed as well as classes C4 and C8.

LLE algorithm (Fig. 3.12) is able to separate five classes: C2, C3, C4, C6, and C8, but classes C1 & C5, and C7 & C9 are mixed.

CCA algorithm (Fig. 3.13) identifies all classes except C1 and C5 that are mixed. However, class C8 appears divided in two very separate clusters.

Isomap algorithm (Fig. 3.14) obtains similar results to LLE where classes C1 & C5 and C7 & C9 appear mixed.

Finally, Fig. 3.15 shows a projection of the same dataset onto the filters found using BHL with $\alpha=4$ and $\beta=20$; where the best separation (by visual inspection) of the individual classes has been achieved, been able to identify 9 out of 9 of classes.

By comparing the results obtained by all methods (Fig. 3.10 to Fig. 3.15), it can be noted that the BHL is the only method out of the six that has clearly separated all 9 classes of algae.

3.5.2.2. <u>Astronomical dataset</u>

The second real dataset consists of 65 color spectra of 115 asteroids of 13 classes (T, D, B, C, X, K, S, L, A, R, Q, V and O) used by [91]. The performance of a variety of artificial neural networks has been previously compared on this dataset [15].

The dataset is composed of a mixture of the 52-colour asteroid survey by [92] together with the 8-colour survey conducted by [93] providing a set of asteroid spectra spanning 0.3–2.5μm. When this extended dataset was compared by [91] to the results in [93] it was found that the additional refinement to the spectra lead to more classes in the taxonomy produced by [93].

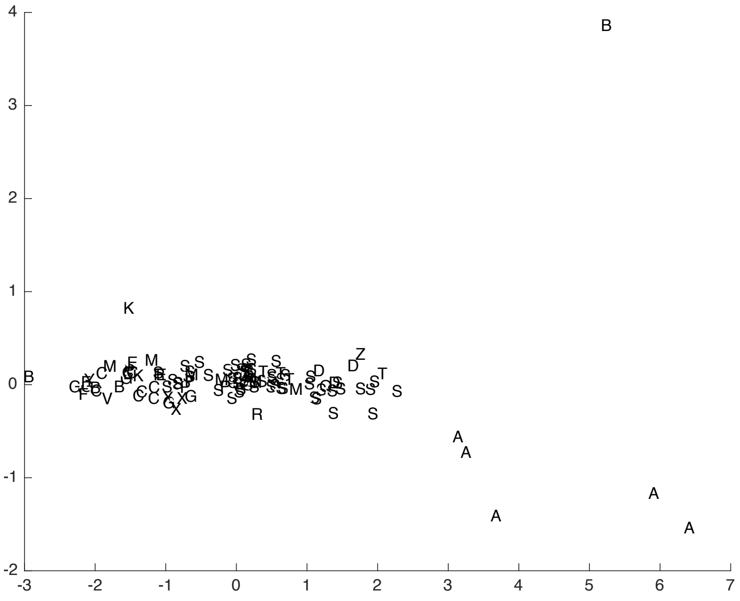


Fig. 3.16 Projection of the astronomical dataset onto the first two principal components using Neural PCA.

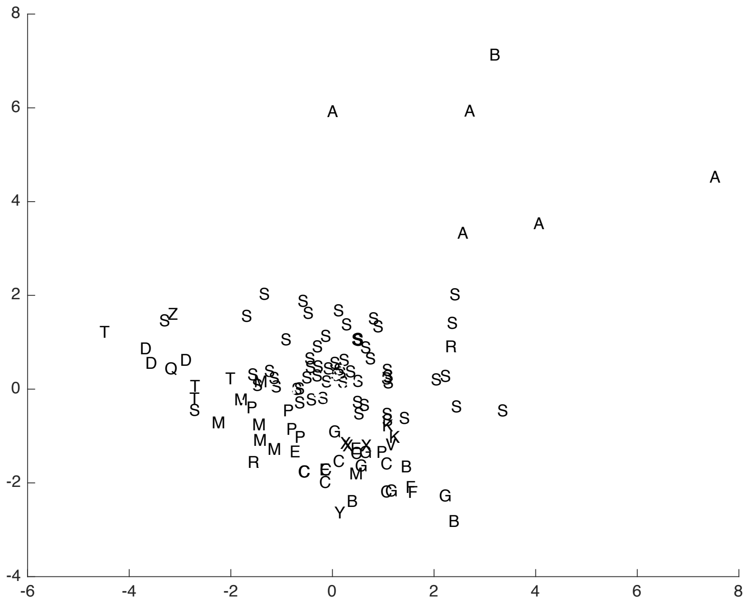


Fig. 3.17 Projection of the astronomical dataset onto the first two filters found using MLHL with *p = 0.5*.

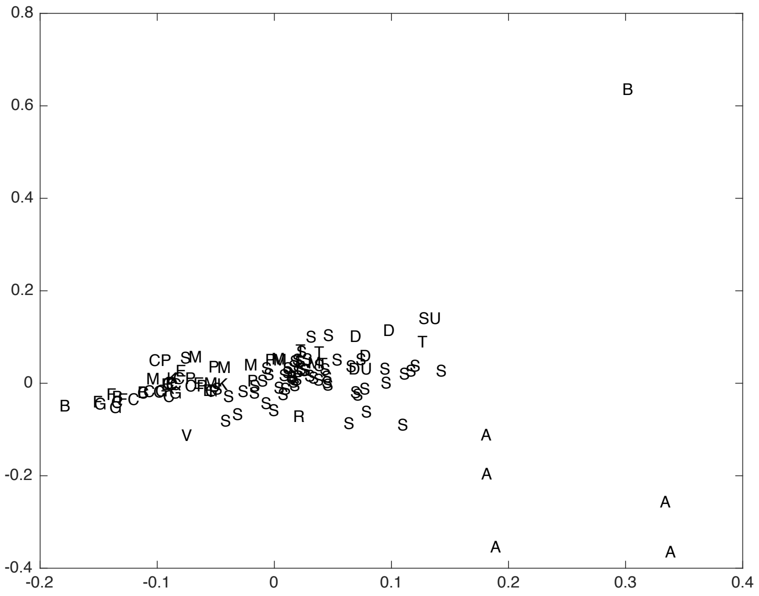


Fig. 3.18 2D projection of the astronomical dataset using LLE algorithm with *k=15*.

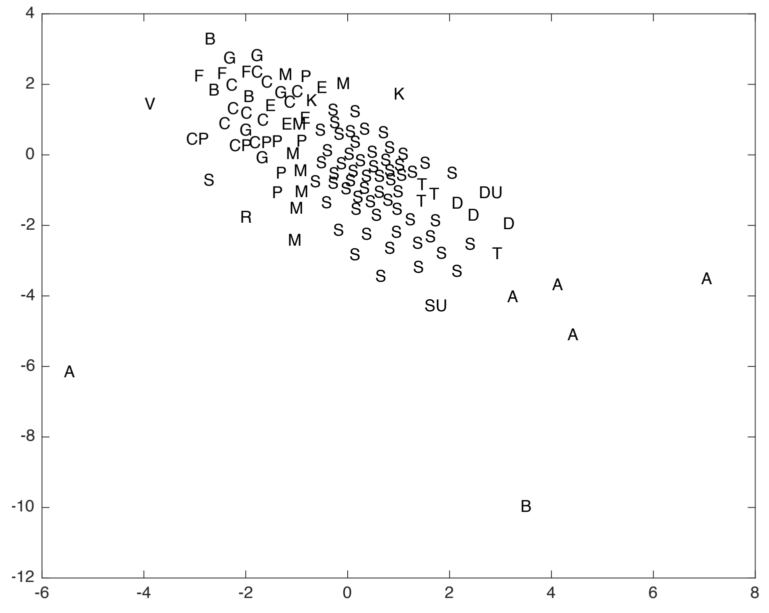


Fig. 3.19 2D projection of the astronomical dataset using CCA algorithm with 5,000 epochs.

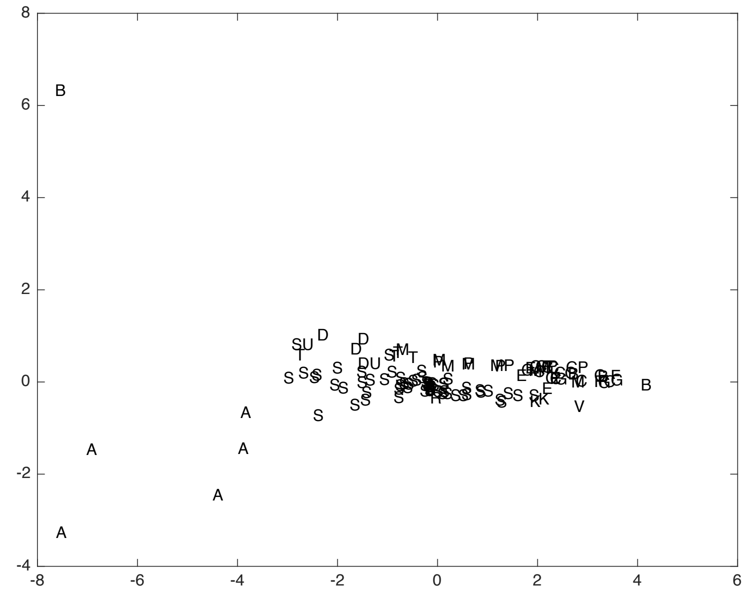


Fig. 3.20 2D projection of the astronomical dataset using Isomap algorithm with *k=12*.

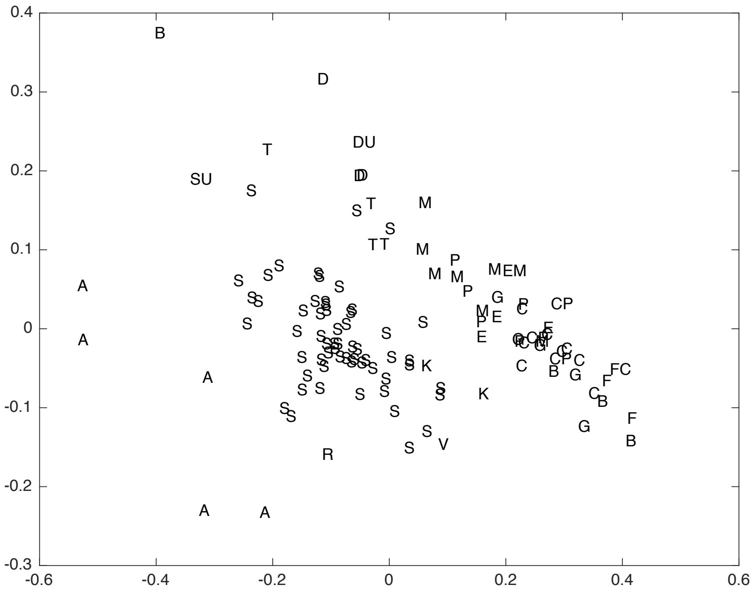


Fig. 3.21 Projection of the astronomical dataset onto the first two filters found using BHL with $\alpha=3.1$ and $\beta=10$.

The visual inspection of the results for this dataset shows that Neural PCA separates out the classes A and (some of) B but leaves most of the others in a single group (Fig. 3.16).

The MLHL algorithm with $p<2$, is able to identify again classes A and some of B, showing a better separation of the central cluster (Fig. 3.17 was from a simulation with $p=0.5$), where, for instance, most of samples of class "S" appear together.

LLE (Fig. 3.18) and Isomap (Fig. 3.20) algorithms present very similar results to Neural PCA, where only classes A and B are separated from the rest of the dataset.

In the case of the CCA algorithm (Fig. 3.19), data samples of the central cluster are more spread out, however there is not a clear separation into different clusters.

However, BHL (with α=*3.1* and β=*10*) obtains a better projection of the central cluster with a more spread out visualization of the data (Fig. 3.21) where most of the samples of class "S" are together and clearly separated from the other classes.

Therefore, comparing the three methods, it can be seen that all identify classes A and (some of) B easily, but only the BHL is able to spread the data out somewhat better than the other methods.

3.5.2.3. Bank dataset

In this experiment, the Neural PCA, MLHL, LLE, CCA, Isomap and BHL methods are applied on a small database of bank customers consisting of 800 records each having 11 fields [15]. Information held includes a unique identifier, age, sex, salary, type of area in which they live, whether they are married or not, number of children, if they have car or not, and then several fields of financial information such as type of bank account, whether they own a Personal Equity Plan (PEP) etc.

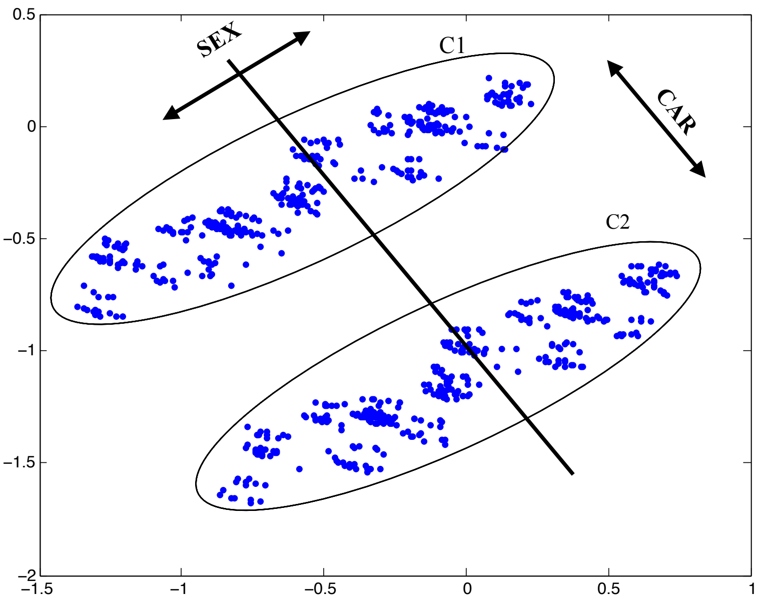


Fig. 3.22 Projection of the bank dataset onto the first two principal components using Neural PCA.

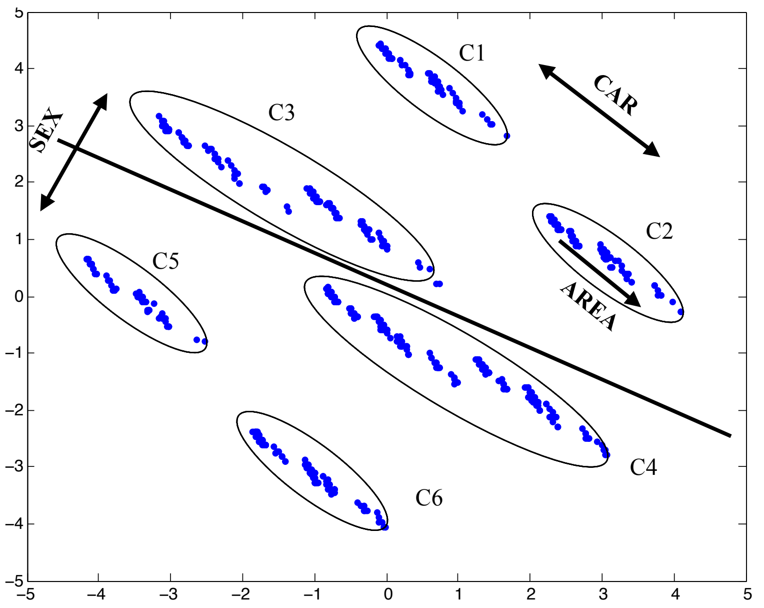


Fig. 3.23 Projection of the bank dataset onto the first two filters found using MLHL with *p = 1*.

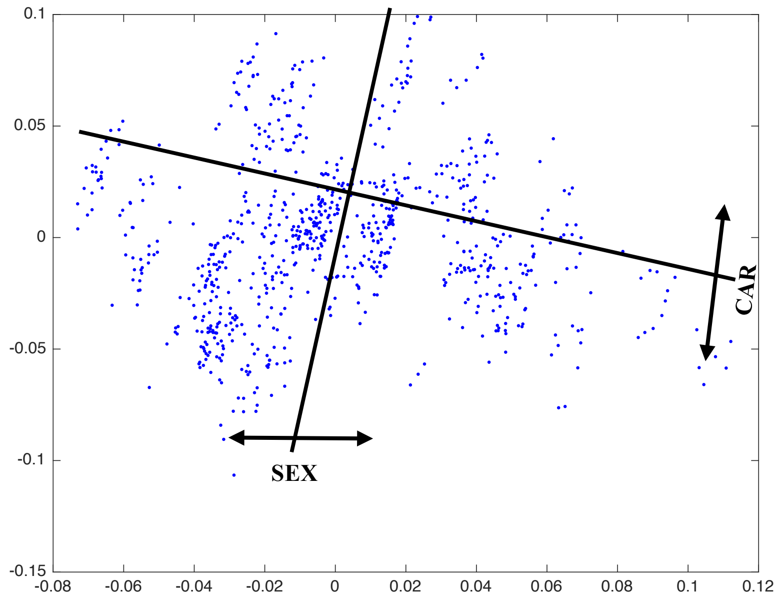


Fig. 3.24 2D projection of the bank dataset using LLE algorithm with *k=40*.

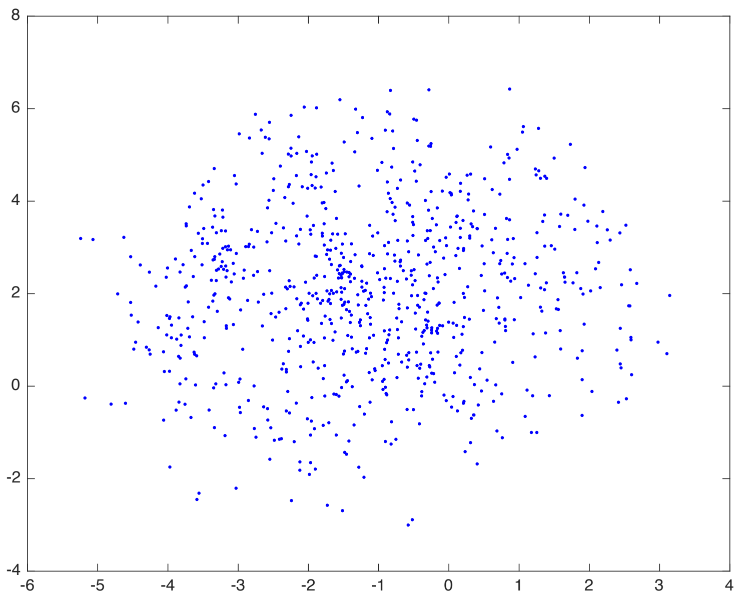


Fig. 3.25 2D projection of the bank dataset using CCA algorithm with 1,000 epochs.

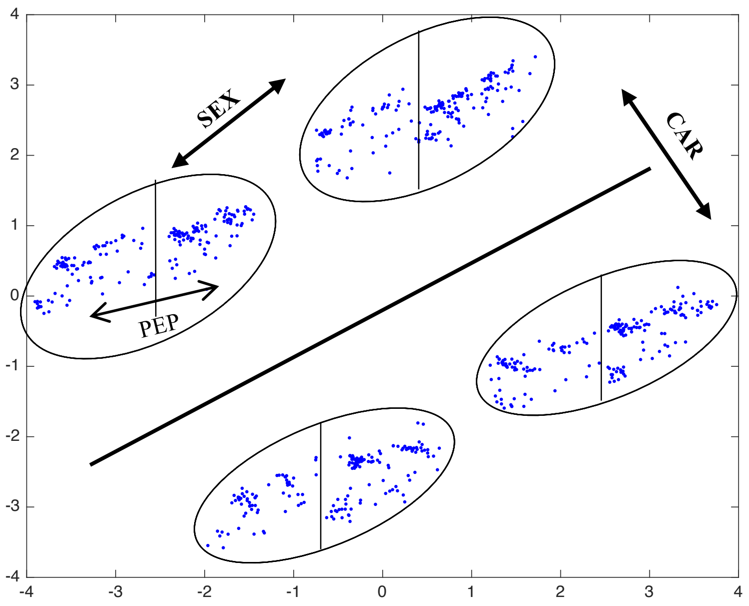


Fig. 3.26 2D projection of the bank dataset using Isomap algorithm with *k=40*.

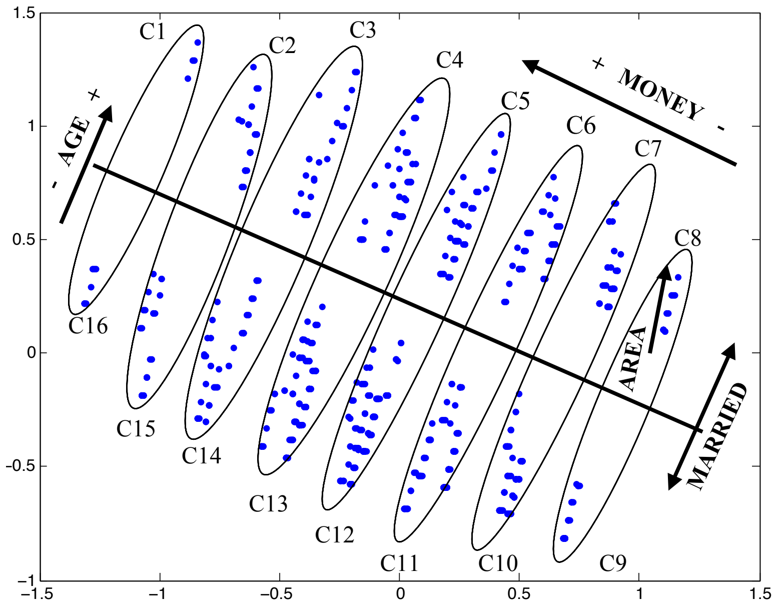


Fig. 3.27 Projection of the bank dataset onto the first two filters found using BHL with *α=β=2.*

Fig. 3.22 shows the two principal components of Neural PCA where it can be seen that Neural PCA is able to clearly identify two main clusters. Visual investigation of the clusters readily reveals that the clusters are formed around the variable CAR, and each cluster is also divided into two halves based on variable SEX. Therefore, C1 groups customers who have a car and C2 groups customers who do not have a car, and the right side of C1 and C2 is related to male customers while the left part are female customers.

In Fig. 3.23, MLHL has clearly identified at least six clusters, based on variables such as CAR and SEX and inside each cluster customers are ordered based on the AREA variable.

In the case of the LLE algorithm (Fig. 3.24), clusters are not clearly separated and data is organized based on variables CAR and SEX. Some clusters can be easily identified (left part of the Fig. 3.24), but data in the central part of Fig. 3.24 cannot be grouped into different clusters.

Results of the CCA algorithm (Fig. 3.25) do not show any kind of clustering or organization of the data, obtaining the worst results among all algorithms.

Fig. 3.26 presents the results of the Isomap algorithm, where 4 main clusters can be easily identified. Separation of these clusters are based on variables SEX and CAR, presenting within each cluster with 2 divisions based on variable PEP.

In the case of BHL (with *α=β=2*), 16 clusters are shown in Fig. 3.27. The BHL method has separated each of the clusters based on variables MARRIED, AGE and MONEY with customers ordered by variable AREA inside each cluster. We can see that the BHL method has identified many more well defined clusters in its projections. Therefore, BHL is able to identify more clusters, generating a more refined analysis of the dataset.

3.5.2.4. Dental milling dataset

Our final dataset was obtained from a dynamic high-precision dental milling machine with five axes. This real industrial use case contains 190 samples obtained by the dental scanner in the manufacturing of dental pieces with different tool types (plane, toric, spherical and drill) and includes eleven input variables (number of pieces, type of dental piece, tool, radius, revolutions, feed rate X, Y and Z, thickness, initial temperature, and initial diameter of the tool), previously used by [82], [94].

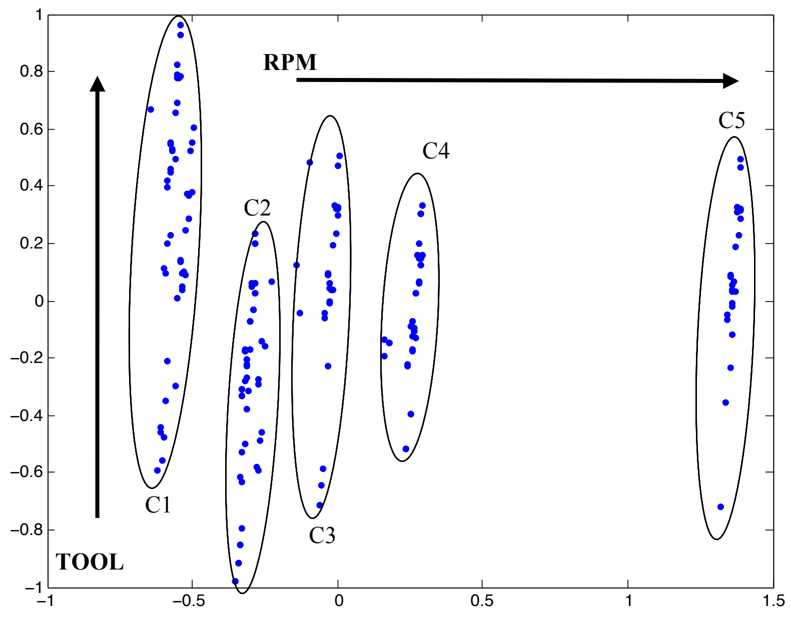


Fig. 3.28 Projection of the dental milling dataset onto the first two principal components using Neural PCA.

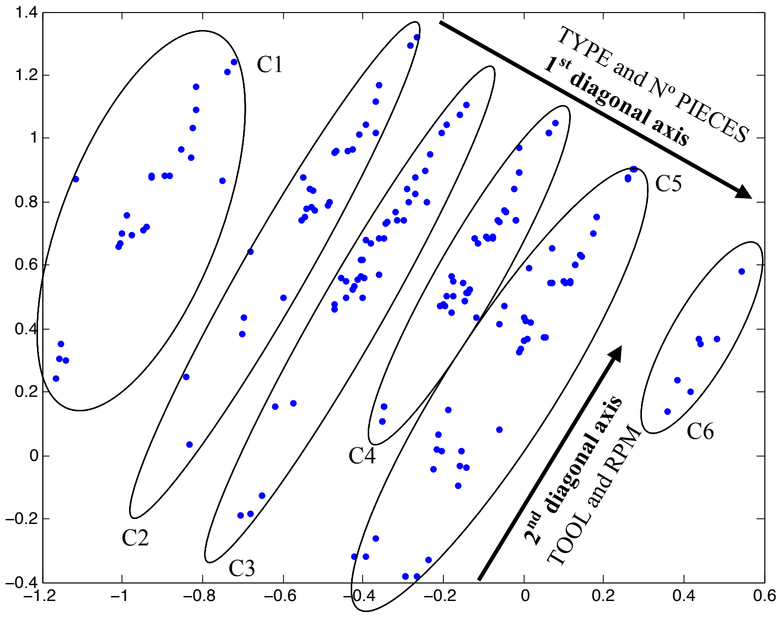


Fig. 3.29 Projection of the dental milling dataset onto the first two filters found using MLHL with *p = 1*.

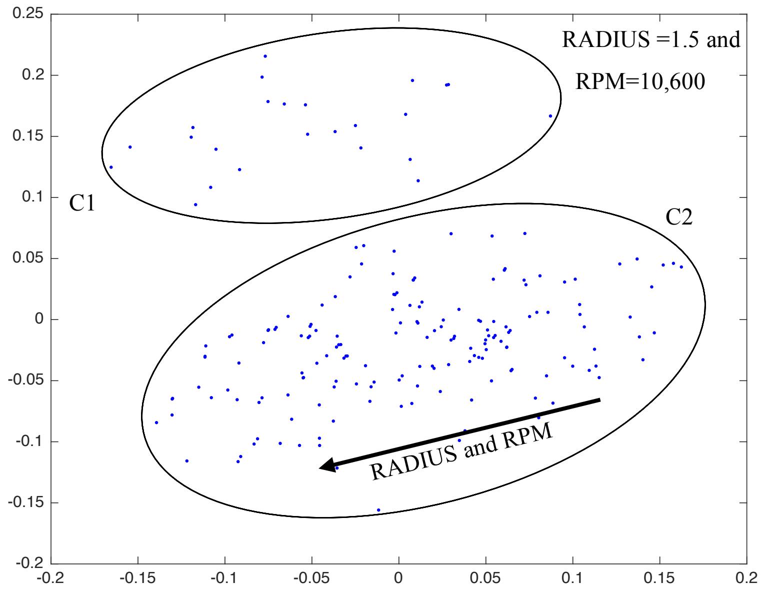


Fig. 3.30 2D projection of the dental milling dataset using LLE algorithm with *k=30*.

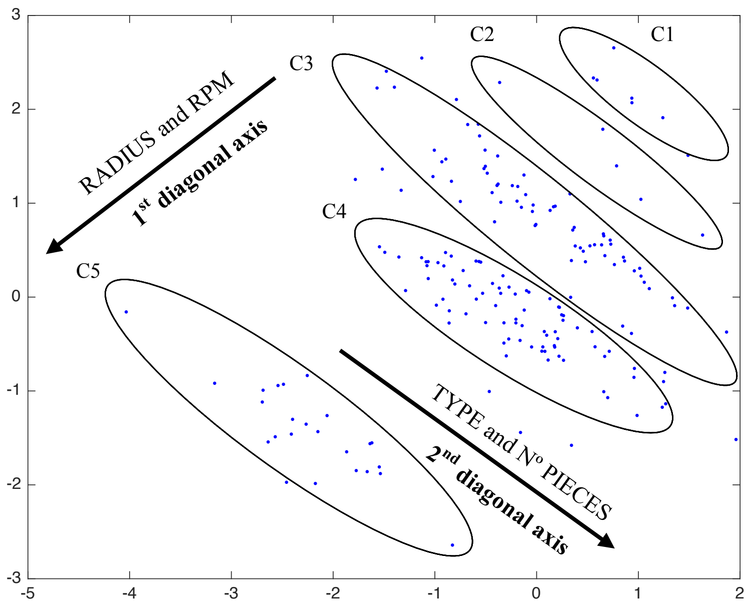


Fig. 3.31 2D projection of the dental milling dataset using CCA algorithm with 1,000 epochs.

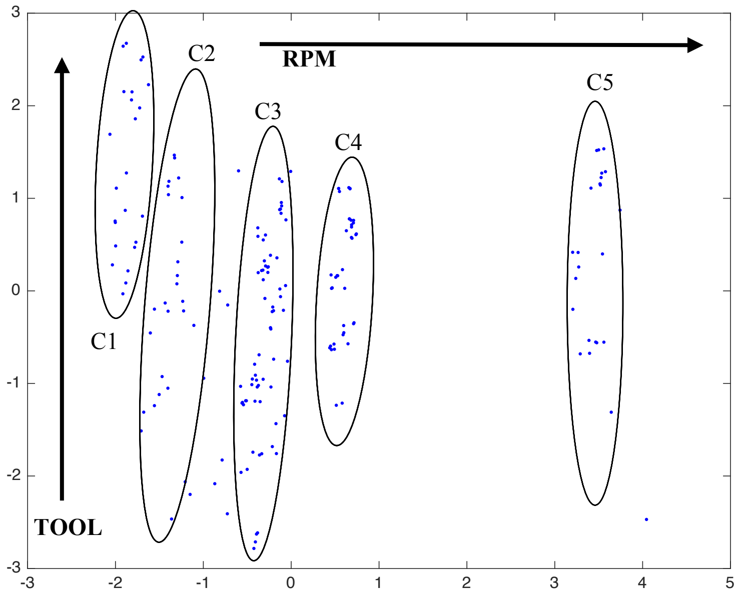


Fig. 3.32 2D projection of the dental milling dataset using Isomap algorithm with *k=30.*

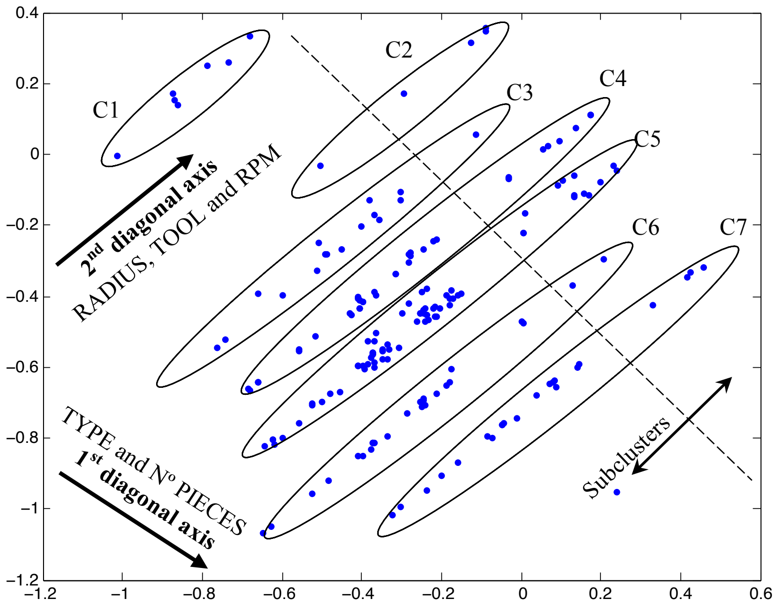


Fig. 3.33 Projection of the dental milling dataset onto the first two filters found using BHL with $\alpha=\beta=2$.

Fig. 3.28 shows the two principal components of Neural PCA where we can see that the model is able to clearly identify at least 5 clusters. Visual investigation of the clusters reveals that they are based on variables RPM on the first axis and TOOL on the second axis.

MLHL (see Fig. 3.29) is able to identify 6 clusters (C1 to C6), however we can identify far more structure in its projection than when we apply Neural PCA, with the clusters defined by variables "TYPE OF DENTAL PIECE" and "NUMBER OF PIECES" on the first diagonal axis and "TOOL" and "RPM" on the second diagonal axis.

Results obtained by LLE algorithm are shown in Fig. 3.30, where it can be seen that LLE only presents 2 differentiated groups. Visual inspection reveals that group C1 is characterized by samples with the same values of "RADIUS=1.5" and "RPM=10,600", and in group C2, samples are organized within the cluster based on the same variables of RADIUS and RPM.

Results obtained by CCA algorithm (Fig. 3.31) are similar to MLHL presenting 2 main diagonal axes defined by variables "TYPE OF DENTAL PIECE" and "NUMBER OF PIECES" on the first diagonal axis and "TOOL" and "RPM" on the other axis. However, CCA is only able to identify 5 clusters and separation among them is not as clear as in the case of MLHL.

Results of the Isomap algorithm (Fig. 3.32) are quite similar to Neural PCA, presenting also 5 clear clusters organized based on variables RPM and TOOL, as is also the case for Neural PCA.

Finally, BHL is able to identify 7 main clusters with relevant internal sub-clusters in the main diagonal (see Fig. 3.33). The main clusters are defined by variables: “TYPE OF DENTAL PIECE” and “NUMBER OF PIECES” on the first diagonal axis, and: “RADIUS”, “TOOL” and “RPM” on the second diagonal axis. The sub-clusters are internally ordered by variables “THICKNESS”, “INITIAL” and “TEMPERATURE”.

Again, BHL has achieved better results than the other algorithms, as it is able to identify far more structure in the projection than that identified by Neural PCA, MLHL, LLE, CCA and Isomap, being able to identify more clusters, generating a more structured analysis of the dataset.

3.6. *Conclusions for Beta Hebbian Learning*

In this chapter, it has been investigated a novel way of extracting information from high dimensional datasets by projecting the datasets onto low dimensional (2 dimensional) subspace. We have initially derived and analyzed a novel family of learning rules, based on Beta distribution, showing that is it very malleable, and outperforms the results obtained by other projection methods such as Neural PCA, LLE, CCA and Isomap or Exploratory Projection Pursuit methods such as MLHL.

The main limitation of previous methods from the family of exponential distributions, such as MLHL or Neural PCA, is that they can generate PDF with different values of kurtosis but always with zero skewness, so the learning rule is not matched to the PDF of the residual as well as it would be desired in case of datasets associated to asymmetric distributions. Then, the use of Beta distribution allows to generate PDF with different values of skewness and kurtosis based on the right combination of α and β parameters. Therefore, it will allow to derive the most appropriate learning rule to the actual PDF of the residual, providing a better visualization of the internal structure of a dataset.

The power of this novel unsupervised method comes from the choice of an appropriate function depending on α and β parameters, to maximize the likelihood of the residuals (e) under particular models of PDFs. A complete analysis of BHL behavior, in terms of α and β parameters, has been performed in Section *“3.5.1 Artificial Datasets for BHL”*, where different artificial datasets have been analyzed. The analyses show that for instance in the case of typical datasets in which one might search for interestingness (in which most of the dimensions are approximately Gaussian and the remaining are more positively kurtotic), the most appropriate combination of values are high values of α and β (i.e. dataset 3 $\alpha=\beta=10$). The opposite is true for dataset 1 (8 leptokurtic dimensions and one Gaussian dimension), which is rather far from being a typical dataset in that most of the directions could be considered interesting, so to identify the Gaussian dimension, the most appropriate combination of values are small values of α and β (i.e. dataset 1 $\alpha=\beta=2$).

In this chapter, it has also been analyzed the stability of the BHL with the general finding that BHL algorithm is only stable when the residuals have positive values lower than 1, as when values of the residuals are beyond this limit, the value of the weights update tends towards infinity.

The BHL maximizes the likelihood of the residual with respect to the actual distribution, thereby matching the learning rule to the PDF of the residual. This is done by identifying adequate values of α and β for the use case. Therefore, BHL is a powerful new tool for the data mining community and should take its place along with existing exploratory methods.

Chapter 4. Beta Scale Invariant Map

4.1. *Introduction*

In this chapter, we present a novel version of the Scale Invariant Map (SIM) called Beta-SIM, developed to facilitate the clustering and visualization of the internal structure of complex datasets effectively and efficiently. It is based on the application of the BHL to the Scale Invariant Map. The Beta-SIM behavior is thoroughly analyzed and successfully demonstrated over 2 artificial and 16 real datasets, comparing its results, in terms of three performance quality measures with other well-known topology preserving models such as Self Organizing Maps (SOM), Scale Invariant Map (SIM), Maximum Likelihood Hebbian Learning-SIM (MLHL-SIM), Visualization Induced SOM (ViSOM), and Growing Neural Gas (GNG). Promising results were found for Beta-SIM, particularly when dealing with highly complex datasets.

4.2. *Beta SIM Learning Rule*

In this section, we present and analyze, for the first time, a novel version of the SIM called Beta-Scale Invariant Map (Beta-SIM), based on the application of a family of learning rules derived from the PDF of the residuals of a Beta distribution when they are applied to the SIM.

The main difference with the SIM is that Beta Hebbian Learning is used to update the weights of all nodes in the neighborhood of the winner, once the winner has been updated. The Beta-SIM model is defined by Eq. (4.1) and (4.2).

$$\text{Feedforward: } y_i = \sum_{j=1}^{N} W_{ij} x_j \,, \ \forall i \tag{4.1}$$

$$\mathit{Feedback}\text{: } e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{4.2}$$

Then, if we apply the BHL method [95] to the SIM to update the weights, we get the following rule for the Beta-SIM (Eq. (4.3)):

$$\begin{gathered}\mathit{Weights\ update}\text{:}\\ \Delta W_i = \eta \cdot h_{ci} \cdot sign(x - W_c) \cdot \left| |x - W_c|^{\alpha-2} \left(1 - |x - W_c|\right)^{\beta-2} \left(1 - \alpha + |x - W_c|(\alpha + \beta - 2)\right) \right|\end{gathered} \tag{4.3}$$

Therefore, by maximizing the likelihood of the residual with respect to the actual distribution, we are matching the learning rule to the pdf of the residual (e).

Based on the analysis about the stability of the BHL learning rule (see "3.4 Stability of the Beta Hebbian Learning") it can be concluded that the Beta-SIM algorithm is only stable when the absolute value of the residuals is lower than 1.

4.3. *Influence of the Choice of α and β Parameters on the Beta-SIM Learning Rule*

In the following, we thoroughly study how the choice of *α* and *β* parameters influences the weights update (Eq. (4.3)) for the different cases: **Case 1: *α=β,* Case 2: *α>β*** and **Case 3: *α<β.***

4.3.1. Case 1: α=β

When *α=β,* the PDF of the beta distribution behavior tends to correspond to a family of exponential distributions (see Fig. 4.1). Therefore it is expected [15] that for leptokurtic residuals, the choice of high values of *α* and *β* (*i.e. α=β=10*) would be more appropriate, while for platykurtic residuals, low values of *α and β (i.e. α=β=2)* would be more appropriate.

Fig. 4.2 presents the weights updates versus the residual providing relevant information about the behavior of the learning rule based on the choice of *α* and *β*. We have analyzed the 3 different possible scenarios related to Case 1: *α=β.*

- ***Case 1.a)*: *α=β*=** high values (*i.e. α=β=10*)
- ***Case 1.b)*: *α=β*=** medium values (*i.e. α=β=5*)
- ***Case 1.c): α=β* =** low values (*i.e. α=β=2*)

In the three cases, it can be seen how the behavior of the weights update (*ΔW*) versus the value of the residual ($e=x-W_c$) is associated to two zones: **zone *a*** where *0<e<0.5* and **zone *b*** where *0.5<e<1*. Such behavior in both areas is symmetric (see Fig. 4.2).

In Case 1.a when *α=β*= high values (*i.e. α=β=10*), in ***zone a,*** as the value of the residual increases, the *ΔW* value (solid blue line in Fig. 4.2) increases until the peak of the function is reached and then it reduces to zero very fast.

This is in line with what it is expected in theory; if a winning neuron is near to the input data (i.e. the error is low) the *ΔW* is also low. As the error increases, the *ΔW* increases up to the peak of ***zone a***. Finally, when the error approaches 0.5, the *ΔW* tends to zero and the neuron is not attracted at all to the input.

As highlighted before, the behavior of *ΔW* versus *e* in ***zone b*** is symmetric to ***zone a*** (see Fig. 4.2).

In both zones the highest values for *ΔW* are related to the peaks of the function, which are based on the choice of *α* and *β*.

***In Case 1.b)* when *α=β*=medium values** (*i.e. α=β=5*), (Fig. 4.2, green dotted line) the behavior resembles the “bell curve" shape. The width of the function is greater and the height is lower than the case of high values of *α=β (case 1.a-* solid blue line*)*.

In Case 1.c) α=β*=low values** (*i.e. α=β=2*), (Fig. 4.2, red dashed line) it can be deduced that in ***zone a, low values of *e* create larger changes in the weights update. As the residual increases, the value *ΔW* decreases (see Fig. 4.2, red dashed line) along ***zone a***. ***Zone b*** has, as in the previous two cases, a symmetric behavior than ***zone a***.

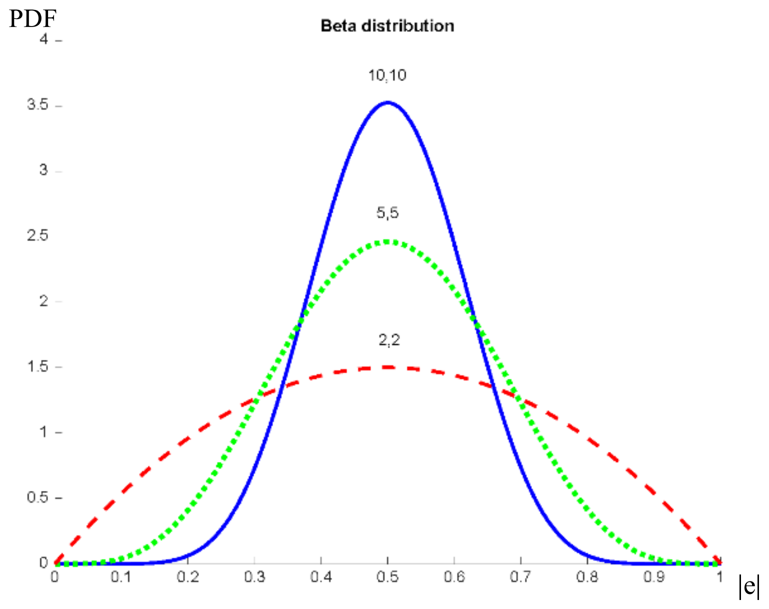


Fig. 4.1 Beta distribution of residuals for values of *α=β:2,2; 5,5; 10,10*.

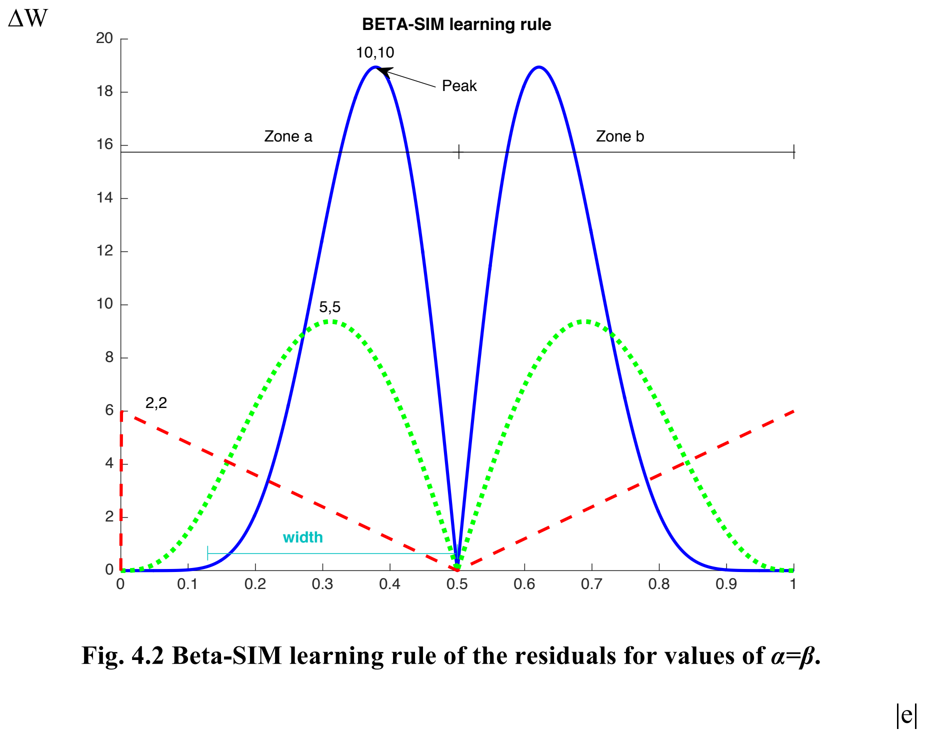


Fig. 4.2 Beta-SIM learning rule of the residuals for values of $\alpha=\beta$.

4.3.2. Case 2: $\alpha>\beta$

When $\alpha\neq\beta$, an asymmetric distribution is obtained, with positive skewness if $\alpha>\beta$, and negative skewness if $\alpha<\beta$. The PDF of the Beta distribution when $\alpha>\beta$ is shown in Fig. 4.3 for several values of both parameters. For values of $\alpha>>>\beta$, leptokurtic and positive skewness residuals are larger [15]. Fig. 4.4 shows the representation of the weights update versus the residual (learning rule Eq. (4.3)) for different values of $\alpha>\beta$.

Again, two zones associated to the magnitude of the residual can be identified. ***Zone a***, where the ΔW increases as the residual increases until reaching the first peak of the function and then decreases quickly to zero (see Fig. 4.4). From this point, the behavior of the learning in ***zone b*** (high values of e) is very interesting as it may be utilized for the identification of sparse clusters or outliers, and based on Fig. 4.4, their influence can be taken into account, or be discarded, depending on the values of α and β. Such effect may be useful depending on the nature of the dataset to be analyzed and on the purpose of the study.

During the training process, the samples from the dataset are selected randomly, so the probability to select a sample of a sparse cluster is lower than the probability to select a sample from a non-sparse cluster (with a high number of samples). This means that normally, the network is trained more often over samples of non-sparse clusters, so the network tends to adapt to these non-sparse clusters. At this point, if we use Beta-SIM with parameters ***$\alpha>\beta$* (Case 2*)***, when a sample of a sparse cluster is selected for training, the distance of this sample to the winning neuron (residual $e=x_j - W_c y_c$) normally will be large. This case is related to *zone b* of the learning rule (see Fig. 4.4). Therefore, the winner node, and its neighbors, are updated attracting the network grid strongly to the sparse cluster. Then, at the end of the training process, at least some nodes of the network grid will be close to the sparse clusters.

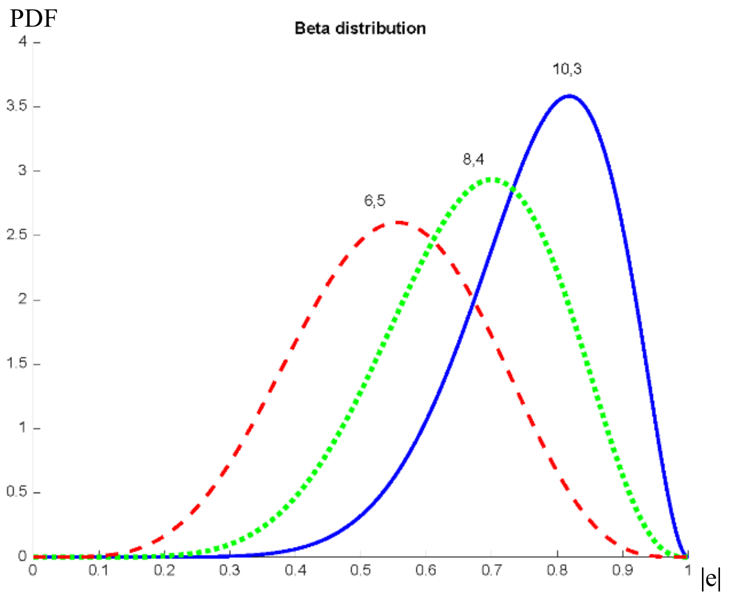


Fig. 4.3 Beta distribution for values of *α>β (6,5; 8,4; 10,3)*.

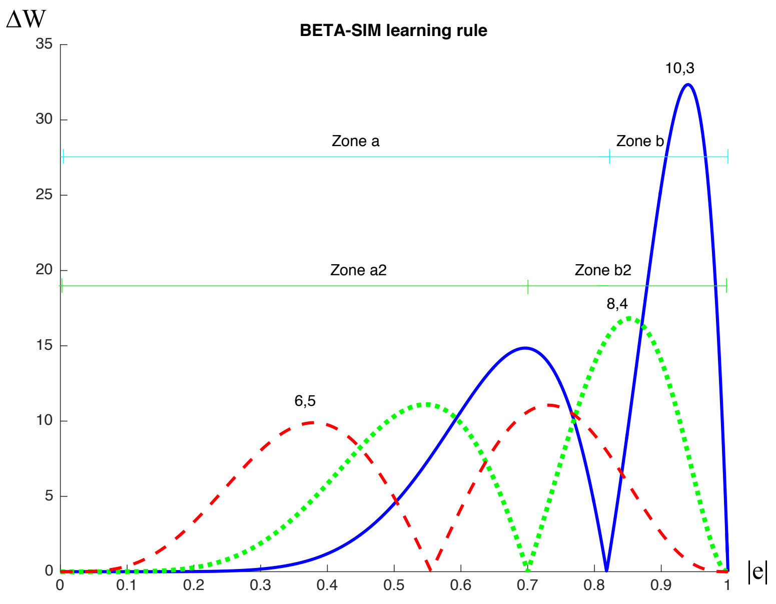


Fig. 4.4 Beta-SIM learning rule for values of *α>β*.

As a conclusion, the Beta-SIM network (specifically when ***α>β (*Case 2*)***) can be seen as a new tool in the data mining community, in the sense that sparse clusters, that are part of high dimensional datasets, can be taken into account and emphasized during clustering tasks instead of being neglected as in many other topology preserving maps.

4.3.3. Case 3: α<β

If $\alpha<\beta$ the effect on the beta distribution (Fig. 4.5) and Beta-SIM learning rule (Fig. 4.6) is the opposite to Case 2: $\alpha>\beta$. This means that the higher values of the error **(*zone b*)** have now low impact on the weights update, and if a winning output vector is far from the input vector (high $e=x-W_c$), this neuron will be less "attracted" to the input than if it was nearer to the input. This is also a useful research finding as it can be seen as a tool to force the learning process to take into account data associated to low residuals, meaning outliers and sparse datasets would have less influence on the weights.

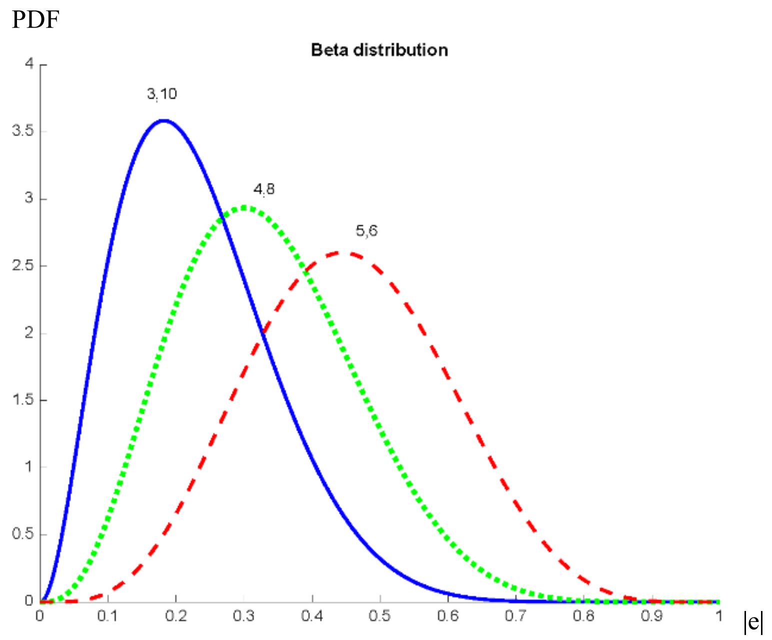


Fig. 4.5 Beta distribution for values of *α<β (5,6; 4,8; 3,10)*.

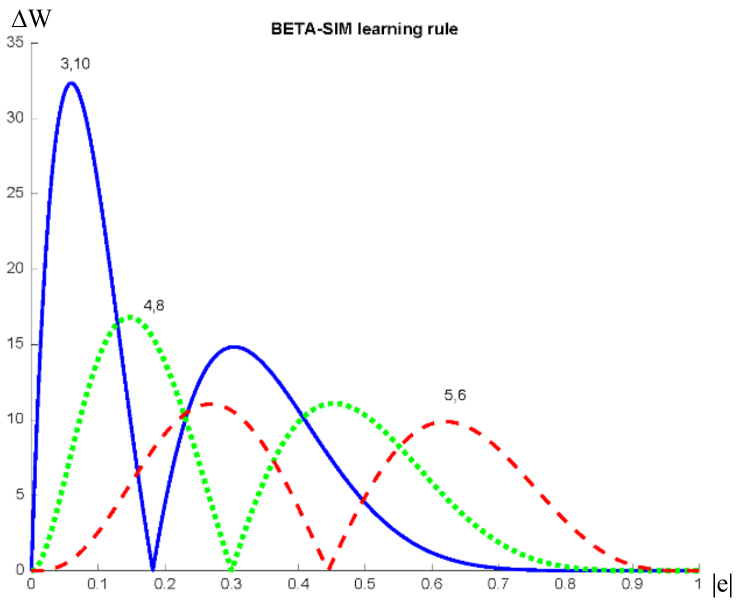


Fig. 4.6 Beta-SIM learning rule for values of *α<β*.

Therefore, the Beta-SIM model can help to model sparse data in highly complex datasets, or limit the influence of outliers and noise by selecting appropriate α and β parameters to create the optimal learning rule.

4.4. *Experiments for Beta-SIM*

In order to test the novel method presented in this thesis, 2 artificial and 16 real datasets were used.

The 2 artificial datasets are used to compare the behavior of the Beta-SIM algorithm with the theoretical analysis described in the previous section *"4.3 Influence of the Choice of α and β Parameters on the Beta-SIM Learning Rule"* and the 16 real datasets, composed of clusters of different sparsity, are used to test the Beta-SIM algorithm on clustering tasks.

Following, 2 real datasets are used to analyze the behavior of the Beta-SIM algorithm to contrast the conclusions obtained in the experiments over the artificial dataset. Once this analysis is performed, the Beta-SIM algorithm is tested over 14 real benchmark datasets by means of a statistical test using three quality measures to compare the novel algorithm results against 5 other well-known topology preserving algorithms (SOM, SIM, MLHL-SIM, ViSOM and GNG).

The objective of the experiments is to show that the novel Beta-SIM network, based on the appropriate choice of α and β parameters, outperforms other topology preserving models, when they are applied to different datasets composed of clusters with different levels of sparsity (imbalanced datasets).

In all experiments, parameters are chosen in an experimental process of trial and error. As parameter selection is a task that is very dependent on the dataset to be used, several initial experiments are conducted with a range of combinations of these parameters.

4.4.1. Artificial Datasets for Beta-SIM

Two artificial datasets were created to measure the adaptation of the network grid to datasets with different sparsity zones.

The objective of these experiments is to analyze the behavior of the Beta-SIM algorithm to contrast the theoretical analysis developed in the previous section *"4.3 Influence of the Choice of α and β Parameters on the Beta-SIM Learning Rule"*.

4.4.1.1. Artificial Dataset 1

In this experiment, a 2-D dataset with radial layout is generated. The dataset consists of a uniform distribution in the shape of an ellipse, where some of the samples were removed to get areas with different levels of sparsity over the *X* axis (see Fig. 4.7).

To generate this sparsity over the *X* axis, a small ellipse is created inside the dataset with an offset in respect to the center of the dataset, with all samples inside this ellipse removed from the main dataset (see Fig. 4.7-red dots).

The objective of this experiment is to measure the adaptation of the different topology preserving networks (SOM, SIM, MLHL-SIM, Beta-SIM) to the dataset, as it presents areas with different levels of sparsity.

As previously mentioned (*"2.4.5 Quality Measures for Topology Preserving Maps"*), the following quality measures are applied to measure the adaptation of the network grids to the dataset: MQE and TE. In these experiments the network grid consisted of 20 neurons.

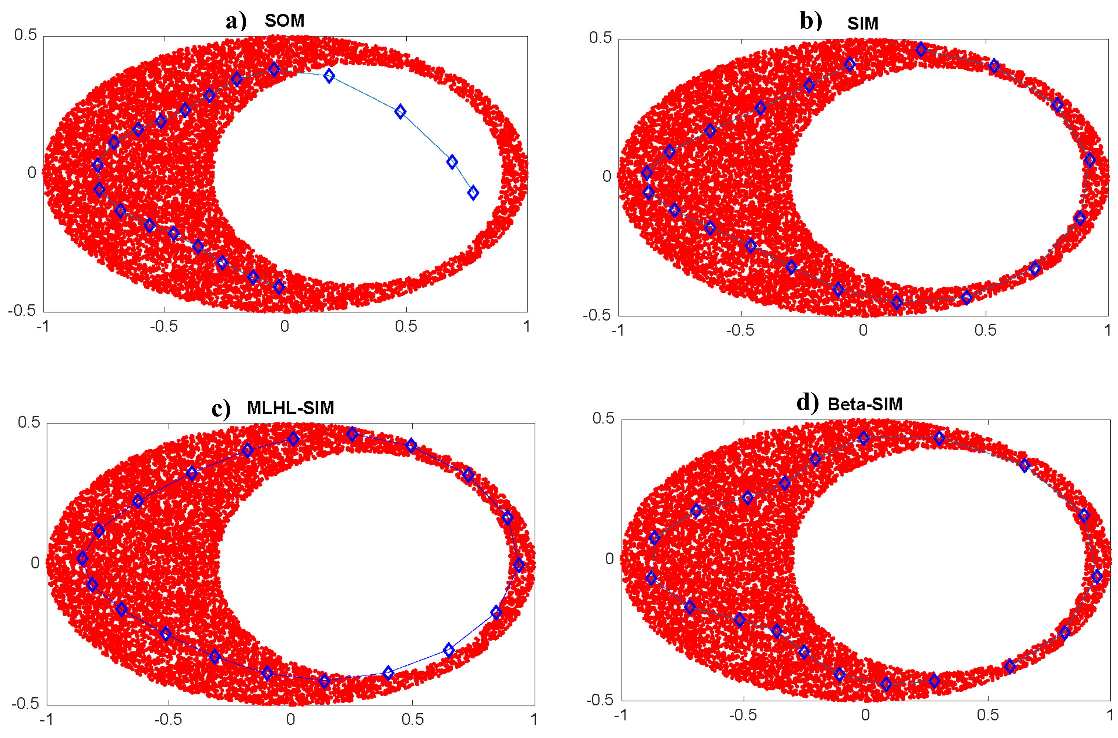


Fig. 4.7 SOM (a), SIM (b), MLHL-SIM (c) and Beta-SIM (d) final network grid (with 20 neurons) for artificial dataset 1.

Table 4.1 shows the parameters used for the four networks (SOM, SIM, MLHL-SIM and Beta-SIM) trained over this 2-D dataset, including the values calculated for their QME and TE.

Table 4.1 SOM, SIM, MLHL-SIM and Beta-SIM parameters and its MQE and TE for the artificial dataset 1.

	SOM	**SIM**	**MLHL-SIM**	**Beta-SIM**
Size	[1,20]	[1,20]	[1,20]	[1,20]
Iterations	10,000	10,000	10,000	10,000
Learning rate	0.1	0.1	0.02	0.1
Neighborhood	10	20	20	15
p	-	-	0.8	-
α;β	-	-	-	*α=3;β=8*
MQE	0.1250	0.0995	0.1102	**0.0972**
TE	**0.0123**	0.0649	0.0526	0.0355

Fig. 4.7 shows the converged weights on the artificial dataset, for each network. The Beta-SIM (Fig. 4.7d), after choosing adequate values for the *α* and *β* parameters, obtained weights that enclosed the data properly, performing better than the SOM (Fig. 4.7a) in terms of preserving the topology on the areas with more data (left side of the dataset), and was able to adapt the weights in the sparser area to be more representative of its properties than the SIM (Fig. 4.7b right side of the data). This research finding is also confirmed by results presented in Table 4.1, where Beta-SIM achieves the lowest MQE in comparison with the other three topology preserving models, and also the second lowest value for the TE. The SOM performed better on the TE measure as the number of winning neurons on the left side of the ellipse (with high-density) is greater than in the Beta-SIM. However, in the case of the SOM, the convergence of the weights fails for areas with sparse data (right side of the Fig. 4.7a).

4.4.1.2. <u>Artificial Dataset 2</u>

This artificial 2-dimensional dataset is created to test the Beta-SIM algorithm, in order to confirm the assertions outlined in section *"4.3 Influence of the Choice of α and β Parameters on the Beta-SIM Learning Rule"*. The dataset is generated using uniform distributions, which are centered at four different points in the 2-D space. Clusters are created with different densities in the range [-1,1]:

- Cluster 1: 10,000 samples; center [-0.8, 0], radius 0.2.
- Cluster 2: 50 samples; center [0, 0], radius 0.2.
- Cluster 3: 10 samples; center [0.4, 0.7], radius 0.1.
- Cluster 4: 5 samples; center [0.8, -0.5], radius 0.1.

Different combinations of values of *α* and *β* are tested in order to analyze the effect on the training, and the results are compared with the SOM, SIM and MLHL-SIM algorithms in order to validate them. Experiments performed with this dataset are organized into 3 cases: where *α=β,* where *α>β* and where *α<β*.

<u>Case 1: α=β</u>

With low values of *α;β* (i.e. *α=β=4*, see Fig. 4.8), which theoretically are more appropriate for platykurtic residuals than for leptokurtic residuals, the behavior of the network outperformed the models where higher values of *α=β* were used. Fig. 4.8h shows the best result of the network for values of *α=β*, which is also compared with the best results of SOM (Fig. 4.8a), SIM (Fig. 4.8b) and MLHL-SIM (Fig. 4.8c,d,e,f,g).

In all cases, the final adaption of the network grid to the dataset is very similar, focusing on the clusters with higher density of samples (non-sparse clusters: cluster 1 (C1) and 2 (C2)), failing to cover and adapt to the other sparse clusters (C3 and C4).

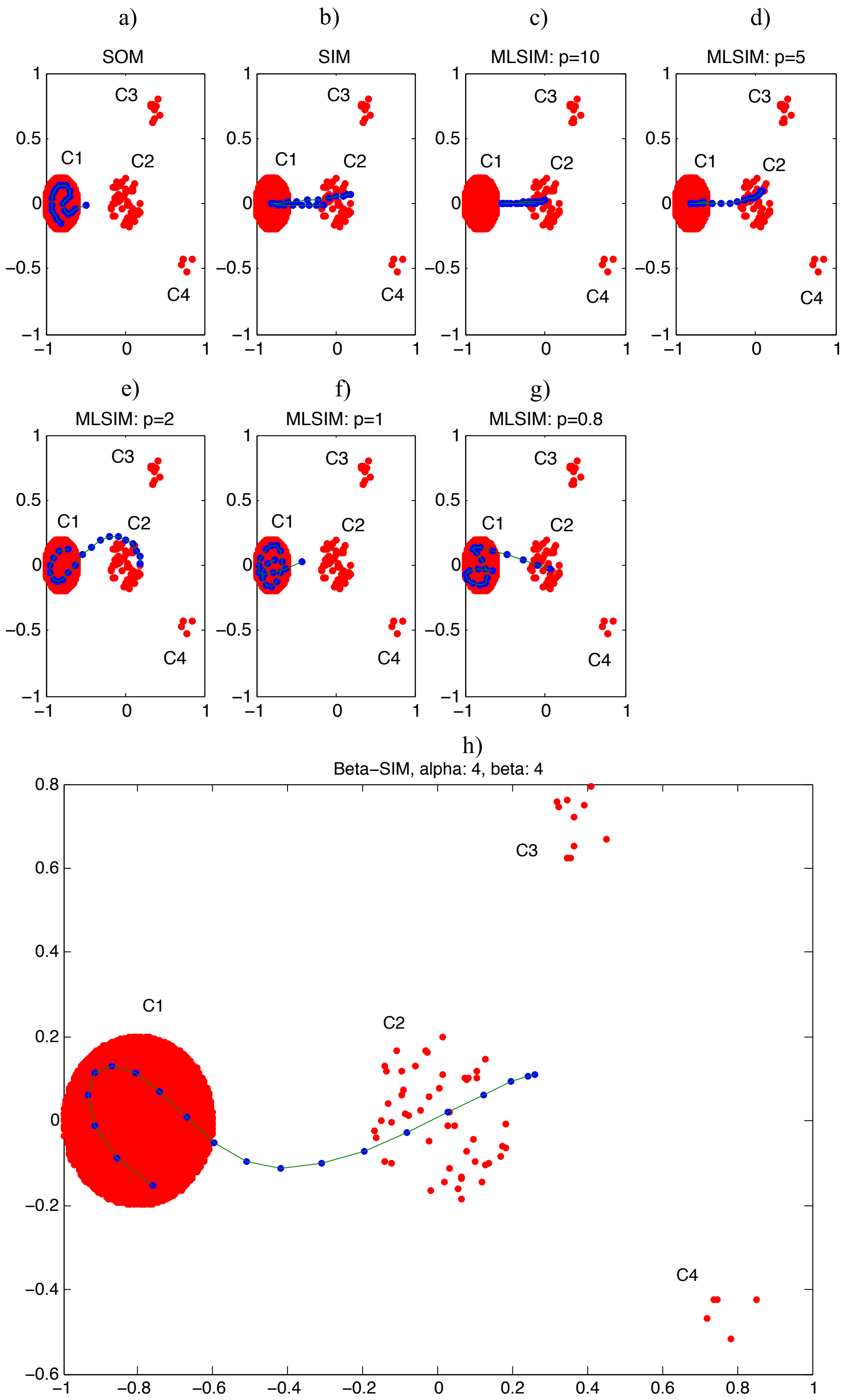


Fig. 4.8 Beta-SIM (h) results for values *α=β*, and its comparison with SOM (a), SIM (b) and MLHL-SIM (c,d,e,f,g) algorithms; 20 neurons; 50,000 iterations; η=0.02; neighborhood function=Gaussian; MLHL-SIM: p=10, 5, 2, 1, 0.8; Beta-SIM *α=β=4*.

4.4.1.3. Case 2: α>β

In this case, residuals with high values should theoretically have more influence on the weights updating (see Fig. 4.4), therefore the adaptation of the network grid should cover all the clusters, even the sparse ones (C3 and C4). In other words, the final positions of some units (neurons of the network grid) will be in closer proximity with the samples of the sparse clusters.

Generally, the learning process is highly conditioned upon samples from clusters of high density (C1 and C2). In general, the final network grid adapts to these high density clusters. By using values of *α>β*, the Beta-SIM algorithm reinforces the learning for the residuals of a sparse cluster sample, as in this dataset where clusters with few samples (low density) are far from clusters with high density of samples. In this case, the final network grid should adapt better over these sparse clusters (C3 and C4).

Fig. 4.9 shows that when *α>β*, the Beta-SIM algorithm is able to assign a small number of units to cover these sparse clusters (C3 and C4), ensuring an effective clustering task even when "imbalanced" datasets, like this dataset, are involved.

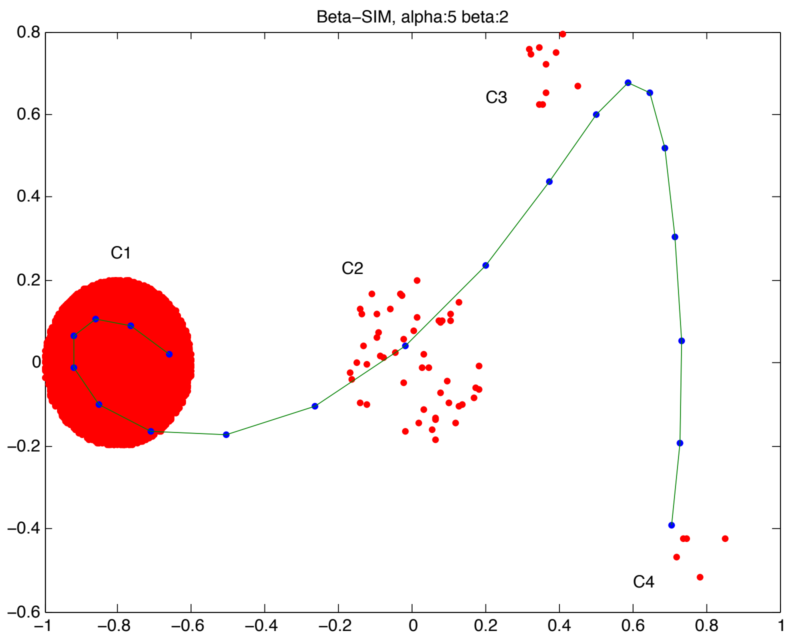


Fig. 4.9 Beta-SIM results for values *α>β*; *α=5, β=2*; η=0.5; iterations=50,000; neighborhood function=Gaussian; neurons grid=[20,1].

4.4.1.4. Case 3: α<β

In this case, by selecting appropriate values of the parameters *α<β* it is possible to make that neurons of the network only react to clusters with higher density of samples (C1, see Fig. 4.10).

When *α<β*, (see Fig. 4.6), low residuals create larger weight updates, making the network grid adapt only to high density clusters (C1, see Fig. 13). Therefore, by selecting values where *α<β*, it is possible to minimize or eliminate the effect of sparse datasets and/or data outliers on the learning process.

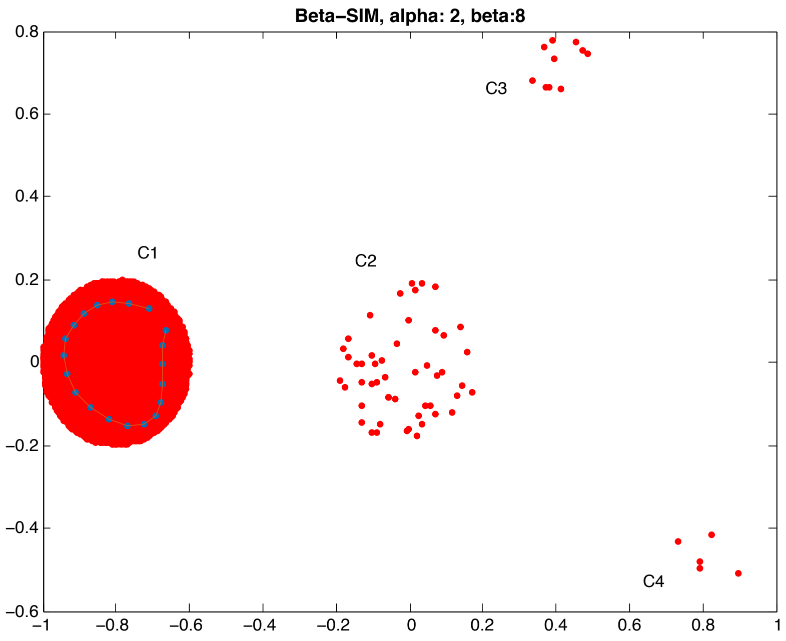


Fig. 4.10 Beta-SIM results for values *α<β*: *α=2, β=8*; η=0.01; iterations=50,000; neighborhood function= Gaussian; neurons grid=[20,1].

Based on the achieved results, it can be concluded that the combination of parameters *α* and *β* allows the selection of how the network grid fits over the dataset. These results seem to confirm that the network takes into account the sparser clusters when *α>β*, and on the contrary, is capable of neglecting, for instance, the existence of outliers (high values of *e*) associated to noise in the data, by using values of *α<β*.

4.4.2. Real Datasets for Beta-SIM

In this subsection (*"4.4.2 Real Datasets for Beta-SIM"*), 2 real datasets are used to analyze the behavior of the Beta-SIM algorithm to contrast the conclusions obtained in the experiments over the artificial datasets. Later, in subsection *"4.4.3 Validation over 14 Real Benchmark Datasets for Beta-SIM"*, the Beta-SIM algorithm is compared to other algorithms using statistical tests over 14 real benchmark datasets. This makes a total of 16 real datasets used to test the behavior of Beta-SIM algorithm.

4.4.2.1. E. Coli dataset

The first real dataset used in these experiments is the well-known E. Coli dataset from UCI repository [96]. The objective of this dataset is to predict the localization site of proteins by utilizing measurements of the cells' characteristics (cytoplasm, inner membrane, periplasm, outer membrane, outer membrane lipoprotein, inner membrane lipoprotein inner membrane, cleavable signal sequence).

SOM, SIM, MLHL-SIM and Beta-SIM algorithms are applied over this dataset. The application of Topology-preserving models to these kind of tasks has previously proved interesting [11].

The dataset consists of 336 instances with 7 attributes, divided in 8 classes with sizes: 143(C1), 77(C2), 2(C3), 2(C4), 35(C5), 20(C6), 5(C7), 52(C8), which create an imbalanced dataset.

In all experiments a normalization of the dataset [-1,1] is performed and a 10-fold-cross validation is used.

Table 4.2 shows the parameters used for the algorithms throughout all the experiments. Parameters are chosen in an experimental process of trial and error. The best performing set of parameters is selected to later conduct all the experiments detailed in the comparison.

Table 4.2 SOM, SIM, MLHL-SIM and Beta-SIM parameters and their associated measures: CE, MQE and TE for E. Coli dataset.

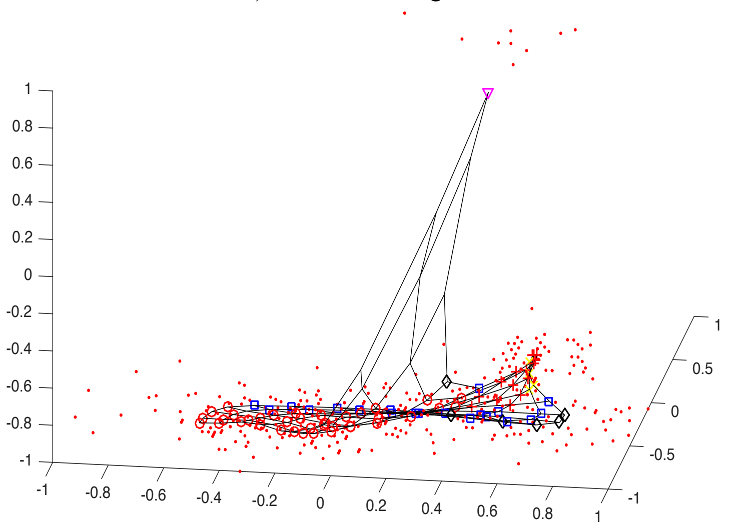
	SOM	SIM	MLHL-SIM	Beta-SIM
Size	[10,10]	[10,10]	[10,10]	[10,10]
Iterations	5,000	10,000	5,000	5,000
Learning rate	0.1	0.01	0.1	0.1
Neighborhood	10	5	10	15
p	-	-	0.9	-
α;β	-	-	-	*α=3;β=2*
CE	11,9%	11,3%	11%	**9,82%**
MQE	0,32	0,26	0,21	**0,20**
TE	**0,11**	0,33	0,63	0,54

Table 4.2 also shows the results for the CE, TE, and MQE obtained for this dataset. It can be seen that Beta-SIM obtains the best CE and MQE. However, the best TE is obtained by SOM as expected.

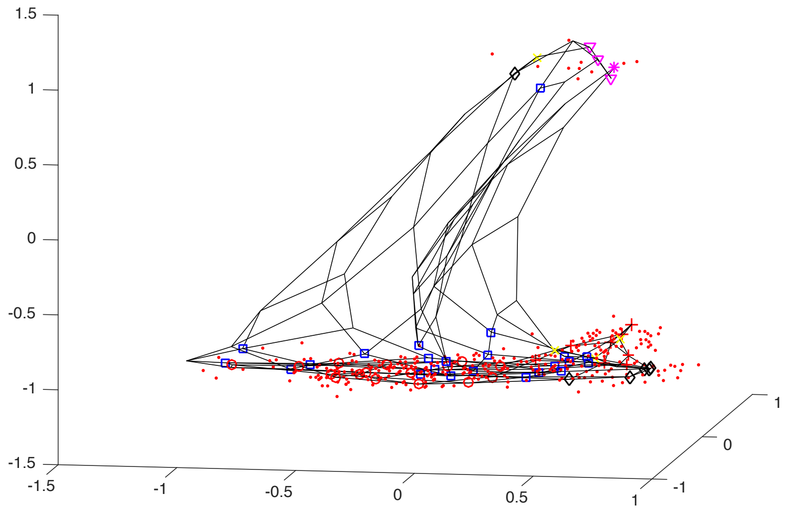
In Fig. 4.11, the final adaptation of the network grids to the dataset is shown for the 4 algorithms (SOM, SIM, MLHL-SIM, Beta-SIM). The Beta-SIM network generates a clear network grid over the dataset, as it is more spread out over the dataset, especially the group at the top of the image (Fig. 4.11d), which seems to be a sparse cluster.

In the case of Beta-SIM, SIM, and MLHL-SIM the topology of the network grid is affected by groups placed at the top of the figure (Fig. 4.11b and Fig. 4.11c) and several neurons are dragged by this group. This effect is clearly reflected in the larger TE compared to the SOM network. For the SOM (Fig. 4.11a), the model does not spread the network grid over the group placed at the top, and hence the effect on the topology of the network is lower than in the case of the other algorithms, leading to a lower TE.

a) SOM network grid



b) SIM network grid



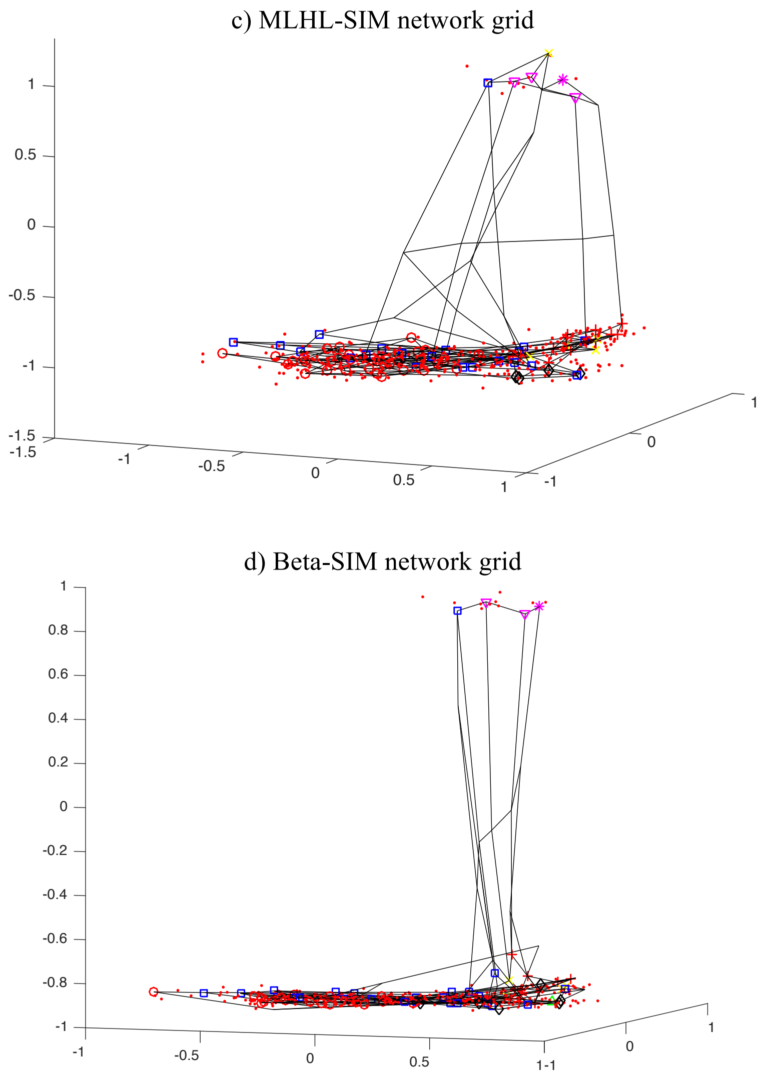


Fig. 4.11 SOM, SIM, MLHL-SIM and Beta-SIM final network grids adaptation over the E. Coli dataset.

If we now modify the values of α and β, Beta-SIM is able to neglect the sparse clusters (top groups of Fig. 4.12), as they have high residual values (e) and the network considers them as outliers (see final network grid adaptation over the dataset in Fig. 4.12).

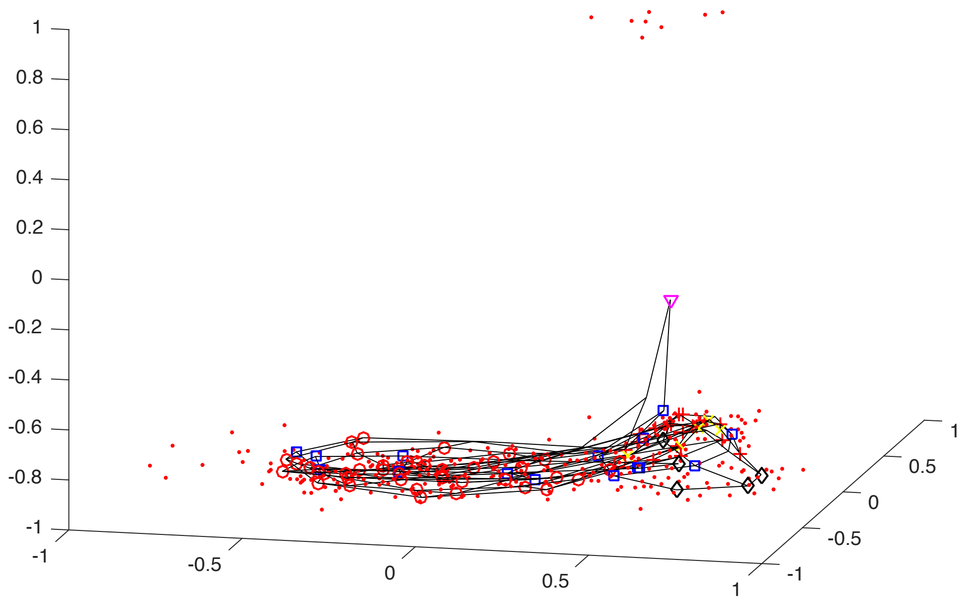
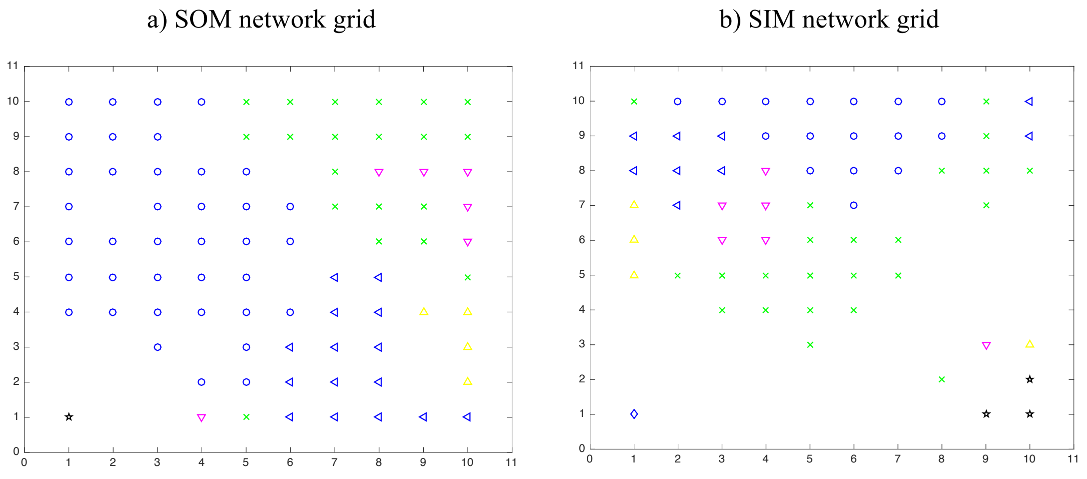


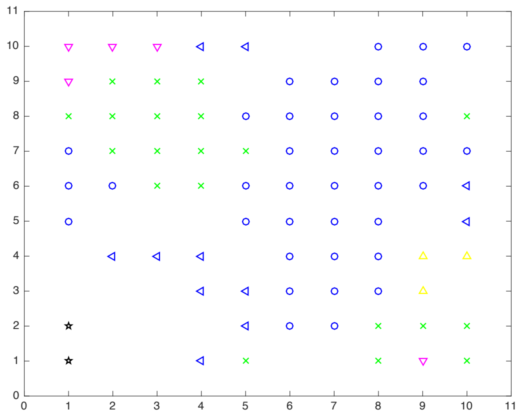
Fig. 4.12 Beta-SIM final network grid adaptation over the E. Coli dataset, with *α=2 and β=4*.

Fig. 4.13 shows the final map for SOM, SIM, MLHL-SIM, and Beta-SIM algorithms, where only the BMUs are displayed. Each BMU is labelled based on the training inputs to which it is reacting. This means that if neuron 10 is activated by 20 training inputs, and 19 of them belongs to class 1, this neuron will be labelled as class 1 (blue circle in Fig. 4.13).

Classes and symbols
○ C1 × C2 □ C3 ◇ C4 ▽ c5 △ C6 ☆ C7 ◁ C8



c) MLHL-SIM network grid



d) Beta-SIM network grid

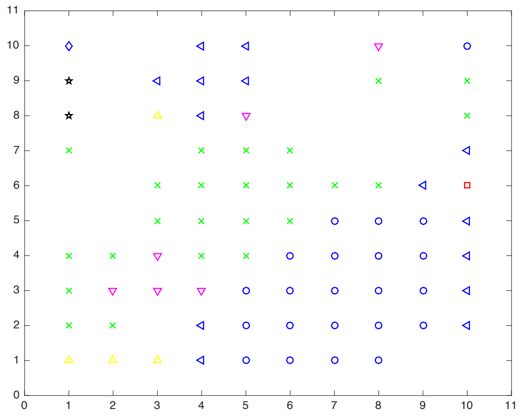


Fig. 4.13 SOM, SIM, MLHL-SIM and Beta-SIM final maps for E. Coli dataset.

Based on the maps from Fig. 4.13, only the Beta-SIM network (Fig. 4.13d) is able to assign at least one neuron to each class. Also, in the Beta-SIM map, the number of neurons assigned to each class is proportional to the size of the classes, so classes C1 and C2 have the highest number of BMUs, followed by classes C8, C5, and C6 respectively, and the lower number of BMUs are associated to sparse classes (C3, C4 and C7).

However, the SOM network (Fig. 4.13a) has the best organization, where neurons are not disordered and the classes are not mixed. In the case of Beta-SIM, several neurons appear disordered, for instance C8 (blue left-pointing triangle in Fig. 4.13d) appears at the top-center and also at the bottom-right side.

Finally, if we present the final Beta-SIM map when *α=2* and *β=4* (see Fig. 4.14), we get as expected the opposite results to Beta-SIM when *α=3* and *β=2*. So now the organization of the map is much clearer, but it is not able to assign BMUs to all classes.

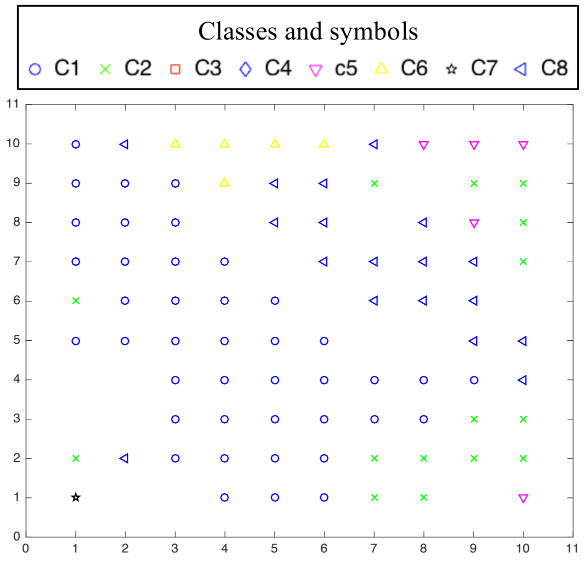


Fig. 4.14 Beta-SIM final map when *α=2* and *β=4* for E. Coli dataset.

4.4.2.2. High Precision Machine Dataset

Our second real dataset was obtained from a dynamic high-precision machinery used for the manufacturing of metal dental pieces [80], [82]. This real industrial use case is described by an initial dataset of 190 samples obtained by a dental scanner in the manufacturing of dental pieces with different tool types (flat, toric, spherical and drill) and is characterized by eleven input variables (number of pieces, type of dental piece, tool, radius, revolutions, feed rate X, Y and Z, thickness, initial temperature, and initial diameter of the tool).

The objective was to test the adaptation of the networks to this real dataset and compare the quality of the Beta-SIM against other algorithms. The MQE and TE were used to measure the quality of the final network grid in 4 algorithms: SOM, SIM, MLHL-SIM and Beta-SIM.

Table 4.3 shows the parameters used for the algorithms throughout all the experiments and the final MQE and TE for each algorithm. Parameters were chosen in an experimental process of trial and error. The best performing set of parameters was selected to later conduct all the experiments detailed in the comparison.

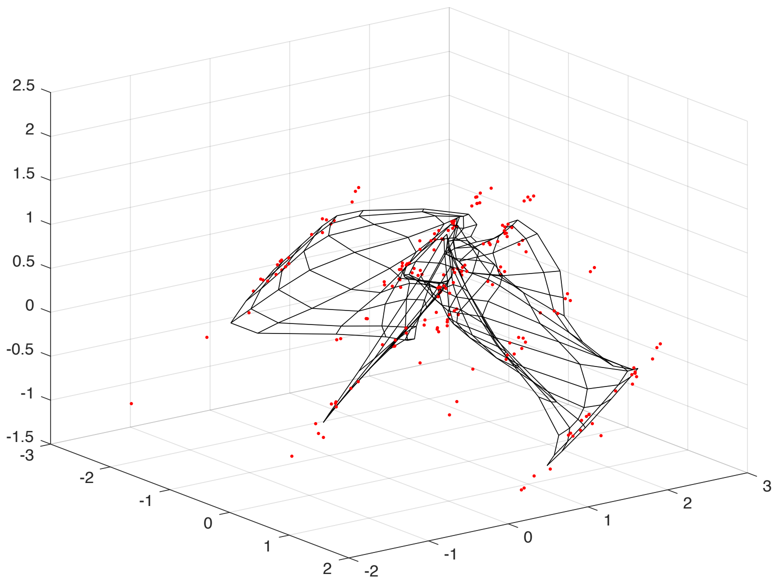
Table 4.3 SOM, SIM, MLHL-SIM and Beta-SIM parameters and their MQE and TE for the high precision machine dataset.

	SOM	**SIM**	**MLHL-SIM**	**Beta-SIM**
Size	[15,15]	[15,15]	[15,15]	[15,15]
Iterations	50,000	100,000	50,000	50,000
Learning rate	0.01	0.01	0.1	0.1
Neighborhood	10	5	10	5
p	-	-	0.5	-
α;β	-	-	-	*α=2;β=3*
MQE	0,55	0,57	0,49	**0,38**
TE	**0,10**	0,36	0,26	0,24

As expected, based on the results of previous experiments over different datasets, the SOM is the one which obtains the best TE, and the Beta-SIM obtains the best MQE. By modifying the parameters *α* and *β* it is possible to reduce the TE to values similar to SOM, but the MQE becomes worse.

In Fig. 4.15, the final adaptation of the network grids to this real dataset is shown for SOM and Beta-SIM. The Beta-SIM network generates a clearer network grid over the dataset, as it is more spread out over the dataspace.

a) SOM network grid



b) Beta-SIM network grid

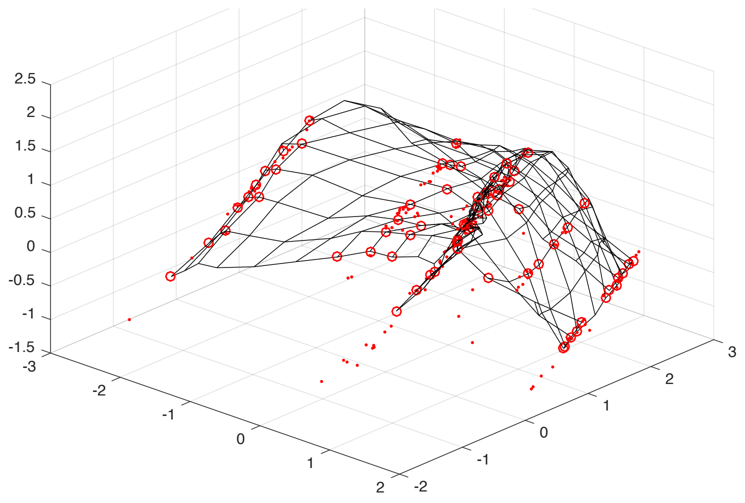


Fig. 4.15 SOM and Beta-SIM final network grids adaptation over the high precision machine dataset.

4.4.3. Validation over 14 Real Benchmark Datasets for Beta-SIM

In this subsection, the Beta-SIM algorithm is validated over 14 diverse real benchmark datasets with diverse number of samples and features by means of a statistical test (ANOVA + post-hoc analysis). The Beta-SIM algorithm is also compared with other well-known topology preserving algorithms such as: SOM, ViSOM, SIM, MLHL-SIM, and GNG in order to validate its capabilities, comparing the methods in terms of CE, MQE and TE.

4.4.3.1. Benchmark datasets description

A total of 14 diverse interesting high dimensional benchmark datasets related to industry, economy and science cases of study were used to validate the performance of the Beta-SIM algorithm. Datasets were taken from the UCI Machine Repository [97] presenting different characteristics, such as number of samples, features and classes. In Table 4.4, a summary of these datasets is presented in terms of samples, features, and number of classes.

Table 4.4 Benchmark datasets for Beta-SIM experiments.

	Name of the dataset	**Samples**	**Features**	**Classes**	**Source**
Dataset 1 (D1)	Liver Disorders	345	6	2	UCI repository
Dataset 2 (D2)	Fertility	100	10	2	UCI repository
Dataset 3 (D3)	Bank Marketing	45211	16	2	UCI repository
Dataset 4 (D4)	Iris	150	4	3	UCI repository
Dataset 5 (D5)	Wine	178	13	3	UCI repository
Dataset 6 (D6)	Contraceptive	1473	9	3	UCI repository
Dataset 7 (D7)	Car	1728	6	4	UCI repository
Dataset 8 (D8)	Dermatology	358	33	6	UCI repository
Dataset 9 (D9)	Image Segmentation	2310	19	7	UCI repository
Dataset 10 (D10)	Landsat Satellite	6435	36	7	UCI repository
Dataset 11 (D11)	Yeast	1484	8	10	UCI repository
Dataset 12 (D12)	Pen-Based Handwritten Digits	10992	16	10	UCI repository
Dataset 13 (D13)	Optical Recog. of Hand. Digits	5620	64	10	UCI repository
Dataset 14 (D14)	Letter Recognition	20000	16	29	UCI repository

4.4.3.2. Results and statistical analysis

In all experiments over all datasets, a normalization of the dataset is performed, and a 10-fold-cross validation is used. Parameters are chosen in an experimental process of trial and error.

Table 8.1 (Chapter 8. Annex) shows the average CE, MQE and TE ± their standard deviation (STD) for each algorithm with each dataset.

Table 8.2 (Chapter 8. Annex) presents the parameters selected for each algorithm over the different experiments (for each dataset). In all cases, the number of neurons were the same for all algorithms in each experiment.

Table 4.5, Table 4.6 and Table 4.7 presents the p-values obtained after applying an ANOVA + post HOC statistical analysis for CE, MQE and TE, respectively (Beta-SIM against each algorithm).

Table 4.5 p-values for CE (Beta-SIM against all algorithms).

Dataset	SOM	ViSOM	SIM	MLHL-SIM	GNG
1	0.6580	1.0000	1.0000	0.9999	0.989
2	1.0000	0.9837	0.9837	1.0000	1.0000
3	1.0000	1.0000	0.9989	1.0000	0.9982
4	0.9999	0.9999	0.9999	0.9973	0.9999
5	1.0000	1.0000	1.0000	0.6053	0.9996
6	1.0000	0.2786	0.9676	0.8666	1.0000
7	0.4614	0.0481*	0.5622	0.4231	0.9602
8	0.9612	0.7831	0.9971	0.2273	0.0139*
9	0.0007*	0.0000*	0.0000*	0.0002*	0.0000*
10	0.0033*	0.0059*	0.0118*	0.1622	0.0048*
11	0.1867	0.4699	0.9950	0.9683	0.7909
12	0.0013*	0.0002*	0.0489*	0.1306	0.3019
13	0.0000*	0.0000*	0.0003*	0.0004*	0.0044*
14	0.0000*	0.0000*	0.0000*	0.0000*	1.0000

* Beta-SIM is significantly better (significance level of 0.05) than the other model

◊ Beta-SIM is significantly worse (significance level of 0.05) than the other model

Table 4.6 p-values for MQE (Beta-SIM against all algorithms).

Dataset	SOM	ViSOM	SIM	MLHL-SIM	GNG
1	0.0031*	0.0003*	0.9847	1.0000	0.9925
2	0.3168	0.5267	0.5422	0.7727	0.0068$^{\Diamond}$
3	0.0231*	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$
4	0.0087*	0.6621	1.0000	0.9987	0.2649
5	0.9627	0.7475	0.5616	0.3808	0.0673
6	0.0000*	0.0565*	0.2433	0.8285	0.1819
7	0.0000*	0.0059$^{\Diamond}$	0.9994	0.0004$^{\Diamond}$	0.0000$^{\Diamond}$
8	0.0001*	0.0000*	0.0051*	0.0007*	0.0000*
9	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
10	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*

11	0.0000*	0.1097	0.8189	0.9515	0.3771
12	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
13	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
14	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*

* Beta-SIM is significantly better (significance level of 0.05) than the other model

$^{\Diamond}$ Beta-SIM is significantly worse (significance level of 0.05) than the other model

Table 4.7 p-values for TE (Beta-SIM against all algorithms).

Dataset	SOM	ViSOM	SIM	MLHL-SIM	GNG
1	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.9999	0.6811	0.0000*
2	0.7557	1.0000	1.0000	0.7557	0.0000*
3	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000*
4	0.6460	0.0001$^{\Diamond}$	0.9989	0.9721	0.0000*
5	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0008$^{\Diamond}$	0.0000*
6	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.7769	0.1592	0.0000*
7	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000*
8	0.0010$^{\Diamond}$	0.0001$^{\Diamond}$	0.7565	1.0000	0.0050$^{\Diamond}$
9	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.9999	1.0000	0.0000*
10	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.9796
11	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.4494	1.0000	0.0000*
12	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.1581	0.0517	0.0000*
13	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0009$^{\Diamond}$	0.0001$^{\Diamond}$	0.0000*
14	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000*

* Beta-SIM is significantly better (significance level of 0.05) than the other model

◊ Beta-SIM is significantly worse (significance level of 0.05) than the other model

After the statistical analysis presented in Table 4.5, Table 4.6 and Table 4.7, it can be concluded that in general terms, the novel Beta-SIM algorithm obtains better results when the number of classes of the datasets increases, that is to say, for complex high dimensional datasets. When the number of classes is larger than 7, in general, Beta-SIM obtains a significant improvement in relation to the CE and MQE measures, with the improvement being of greater magnitude in the MQE measures. However, as can be expected, the TE measure is worse than the other algorithms (except GNG). This is due to the better adaptation to the datasets forcing a deformation of the final neural grid. Nevertheless, when the number of classes is lower than 7, Beta-SIM results are statistically similar to the ones obtained by the other state-of-the-art algorithms.

The results of the experiments performed over these benchmark datasets help to draw broader conclusions about the behavior of the novel proposed method, showing how it improves the other models based on such three well-known quality measures (CE, MQE and TE).

Table 8.3 (Chapter 8. Annex) summarizes the results of Table 4.5, Table 4.6 and Table 4.7 in terms of percentage (%) of improvement:

- Positive values are the "%" of improvement of Beta-SIM in comparison with a specific algorithm, in relation to the three well-known quality measures.
- Negative values are the "%" of deterioration of Beta-SIM in comparison with a specific algorithm, in relation to the three well-known quality measures.

Finally, based on the parameters selection for the Beta-SIM algorithm (Table 8.2 in Chapter 8. Annex), it can be observed that when the number of classes is large (more than 7 classes) the best combination of α and β parameters is obtained for $\alpha>\beta$, and otherwise when the number of classes is small (less than 7 classes), the best combination is obtained when $\alpha=\beta$, or even $\alpha<\beta$ when the number of classes are very small (i.e. 2 or 3 classes). Therefore, it is worthy to apply Beta-SIM when analyzing complex high dimensional datasets.

4.5. *Conclusions for Beta Scale Invariant Map*

In this chapter, a novel algorithm called Beta-SIM has been presented and thoroughly analyzed. Beta-SIM aims to obtain the best topology preserving map possible, in order to be used as a reliable tool in data visualization. Due to the inherent capabilities of the SIM, their combination with the BHL algorithm improves adaptation and visualization of datasets with a radial structure, as has been successfully shown in the tests. The main improvement of the algorithm is the capacity to adapt to sparse clusters or to neglect outliers depending on the combination of values of α and β and the task to be carried out.

Beta-SIM is therefore a powerful new tool for the data mining and big data communities and should take its place along with existing topology preserving maps.

Chapter 5. Weighted Voting Superposition Beta Scale Invariant Map

5.1. *Introduction*

In this section, it is presented a novel topology preserving map called WeVoS-Beta-Scale Invariant Map (WeVoS-Beta-SIM), based on the application of the Weighted Voting Supervision (WeVoS) meta-algorithm to a novel family of learning rules called Beta-Scale Invariant Map (Beta-SIM). The aim of applying WeVoS to Beta-SIM is to improve the visual representation of high dimensional datasets and to increase the stability of the original model (Beta-SIM). This is done by WeVoS-Beta-SIM generating accurate topology maps in an effectively and efficiently way. WeVoS is based on the training of an ensemble of networks and the combination of them to obtain a single one that includes the best features of each one of the networks in the ensemble. WeVoS-Beta-SIM is thoroughly analyzed and successfully demonstrated over the same 14 real datasets used in section "4.4.3 Validation over 14 Real Benchmark Datasets for Beta-SIM", using the three same quality measures. In order to present a complete study of its capabilities, results are compared with other topology preserving models such as Self Organizing Maps (SOM), Scale Invariant Map (SIM), Maximum Likelihood Hebbian Learning-SIM (MLHL-SIM), Visualization Induced SOM (ViSOM), Growing Neural Gas (GNG) and Beta-SIM. The results obtained confirm that the WeVoS-Beta-SIM algorithm presented in this thesis outperforms the classical topology preserving maps and other WeVoS versions in terms of organization and visualization of the presented information.

5.2. *WeVoS-Beta-SIM*

The capability of Beta-SIM algorithm to adapt to sparse clusters or to neglect them, based on combinations of parameters α and β [98], provides to the units of Beta-SIM network more freedom than other topology preserving maps to adapt to datasets, however, it also potentially adds instability to the training. So, the use of ensembles and specifically WeVoS fusion algorithm [61], seems to be the most appropriated method to correct this effect.

WeVoS [99] has been previously applied to other well-known topology preserving maps [11], [51], [99]. Beta-SIM has proved overcome such topology preserving algorithms [98] in the main aspects aimed for topology preserving maps. Then, this thesis chapter it is analyzed the combination of Beta-SIM and WeVoS and compare it with other well-known topology preserving maps and some of the previous WeVoS versions, such as WeVoS-SOM, to study its impact on aspects such as the stability and topology preservation conditions.

WeVoS-Beta-SIM obtains a final map as combination of different Beta-SIM maps by fusion of the neurons in the same position based on a weighted voting. Eq. (**5.1**) is applied for this voting process:

$$V_{p,m} = \frac{\sum b_{p,m}}{\sum_{i=1}^{M} b_{p,i}} \cdot \frac{q_{p,m}}{\sum_{i=1}^{M} q_{p,i}} \tag{5.1}$$

where $V_{p,m}$ is the weight of the vote for the unit included in map m of the ensemble, in its position p, M is the total number of Beta-SIM maps, $b_{p,m}$ is the binary vector used for marking the dataset entries recognized by unit in position p of map m, and $q_{p,m}$ is the value of the desired quality measure for unit in position p of map m.

b is a binary vector of the same length as data samples are in the dataset. It is used to store the samples recognized by a single unit.

Fusion of neurons of the different Beta-SIM maps for WeVoS-Beta-SIM, during the training process, is done based on a quality measure [11] calculated for each Beta-SIM map. This quality measure is considered during the fusion process where the weights of each neuron is proportional to the value of such quality measure, as to modify the position of the neuron in the fused map, the weights of each of the neurons in that position are fed to the final map (see Fig. 5.1).

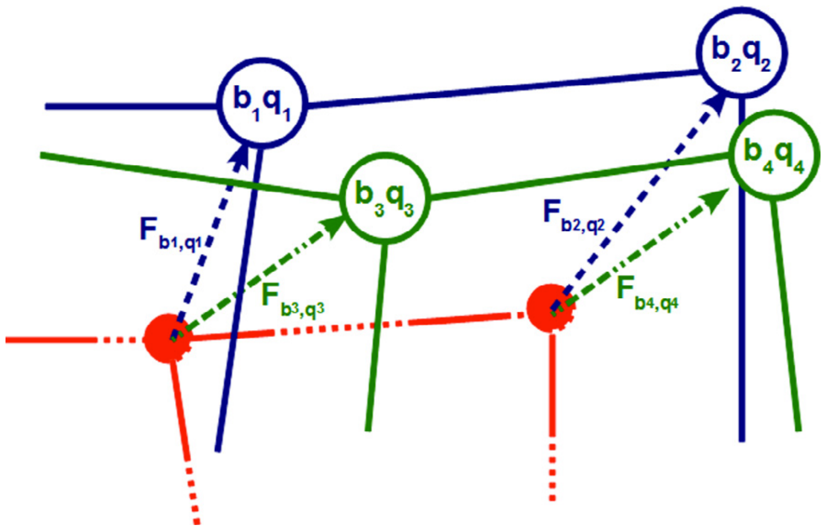


Fig. 5.1 Schematic diagram of the weight voting in WeVoS in a 2-D map.

Briefly, WeVoS-Beta-SIM meta-algorithm works in the following way:

- First of all, an ensemble of Beta-SIM maps is trained.
- Then, the chosen quality/error measure is calculated for each neuron in all Beta-SIM maps.
- The fused map is initialized by calculating the centroids of the neurons in the same position of all the maps, by calculating the superposition of the ensemble.

- For each of the neurons in the fused map, the average neuron quality and the number of total samples recognized in that position for the Beta-SIM maps, are calculated.
- The weight of the vote for each neuron can be calculated with this information by using Eq. (**5.1**).
- To modify the position of the neuron in the fused map, the weights of each of the neurons in that position are fed to the final map.
- Finally, the learning rate in each case will be the weight of the vote for that neuron.

5.3. *Experiments and results for WeVoS-Beta-SIM*

Several experiments have been designed and performed to investigate the capabilities of WeVoS-Beta-SIM and also to compare it with other well-known topology preserving maps such as of SIM, Beta-SIM, SOM, WeVoS-SOM, ViSOM, MLHL-SIM and GNG.

The first type of experiments was designed to present visually some of the main characteristics of the used algorithms such as topology of the grid maps and spread of the grid maps over the data. In the second type, a thoroughly analysis, in terms of the three quality measures used in this thesis (CE, MQE and TE), was performed to validate the visual results obtained previously.

All the tests were run using a classic tenfold cross-validation to use the complete dataset for training and testing. The ensembles were trained using one of the simplest meta-algorithms for ensemble training: the bagging meta-algorithm [100].

In the case of WeVoS-Beta-SIM and WeVoS-SOM, the datasets have been reduced to 1/5 of its original size, and a single model and an ensemble of 5 maps are calculated for each one, comparing the performance of the models over datasets with the same inner structure.

5.3.1. Benchmark datasets description for WeVos-Beta-SIM

The same 14 benchmark datasets used to validate the Beta-SIM algorithm (see Table 4.4) are applied here to validate the WeVoS-Beta-SIM algorithm. The WeVoS-Beta-SIM algorithm is also compared with other well-known topology preserving algorithms such as: SOM, ViSOM, WeVoS-SOM, SIM, MLHL-SIM, Beta-SIM and GNG in order to validate its capabilities, comparing the methods in terms of CE, MQE and TE. Experiments are running using the same parameters selected previously over the 14 benchmark datasets (Table 8.2, Chapter 8. Annex). WeVoS-Beta-SIM and WeVoS-SOM use the same parameters as their respective single models, Beta-SIM and SOM.

5.3.2. Visualization results

In this subsection are presented the visualization results obtained for 2 of the benchmark datasets used in this thesis ("Iris" dataset and "Landsat Satellite" dataset). The aim is to present visually some of the main characteristics of the used algorithms such as topology of the grid maps and spread of the grid maps over the data. Then, a complete analysis of the results, in terms of three quality measures (see Section 4), is performed over all benchmark datasets.

The 2 datasets used in this subsection were selected due to their different levels of complexity. The first dataset is the well-known "Iris" dataset (with low complexity; only 3 classes) and the second is the "Landsat Satellite" dataset (with high complexity; 7 classes). The graphs presented in Fig. 5.2, Fig. 5.3, Fig. 5.4 and Fig. 5.5 illustrate the performance of each model for each dataset, presenting the analytical results in the next subsection ("5.3.3 Analytical results").

Fig. 5.2 and Fig. 5.4 represent the adaptation of each map to its structure in representation of the dataset under analysis ("Iris" and "Landsat Satellite" datasets). It depicts the lattices composing the maps embedded in a 2D input space. All figures, from Fig. 5.2 to Fig. 5.4, present the datasets projected onto their first three principal components and the final grid maps of the models are also embedded in the space of the three principal components.

Fig. 5.3 and Fig. 5.5 show the final unit map for each algorithm where only BMUs are displayed. Each BMU of each map is labelled based on the training inputs to which they are reacting. This means that if one neuron (BMU) is activated by 20 inputs samples and 19 of them belongs to class 1, this neuron is labelled as class 1 (red circles in Fig. 5.4 and Fig. 5.5). GNG is not suitable for this 2D map representation, as some units are disregarded from the final model and therefore the topology preservation is lost.

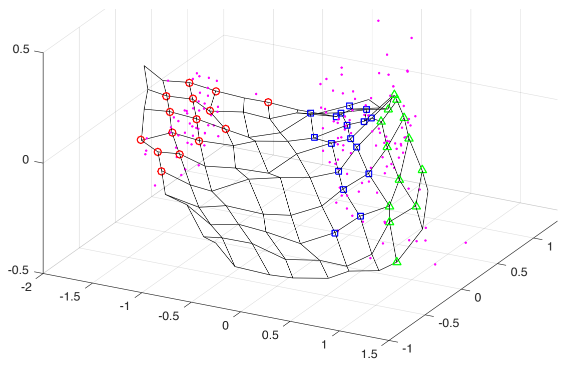
5.3.2.1. Visualization Results for Iris Dataset

It is easily observed in Fig. 5.2 that Beta-SIM and WeVoS-Beta-SIM grid maps are more widely spread throughout the Iris dataset (represented as magenta dots), covering the input space better than the other algorithms. This better coverage over the dataset corresponds to a better MQE result (see Table 8.8 Annex).

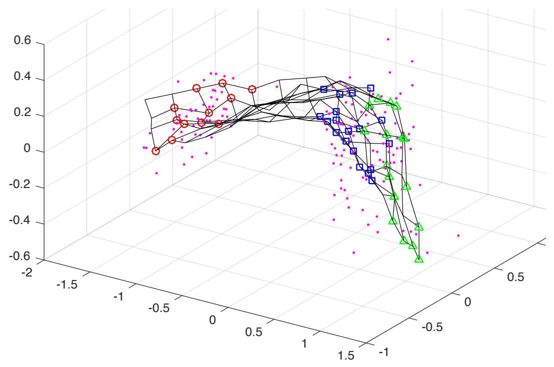
However, SOM, WeVoS-SOM and ViSOM conserve the topology of the map very well as their grid maps contains just few twists and folds (Fig. 5.2d, Fig. 5.2e and Fig. 5.2g). Therefore, they have the lowest TE values among all algorithms (see Table 8.9Annex).

Comparing Beta-SIM and WeVoS-Beta-SIM, Fig. 5.2b and Fig. 5.2c show how WeVoS-Beta-SIM obtains a better topology of the grid map versus Beta-SIM (which means better TE results), due to the fact that WeVoS-Beta-SIM grid map contains less twists and folds than Beta-SIM. At the same time, the WeVoS-Beta-SIM grid map adapts slightly better their structure to the dataset, covering the Iris dataset more adequately (which means a lower MQE value).

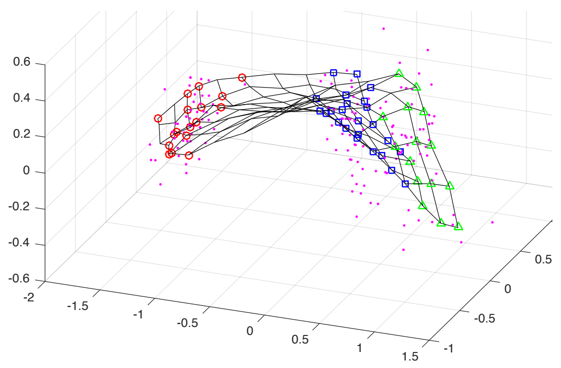
a) SIM



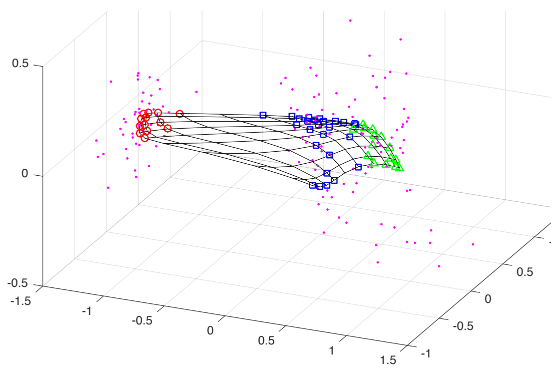
b) Beta-SIM



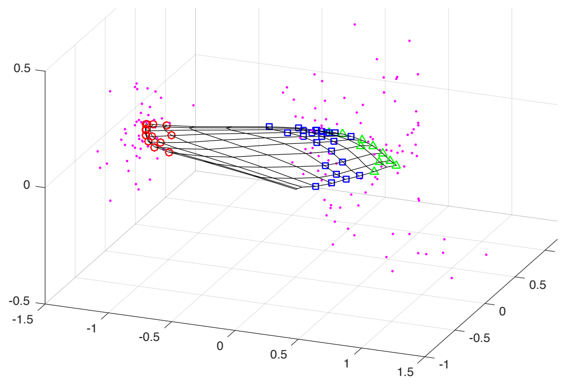
c) WeVoS-Beta-SIM



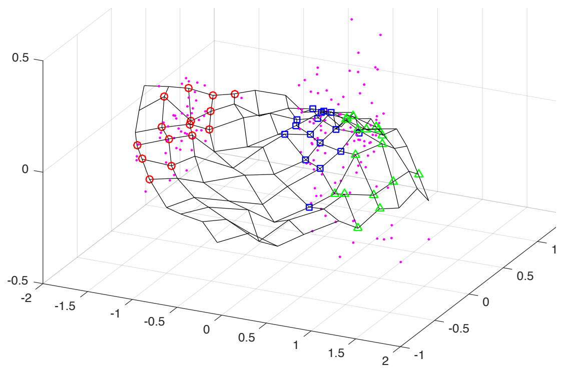
d) SOM



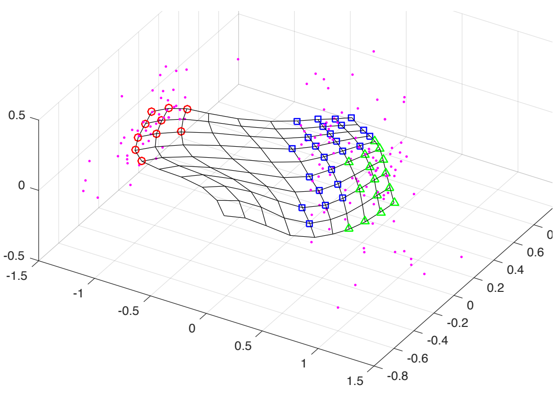
e) WeVoS-SOM



f) MLHL-SIM



g) ViSOM



h) GNG

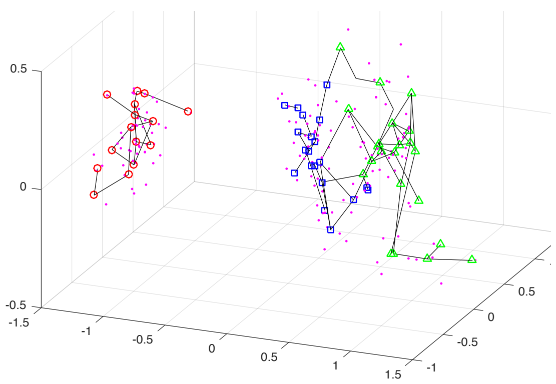
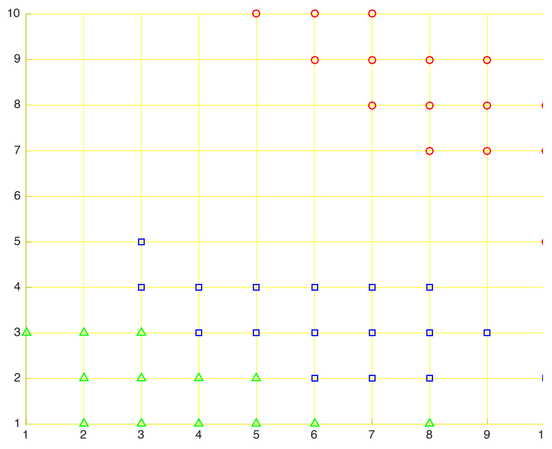


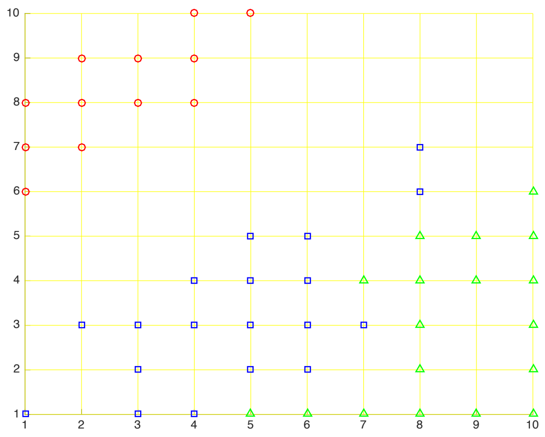
Fig. 5.2 Algorithms' final network grids adaptation over the Iris dataset.

Fig. 5.3 shows that, in general, the WeVoS-Beta-SIM algorithm (Fig. 5.3c) provides the map with more compact and clearly separated groups. However, differences with the other algorithms are minor. All algorithms obtain maps where class 1 (red circles Fig. 5.3) is clearly separated from the other 2 classes (class 2 –blue squares– and class 3 –green triangles–). Differences between maps are only appreciable when they are compared in terms of separation of classes 2 and 3, where WeVoS-Beta-SIM algorithm presents these 2 classes in more compact and clearly defined groups (Fig. 5.3c).

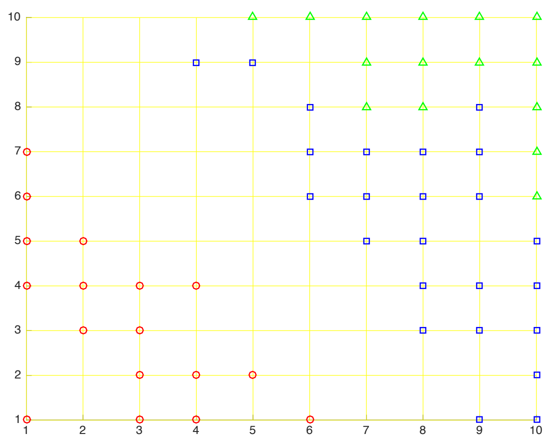
a) SIM



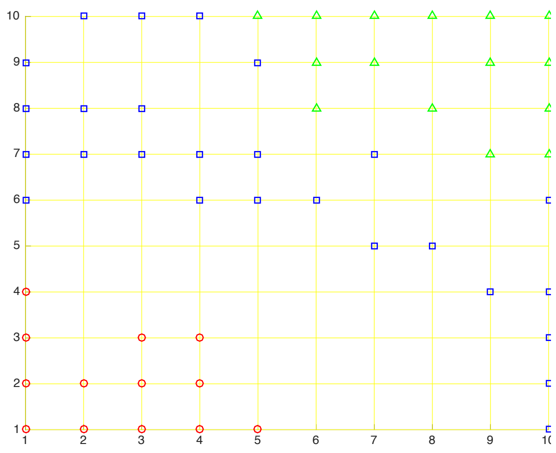
b) Beta-SIM



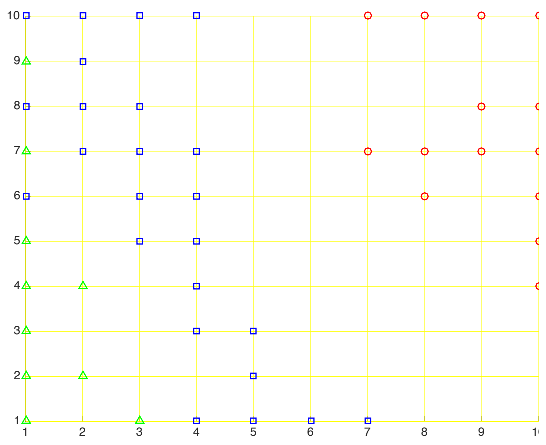
c) WeVoS-Beta-SIM



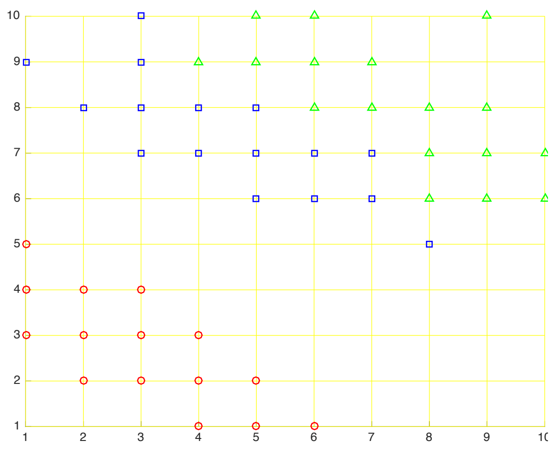
d) SOM



e) WeVoS-SOM



f) MLHL-SIM



g) ViSOM

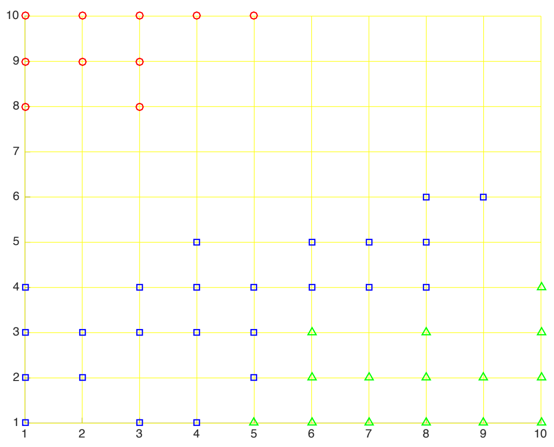


Fig. 5.3 Algorithms' final unit maps for Iris dataset.

5.3.2.2. Visualization Results for Landsat Satellite Dataset

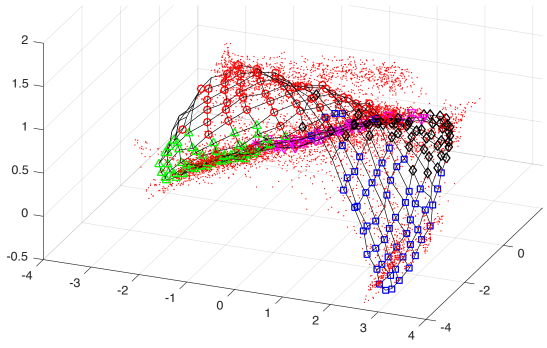
Fig. 5.4 shows how Beta-SIM (Fig. 5.4b), GNG (Fig. 5.4h) and WeVoS-Beta-SIM (Fig. 5.4c) algorithms outperform the other algorithms by distributing the units of their grid maps over the Landsat Satellite dataset (represented as red dots) in the best possible way. Units of the grid maps are close to the input samples over the whole dataset, which leads to lower MQE values than the other algorithms.

A similar situation occurs with SIM and MLHL-SIM algorithms (Fig. 5.4a and Fig. 5.4f) as they obtained grid maps which adapt well their structure to the dataset but not as well as the previously mentioned algorithms.

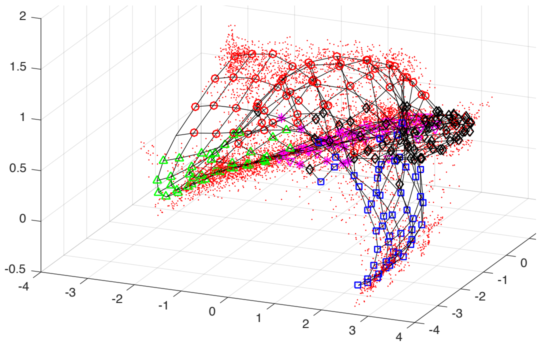
Again, SOM, WeVoS-SOM and ViSOM algorithms are the ones which better preserve the topology of the grid maps (Fig. 5.4, Fig. 5.4e and Fig. 5.4g) as their maps have less twist and folds than the other algorithms. Therefore, they obtain the lowest TE values (see Table 8.9Annex).

WeVos-Beta-SIM (Fig. 5.4c) and Beta-SIM (Fig. 5.4b) algorithms obtain similar final grid maps, but WeVoS-Beta-SIM preserves the topology of the grid map better than the Beta-SIM algorithm. Comparing both figures (Fig. 5.4b and Fig. 5.4c), the WeVoS-Beta-SIM grid map presents less twists and folds than the Beta-SIM grid map, having then lower TE values.

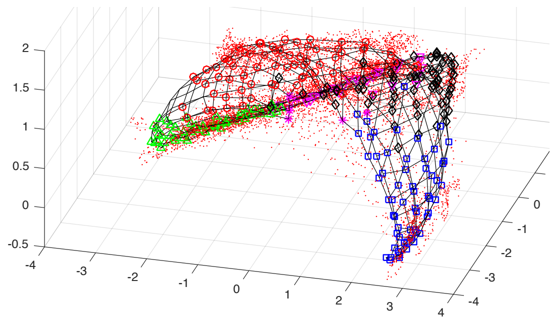
a) SIM



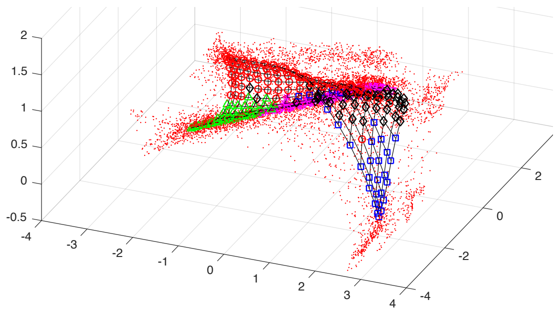
b) Beta-SIM



c) WeVoS-Beta-SIM



d) SOM



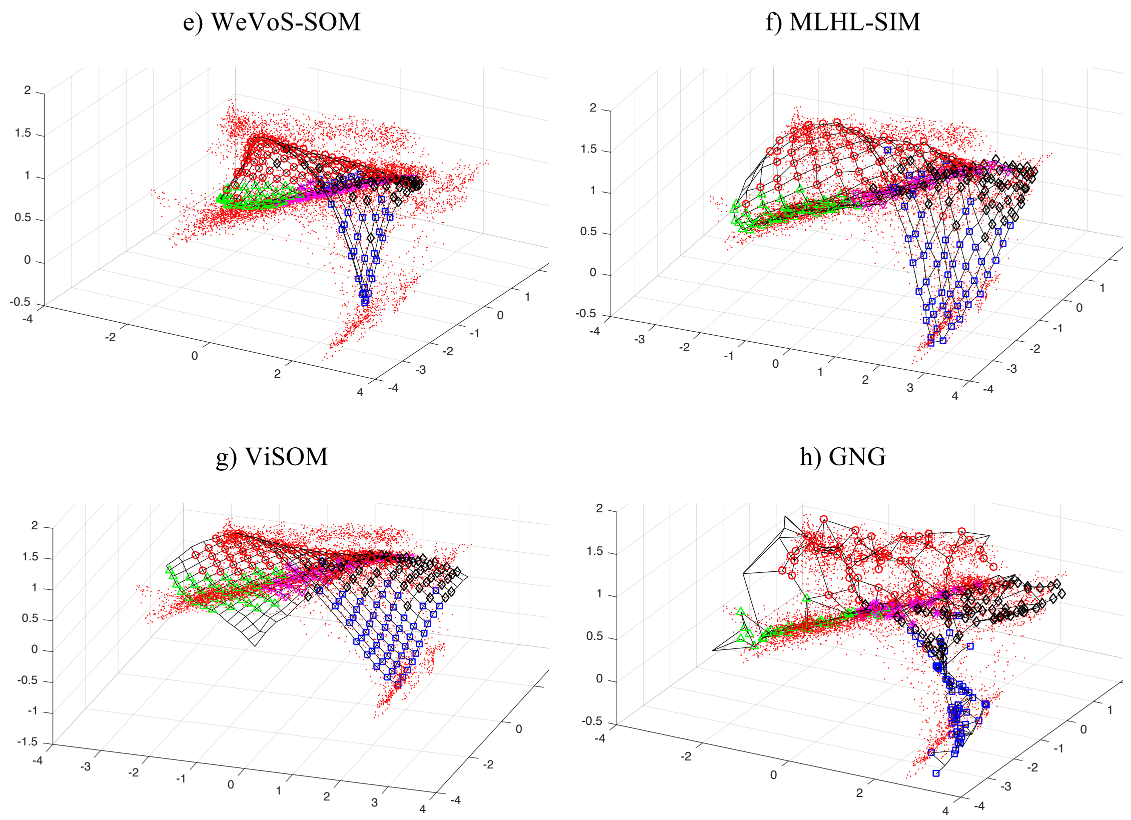
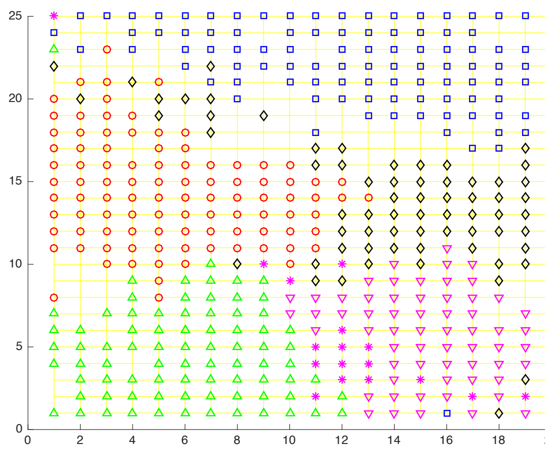
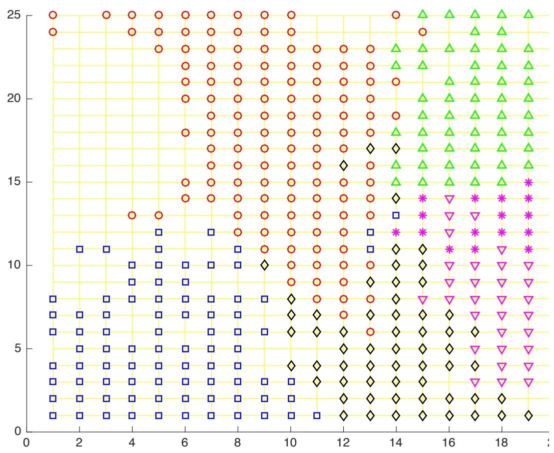


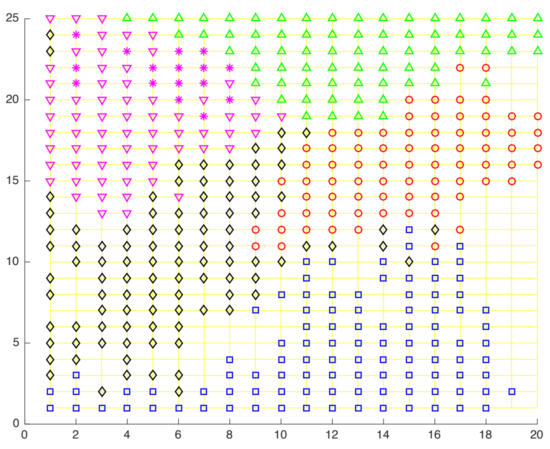
Fig. 5.4 Algorithms' final network grid adaptations over the Landsat Satellite dataset.

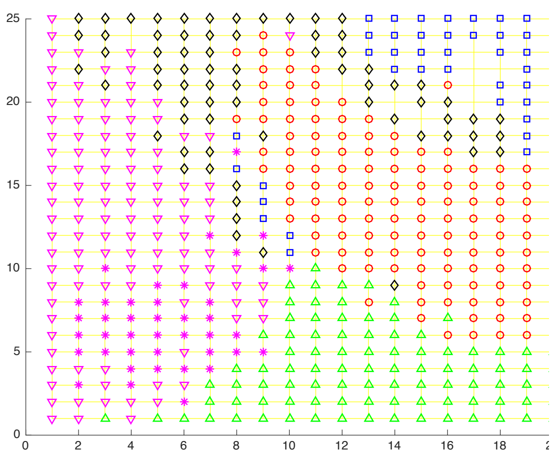
It can be seen in Fig. 5.5 that WeVoS-Beta-SIM (Fig. 5.5c) provides the best visual representation through a smoother map compared to the rest of the algorithms. The map presents, in general, compact and unmixed groups (there is no mixing of BMUs from different classes). Beta-SIM also obtains compact groups (Fig. 5.5b), but some of the neurons associated to different classes are mixed. This fact was observed in Fig. 5.4 where the Beta-SIM grid map contained more twists and folds than WeVoS-Beta-SIM.

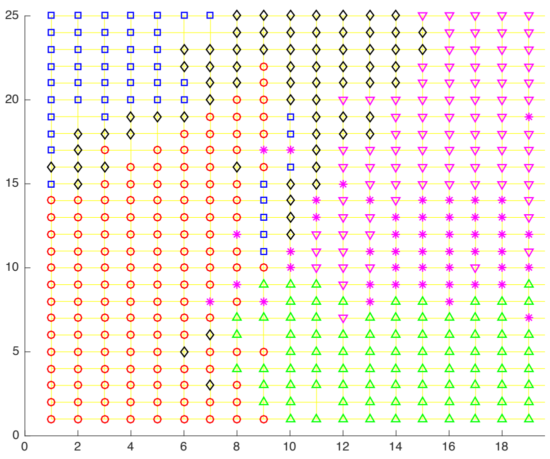
In the case of SOM and WeVoS-SOM (Fig. 5.5d and Fig. 5.5e), both algorithms produced maps where groups are clearly defined, however some groups present mixed classes. For instance, in the top of both maps (Fig. 5.5d and Fig. 5.5e), class 2 (blue squares) is divided into two groups separated by a group of class 1 (red circles). In the case of classes 2 and 6 (blues squares and magenta asterisks respectively), in both maps several neurons appear lost in the middle of the map.

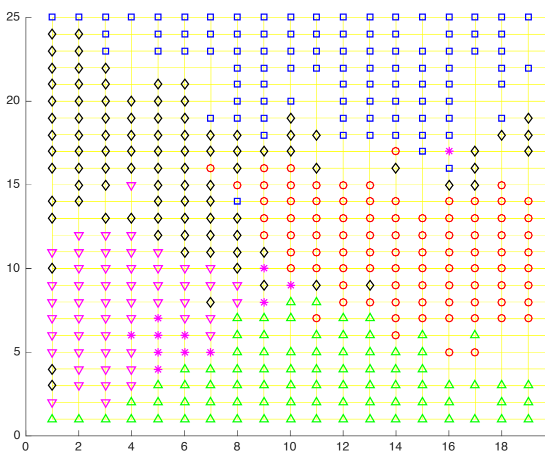

a) SIM


b) Beta-SIM


c) WeVoS-Beta-SIM


d) SOM


e) WeVoS-SOM


f) MLHL-SIM

g) ViSOM

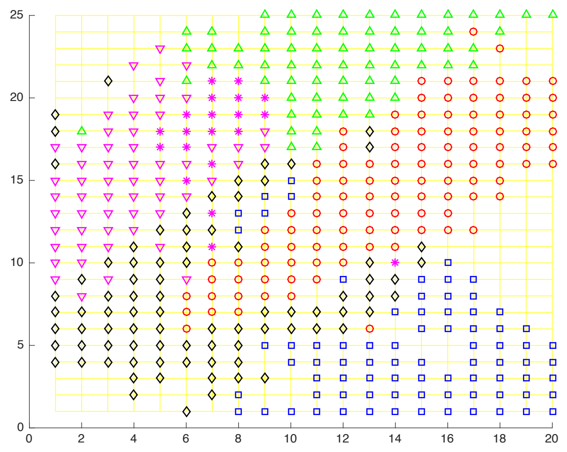


Fig. 5.5 Algorithms final network grid adaptations over the Landsat Satellite dataset.

5.3.2.3. Conclusions of the Visualization Results

The results suggest that WeVoS-Beta-SIM provides a better visual representation of the datasets than the other algorithms, as it is able to widely spread its grid map covering the input space better than the other tested models. At the same time, WeVoS-Beta-SIM obtains grid maps with less twists and folds than the Beta-SIM algorithm, which signifies a better topology of the map.

The improvement on visual representation, achieved by WeVoS-Beta-SIM, is notably higher when the complexity of the datasets increases. Using the previous examples, differences between maps were minor for the Iris dataset (low complexity), whereas differences were higher for the Landsat Satellite dataset (high complexity).

5.3.3. Analytical results

In order to validate the results obtained, statistical tests for the three quality measures were performed consisting of an ANOVA + post-hoc analyses. The statistical results for CE, MQE and TE are presented from Table 8.4 to Table 8.6 (p-values) and from Table 8.7 to Table 8.9 (average testing values ± standard deviation).

All measures presented are error measures for the testing dataset, so the desired value is always as close to 0 as possible. The CE is presented in percentage form and normalized between 0 and 1, while the rest of the measures are expressed as absolute values.

These series of experiments analyze 2 different aspects of the novel WeVoS-Beta-SIM and the other tested algorithms:

- The performance of WeVoS-Beta-SIM in comparison with the other 7 topology preserving models, in terms of CE, MQE and TE quality measures.
- The effect of modifying the number of data samples used during the training process for all algorithms under study. This was done in order to emulate the addition of noise or instability in the datasets [51].

5.3.3.1. Analysis of results in terms of CE

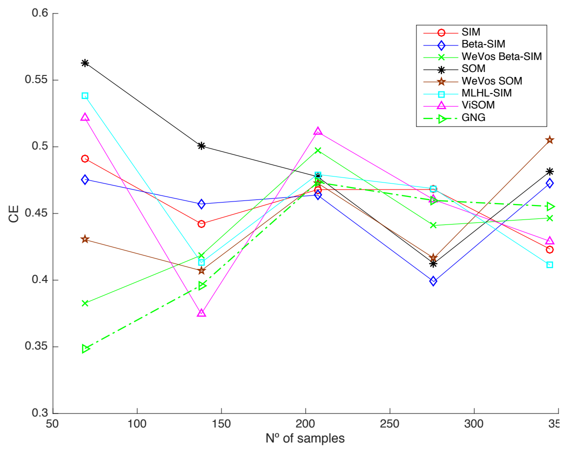
Results presented in Fig. 5.6 show that GNG, Beta-SIM and WeVoS-Beta-SIM often obtain better results than the other algorithms (SOM, ViSOM, WeVoS-SOM, SIM, MLHL-SIM), in terms of CE values. It should be noted that these differences in CE can only be seen when the complexity of the datasets is high. For low complexity datasets (in this research those having less than 5 classes; datasets from D1 to D8), CE results obtained by the different algorithms were not statistically significant (see results of Table 8.4 Annex).

Results for this experiment confirm the conclusions obtained by the visual representation test (figures from Fig. 5.2 to Fig. 5.5). For example, in the case of the Iris dataset (a dataset with low complexity), all algorithms presented very similar final maps (Fig. 5.3), so similar CE values for all algorithms were expected. In the case of the Landsat Satellite dataset (high complexity), the WeVoS-Beta-SIM algorithm obtained a map (Fig. 5.5c) with more compact groups and less mixed classes, therefore obtaining a better CE value than the other algorithms.

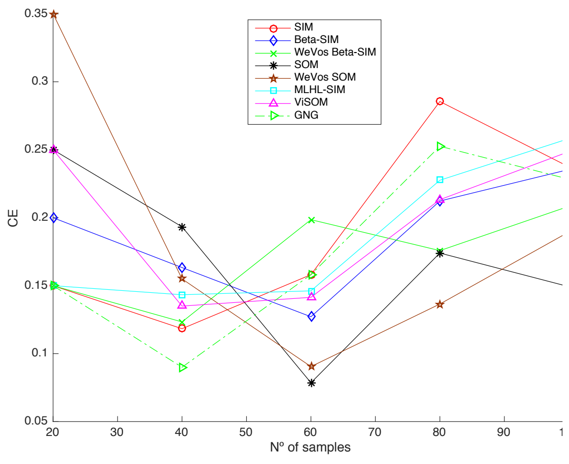
It can also be seen in Fig. 5.6 that when the complexity of the datasets increases, the WeVoS-Beta-SIM obtains better CE results than the Beta-SIM algorithm. However, when a statistical test of the results is performed (Table 8.4 Annex) differences between CE values were not statistically significant.

Finally, the change in CE values when the number of samples is increased was analyzed (Fig. 5.6). The effect of adding such instability in this case is not particularly evident, as the change in CE values did not follow a clear tendency.

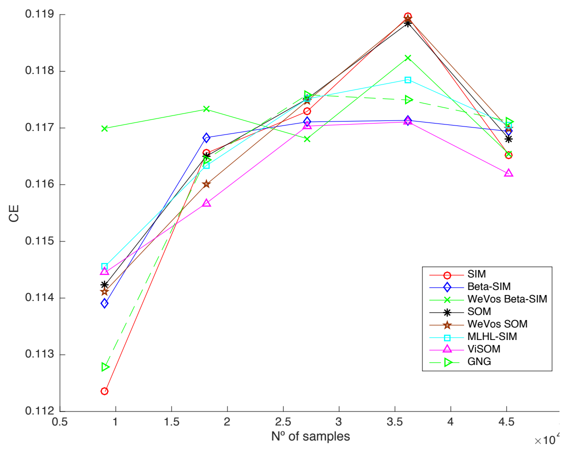
a) Dataset 1



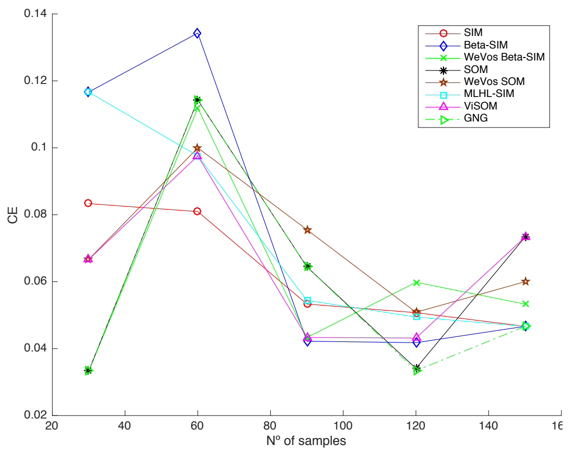
b) Dataset 2



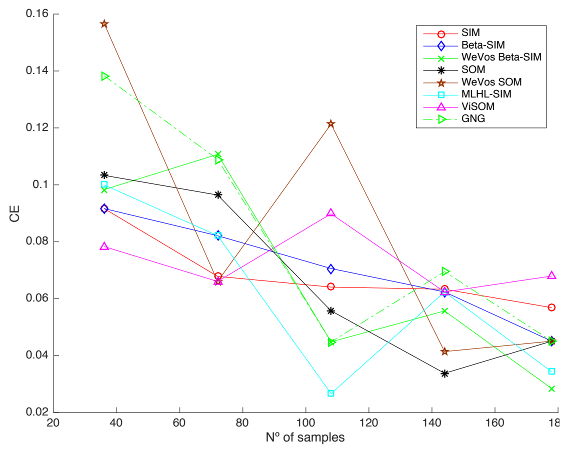
c) Dataset 3



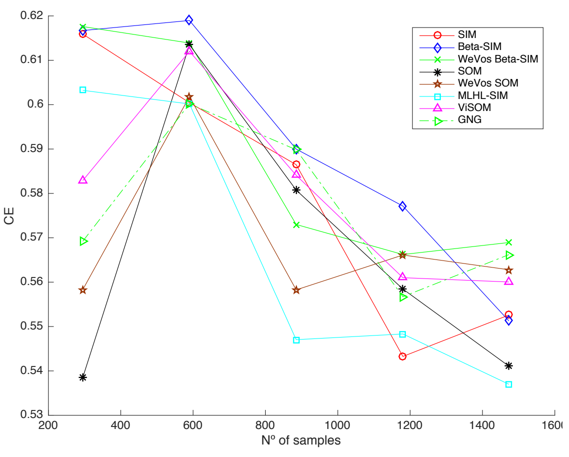
d) Dataset 4



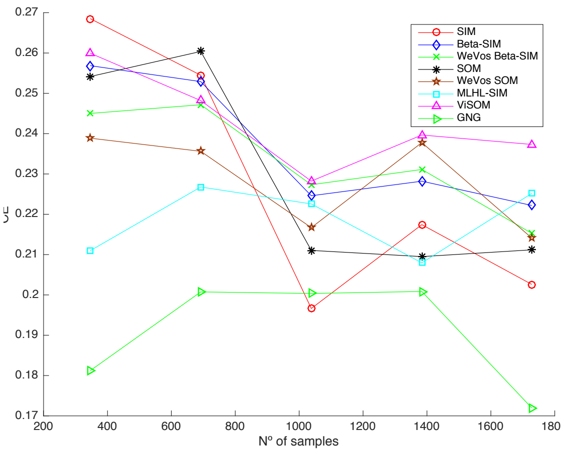
e) Dataset 5



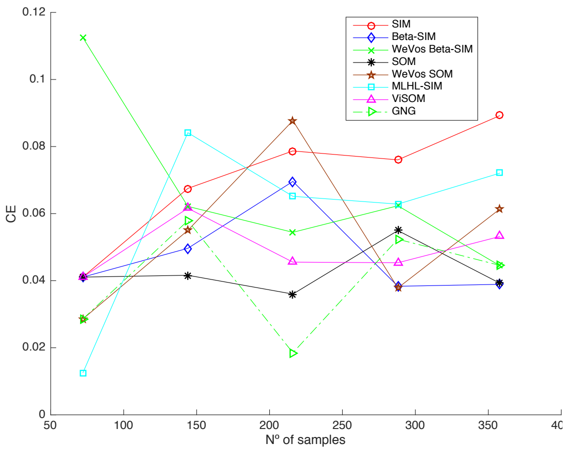
f) Dataset 6



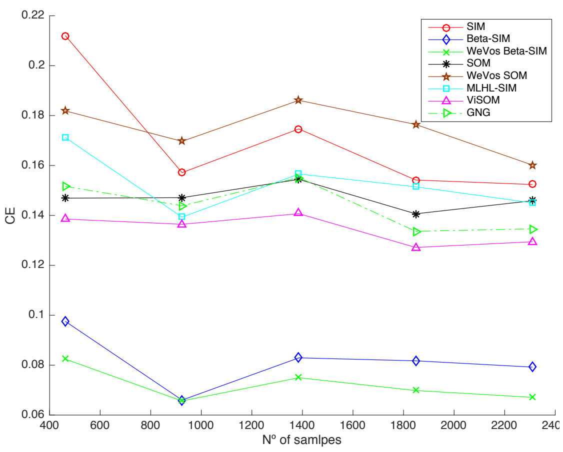
g) Dataset 7



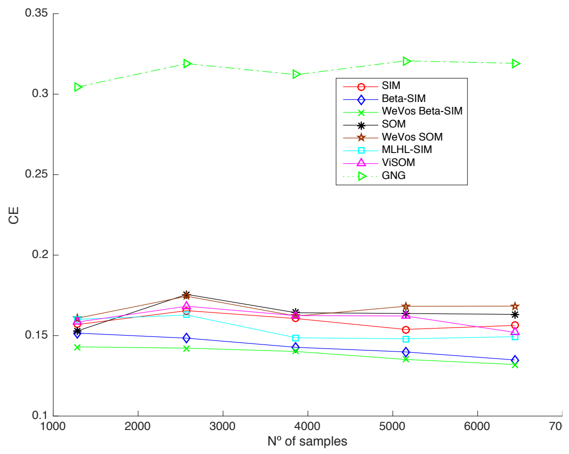
h) Dataset 8



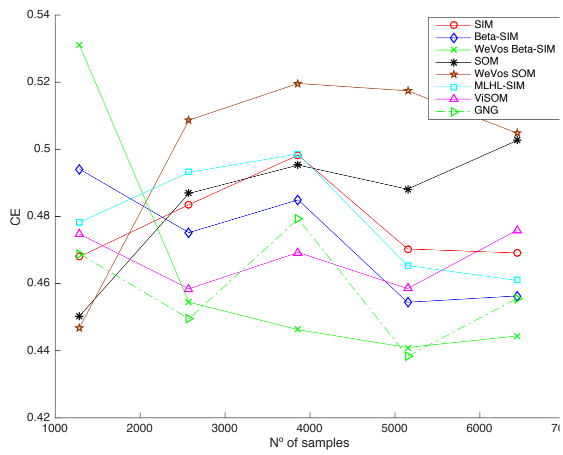
i) Dataset 9



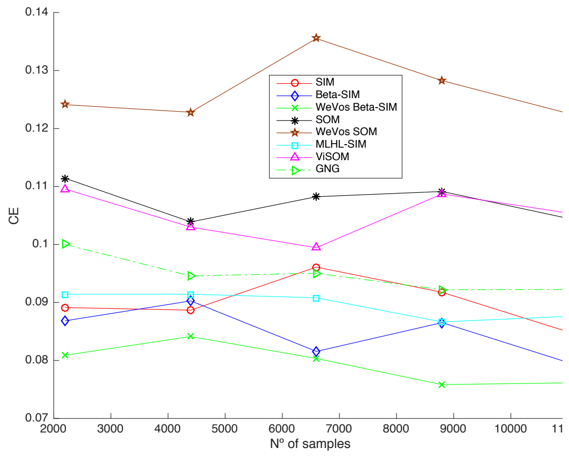
j) Dataset 10



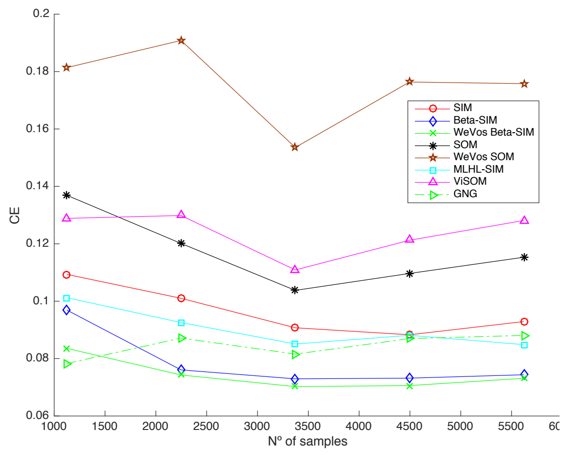
k) Dataset 11



l) Dataset 12



m) Dataset 13



n) Dataset 14

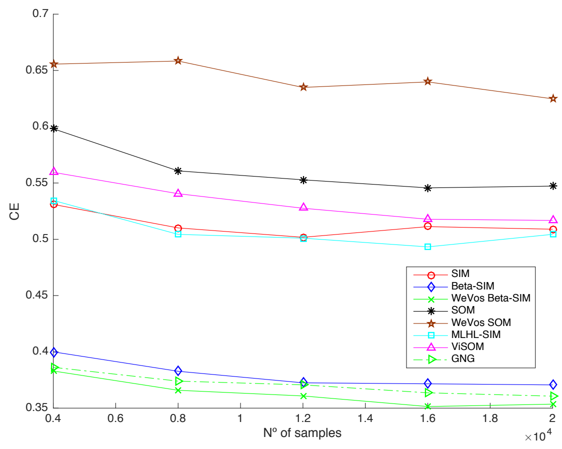


Fig. 5.6 CE measures vs number of samples of each algorithm for each of the 14 datasets.

5.3.3.2. Analysis of results in terms of MQE

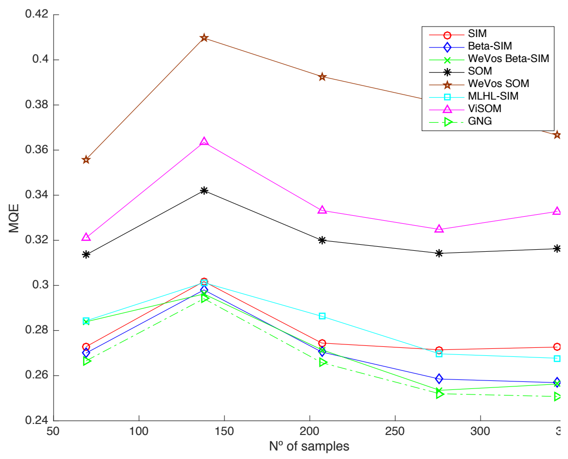
Results for MQE values are presented in Fig. 5.7, where it is clear that SOM and WeVoS-SOM algorithms obtain the highest MQE values for all datasets. This is expected based on results obtained in the visual representation test (Fig. 5.2 and Fig. 5.4), where WeVoS-SOM and SOM grid maps do not spread over the datasets as well as the other algorithms.

The rest of the algorithms behave in a similar way, in terms of MQE, where none of which outperform the others when complexity of the dataset is low. However, when complexity of the datasets increases, 2 algorithms stand out over the others in terms of MQE results: WeVoS-Beta-SIM and Beta-SIM algorithms. Again, these results confirm the conclusions obtained by the visual representation tests ("5.3.2 Visualization results"), where when the complexity of the datasets is high, the maps of these algorithms cover the input space better than the other algorithms.

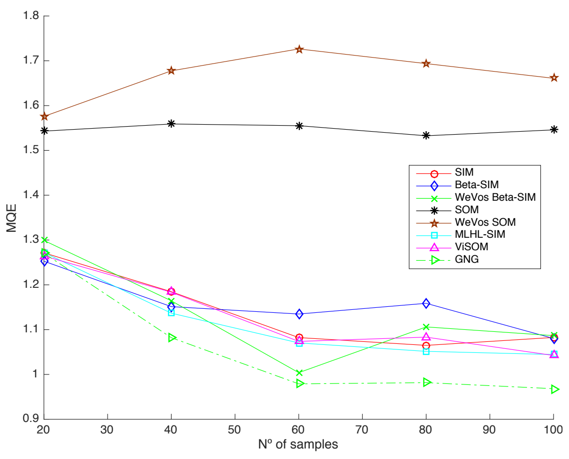
Fig. 5.7 shows that the MQE results obtained by WeVoS-Beta-SIM are often better than the simple model Beta-SIM, even when the complexity of the dataset is low. However, these differences are not always statistically significant (see Table 8.5 Annex).

Finally, the change in MQE values when the number of samples is increased was analyzed. In this case, adding such instability does not present a particular effect on the algorithms, with the MQE being more dependent on the total size of each dataset.

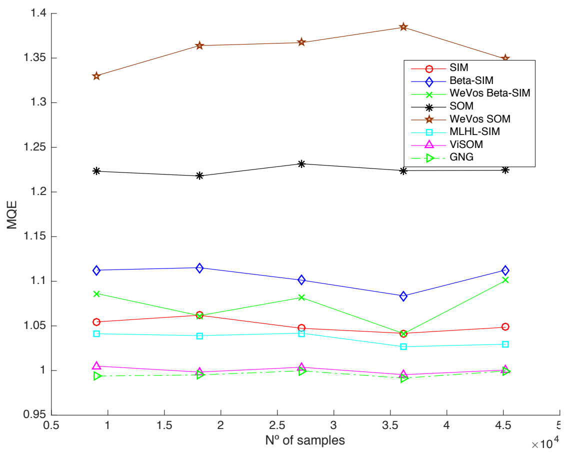
a) Dataset 1



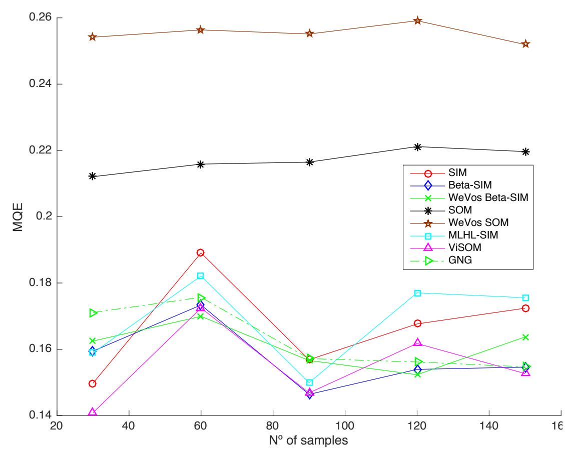
b) Dataset 2



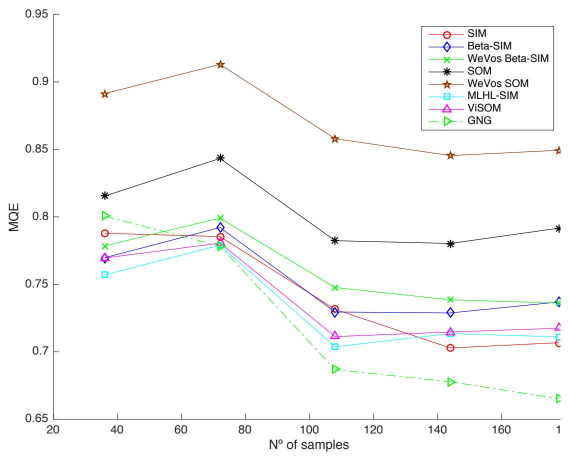
c) Dataset 3



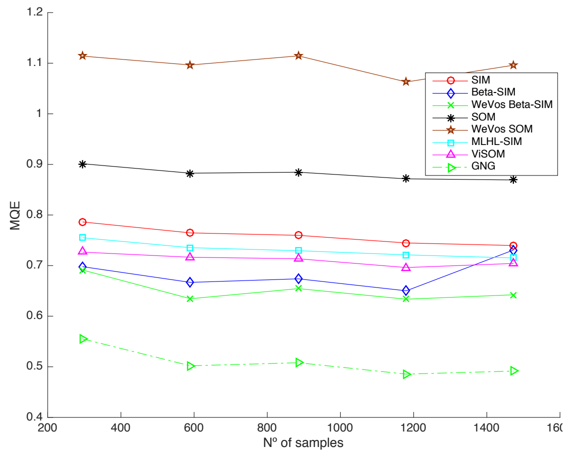
d) Dataset 4



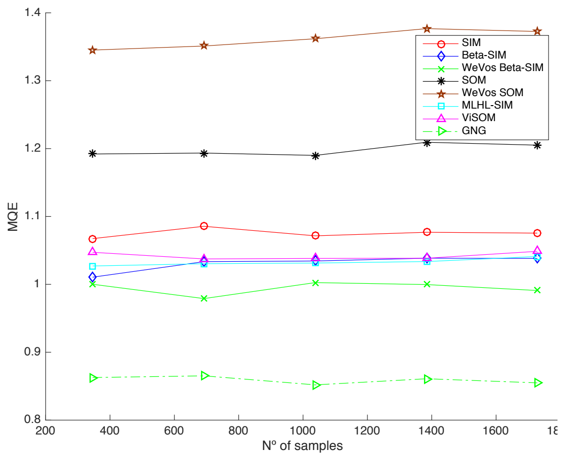
e) Dataset 5



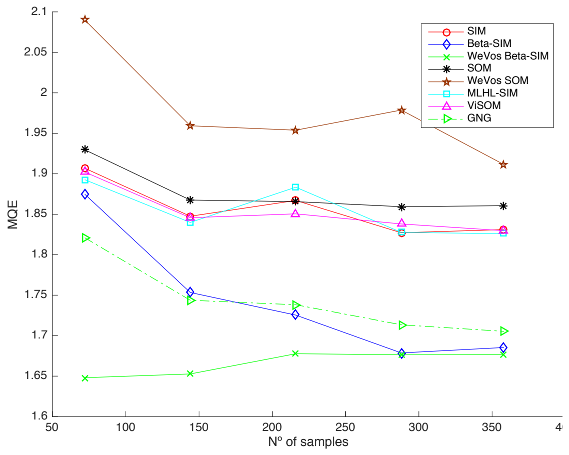
f) Dataset 6



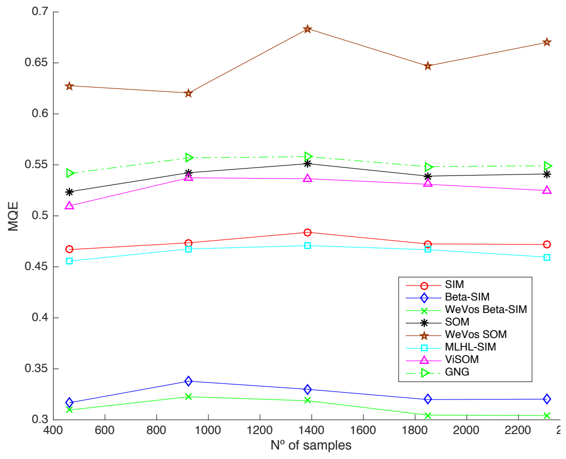
g) Dataset 7



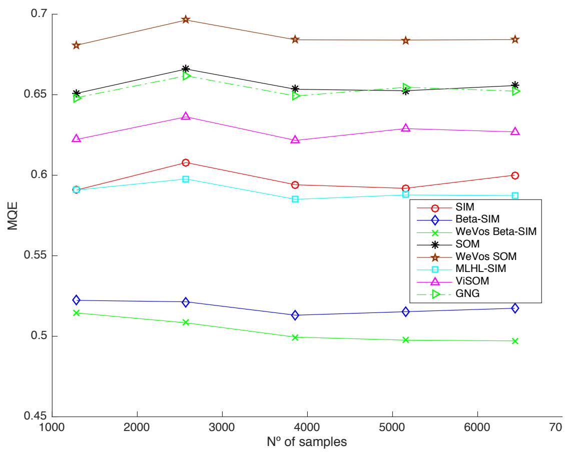
h) Dataset 8



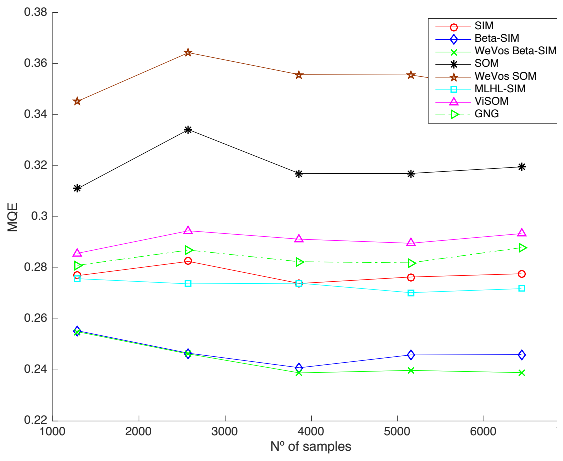
i) Dataset 9



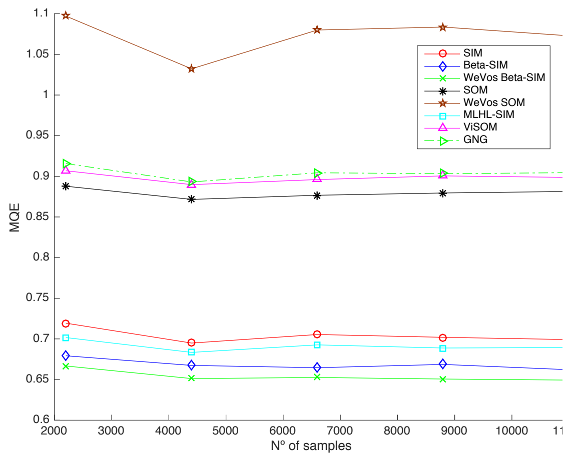
j) Dataset 10



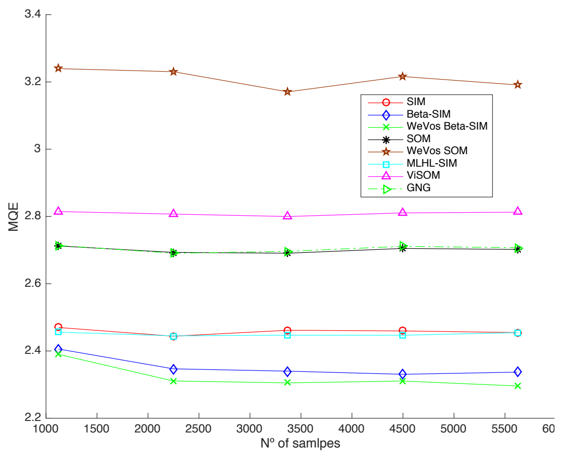
k) Dataset 11



l) Dataset 12



m) Dataset 13



n) Dataset 14

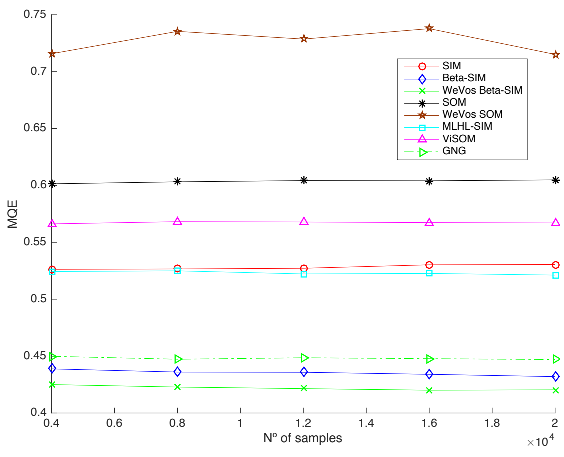


Fig. 5.7 MQE measures vs number of samples of each algorithm for each of the 14 datasets.

5.3.3.3. Analysis of results in terms of TE

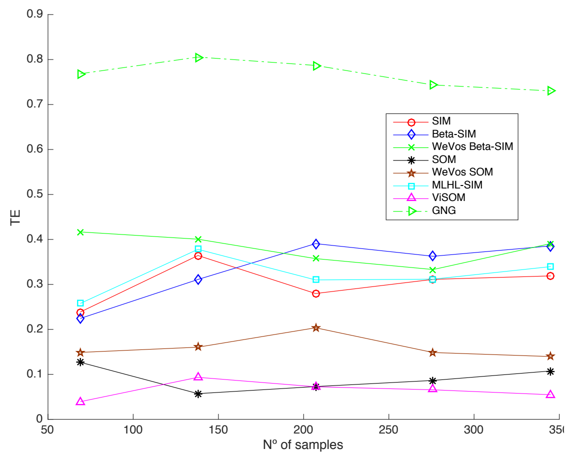
Finally, the TE results are analyzed and presented in Fig. 5.8. The TE is related with the topology of the final maps, as was shown in the visual representation section ("5.3.2 Visualization results"). Based on those results, it was expected that WeVoS-SOM, SOM and ViSOM would obtain the lowest TE, as these algorithms produced maps with less twists and folds. Results of Fig. 5.8 confirm it, but differences can only be seen when the complexity of the dataset is high.

When complexity of the datasets increases and differences between algorithms are clear (datasets from D8 to D14), in terms of TE values, GNG and Beta-SIM algorithm obtain the worst TE values. In the case of GNG, this is due to the fact that some units of the final map are disregarded from the final model and therefore the topology preservation is lost. In the case of the Beta-SIM algorithm, it focuses on distributing the units of the grid map over the dataset, so the final map contains several twists and folds (Fig. 5.2b and Fig. 5.4b).

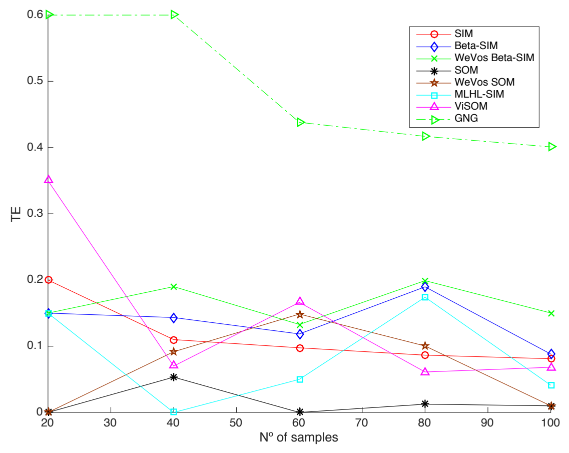
WeVoS-Beta-SIM (Fig. 5.8) consistently improves the TE results of the simple model Beta-SIM, especially when the complexity of the datasets increases. This means that the final map provides a better visualization regarding the topology preservation of the map (better TE) while keeping similar MQE values. The statistical tests presented in Table 8.6 (Annex) validate these results when complexity of the datasets is high.

The effect of adding instability by decreasing the number of samples for the training process was analyzed for WeVoS-Beta-SIM and its simple version Beta-SIM. When the complexity of the datasets is high, WeVoS-Beta-SIM presents very similar TE values as the number of samples is increases. However, Beta-SIM algorithm (Fig. 5.8) obtain unstable TE values (increasing and decreasing without a clear tendency). Therefore, it can be concluded that WeVoS-Beta-SIM, in terms of TE, is less sensitive to noise than the Beta-SIM algorithm. Therefore, WeVoS-Beta-SIM obtains maps with less distortion which in turn provides a better visual representation of the internal dataset structure.

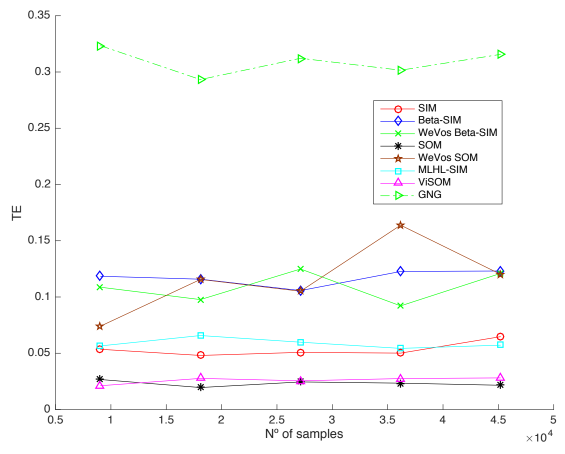
a) Dataset 1



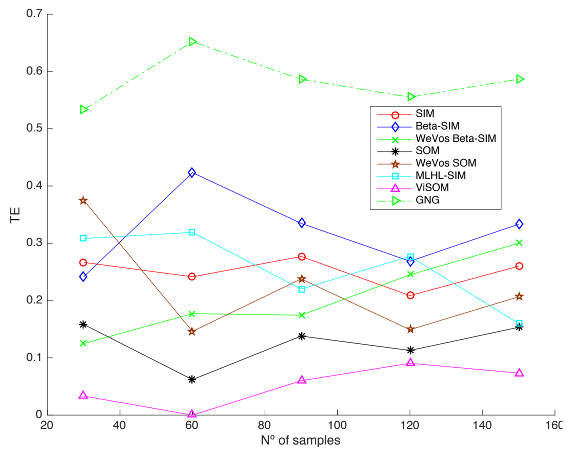
b) Dataset 2



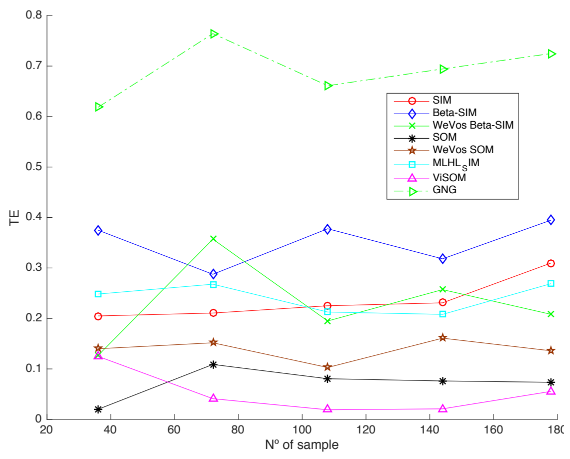
c) Dataset 3



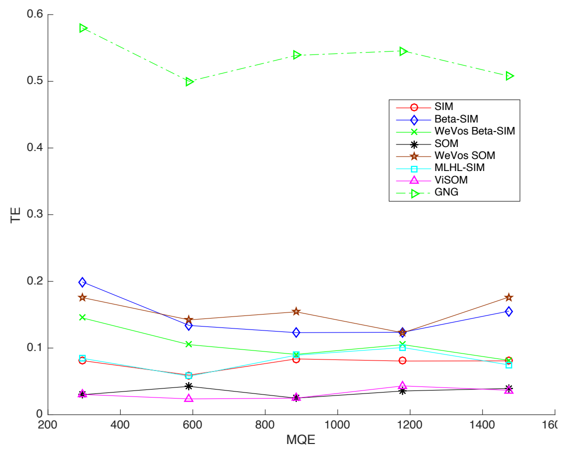
d) Dataset 4



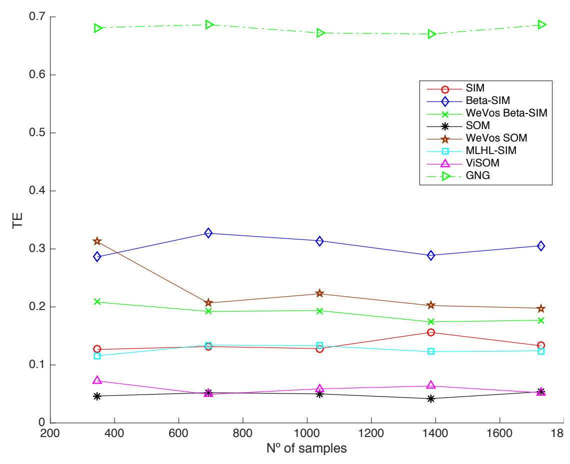
e) Dataset 5



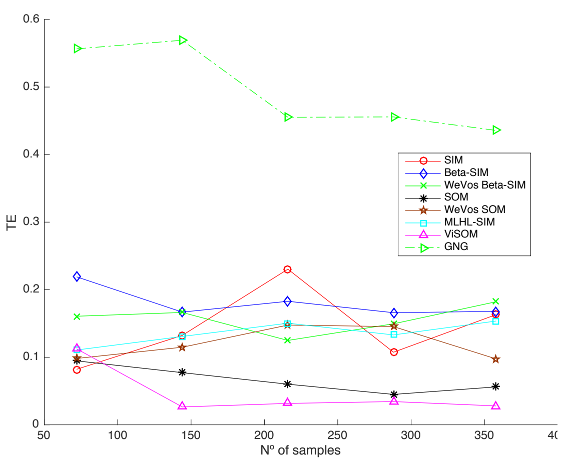
f) Dataset 6



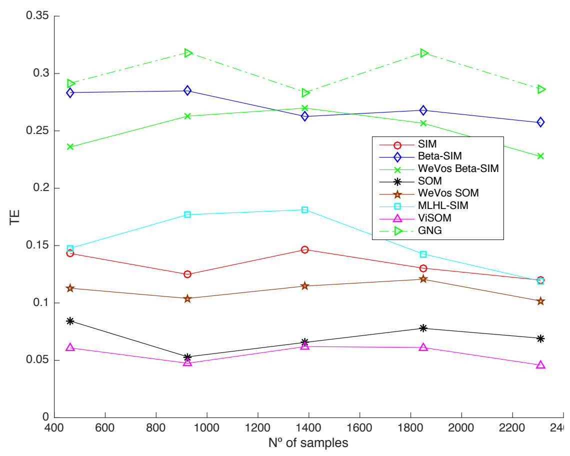
g) Dataset 7



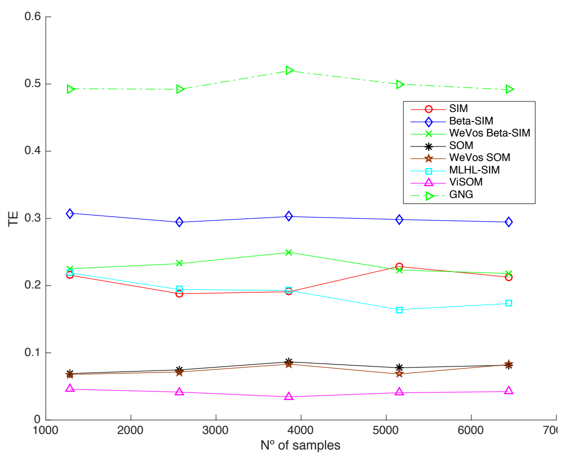
h) Dataset 8



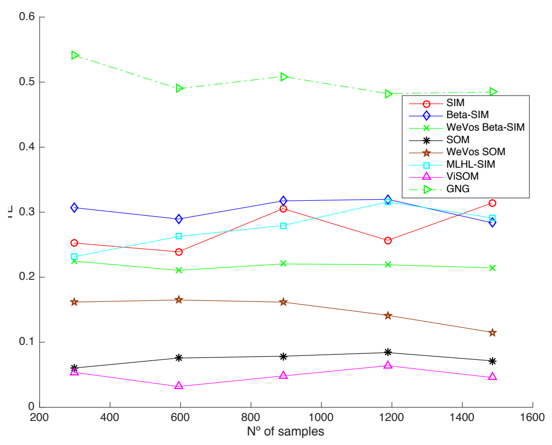
i) Dataset 9



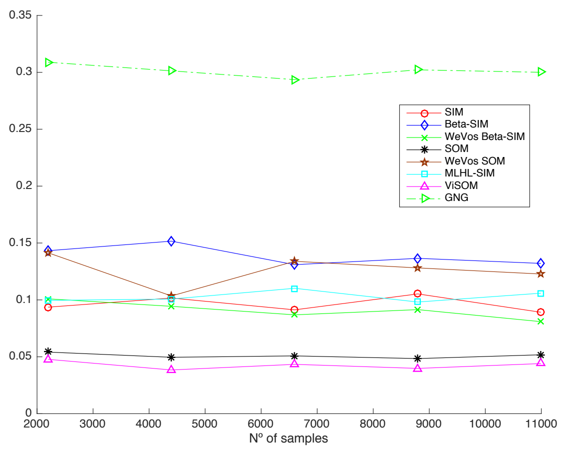
j) Dataset 10



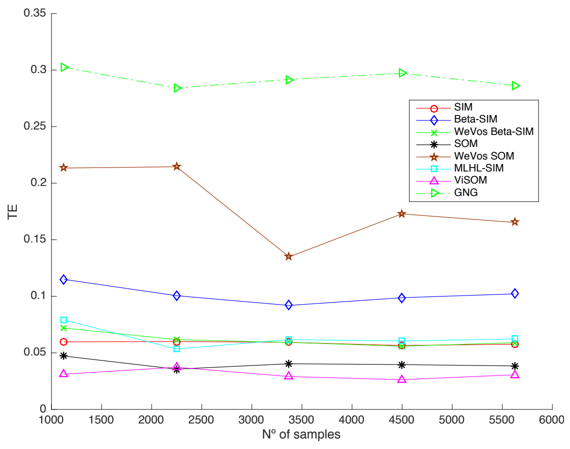
k) Dataset 11



l) Dataset 12



m) Dataset 13



n) Dataset 14

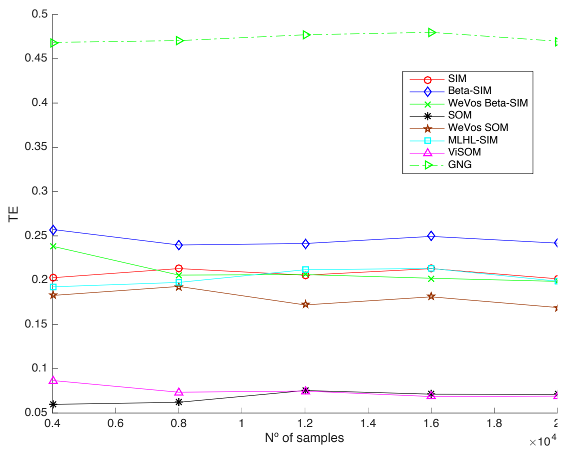


Fig. 5.8 TE measures vs number of samples of each algorithm for each of the 14 datasets.

5.3.3.4. <u>Conclusion of the analytical results</u>

Analytical results of this subsection confirm the conclusions obtained by the visual representation tests (Section 5.2). When complexity of the datasets is high, WeVoS-Beta-SIM algorithm often obtains the best MQE results, which corresponds to a more open and widely spread grid map over the dataset.

WeVoS-Beta-SIM also obtains the best CE results providing final maps with clearly defined groups, again when complexity of the datasets is high.

At the same time, the TE results of WeVoS-Beta-SIM are better than those obtained by the simple model Beta-SIM, providing to the final map a better topology preservation. Therefore, it can be concluded that the WeVoS-Beta-SIM is able to provide the best visual representation of the internal structure of datasets when their complexity is high (i.e. in this research means more than 5 classes).

Finally, in terms of stability, WeVoS-Beta-SIM is less sensitive to noise in terms of TE than the simple model Beta-SIM, which lead to obtain maps with less distortion effect which in turn provides a better visual representation of the internal dataset structure. However, CE and MQE results do not show a clear tendency when instability is added.

5.4. *Conclusions for WeVoS-Beta-SIM*

In this chapter, a novel topology-preserving model known as WeVoS-Beta-SIM has been presented, analyzed and compared with other well-known topology preserving models. This algorithm aims to obtain the best topology preserving summary as possible in order to improve the visual representation of high dimensional datasets and to increase the stability of the original model (Beta-SIM).

Therefore, the use of ensemble WeVoS when applying to the Beta-SIM algorithm improves the visual representation of the internal structure of high complex datasets (more than 5 classes), generating grid maps widely spread and that covers the input space better than the other models (better MQE values). At the same time, WeVoS-Beta-SIM obtains maps with less twists and folds than the simple model Beta-SIM algorithm (better TE values), which signifies a better topology of the map.

As can be seen in the results, the improvement on visual representation, created by WeVoS-Beta-SIM, is notably higher when complexity of the datasets increases. With very simple datasets, it only makes slight improvements or can even obtain worse results. That said, its usefulness has been proven in the case of more complex datasets (more than 5 classes), where the extra complexity of the calculation of the ensemble leads to the considerable increase of performance, obtaining a better organization and visualization of the presented information.

Results also show that WeVoS-Beta-SIM is less sensitive to noise in terms of TE than the simple model Beta-SIM. It leads to maps with less distortion which in turn provides a better visual representation of the internal dataset structure.

Chapter 6. Conclusions & Future Work

In this thesis, a novel family of learning rules called Beta Hebbian Learning (BHL) has been explored. It has been the base for the development of 3 novel algorithms of unsupervised learning. The main objectives were to facilitate the information extraction, clustering and visualization of the internal structure of high complex datasets.

A complete study of the algorithms in combination with several well-known algorithms such as PCA, MLHL, LLE, CCA, Isomap, SOM, ViSOM, SIM, MLHL-SIM, GNG and WeVoS-SOM, was presented. In this study the behavior of the novel algorithms was thoroughly analyzed and successfully demonstrated over artificial and real datasets, comparing their results, in terms of three performance quality measures such as CE, MQE and TE, with other well-known algorithms. The main conclusions extracted from the tests are:

- BHL: A new family of learning rules have been derived which are based on maximising the likelihood of the residual from a negative feedback network when such residual is deemed to come from the Beta Distribution The power of BHL comes from the choice of an appropriate function depending on α and β parameters, to maximize the likelihood of the residuals (e) under particular models of PDFs. Therefore, it will allow to derive the most appropriate learning rule to the actual PDF of the residual, providing a better visualization of the internal structure of a dataset.

- BETA-SIM: Beta-SIM is capable to obtain better topology preserving map than other well-known algorithms, in order to be used as a reliable tool in data visualization. Due to the inherent capabilities of the SIM, their combination with the BHL algorithm improves adaptation and visualization of datasets with a radial structure, as has been successfully shown in the tests. The main improvement of the algorithm is the capacity to adapt to sparse clusters or to neglect outliers depending on the combination of values of α and β and the task to be carried out.

- WeVoS-Beta-SIM: This novel algorithm improves the visual representation of the internal structure of high complex datasets, generating grid maps widely spread and that covers the input space better than the other models. At the same time, WeVoS-Beta-SIM obtains maps with less twists and folds than the simple model Beta-SIM algorithm, which signifies a better topology of the map and improving its stability when using complex datasets.

Future work includes the application of BHL rule to other topology preserving models, such as Self Organizing Maps (SOM) and Visualization Inducted SOM (ViSOM) [51], to generate new algorithms for boosting performance. Also BHL, Beta-SIM and WeVoS-Beta-SIM should be applied to analyze challenging real datasets to solve Data Mining and Big Data problems in the field of the Electric Vehicle, Energy Efficiency, Cybersecurity, etc.

Conclusiones y trabajo futuro

En esta tesis se ha investigado una nueva familia de reglas de aprendizaje llamada Beta Hebbian Learning (BHL) que ha sido la base para el desarrollo de 3 nuevos algoritmos de aprendizaje no supervisado. Los principales objetivos eran facilitar la extracción de información, la agrupación y la visualización de la estructura interna de conjuntos de datos complejos.

Se ha presentado un estudio completo de los 3 nuevos algoritmos desarrollados en comparación con varios algoritmos ampliamente conocidos como *PCA, MLHL, LLE, CCA, Isomap, SOM, ViSOM, SIM, MLHL-SIM, GNG* y *WeVoS-SOM*. En este estudio, el comportamiento de los nuevos algoritmos ha sido analizado en profundidad y demostrado con éxito sobre conjuntos de datos artificiales y reales. Los resultados fueron comparados en términos de tres medidas de calidad de rendimiento, como son el *CE*, el *MQE* y el *TE* con otros algoritmos ampliamente conocidos. Las principales conclusiones extraídas de los experimentos son:

- BHL: Se ha derivado una nueva familia de reglas de aprendizaje que se basan en maximizar la probabilidad del residuo de una red de retroalimentación negativa cuando se considera que dicho residuo procede de la Distribución Beta. El potencial de BHL viene de la elección de una función apropiada en base a los parámetros α y β, para maximizar la probabilidad de los residuos (*e*) en modelos particulares de PDFs. Por lo tanto, permite derivar la regla de aprendizaje más apropiada al PDF real del residuo, proporcionando una mejor visualización de la estructura interna de un conjunto de datos.

- BETA-SIM: Beta-SIM es capaz de obtener mejores mapas de preservación de topología que otros algoritmos ampliamente usados, pudiendo ser utilizado como una herramienta fiable en la visualización de datos. Debido a las capacidades inherentes del SIM, su combinación con el algoritmo BHL mejora la adaptación y visualización de conjuntos de datos con una estructura radial, como se ha demostrado con éxito en los experimentos realizados. La principal mejora del algoritmo es la capacidad de adaptación a grupos dispersos o de no tener en consideración los "outliers" dependiendo de la combinación de valores de α y β y la tarea a realizar.

- WeVoS-Beta-SIM: Este nuevo algoritmo mejora la representación visual de la estructura interna de conjuntos de datos complejos, generando mapas con estructuras ampliamente extendidas y que cubre el espacio de entrada mejor que los otros modelos. Al mismo tiempo, WeVoS-Beta-SIM obtiene mapas con menos giros y pliegues que el algoritmo de su modelo simple, el Beta-

SIM, lo que significa una mejor topología del mapa, mejorando su estabilidad cuando se utilizan conjuntos de datos complejos.

El trabajo futuro incluye la aplicación de la regla de BHL a otros modelos de preservación de topología, tales como Mapas Auto organizados (SOM) y *"Visual Inducted SOM"* (ViSOM) [51], para generar nuevos algoritmos que mejoren los resultados. También se aplicará BHL, Beta-SIM y WeVoS-Beta-SIM al análisis de conjuntos de datos reales para resolver problemas de *Data Mining* y *Big Data* en el campo del Vehículo Eléctrico, Eficiencia Energética, Ciberseguridad, etc.

Chapter 7. References

[1] G. G. Sundarkumar and V. Ravi, ‘A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance’, *Eng. Appl. Artif. Intell.*, vol. 37, pp. 368–377, Jan. 2015.

[2] D. T. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[3] X. Yao and B. Van Durme, ‘Information Extraction over Structured Data: Question Answering with Freebase.’, in *ACL (1)*, 2014, pp. 956–966.

[4] J. Piskorski and R. Yangarber, ‘Information extraction: Past, present and future’, in *Multi-source, multilingual information extraction and summarization*, Springer, 2013, pp. 23–49.

[5] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[6] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, ‘Data Mining with Big Data’, *Ieee Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[7] A. Moran *et al.*, ‘Analysis of electricity consumption profiles in public buildings with dimensionality reduction techniques’, *Eng. Appl. Artif. Intell.*, vol. 26, no. 8, pp. 1872–1880, Sep. 2013.

[8] D. Perez *et al.*, ‘Visual analysis of a cold rolling process using a dimensionality reduction approach’, *Eng. Appl. Artif. Intell.*, vol. 26, no. 8, pp. 1865–1871, Sep. 2013.

[9] P. Sarlin, ‘Exploiting the self-organizing financial stability map’, *Eng. Appl. Artif. Intell.*, vol. 26, no. 5–6, pp. 1532–1539, Jun. 2013.

[10] Á. Herrero, L. S. Bárcena, M. A. Manzanedo, and E. Corchado, ‘A hybrid proposal for cross-sectoral analysis of knowledge management.’, *Soft Comput*, vol. 20, no. 11, pp. 4271–4285, 2016.

[11] B. Baruque and E. Corchado, ‘WeVoS scale invariant map’, *Inf Sci*, vol. 280, pp. 307–321, 2014.

[12] M. Ahmadlou and H. Adeli, ‘Enhanced Probabilistic Neural Network with Local Decision Circles: A Robust Classifier’, *Integr Comput-Aided Eng*, vol. 17, no. 3, pp. 197–210, Aug. 2010.

[13] N. Siddique and H. Adeli, *Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing*. John Wiley & Sons, 2013.

[14] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] E. Corchado, D. MacDonald, and C. Fyfe, ‘Maximum and minimum likelihood Hebbian learning for exploratory projection pursuit’, *Data Min. Knowl. Discov.*, vol. 8, no. 3, pp. 203–225, 2004.

[16] S. Espezua, E. Villanueva, and C. D. Maciel, ‘Towards an efficient genetic algorithm optimizer for sequential projection pursuit’, *Neurocomputing*, vol. 123, pp. 40–48, Jan. 2014.

[17] A. Berro, S. Larabi Marie-Sainte, and A. Ruiz-Gazen, ‘Genetic algorithms and particle swarm optimization for exploratory projection pursuit’, *Ann. Math. Artif. Intell.*, vol. 60, no. 1, pp. 153–178, 2010.

[18] S. Hou and P. D. Wentzell, ‘Re-centered kurtosis as a projection pursuit index for multivariate data analysis’, *J. Chemom.*, vol. 28, no. 5, pp. 370–384, 2014.

[19] N. Chen, B. Ribeiro, A. Vieira, and A. Chen, ‘Clustering and visualization of bankruptcy trajectory using self-organizing map’, *Expert Syst. Appl.*, vol. 40, no. 1, pp. 385–393, Jan. 2013.

[20] J. J. Fuertes, M. Domínguez, P. Reguera, M. A. Prada, I. Díaz, and A. A. Cuadrado, ‘Visual dynamic model based on self-organizing maps for supervision and fault detection in industrial processes’, *Eng. Appl. Artif. Intell.*, vol. 23, no. 1, pp. 8–17, 2010.

[21] T. Kohonen, ‘The self-organizing map’, *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

[22] E. Mohebi and A. Bagirov, ‘Constrained Self Organizing Maps for Data Clusters Visualization’, *Neural Process. Lett.*, vol. 43, no. 3, pp. 849–869, Jun. 2016.

[23] Y. Wu, T. K. Doyle, and C. Fyfe, ‘Multi-layer Topology Preserving Mapping for K-Means Clustering’, in *Intelligent Data Engineering and Automated Learning - Ideal 2011*, vol. 6936, H. Yin, W. Wang, and V. RaywardSmith, Eds. Berlin: Springer-Verlag Berlin, 2011, pp. 84–91.

[24] E. K. Haimoudi, H. Fakhouri, L. Cherrat, and M. Ezziyyani, ‘Towards a New Approach to Improve the Classification Accuracy of the Kohonen’s Self-Organizing Map During Learning Process’, *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 3, pp. 230–236, Mar. 2016.

[25] T. Kohonen, ‘Essentials of the self-organizing map’, *Neural Netw.*, vol. 37, pp. 52–65, 2013.

[26] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. J. Hudspeth, *Principles of neural science*, vol. 4. McGraw-hill New York, 2000.

[27] S. Jacobs, ‘Brain Images: New Techniques And Bright Colors’. [Online]. Available: http://www.wbur.org/news/2014/06/12/brain-images. [Accessed: 27-Jan-2017].

[28] S. R. y Cajal, *Histologie du système nerveux de l’homme & des vertébrés*. Consejo Superior de Investigaciones Científicas. Instituto Ramón y Cajal, 1952.

[29] ‘Brain Function’. [Online]. Available: http://www.pointofreturn.com/brain_function.html. [Accessed: 27-Jan-2017].

[30] S. L. Hung and H. Adeli, ‘Parallel backpropagation learning algorithms on {CRAY} Y-MP8/864

supercomputer', *Neurocomputing*, vol. 5, no. 6, pp. 287–302, 1993.

[31] S. L. Hung and H. Adeli, 'A parallel genetic/neural network learning algorithm for MIMD shared memory machines', *IEEE Trans. Neural Netw. Publ. IEEE Neural Netw. Counc.*, vol. 5, no. 6, pp. 900–909, 1994.

[32] W. S. McCulloch and W. Pitts, 'A logical calculus of the ideas immanent in nervous activity', *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[34] B. Kosko, 'Adaptive bidirectional associative memories', *Appl. Opt.*, vol. 26, no. 23, pp. 4947–4960, 1987.

[35] J. Schmidhuber, 'Deep learning in neural networks: An overview', *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[36] T. Hastie, R. Tibshirani, and J. Friedman, 'Unsupervised Learning', in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, and J. Friedman, Eds. New York, NY: Springer New York, 2009, pp. 485–585.

[37] B. Baruque and E. Corchado, *Fusion Methods for Unsupervised Learning Ensembles*, vol. 322. Springer, 2011.

[38] S. Ghosh-Dastidar and H. Adeli, 'A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection', *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, Dec. 2009.

[39] D. O. Hebb, *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[40] S. Haykin, 'Neural Networks: A Comprehensive Foundation: Macmillan College Publishing Company', *N. Y.*, 1994.

[41] K. Peason, 'On lines and planes of closest fit to systems of point in space', *Philos. Mag.*, vol. 2, pp. 559–572, 1901.

[42] H. Hotelling, 'Analysis of a complex of statistical variables into principal components.', *J. Educ. Psychol.*, vol. 24, no. 6, p. 417, 1933.

[43] E. Oja, 'Simplified neuron model as a principal component analyzer', *J. Math. Biol.*, vol. 15, no. 3, pp. 267–273, 1982.

[44] E. Oja, 'Neural networks, principal components, and subspaces', *Int. J. Neural Syst.*, vol. 01, no.

01, pp. 61–68, Jan. 1989.

[45] E. Oja, ‘Principal components, minor components, and linear neural networks’, *Neural Netw.*, vol. 5, no. 6, pp. 927–935, 1992.

[46] C. Fyfe, ‘Negative feedback as an organising principle for artificial neural networks. Ph.D. Thesis’, Strathclyde University, 1995.

[47] D. Charles and C. Fyfe, ‘Modelling multiple-cause structure using rectification constraints’, *Netw. Comput. Neural Syst.*, vol. 9, no. 2, pp. 167–182, 1998.

[48] D. E. Rumelhart and D. Zipser, ‘Feature discovery by competitive learning’, *Cogn. Sci.*, vol. 9, no. 1, pp. 75–112, 1985.

[49] T. Kohonen, P. Lehtiö, J. Rovamo, J. Hyvärinen, K. Bry, and L. Vainio, ‘A principle of neural associative memory’, *Neuroscience*, vol. 2, no. 6, pp. 1065–1076, 1977.

[50] H. Yin, ‘Data visualisation and manifold mapping using the ViSOM’, *Neural Netw.*, vol. 15, no. 8, pp. 1005–1016, 2002.

[51] E. Corchado and B. Baruque, ‘WeVoS-ViSOM: An ensemble summarization algorithm for enhanced data visualization’, *Neurocomputing*, vol. 75, no. 1, pp. 171–184, 2012.

[52] B. Baruque and E. Corchado, ‘A Novel Ensemble of Scale-Invariant Feature Maps’, in *Computer Recognition Systems 3*, M. Kurzynski and M. Wozniak, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 265–273.

[53] M. Zapater, D. Fraga, P. Malagon, Z. Bankovic, and J. M. Moya, ‘Self-organizing Maps versus Growing Neural Gas in Detecting Anomalies in Data Centres’, *Log. J. Igpl*, vol. 23, no. 3, pp. 495–505, Jun. 2015.

[54] B. Fritzke, ‘A Growing Neural Gas Network Learns Topologies’, in *Advances in Neural Information Processing Systems 7*, 1995, pp. 625–632.

[55] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, ‘Neural-gas’ network for vector quantization and its application to time-series prediction’, *Neural Netw. IEEE Trans. On*, vol. 4, no. 4, pp. 558–569, 1993.

[56] B. Fritzke, ‘Growing cell structures—A self-organizing network for unsupervised and supervised learning’, *Neural Netw.*, vol. 7, no. 9, pp. 1441–1460, 1994.

[57] T. Martinetz, ‘Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps’, in *ICANN ’93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993*, S. Gielen and B. Kappen, Eds. London: Springer London, 1993, pp. 427–434.

[58] G. Pölzlbauer, ‘Survey and Comparison of Quality Measures for Self-Organizing Maps’, in *Proceedings of the Fifth Workshop on Data Analysis (WDA’04)*, Sliezsky dom, Vysoké Tatry, Slovakia, 2004, pp. 67–82.

[59] G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, ‘BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification’, *Eng. Appl. Artif. Intell.*, vol. 49, pp. 176–193, Mar. 2016.

[60] H. Elghazel and A. Aussem, ‘Unsupervised feature selection with ensemble learning’, *Mach. Learn.*, vol. 98, no. 1–2, pp. 157–180, 2015.

[61] B. Baruque, E. Corchado, and H. Yin, ‘The s2-ensemble fusion algorithm’, *Int. J. Neural Syst.*, vol. 21, no. 06, pp. 505–525, Dec. 2011.

[62] A. Georgakis, H. Li, and M. Gordan, ‘An Ensemble of SOM Networks for Document Organization and Retrieval’, in *Int. Conf. on Adaptive Knowledge Representation and Reasoning (AKRR’05)*, 2005, p. 6.

[63] C. Saavedra, R. Salas, S. Moreno, and H. Allende, ‘Fusion of self organizing maps’, in *International Work-Conference on Artificial Neural Networks*, 2007, pp. 227–234.

[64] J.W. Tukey and J.H. Friedman, ‘A Projection Pursuit Algorithm for Exploratory Data Analysis’, *IEEE Trans. Comput.*, vol. 23, no. undefined, pp. 881–890, 1974.

[65] P. Diaconis and D. Freedman, ‘Asymptotics of graphical projection pursuit’, *Ann. Stat.*, vol. 12, no. 3, pp. 793–815, 1984.

[66] J. Karhunen and J. Joutsensalo, ‘Representation and separation of signals using nonlinear PCA type learning’, *Neural Netw.*, vol. 7, no. 1, pp. 113–127, 1994.

[67] E. Oja, ‘Learning in nonlinear constrained Hebbian networks’, *Artif. Neural Netw.*, pp. 385–390, 1991.

[68] L. Xu, ‘Least mean square error reconstruction principle for self-organizing neural-nets’, *Neural Netw.*, vol. 6, no. 5, pp. 627–648, 1993.

[69] A. J. Bell and T. J. Sejnowski, ‘An information-maximization approach to blind separation and blind deconvolution’, *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, 1995.

[70] M. Girolami and C. Fyfe, ‘Stochastic ICA contrast maximisation using Oja’s nonlinear PCA algorithm’, *Int. J. Neural Syst.*, vol. 8, no. 05n06, pp. 661–678, 1997.

[71] J. Karhunen and J. Joutsensalo, ‘Nonlinear Hebbian algorithm for sinusoidal frequency estimation’, *Artif. Neural Netw.*, vol. 2, pp. 1099–1102, 1992.

[72] E. Oja and J. Karhunen, ‘Signal separation by nonlinear Hebbian learning’, *Comput. Intell. Dyn. Syst. Perspect.*, pp. 83–97, 1995.

[73] C. Fyfe, D. R. McGregor, and R. Baddeley, *Exploratory projection pursuit: an artificial neural network approach*. Department of Computer Science, University of Strathclyde, 1994.

[74] E. Corchado, M. A. Pellicer, and M. L. B. Diz, ‘A maximum likelihood Hebbian learning-based method to an agent-based architecture’, *Int J Comput Math*, vol. 86, no. 10&11, pp. 1760–1768, 2009.

[75] Á. Herrero, M. Navarro, E. Corchado, and V. Julián, ‘RT-MOVICAB-IDS: Addressing real-time intrusion detection’, *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 250–261, Jan. 2013.

[76] J. M. Mendel, Ed., *A Prelude to Neural Networks: Adaptive and Learning Systems*. Upper Saddle River, NJ, USA: Prentice Hall Press, 1994.

[77] E. Corchado and C. Fyfe, ‘Connectionist techniques for the identification and suppression of interfering underlying factors’, *Int. J. Pattern Recognit. Artif. Intell.*, vol. 17, no. 08, pp. 1447–1466, 2003.

[78] E. Corchado, Y. Han, and C. Fyfe, ‘Structuring global responses of local filters using lateral connections’, *J. Exp. Theor. Artif. Intell.*, vol. 15, no. 4, pp. 473–487, 2003.

[79] N. D. Socci, D. D. Lee, and H. Sebastian Seung, ‘The rectified Gaussian distribution’, *Adv. Neural Inf. Process. Syst.*, pp. 350–356, 1998.

[80] R. Redondo, J. Sedano, V. Vera, B. Hernando, and E. Corchado, ‘A novel hybrid intelligent system for multi-objective machine parameter optimization’, *Pattern Anal Appl*, vol. 18, no. 1, pp. 31–44, 2015.

[81] R. M. Sánchez, Á. Herrero, and E. Corchado, ‘Visualization and Clustering for SNMP Intrusion Detection’, *Cybern. Syst.*, vol. 44, no. 6–7, pp. 505–532, 2013.

[82] V. Vera, E. Corchado, R. Redondo, J. Sedano, and Á. E. García, ‘Applying soft computing techniques to optimise a dental milling process’, *Neurocomputing*, vol. 109, pp. 94–104, Jun. 2013.

[83] C. Pinzón, J. F. de Paz, Á. Herrero, E. Corchado, J. Bajo, and J. M. Corchado, ‘idMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL injection through data mining.’, *Inf Sci*, vol. 231, pp. 15–31, 2013.

[84] E. Corchado and C. Fyfe, ‘The Scale Invariant Map And Maximum Likelihood Hebbian Learning’, presented at the KES 2002 , Knowledge-based intelligent information engineering systems & allied technologies., 2002, vol. 82, pp. 245–249.

[85] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent component analysis*. New York, Chichester, Weinheim: John Wiley, 2001.

[86] F. J. Bremner, S. J. Gotts, and D. L. Denham, ‘Hinton diagrams: Viewing connection strengths in neural networks’, *Behav. Res. Methods Instrum. Comput.*, vol. 26, no. 2, pp. 215–218, 1994.

[87] E. Corchado, ‘Early vision and artificial neural networks’, Universidad de Salamanca, 2002.

[88] N. Qi, Z. Zhang, Y. Xiang, and P. de B. Harrington, ‘Locally linear embedding method for dimensionality reduction of tissue sections of endometrial carcinoma by near infrared spectroscopy’, *Anal. Chim. Acta*, vol. 724, pp. 12–19, Apr. 2012.

[89] G. Cirrincione, J. Hérault, and V. Randazzo, ‘The on-line curvilinear component analysis (onCCA) for real-time data reduction’, in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[90] T. Zhang, Y. Du, T. Huang, J. Yang, and X. Li, ‘Stochastic simulation of patterns using ISOMAP for dimensionality reduction of training images’, *Comput. Geosci.*, vol. 79, pp. 82–93, Jun. 2015.

[91] E. S. Howell, E. Merényi, and L. A. Lebofsky, ‘Classification of asteroid spectra using a neural network’, *J. Geophys. Res. Planets*, vol. 99, no. E5, pp. 10847–10865, May 1994.

[92] J. F. Bell, P. D. Owensby, B. R. Hawke, and M. J. Gaffey, ‘The 52-Color Asteroid Survey: Final Results and Interpretation’, presented at the Lunar and Planetary Science Conference, 1988, vol. 19.

[93] B. Zellner, D. J. Tholen, and E. F. Tedesco, ‘The eight-color asteroid survey: Results for 589 minor planets’, *Icarus*, vol. 61, no. 3, pp. 355–416, Mar. 1985.

[94] P. Krömer *et al.*, ‘Prediction of Dental Milling Time-Error by Flexible Neural Trees and Fuzzy Rules’, in *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, H. Yin, J. A. F. Costa, and G. Barreto, Eds. Springer Berlin Heidelberg, 2012, pp. 842–849.

[95] H. Quintián and E. Corchado, ‘Beta Hebbian Learning as a New Method for Exploratory Projection Pursuit’, *Int. J. Neural Syst.*, vol. Accepted, In press, 2017.

[96] M. Lichman, *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[97] M. Lichman, *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[98] H. Quintián and E. Corchado, ‘Beta Scale Invariant Map’, *Eng. Appl. Artif. Intell.*, vol. 59, pp. 218–235, 2017.

[99] B. Baruque and E. Corchado, ‘A weighted voting summarization of SOM ensembles’, *Data Min. Knowl. Discov.*, vol. 21, no. 3, pp. 398–426, 2010.

[100] H. Parvin, M. MirnabiBaboli, and H. Alinejad-Rokny, ‘Proposing a classifier ensemble framework based on classifier selection and decision tree’, *Eng. Appl. Artif. Intell.*, vol. 37, pp. 34–42, Jan. 2015.

Chapter 8. Annex

Table 8.1 Average testing CE/MQE/TE ± STD over the 14 benchmark datasets for Beta-SIM experiments.

Dataset	Quality Measure	SOM	ViSOM	SIM	MLHL-SIM	GNG	Beta-SIM
	CE	0.412 ± 0.0866	0.472 ± 0.1004	0.474 ± 0.0904	0.461 ± 0.0858	0.446 ± 0.0690	0.469 ± 0.0725
1	MQE	0.314 ± 0.0352	0.326 ± 0.0320	0.272 ± 0.0282	0.265 ± 0.0225	0.256 ± 0.0223	0.263 ± 0.0308
	TE	0.090 ± 0.0456	0.064 ± 0.0467	0.313 ± 0.0735	0.348 ± 0.0646	0.713 ± 0.0724	0.308 ± 0.0595
	CE	0.210 ± 0.1595	0.180 ± 0.1033	0.180 ± 0.1229	0.210 ± 0.1287	0.220 ± 0.1476	0.210 ± 0.1229
2	MQE	1.226 ± 0.0824	1.035 ± 0.1118	1.037 ± 0.1110	1.055 ± 0.1264	0.937 ± 0.0968	1.121 ± 0.1381
	TE	0.030 ± 0.0483	0.090 ± 0.1287	0.090 ± 0.0568	0.030 ± 0.0675	0.440 ± 0.1430	0.090 ± 0.1101
	CE	0.117 ± 0.0039	0.117 ± 0.0046	0.116 ± 0.0046	0.117 ± 0.0040	0.116 ± 0.0047	0.117 ± 0.0048
3	MQE	1.223 ± 0.0103	1.005 ± 0.0094	1.057 ± 0.0170	1.037 ± 0.016	0.993 ± 0.0131	1.178 ± 0.0701
	TE	0.023 ± 0.0123	0.026 ± 0.0065	0.053 ± 0.0148	0.055 ± 0.0297	0.309 ± 0.0425	0.136 ± 0.0466
	CE	0.047 ± 0.0632	0.047 ± 0.0549	0.060 ± 0.0734	0.067 ± 0.0544	0.047 ± 0.0549	0.046 ± 0.0820
4	MQE	0.222 ± 0.0388	0.157 ± 0.0209	0.177 ± 0.0310	0.181 ± 0.0331	0.149 ± 0.0143	0.166 ± 0.0250
	TE	0.200 ± 0.0770	0.120 ± 0.0322	0.260 ± 0.1313	0.320 ± 0.1209	0.587 ± 0.1565	0.280 ± 0.1363
	CE	0.045 ± 0.0353	0.045 ± 0.0365	0.045 ± 0.0450	0.079 ± 0.0614	0.051 ± 0.0513	0.044 ± 0.0511
5	MQE	0.795 ± 0.0836	0.720 ± 0.0743	0.710 ± 0.0822	0.700 ± 0.0807	0.669 ± 0.0422	0.767 ± 0.0909
	TE	0.090 ± 0.0390	0.062 ± 0.0720	0.190 ± 0.0865	0.225 ± 0.0950	0.668 ± 0.1129	0.400 ± 0.1124
	CE	0.538 ± 0.0448	0.541 ± 0.0571	0.546 ± 0.0416	0.538 ± 0.0536	0.560 ± 0.0432	0.550 ± 0.0417
6	MQE	0.870 ± 0.0342	0.699 ± 0.0338	0.742 ± 0.0378	0.717 ± 0.0342	0.493 ± 0.0242	0.654 ± 0.0360
	TE	0.040 ± 0.0255	0.028 ± 0.0088	0.081 ± 0.0286	0.073 ± 0.0505	0.529 ± 0.0469	0.125 ± 0.0364
	CE	0.210 ± 0.0499	0.231 ± 0.0413	0.207 ± 0.0287	0.211 ± 0.0413	0.192 ± 0.0491	0.177 ± 0.0268
7	MQE	1.206 ± 0.0243	1.045 ± 0.0305	1.083 ± 0.0257	1.036 ± 0.0252	0.851 ± 0.0097	1.086 ± 0.0286
	TE	0.046 ± 0.0142	0.063 ± 0.0177	0.136 ± 0.0266	0.120 ± 0.0243	0.667 ± 0.0217	0.443 ± 0.0393
	CE	0.031 ± 0.0406	0.081 ± 0.0403	0.061 ± 0.0677	0.104 ± 0.0722	0.131 ± 0.0514	0.050 ± 0.0318
8	MQE	1.818 ± 0.0638	1.834 ± 0.0683	1.783 ± 0.0585	1.800 ± 0.0677	2.261 ± 0.0606	1.679 ± 0.0452
	TE	0.053 ± 0.0243	0.031 ± 0.0337	0.132 ± 0.0708	0.162 ± 0.0721	0.070 ± 0.0509	0.167 ± 0.0818
	CE	0.143 ± 0.0262	0.195 ± 0.0237	0.165 ± 0.0249	0.147 ± 0.0246	0.151 ± 0.0187	0.098 ± 0.0162
9	MQE	0.553 ± 0.0197	0.532 ± 0.0298	0.487 ± 0.0227	0.464 ± 0.0265	0.511 ± 0.0183	0.381 ± 0.0206
	TE	0.071 ± 0.0195	0.040 ± 0.0246	0.162 ± 0.0350	0.161 ± 0.0364	0.345 ± 0.0362	0.158 ± 0.0404
	CE	0.158 ± 0.0106	0.157 ± 0.0100	0.155 ± 0.0216	0.148 ± 0.0204	0.157 ± 0.0081	0.132 ± 0.0125
10	MQE	0.644 ± 0.0084	0.655 ± 0.0103	0.597 ± 0.0197	0.592 ± 0.0083	0.681 ± 0.0114	0.489 ± 0.0075
	TE	0.120 ± 0.0184	0.063 ± 0.0132	0.197 ± 0.0460	0.243 ± 0.0606	0.380 ± 0.0252	0.392 ± 0.0600
	CE	0.500 ± 0.0359	0.489 ± 0.0472	0.462 ± 0.0538	0.468 ± 0.0394	0.478 ± 0.0427	0.451 ± 0.0530
11	MQE	0.321 ± 0.0274	0.292 ± 0.0170	0.279 ± 0.0207	0.276 ± 0.0190	0.286 ± 0.0203	0.267 ± 0.0217
	TE	0.059 ± 0.0174	0.051 ± 0.0193	0.297 ± 0.0383	0.333 ± 0.0598	0.520 ± 0.0527	0.330 ± 0.0340
	CE	0.106 ± 0.0123	0.109 ± 0.0147	0.098 ± 0.0150	0.096 ± 0.0101	0.093 ± 0.0135	0.081 ± 0.0114
12	MQE	0.883 ± 0.0102	0.897 ± 0.0115	0.753 ± 0.0129	0.741 ± 0.0138	0.901 ± 0.0123	0.668 ± 0.0152
	TE	0.055 ± 0.0090	0.042 ± 0.0037	0.088 ± 0.0136	0.082 ± 0.0135	0.290 ± 0.0268	0.105 ± 0.0176
	CE	0.105 ± 0.0098	0.120 ± 0.0146	0.092 ± 0.0103	0.091 ± 0.0115	0.087 ± 0.0178	0.064 ± 0.0128
13	MQE	2.703 ± 0.0228	2.811 ± 0.0173	2.450 ± 0.0203	2.454 ± 0.0213	2.701 ± 0.0229	2.247 ± 0.0289
	TE	0.038 ± 0.0092	0.037 ± 0.0082	0.063 ± 0.0123	0.058 ± 0.0121	0.293 ± 0.0273	0.091 ± 0.0095
	CE	0.545 ± 0.0169	0.516 ± 0.0141	0.503 ± 0.0101	0.497 ± 0.0139	0.363 ± 0.0175	0.362 ± 0.0122
14	MQE	0.605 ± 0.0044	0.568 ± 0.0045	0.527 ± 0.0028	0.521 ± 0.0044	0.448 ± 0.0031	0.432 ± 0.0043
	TE	0.074 ± 0.0109	0.075 ± 0.0109	0.193 ± 0.0129	0.204 ± 0.0196	0.483 ± 0.0154	0.247 ± 0.0191

Table 8.2 Algorithms parameters for each benchmark dataset for Beta-SIM and WeVoS-Beta-SIM experiments.

Dataset	Parameters	SOM	ViSOM	SIM	MLHL-SIM	GNG	Beta-SIM
	Size	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	3,000	3,000	3,000	3,000	3,000	3,000
	Learning rate	0.1	0.05	0.05	0.05	-	0.05
	Neighborhood	12	10	10	10	-	6
1	***p***	-	-	-	0.5	-	-
	$\alpha;\beta$	-	-	-	-	-	$\alpha=3;\beta=4$
	λ_{ViSOM}	-	0,25	-	-	-	-
	$\lambda_{GNG}; \alpha_{GNG};\beta_{GNG}; a_{max}; Wl; Wn$	-	-	-	-	200; 0.5; 0.9; 5; 0.01; 0.001	-
2	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	5,000	5,000	5,000	5,000	10,000	5,000
	Learning rate	0.1	0.05	0.1	0.1	-	0.1
	Neighborhood	12	10	10	10	-	6
	p	-	-	-	0.9	-	-
	$\alpha;\beta$	-	-	-	-	-	$\alpha=3;\beta=3.5$
	λ_{ViSOM}	-	0,25	-	-	-	-
	$\lambda_{GNG}; \alpha_{GNG};\beta_{GNG}; a_{max}; Wl; Wn$	-	-	-	-	200; 0.5; 0.9; 5; 0.01; 0.001	-
3	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	30,000	30,000	30,000	30,000	30,000	50,000
	Learning rate	0.05	0.05	0.05	0.05	-	0.05
	Neighborhood	12	10	10	10	-	6
	p	-	-	-	0.5	-	-
	$\alpha;\beta$	-	-	-	-	-	$\alpha=3;\beta=3.5$
	λ_{ViSOM}	-	0,25	-	-	-	-
	$\lambda_{GNG}; \alpha_{GNG};\beta_{GNG}; a_{max}; Wl; Wn$	-	-	-	-	100; 0.5; 0.9; 5; 0.01; 0.001	-
4	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	5,000	5,000	5,000	5,000	5,000	5,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	10		5	10	-	12
	p	-	-	-	0.9	-	-
	$\alpha;\beta$	-	-	-	-	-	$\alpha=3;\beta=3$
	λ_{ViSOM}	-	0,1	-	-	-	-
	$\lambda_{GNG}; \alpha_{GNG};\beta_{GNG}; a_{max}; Wl; Wn$	-	-	-	-	100; 0.5; 0.9; 5; 0.1; 0.001	-
5	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	5,000	5,000	5,000	5,000	5,000	5,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	12	10	-	12
	p	-	-	-	0.9	-	-
	$\alpha;\beta$	-	-	-	-	-	$\alpha=4;\beta=4$
	λ_{ViSOM}	-	0.1	-	-	-	-
	$\lambda_{GNG}; \alpha_{GNG};\beta_{GNG}; a_{max}; Wl; Wn$	-	-	-	-	50; 0.5; 0.9; 5 0.1; 0.0001	-
6	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,15]
	Iterations	10,000	10,000	10,000	10,000	10,000	50,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	12	10	-	5
	p	-	-	-	1.2	-	-

	α;β	-	-	-	-	-	*α=4;β=4*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.2	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	50; 0.5; 0.9; 5 0.1; 0.0001	-
7	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	10,000	5,000	10,000	10,000	10,000	10,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	10	10	12	10	-	6
	p	-		-	0.9	-	-
	α;β	-		-	-	-	*α=5;β=3*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.2	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	100; 0.5; 0.9; 5;0.01; 0.0001	-
8	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	5,000	5,000	5,000	5,000	5,000	5,000
	Learning rate	0.1	0.1	0.05	0.05	-	0.1
	Neighborhood	10	10	10	10	-	6
	p	-	-	-	0.9	-	-
	α;β	-	-	-	-	-	*α=4;β=3*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.3	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	50; 0.5; 0.9; 5;0.01; 0.0001	-
9	**Size**	[20,15]	[20,15]	[20,15]	[20,15]	-	[20,15]
	Iterations	10,000	5,000	10,000	10,000	10,000	10,000
	Learning rate	0.1	0.1	0.05	0.1	-	0.1
	Neighborhood	10	10	10	10	-	6
	p	-		-	0.9	-	-
	α;β	-		-	-	-	*α=4;β=3*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.25	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	100; 0.5; 0.9; 5;0.01; 0.0001	-
10	**Size**	[20,15]	[20,15]	[20,15]	[20,15]	-	[20,15]
	Iterations	30,000	30,000	30,000	30,000	30,000	30,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	10	10	10	10	-	6
	p	-	-	-	1.1	-	-
	α;β	-	-	-	-	-	*α=4;β=3*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.25	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	100; 0.5; 0.9; 5;0.01; 0.0001	-
11	**Size**	[15,10]	[15,10]	[15,10]	[15,10]	-	[15,10]
	Iterations	10,000	10,000	10,000	10,000	10,000	10,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	10	10	-	6
	p	-	-	-	0.5	-	-
	α;β	-	-	-	-	-	*α=4;β=3*
	$\boldsymbol{\lambda_{ViSOM}}$	-	0.1	-	-	-	-
	$\boldsymbol{\lambda_{GNG};\ \alpha_{GNG};\beta_{GNG};\ a_{max};\ Wl;\ Wn}$	-	-	-	-	50; 0.5; 0.9; 5;0.01;0.0001	-
12	**Size**	[20,15]	[20,15]	[20,15]	[20,15]	-	[20,15]
	Iterations	50,000	50,000	50,000	50,000	50,000	50,000
	Learning rate	0. 1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	8	8	-	6
	p	-	-	-	0.9	-	-

	α;β	-	-	-	-	-	*α=4;β=3*
	λ_{ViSOM}	-	0.1	-	-	-	-
	λ_{GNG}; α_{GNG};β_{GNG}; a_{max}; Wl; Wn	-	-	-	-	100; 0.5; 0.9; 5;0.001;0.000	-
13	**Size**	[20,10]	[20,10]	[20,10]	[20,10]	-	[20,10]
	Iterations	40,000	40,000	40,000	40,000	40,000	40,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	12	12	-	6
	p	-	-	-	0.9	-	-
	α;β	-	-	-	-	-	*α=4;β=3*
	λ_{ViSOM}						
	λ_{GNG}; α_{GNG};β_{GNG}; a_{max}; Wl; Wn	-	-	-	-	100; 0.5; 0.9; 5;0.001;0.000	-
14	**Size**	[25,20]	[25,20]	[25,20]	[25,20]	-	[25,20]
	Iterations	80,000	80,000	80,000	80,000	80,000	80,000
	Learning rate	0.1	0.1	0.1	0.1	-	0.1
	Neighborhood	12	10	12	12	-	6
	p	-	-	-	0.5	-	-
	α;β	-	-	-	-	-	*α=4;β=3*
	λ_{ViSOM}	-	0.1	-	-	-	-
	λ_{GNG}; α_{GNG};β_{GNG}; a_{max}; Wl; Wn	-	-	-	-	100; 0.5; 0.9; 5;0.005;0.000	-

Table 8.3 Summary of the results in terms of the three well-known quality measures for each benchmark dataset for Beta-SIM experiments.

Dataset	Quality Measure	SOM	ViSOM	SIM	MLHL-SIM	GNG
1	CE	-6%	0%	1%	-1%	-2%
	MQE	**16%**	**19%**	3%	1%	-3%
	TE	-22%	-24%	1%	4%	**41%**
2	CE	0%	-3%	-3%	0%	**1%**
	MQE	9%	-8%	-8%	-6%	-20%
	TE	-6%	0%	0%	-6%	**35%**
3	CE	0%	0%	0%	0%	0%
	MQE	**4%**	**-17%**	**-11%**	**-14%**	**-19%**
	TE	-11%	-11%	-8%	-8%	**17%**
4	CE	0%	0%	1%	2%	0%
	MQE	**25%**	-6%	6%	8%	-11%
	TE	-8%	-16%	-2%	4%	**31%**
5	CE	0%	0%	0%	4%	1%
	MQE	4%	-7%	-8%	-10%	-15%
	TE	-31%	-34%	-21%	-18%	**27%**
6	CE	-1%	-1%	0%	-1%	1%
	MQE	**25%**	**6%**	12%	9%	-33%
	TE	-9%	-10%	-4%	-5%	**40%**
7	CE	3%	**5%**	3%	3%	2%
	MQE	**10%**	**-4%**	0%	**-5%**	**-28%**
	TE	-40%	-38%	-31%	-32%	**22%**
8	CE	-2%	3%	1%	5%	**8%**
	MQE	**8%**	**8%**	**6%**	**7%**	**26%**
	TE	-11%	-14%	-4%	-1%	-10%
9	CE	**5%**	**10%**	**7%**	**5%**	**5%**
	MQE	**31%**	**28%**	**22%**	**18%**	**25%**
	TE	-9%	-12%	0%	0%	**19%**
10	CE	**3%**	**3%**	**2%**	**2%**	**3%**
	MQE	**24%**	**25%**	**18%**	**17%**	**28%**

	TE	-27%	-33%	-20%	-15%	-1%
11	CE	5%	4%	1%	2%	3%
	MQE	**17%**	9%	4%	3%	7%
	TE	-27%	-28%	-3%	0%	**19%**
12	CE	**3%**	**3%**	**2%**	2%	1%
	MQE	**24%**	**26%**	**11%**	**10%**	**26%**
	TE	-5%	-6%	-2%	-2%	**19%**
13	CE	**4%**	**6%**	**3%**	**3%**	**2%**
	MQE	**17%**	**20%**	**8%**	**8%**	**17%**
	TE	-5%	-5%	-3%	-3%	**20%**
14	CE	**18%**	**15%**	**14%**	**14%**	0%
	MQE	**29%**	**24%**	**18%**	**17%**	**4%**
	TE	-17%	-17%	-5%	-4%	**24%**

Red color: Beta-SIM is significantly better (significance level of 0.05) than the other model
Blue color: Beta-SIM is significantly worse (significance level of 0.05) than the other model

Table 8.4 p-values for CE (WeVoS-Beta-SIM against all algorithms)

Dataset	Nº samples	SIM	Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
	69	0.8958	0.9516	0.3850	0.9991	0.5790	0.7094	0.9999
	138	0.9999	0.9983	0.8807	1.0000	1.0000	0.9962	1.0000
1	207	0.9986	0.9968	0.9999	0.9996	0.9999	1.0000	0.9996
	276	0.9983	0.9746	0.9976	0.9991	0.9980	0.9998	0.9998
	345	0.9989	0.9979	0.9873	0.8236	0.9861	0.9998	1.0000
	20	1.0000	0.9999	0.9938	0.7762	1.0000	0.9938	1.0000
	40	1.0000	0.9990	0.9711	0.9998	1.0000	1.0000	0.9997
2	60	0.9992	0.9754	0.7170	0.8110	0.9961	0.9933	0.9992
	80	0.4724	0.9974	1.0000	0.9959	0.9790	0.9970	0.8462
	100	0.9996	0.9998	0.9705	1.0000	0.9889	0.9970	1.0000
	9043	0.0446$^{\Diamond}$	0.4195	0.5673	0.5171	0.7138	0.6666	0.0960
	18086	0.9885	0.9993	0.9828	0.8168	0.9530	0.5913	0.9752
3	27129	0.9951	0.9998	0.9559	0.9670	0.9559	1.0000	0.9263
	36172	0.9964	0.9557	0.9987	0.9972	0.9999	0.9495	0.9954
	45211	1.0000	0.9907	0.9993	0.9827	0.9624	0.9954	0.9290
	30	0.9952	0.9162	1.0000	0.9996	0.9162	0.9996	1.0000
	60	0.9974	0.9997	1.0000	1.0000	1.0000	1.0000	1.0000
4	90	1.0000	1.0000	0.9991	0.9871	1.0000	1.0000	0.9991
	120	0.9999	0.9934	0.9505	0.9999	0.9998	0.9960	0.9438
	150	1.0000	1.0000	0.9942	1.0000	1.0000	0.9942	1.0000
	36	1.0000	1.0000	1.0000	0.9905	1.0000	1.0000	0.9991
	72	0.9897	0.9992	1.0000	0.9869	0.9992	0.9869	1.0000
5	108	0.9994	0.9962	1.0000	0.4038	0.9996	0.9120	1.0000
	144	1.0000	1.0000	0.9937	0.9996	1.0000	1.0000	0.9997
	178	0.8318	0.9895	0.9895	0.9895	1.0000	0.4805	0.9895
	295	1.0000	1.0000	0.5788	0.8551	1.0000	0.9915	0.9453
	590	0.9998	1.0000	1.0000	0.9999	0.9997	1.0000	0.9997
6	885	0.9995	0.9976	1.0000	0.9990	0.9701	0.9999	0.9977
	1180	0.9430	0.9993	0.9999	1.0000	0.9855	1.0000	0.9997
	1473	0.9897	0.9838	0.8343	1.0000	0.7163	0.9998	1.0000
	346	0.9946	0.9999	1.0000	1.0000	0.9546	0.9997	0.4479
	692	1.0000	1.0000	0.9988	0.9995	0.9828	1.0000	0.4248
7	1038	0.7513	1.0000	0.9896	0.9993	1.0000	1.0000	0.8534
	1384	0.9932	1.0000	0.9154	0.9999	0.8827	0.9997	0.6597
	1728	0.9879	0.9997	1.0000	1.0000	0.9973	0.7986	0.0737
8	72	0.5625	0.5625	0.5625	0.3524	0.1571	0.5625	0.3524
	144	1.0000	0.9997	0.9928	1.0000	0.9892	1.0000	1.0000

	216	0.9345	0.9954	0.9854	0.7354	0.9995	0.9999	0.6554
	288	0.9923	0.8485	0.9999	0.8346	1.0000	0.9730	0.9989
	358	0.1169	1.0000	1.0000	0.9646	0.6784	0.9994	1.0000
	462	0.0000*	0.9978	0.1091	0.0012*	0.0056*	0.2371	0.0662
	924	0.0000*	1.0000	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
9	1386	0.0000*	0.9957	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	1848	0.0000*	0.9613	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	2310	0.0000*	0.9228	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	1287	0.9713	0.9860	0.9864	0.8818	0.9265	0.8279	0.0000*
	2574	0.4136	0.8679	0.0014*	0.0927	0.4720	0.1590	0.0000*
10	3861	0.8162	1.0000	0.4380	0.2670	0.9873	0.8016	0.0000*
	5148	0.0188*	1.0000	0.0003*	0.0015*	0.0348*	0.1331	0.0000*
	6435	0.0000*	0.9967	0.0000*	0.0000*	0.0005*	0.0001*	0.0000*
	297	0.7997	0.9865	0.5333	0.4788	0.9089	0.8765	0.8101
	594	0.9817	0.9976	0.9663	0.6593	0.9159	1.0000	1.0000
11	891	0.4254	0.7712	0.5029	0.0757	0.4165	0.9821	0.8832
	1188	0.9159	0.9991	0.4903	0.0364*	0.9675	0.9953	1.0000
	1484	0.9475	0.9993	0.1509	0.1201	0.9947	0.8352	0.9996
	2199	0.9973	0.9997	0.2056	0.0144*	0.9881	0.2753	0.7632
	4398	0.9982	0.9883	0.1169	0.0000*	0.9691	0.1542	0.8215
12	6597	0.2249	1.0000	0.0010*	0.0000*	0.7232	0.0692	0.3049
	8796	0.2290	0.7177	0.0001*	0.0000*	0.7059	0.0001*	0.2002
	10992	0.7618	0.9983	0.0001*	0.0000*	0.4568	0.0000*	0.0926
	1124	0.7862	0.9929	0.0434*	0.0000*	0.9645	0.1407	1.0000
	2248	0.0487*	1.0000	0.0010*	0.0000*	0.6557	0.0000*	0.9196
13	3372	0.2678	1.0000	0.0052*	0.0000*	0.6724	0.0003*	0.8906
	4496	0.4075	1.0000	0.0001*	0.0000*	0.4241	0.0000*	0.5245
	5620	0.0379*	1.0000	0.0000*	0.0000*	0.6031	0.0000*	0.2933
	4000	0.0000*	0.8880	0.0000*	0.0000*	0.0000*	0.0000*	1.0000
	8000	0.0000*	0.4548	0.0000*	0.0000*	0.0000*	0.0000*	0.9745
14	12000	0.0000*	0.7614	0.0000*	0.0000*	0.0000*	0.0000*	0.8839
	16000	0.0000*	0.2165	0.0000*	0.0000*	0.0000*	0.0000*	0.8033
	20000	0.0000*	0.4712	0.0000*	0.0000*	0.0000*	0.0000*	0.9900

* WeVoS-Beta-SIM is significantly better (significance level of 0.05)

◊ WeVoS-Beta-SIM is significantly worse (significance level of 0.05)

Table 8.5 p-values for MQE (WeVoS-Beta-SIM against all algorithms)

Dataset	Nº samples	SIM	Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
1	69	0.9999	0.9995	0.9493	0.1308	1.0000	0.8491	0.9978
	138	1.0000	1.0000	0.7775	0.0061*	1.0000	0.3215	1.0000
	207	1.0000	1.0000	0.0935	0.0000*	0.9871	0.0108*	1.0000
	276	0.9384	1.0000	0.0043*	0.0000*	0.9646	0.0004*	1.0000
	345	0.9972	1.0000	0.2095	0.0004*	0.9997	0.0409*	1.0000
2	20	1.0000	1.0000	0.9206	0.8572	1.0000	1.0000	1.0000
	40	1.0000	1.0000	0.0018*	0.0000*	1.0000	1.0000	0.9872
	60	0.9768	0.7329	0.0000*	0.0000*	0.9911	0.9880	1.0000
	80	0.9992	0.9962	0.0000*	0.0000*	0.9954	1.0000	0.6938
	100	1.0000	1.0000	0.0000*	0.0000*	0.9984	0.9978	0.6521
3	9043	0.4056	0.6412	0.0000*	0.0000*	0.0639	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$
	18086	1.0000	0.0325*	0.0000*	0.0000*	0.8685	0.0058$^{\Diamond}$	0.0030$^{\Diamond}$
	27129	0.1249	0.7730	0.0000*	0.0000*	0.0421$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$
	36172	1.0000	0.0363*	0.0000*	0.0000*	0.9498	0.0162$^{\Diamond}$	0.0065$^{\Diamond}$
	45211	0.0027$^{\Diamond}$	0.9836	0.0000*	0.0000*	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$	0.0000$^{\Diamond}$
4	30	0.9994	1.0000	0.4264	0.0054*	1.0000	0.9837	1.0000
	60	0.9821	1.0000	0.3470	0.0019*	0.9989	1.0000	1.0000
	90	1.0000	0.9985	0.0114*	0.0000*	0.9999	0.9988	1.0000
	120	0.9118	1.0000	0.0000*	0.0000*	0.4734	0.9940	1.0000
	150	0.9995	0.9993	0.0234*	0.0000*	0.9962	0.9975	0.9993
5	36	1.0000	1.0000	0.9999	0.9471	1.0000	1.0000	1.0000
	72	1.0000	1.0000	0.9914	0.4144	0.9999	1.0000	0.9999
	108	0.9997	0.9995	0.9713	0.0387*	0.9025	0.9626	0.6379
	144	0.1953	0.9969	0.0727	0.0000*	0.6331	0.6876	0.0012$^{\Diamond}$
	178	0.9605	1.0000	0.4684	0.0023*	0.9836	0.9973	0.1763
6	295	0.0018*	1.0000	0.0000*	0.0000*	0.1054	0.7699	0.0000$^{\Diamond}$
	590	0.0000*	0.8916	0.0000*	0.0000*	0.0026*	0.0285*	0.0000$^{\Diamond}$
	885	0.0010*	0.9922	0.0000*	0.0000*	0.0504	0.2314	0.0000$^{\Diamond}$
	1180	0.0000*	0.9721	0.0000*	0.0000*	0.0000*	0.0069*	0.0000$^{\Diamond}$
	1473	0.0197*	0.0475*	0.0000*	0.0000*	0.1789	0.3657	0.0000$^{\Diamond}$
7	346	0.0066*	0.9989	0.0000*	0.0000*	0.7893	0.1484	0.0000$^{\Diamond}$
	692	0.0000*	0.0074*	0.0000*	0.0000*	0.0141*	0.0030*	0.0000$^{\Diamond}$
	1038	0.0000*	0.1972	0.0000*	0.0000*	0.3101	0.1014	0.0000$^{\Diamond}$
	1384	0.0000*	0.0219*	0.0000*	0.0000*	0.0713	0.0224*	0.0000$^{\Diamond}$
	1728	0.0000*	0.0005*	0.0000*	0.0000*	0.0002*	0.0000*	0.0000$^{\Diamond}$

	72	0.3207	0.4928	0.2220	0.0052*	0.3928	0.3437	0.7936
	144	0.0007*	0.2966	0.0001*	0.0000*	0.0012*	0.0008*	0.4246
8	216	0.0000*	0.9768	0.0000*	0.0000*	0.0000*	0.0001*	0.8949
	288	0.0000*	1.0000	0.0000*	0.0000*	0.0000*	0.0000*	0.9927
	358	0.0000*	1.0000	0.0000*	0.0000*	0.0000*	0.0000*	0.9865
	462	0.0000*	0.9997	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	924	0.0000*	0.9759	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
9	1386	0.0000*	0.9898	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	1848	0.0000*	0.9420	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	2310	0.0000*	0.9067	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	1287	0.0000*	0.9907	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	2574	0.0000*	0.9386	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
10	3861	0.0000*	0.8347	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	5148	0.0000*	0.0458*	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	6435	0.0000*	0.0030*	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	297	0.6356	1.0000	0.0006*	0.0000*	0.6983	0.2171	0.4205
	594	0.0305*	1.0000	0.0000*	0.0000*	0.2113	0.0009*	0.0092*
11	891	0.0111*	0.9991	0.0000*	0.0000*	0.0107*	0.0000*	0.0005*
	1188	0.0060*	0.9982	0.0000*	0.0000*	0.0404*	0.0000*	0.0008*
	1484	0.0000*	0.9394	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	2199	0.2761	0.9992	0.0000*	0.0000*	0.7677	0.0000*	0.0000*
	4398	0.2332	0.9844	0.0000*	0.0000*	0.6176	0.0000*	0.0000*
12	6597	0.4586	0.9998	0.0000*	0.0000*	0.7756	0.0000*	0.0000*
	8796	0.3557	0.9939	0.0000*	0.0000*	0.7204	0.0000*	0.0000*
	10992	0.3036	0.9989	0.0000*	0.0000*	0.5867	0.0000*	0.0000*
	1124	0.3653	0.9999	0.0000*	0.0000*	0.6107	0.0000*	0.0000*
	2248	0.0000*	0.4692	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
13	3372	0.0000*	0.4019	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	4496	0.0000*	0.9553	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	5620	0.0000*	0.5695	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	4000	0.0000*	0.2528	0.0000*	0.0000*	0.0000*	0.0000*	0.0011*
	8000	0.0000*	0.5156	0.0000*	0.0000*	0.0000*	0.0000*	0.0114*
14	12000	0.0000*	0.0120*	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	16000	0.0000*	0.0443*	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
	20000	0.0000*	0.4932	0.0000*	0.0000*	0.0000*	0.0000*	0.0005*

* WeVoS-Beta-SIM is significantly better (significance level of 0.05)

◊ WeVoS-Beta-SIM is significantly worse (significance level of 0.05)

Table 8.6 p-values for TE (WeVoS-Beta-SIM against all algorithms)

Dataset	Nº samples	SIM	Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
	69	0.1601	0.0993	0.0013$^{\lozenge}$	0.0040$^{\lozenge}$	0.2797	0.0000$^{\lozenge}$	0.0000*
	138	0.9981	0.7576	0.0000$^{\lozenge}$	0.0016$^{\lozenge}$	0.9999	0.0000$^{\lozenge}$	0.0000*
1	207	0.6231	0.9940	0.0000$^{\lozenge}$	0.0143$^{\lozenge}$	0.9553	0.0000$^{\lozenge}$	0.0000*
	276	0.9989	0.9926	0.0000$^{\lozenge}$	0.0001$^{\lozenge}$	0.9990	0.0000$^{\lozenge}$	0.0000*
	345	0.3572	1.0000	0.0000$^{\lozenge}$	0.0000$^{\lozenge}$	0.7626	0.0000$^{\lozenge}$	0.0000*
	20	0.9998	1.0000	0.8518	0.8518	1.0000	0.5728	0.0017*
	40	0.9617	0.9985	0.6080	0.8920	0.1990	0.7492	0.0000*
2	60	0.9985	1.0000	0.2883	1.0000	0.8267	0.9989	0.0000*
	80	0.5303	1.0000	0.0392$^{\lozenge}$	0.6881	0.9999	0.2693	0.0080*
	100	0.6678	0.7758	0.0168$^{\lozenge}$	0.0168$^{\lozenge}$	0.1298	0.4541	0.0000*
	9043	0.0022$^{\lozenge}$	0.9957	0.0000$^{\lozenge}$	0.1702	0.0047$^{\lozenge}$	0.0000$^{\lozenge}$	0.0000*
	18086	0.0373$^{\lozenge}$	0.9351	0.0001$^{\lozenge}$	0.9390	0.4378	0.0005$^{\lozenge}$	0.0000*
3	27129	0.0006$^{\lozenge}$	0.9370	0.0000$^{\lozenge}$	0.9267	0.0039$^{\lozenge}$	0.0000$^{\lozenge}$	0.0000*
	36172	0.3571	0.7424	0.0121$^{\lozenge}$	0.0074*	0.4952	0.0229$^{\lozenge}$	0.0000*
	45211	0.0116$^{\lozenge}$	1.0000	0.0000$^{\lozenge}$	1.0000	0.0024$^{\lozenge}$	0.0000$^{\lozenge}$	0.0000*
	30	0.9603	0.9866	1.0000	0.5595	0.8594	0.9969	0.0539
	60	0.9879	0.0284*	0.7728	0.9999	0.5427	0.2593	0.0000*
4	90	0.7122	0.1744	0.9989	0.9689	0.9955	0.5864	0.0000*
	120	0.9986	0.9999	0.3425	0.7344	0.9993	0.1675	0.0000*
	150	0.9931	0.9978	0.0879	0.5923	0.1198	0.0007$^{\lozenge}$	0.0000*
	36	0.9876	0.1204	0.9329	1.0000	0.8742	1.0000	0.0000*
	72	0.3931	0.9694	0.0111◊	0.0655	0.8914	0.0004$^{\lozenge}$	0.0000*
5	108	0.9978	0.0038*	0.2146	0.4834	0.9999	0.0062$^{\lozenge}$	0.0000*
	144	0.9992	0.8896	0.0049$^{\lozenge}$	0.4388	0.9630	0.0001$^{\lozenge}$	0.0000*
	178	0.3531	0.0028*	0.0796	0.7623	0.8855	0.0282$^{\lozenge}$	0.0000*
	295	0.1927	0.3963	0.0005$^{\lozenge}$	0.9257	0.2522	0.0005$^{\lozenge}$	0.0000*
	590	0.1917	0.7446	0.0177$^{\lozenge}$	0.4426	0.1582	0.0006$^{\lozenge}$	0.0000*
6	885	1.0000	0.7656	0.0446$^{\lozenge}$	0.0563	1.0000	0.0445$^{\lozenge}$	0.0000*
	1180	0.8835	0.9722	0.0078$^{\lozenge}$	0.9782	1.0000	0.0267$^{\lozenge}$	0.0000*
	1473	1.0000	0.0056*	0.3620	0.0001*	1.0000	0.2847	0.0000*
	346	0.0804	0.1081	0.0000$^{\lozenge}$	0.0080*	0.0288$^{\lozenge}$	0.0002$^{\lozenge}$	0.0000*
	692	0.0771	0.0000*	0.0000$^{\lozenge}$	0.9967	0.1065	0.0000$^{\lozenge}$	0.0000*
7	1038	0.0236$^{\lozenge}$	0.0000*	0.0000$^{\lozenge}$	0.8051	0.0471$^{\lozenge}$	0.0000$^{\lozenge}$	0.0000*
	1384	0.9576	0.0000*	0.0000$^{\lozenge}$	0.6972	0.0562	0.0000$^{\lozenge}$	0.0000*
	1728	0.0311$^{\lozenge}$	0.0000*	0.0000$^{\lozenge}$	0.7735	0.0043$^{\lozenge}$	0.0000$^{\lozenge}$	0.0000*

8	72	0.9396	0.9883	0.9761	0.9826	0.9954	0.9963	0.0000*
	144	0.4076	1.0000	0.0248◊	0.2021	0.3858	0.0006◊	0.0000*
	216	0.1169	0.7906	0.6833	0.9988	0.9975	0.2318	0.0000*
	288	0.8899	0.9554	0.0000◊	1.0000	0.9739	0.0000◊	0.0000*
	358	0.9985	0.9998	0.0025◊	0.1206	0.9817	0.0001◊	0.0000*
9	462	0.0579	0.7664	0.0001◊	0.0029◊	0.0842	0.0000◊	0.5963
	924	0.0000◊	0.9797	0.0000◊	0.0000◊	0.0091◊	0.0000◊	0.2574
	1386	0.0000◊	1.0000	0.0000◊	0.0000◊	0.0028◊	0.0000◊	0.9982
	1848	0.0000◊	0.8863	0.0000◊	0.0000◊	0.0000◊	0.0000◊	0.0151*
	2310	0.0000◊	0.6161	0.0000◊	0.0000◊	0.0000◊	0.0000◊	0.0122*
10	1287	1,0000	0,0459*	0,0000◊	0,0000◊	1,0000	0,0000◊	0,0000*
	2574	0,6093	0,2119	0,0000◊	0,0000◊	0,7702	0,0000◊	0,0000*
	3861	0,0426◊	0,0461*	0,0000◊	0,0000◊	0,0555	0,0000◊	0,0000*
	5148	1,0000	0,0198*	0,0000◊	0,0000◊	0,1377	0,0000◊	0,0000*
	6435	1,0000	0,0276*	0,0000◊	0,0000◊	0,5172	0,0000◊	0,0000*
11	297	0,9867	0,1683	0,0000◊	0,4861	1,0000	0,0000◊	0,0000*
	594	0,8594	0,0126*	0,0000◊	0,5720	0,2472	0,0000◊	0,0000*
	891	0,0860	0,0204*	0,0000◊	0,0162◊	0,6542	0,0000◊	0,0000*
	1188	0,6577	0,0003*	0,0000◊	0,0121◊	0,0007*	0,0000◊	0,0000*
	1484	0,0000*	0,0013*	0,0000◊	0,0000◊	0,0002*	0,0000◊	0,0000*
12	2199	0,9997	0,1034	0,0489◊	0,1372	1,0000	0,0146◊	0,0000*
	4398	0,9987	0,0002*	0,0084◊	0,9946	0,9994	0,0004◊	0,0000*
	6597	0,9898	0,0439*	0,7897	0,0272*	0,5190	0,5132	0,0000*
	8796	0,9971	0,2112	0,1674	0,4805	1,0000	0,0506	0,0000*
	10992	0,9967	0,0011*	0,2249	0,0148*	0,4110	0,0494◊	0,0000*
13	1124	0,9922	0,1021	0,7303	0,0000*	0,9997	0,1430	0,0000*
	2248	0,9996	0,0470*	0,1349	0,0000*	0,9522	0,1879	0,0000*
	3372	1,0000	0,0355*	0,5704	0,0000*	1,0000	0,0682	0,0000*
	4496	1,0000	0,0000*	0,5283	0,0000*	0,9963	0,0114◊	0,0000*
	5620	1,0000	0,0018*	0,4892	0,0000*	1,0000	0,1211	0,0000*
14	4000	0,0630	0,7376	0,0000◊	0,0003◊	0,0049◊	0,0000◊	0,0000*
	8000	1,0000	0,3346	0,0000◊	0,6199	0,8500	0,0000◊	0,0000*
	12000	0,9996	0,0008*	0,0000◊	0,0001◊	1,0000	0,0000◊	0,0000*
	16000	0,8154	0,0000*	0,0000◊	0,0976	0,8015	0,0000◊	0,0000*
	20000	1,0000	0,0003*	0,0000◊	0,0404◊	1,0000	0,0000◊	0,0000*

* WeVoS-Beta-SIM is significantly better (significance level of 0.05)

◊ WeVoS-Beta-SIM is significantly worse (significance level of 0.05)

Table 8.7 Average testing CE ± STD over the 14 benchmark datasets

Dataset	Nº samples	SIM	Beta-SIM	WeVoS-Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
	69	0.491±0.2007	0.476±0.1791	0.383±0.2025	0.563±0.2089	0.430±0.1539	0.538±0.2178	0.521±0.1393	0.349±0.1735
	138	0.442±0.0860	0.457±0.1524	0.419±0.0938	0.501±0.1204	0.407±0.1761	0.413±0.1482	0.375±0.1206	0.396±0.1676
1	207	0.468±0.1189	0.464±0.0761	0.497±0.0994	0.478±0.1292	0.473±0.1099	0.479±0.1254	0.511±0.0820	0.473±0.1086
	276	0.468±0.0717	0.399±0.0775	0.441±0.1164	0.413±0.0792	0.417±0.1062	0.469±0.1010	0.460±0.0981	0.460±0.1015
	345	0.423±0.0720	0.473±0.1093	0.447±0.0832	0.482±0.1008	0.505±0.0835	0.411±0.0815	0.429±0.0852	0.455±0.0906
	20	0.150±0.2415	0.200±0.3496	0.150±0.2415	0.250±0.3536	0.350±0.3375	0.150±0.2415	0.250±0.2635	0.150±0.2415
	40	0.118±0.1306	0.163±0.1927	0.123±0.1350	0.193±0.1607	0.155±0.1442	0.143±0.1294	0.135±0.1667	0.090±0.1663
2	60	0.158±0.1660	0.127±0.1568	0.199±0.2467	0.079±0.0833	0.090±0.1405	0.146±0.1168	0.141±0.1733	0.158±0.1660
	80	0.286±0.1802	0.212±0.1288	0.176±0.1071	0.174±0.1185	0.137±0.0664	0.228±0.1084	0.213±0.1043	0.253±0.1256
	100	0.237±0.1530	0.235±0.1499	0.208±0.0957	0.149±0.0958	0.189±0.1078	0.258±0.1784	0.249±0.1219	0.228±0.1131
	9043	0.112±0.0048	0.114±0.0023	0.117±0.0026	0.114±0.0013	0.114±0.0007	0.115±0.0053	0.114±0.0022	0.113±0.0040
	18086	0.117±0.0008	0.117±0.0019	0.117±0.0029	0.116±0.0011	0.116±0.0011	0.116±0.0010	0.116±0.0026	0.116±0.0030
3	27129	0.117±0.0015	0.117±0.0017	0.117±0.0029	0.118±0.0004	0.117±0.0007	0.118±0.0005	0.117±0.0013	0.118±0.0005
	36172	0.119±0.0005	0.117±0.0037	0.118±0.0026	0.119±0.0003	0.119±0.0003	0.118±0.0023	0.117±0.0026	0.117±0.0027
	45211	0.117±0.0011	0.117±0.0013	0.117±0.0009	0.117±0.0009	0.117±0.0002	0.117±0.0009	0.116±0.0017	0.117±0.0010
	30	0.083±0.1800	0.117±0.1933	0.033±0.1054	0.033±0.1054	0.067±0.1405	0.117±0.1933	0.067±0.1405	0.033±0.1054
	60	0.081±0.0856	0.134±0.1066	0.112±0.1106	0.114±0.1116	0.100±0.1165	0.098±0.0843	0.098±0.0843	0.114±0.1116
4	90	0.053±0.1068	0.042±0.0892	0.043±0.0767	0.064±0.0898	0.076±0.0740	0.054±0.0778	0.043±0.0560	0.064±0.0764
	120	0.051±0.0588	0.042±0.0442	0.060±0.0693	0.034±0.0441	0.051±0.0440	0.049±0.0561	0.043±0.0456	0.033±0.0433
	150	0.047±0.0549	0.047±0.0632	0.053±0.0422	0.073±0.0584	0.060±0.0663	0.047±0.0450	0.073±0.0663	0.047±0.0632
	36	0.092±0.1493	0.092±0.1493	0.098±0.1622	0.103±0.1372	0.157±0.1421	0.100±0.1610	0.078±0.1301	0.138±0.2061
	72	0.068±0.1147	0.082±0.1487	0.111±0.0832	0.096±0.1117	0.066±0.1237	0.082±0.1142	0.066±0.0915	0.109±0.1066
5	108	0.064±0.0941	0.071±0.0777	0.045±0.0475	0.056±0.0781	0.121±0.0971	0.027±0.0584	0.090±0.1074	0.045±0.0616
	144	0.063±0.0710	0.062±0.0677	0.056±0.0560	0.034±0.0475	0.041±0.0357	0.062±0.0775	0.062±0.0612	0.070±0.0744
	178	0.057±0.0547	0.045±0.0440	0.028±0.0300	0.045±0.0354	0.045±0.0440	0.034±0.0405	0.068±0.0648	0.045±0.0238
	295	0.616±0.1162	0.617±0.1116	0.618±0.0650	0.539±0.0783	0.558±0.1048	0.603±0.0929	0.583±0.0861	0.569±0.0927
	590	0.600±0.0717	0.619±0.0551	0.614±0.0584	0.613±0.0757	0.602±0.0815	0.600±0.0415	0.612±0.0726	0.600±0.0463
6	885	0.586±0.0613	0.590±0.0533	0.573±0.0514	0.581±0.0421	0.558±0.0899	0.547±0.0483	0.584±0.0597	0.590±0.0318
	1180	0.543±0.0385	0.577±0.0338	0.566±0.0464	0.558±0.0302	0.566±0.0413	0.548±0.0526	0.561±0.0479	0.557±0.0591
	1473	0.553±0.0475	0.551±0.0158	0.569±0.0510	0.541±0.0404	0.563±0.0438	0.537±0.0381	0.560±0.0391	0.566±0.0575
	346	0.269±0.0497	0.257±0.0753	0.245±0.0529	0.254±0.0387	0.239±0.0754	0.211±0.0785	0.260±0.0750	0.181±0.0914
	692	0.254±0.0283	0.253±0.0645	0.247±0.0544	0.260±0.0713	0.236±0.0369	0.227±0.0469	0.248±0.0446	0.201±0.0314
7	1038	0.197±0.0296	0.225±0.0545	0.227±0.0545	0.211±0.0443	0.217±0.0392	0.223±0.0337	0.228±0.0333	0.200±0.0474
	1384	0.217±0.0332	0.228±0.0304	0.231±0.0531	0.210±0.0373	0.238±0.0262	0.208±0.0398	0.240±0.0455	0.201±0.0386
	1728	0.203±0.0262	0.222±0.0360	0.215±0.0360	0.211±0.0283	0.214±0.0265	0.225±0.0385	0.237±0.0292	0.172±0.0372
	72	0.041±0.0663	0.041±0.0663	0.113±0.1754	0.041±0.0663	0.029±0.0602	0.013±0.0395	0.041±0.0663	0.029±0.0602
	144	0.068±0.0614	0.050±0.0482	0.062±0.0404	0.042±0.0361	0.055±0.0535	0.084±0.0811	0.062±0.0767	0.058±0.0476
8	216	0.079±0.0453	0.070±0.0557	0.054±0.0410	0.036±0.0347	0.088±0.0537	0.065±0.0561	0.046±0.0472	0.019±0.0240
	288	0.076±0.0553	0.038±0.0306	0.062±0.0141	0.055±0.0400	0.038±0.0407	0.063±0.0430	0.045±0.0369	0.052±0.0299

	358	0.089±0.0389	0.039±0.0322	0.044±0.0338	0.039±0.0269	0.061±0.0362	0.072±0.0458	0.053±0.0304	0.045±0.0295
9	462	0.212±0.0723	0.098±0.0297	0.082±0.0270	0.147±0.0555	0.182±0.0436	0.171±0.0620	0.139±0.0491	0.152±0.0551
	924	0.157±0.0359	0.066±0.0164	0.066±0.0245	0.147±0.0313	0.170±0.0389	0.139±0.0427	0.136±0.0300	0.144±0.0256
	1386	0.175±0.0287	0.083±0.0147	0.075±0.0128	0.154±0.0291	0.186±0.0404	0.157±0.0149	0.141±0.0175	0.155±0.0231
	1848	0.154±0.0277	0.082±0.0205	0.070±0.0119	0.141±0.0248	0.176±0.0267	0.151±0.0309	0.127±0.0245	0.134±0.0276
	2310	0.152±0.0210	0.079±0.0189	0.067±0.0146	0.146±0.0247	0.160±0.0289	0.145±0.0178	0.129±0.0293	0.135±0.0174
10	1287	0.158±0.0280	0.156±0.0325	0.145±0.0358	0.156±0.0194	0.162±0.0249	0.160±0.0329	0.163±0.0218	0.304±0.0195
	2574	0.160±0.0231	0.154±0.0133	0.143±0.0220	0.178±0.0198	0.167±0.0153	0.160±0.0153	0.165±0.0196	0.300±0.0171
	3861	0.158±0.0163	0.144±0.0184	0.144±0.0255	0.164±0.0199	0.167±0.0238	0.152±0.0231	0.158±0.0225	0.314±0.0183
	5148	0.157±0.0133	0.139±0.0071	0.139±0.0106	0.164±0.0149	0.162±0.0140	0.156±0.0088	0.153±0.0162	0.309±0.0090
	6435	0.159±0.0131	0.135±0.0088	0.131±0.0110	0.158±0.0145	0.163±0.0083	0.154±0.0135	0.157±0.0112	0.312±0.0086
11	297	0.468±0.0729	0.494±0.1023	0.531±0.1070	0.450±0.1084	0.447±0.0723	0.478±0.0971	0.475±0.0697	0.469±0.1051
	594	0.483±0.0634	0.475±0.0942	0.454±0.0788	0.487±0.0499	0.509±0.0464	0.493±0.0721	0.458±0.0803	0.450±0.0578
	891	0.498±0.0552	0.485±0.0302	0.446±0.0542	0.495±0.0662	0.520±0.0502	0.499±0.0646	0.469±0.0485	0.479±0.0654
	1188	0.470±0.0433	0.454±0.0297	0.441±0.0666	0.488±0.0351	0.517±0.0612	0.465±0.0568	0.459±0.0530	0.438±0.0640
	1484	0.469±0.0471	0.456±0.0462	0.444±0.0470	0.503±0.0569	0.505±0.0372	0.461±0.0425	0.476±0.0594	0.456±0.0508
12	2199	0.089±0.0154	0.087±0.0312	0.081±0.0291	0.111±0.0276	0.124±0.0333	0.091±0.0293	0.110±0.0179	0.100±0.0278
	4398	0.089±0.0178	0.090±0.0132	0.084±0.0079	0.104±0.0187	0.123±0.0184	0.091±0.0176	0.103±0.0159	0.095±0.0150
	6597	0.096±0.0079	0.082±0.0085	0.080±0.0160	0.108±0.0110	0.136±0.0266	0.091±0.0135	0.099±0.0104	0.095±0.0110
	8796	0.092±0.0173	0.087±0.0103	0.076±0.0103	0.109±0.0120	0.128±0.0213	0.087±0.0125	0.109±0.0178	0.092±0.0096
	10992	0.085±0.0126	0.080±0.0118	0.076±0.0102	0.105±0.0151	0.123±0.0163	0.088±0.0125	0.105±0.0104	0.092±0.0094
13	1124	0.109±0.0354	0.097±0.0333	0.084±0.0292	0.137±0.0408	0.181±0.0515	0.101±0.0431	0.129±0.0327	0.078±0.0289
	2248	0.101±0.0165	0.076±0.0192	0.074±0.0191	0.120±0.0189	0.191±0.0382	0.093±0.0187	0.130±0.0236	0.087±0.0251
	3372	0.091±0.0097	0.073±0.0095	0.070±0.0193	0.104±0.0170	0.154±0.0338	0.085±0.0147	0.111±0.0185	0.082±0.0206
	4496	0.088±0.0155	0.073±0.0077	0.070±0.0158	0.110±0.0132	0.176±0.0230	0.088±0.0167	0.121±0.0199	0.087±0.0144
	5620	0.093±0.0109	0.074±0.0056	0.073±0.0131	0.115±0.0120	0.176±0.0245	0.085±0.0146	0.128±0.0167	0.088±0.0094
14	4000	0.531±0.0199	0.400±0.0176	0.383±0.0313	0.598±0.0404	0.656±0.0357	0.534±0.0322	0.559±0.0199	0.386±0.0245
	8000	0.510±0.0172	0.383±0.0168	0.366±0.0165	0.561±0.0114	0.658±0.0238	0.504±0.0227	0.541±0.0181	0.374±0.0179
	12000	0.502±0.0183	0.372±0.0164	0.361±0.0126	0.553±0.0235	0.635±0.0140	0.501±0.0134	0.528±0.0178	0.371±0.0133
	16000	0.511±0.0149	0.372±0.0223	0.351±0.0146	0.546±0.0218	0.640±0.0130	0.493±0.0195	0.518±0.0158	0.364±0.0204
	20000	0.509±0.0198	0.371±0.0205	0.354±0.0182	0.547±0.0193	0.625±0.0233	0.504±0.0171	0.517±0.0202	0.361±0.0090

Table 8.8 Average testing MQE ± STD over the 14 benchmark datasets

Dataset	Nº samples	SIM	Beta-SIM	WeVoS-Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
	69	0.273±0.0581	0.270±0.0588	0.284±0.0547	0.314±0.0601	0.356±0.0836	0.284±0.0529	0.321±0.0582	0.267±0.0349
	138	0.302±0.0593	0.298±0.0544	0.296±0.0575	0.342±0.0771	0.410±0.0949	0.301±0.0557	0.363±0.0693	0.294±0.0482
1	207	0.274±0.0342	0.270±0.0392	0.271±0.0355	0.320±0.0427	0.393±0.0469	0.286±0.0393	0.333±0.0325	0.266±0.0285
	276	0.271±0.0234	0.258±0.0368	0.254±0.0319	0.314±0.0424	0.381±0.0540	0.270±0.0301	0.325±0.0188	0.252±0.0240
	345	0.273±0.0452	0.257±0.0502	0.256±0.0512	0.316±0.0632	0.367±0.0707	0.268±0.0453	0.333±0.0586	0.251±0.0355
	20	1.271±0.4257	1.253±0.4362	1.299±0.4564	1.543±0.3992	1.576±0.4144	1.272±0.4763	1.264±0.4367	1.271±0.4981
	40	1.185±0.2158	1.151±0.3075	1.164±0.1639	1.559±0.1349	1.678±0.1624	1.137±0.2281	1.184±0.2166	1.082±0.2010
2	60	1.082±0.1299	1.135±0.1129	1.004±0.3907	1.555±0.0644	1.726±0.1009	1.070±0.1453	1.074±0.1206	0.979±0.1642
	80	1.065±0.1432	1.159±0.1951	1.106±0.2257	1.533±0.1191	1.694±0.1569	1.052±0.1598	1.083±0.1516	0.982±0.1406
	100	1.083±0.1255	1.080±0.1832	1.087±0.1921	1.546±0.1504	1.661±0.1187	1.044±0.1568	1.042±0.1540	0.968±0.1053
	9043	1.055±0.0212	1.112±0.0442	1.086±0.0461	1.223±0.0185	1.330±0.0460	1.041±0.0311	1.005±0.0259	0.994±0.0142
	18086	1.062±0.0102	1.115±0.0473	1.061±0.0321	1.218±0.0136	1.364±0.0798	1.039±0.0211	0.998±0.0109	0.995±0.0156
3	27129	1.047±0.0226	1.101±0.0257	1.082±0.0361	1.232±0.0111	1.367±0.0560	1.042±0.0153	1.004±0.0116	1.000±0.0133
	36172	1.042±0.0216	1.084±0.0194	1.041±0.0218	1.224±0.0181	1.384±0.0683	1.027±0.0104	0.995±0.0126	0.991±0.0155
	45211	1.048±0.0194	1.113±0.0304	1.101±0.0406	1.224±0.0112	1.349±0.0547	1.030±0.0159	1.001±0.0079	0.999±0.0138
	30	0.150±0.0471	0.159±0.0429	0.162±0.0530	0.212±0.0698	0.254±0.0781	0.159±0.0398	0.141±0.0309	0.171±0.0446
	60	0.189±0.0424	0.173±0.0337	0.170±0.0367	0.216±0.0569	0.256±0.0637	0.182±0.0415	0.172±0.0481	0.176±0.0368
4	90	0.157±0.0297	0.146±0.0265	0.157±0.0260	0.216±0.0513	0.255±0.0654	0.150±0.0228	0.147±0.0257	0.157±0.0214
	120	0.168±0.0182	0.154±0.0221	0.152±0.0273	0.221±0.0430	0.259±0.0332	0.177±0.0259	0.162±0.0212	0.156±0.0180
	150	0.172±0.0232	0.155±0.0255	0.164±0.0251	0.220±0.0536	0.252±0.0694	0.176±0.0224	0.153±0.0208	0.155±0.0199
	36	0.788±0.2192	0.770±0.2263	0.779±0.2192	0.815±0.2219	0.891±0.2175	0.757±0.2419	0.770±0.2309	0.800±0.2016
	72	0.785±0.1029	0.792±0.1327	0.799±0.1349	0.843±0.1338	0.913±0.1594	0.779±0.0977	0.781±0.1124	0.778±0.0630
5	108	0.731±0.0834	0.729±0.0765	0.748±0.0813	0.782±0.0741	0.858±0.0710	0.704±0.0747	0.711±0.0885	0.687±0.0604
	144	0.703±0.0165	0.729±0.0248	0.739±0.0268	0.780±0.0322	0.845±0.0562	0.713±0.0321	0.715±0.0233	0.678±0.0225
	178	0.707±0.0532	0.737±0.0635	0.736±0.0757	0.792±0.0591	0.849±0.0762	0.711±0.0538	0.717±0.0567	0.665±0.0436
	295	0.786±0.0393	0.698±0.0493	0.691±0.0567	0.901±0.0606	1.114±0.0607	0.755±0.0496	0.727±0.0491	0.555±0.0309
	590	0.765±0.0540	0.667±0.0320	0.635±0.0518	0.883±0.0580	1.096±0.0929	0.735±0.0453	0.717±0.0436	0.502±0.0413
6	885	0.760±0.0463	0.674±0.0404	0.655±0.0391	0.885±0.0516	1.115±0.1007	0.729±0.0433	0.713±0.0498	0.508±0.0247
	1180	0.745±0.0314	0.650±0.0194	0.634±0.0243	0.871±0.0220	1.063±0.0822	0.721±0.0288	0.696±0.0215	0.486±0.0191
	1473	0.740±0.0241	0.731±0.1435	0.642±0.0323	0.869±0.0376	1.096±0.0805	0.715±0.0220	0.704±0.0284	0.492±0.0189
	346	1.067±0.0433	1.011±0.0413	1.000±0.0616	1.192±0.0176	1.345±0.0421	1.027±0.0317	1.047±0.0273	0.863±0.0349
	692	1.086±0.0440	1.033±0.0300	0.979±0.0222	1.193±0.0353	1.351±0.0326	1.030±0.0342	1.037±0.0272	0.865±0.0270
7	1038	1.071±0.0166	1.034±0.0362	1.003±0.0280	1.190±0.0302	1.362±0.0367	1.031±0.0275	1.038±0.0259	0.852±0.0153
	1384	1.077±0.0333	1.039±0.0408	1.000±0.0182	1.209±0.0232	1.377±0.0212	1.034±0.0268	1.038±0.0145	0.861±0.0119
	1728	1.075±0.0247	1.038±0.0235	0.991±0.0206	1.205±0.0234	1.373±0.0307	1.041±0.0244	1.049±0.0239	0.855±0.0069
	72	1.907±0.1526	1.875±0.1530	1.648±0.5929	1.930±0.1503	2.090±0.1507	1.893±0.1591	1.902±0.1489	1.821±0.1609
8	144	1.848±0.1067	1.753±0.0817	1.653±0.1058	1.868±0.0886	1.959±0.1094	1.840±0.1073	1.846±0.0767	1.744±0.0932
	216	1.867±0.1031	1.725±0.0808	1.677±0.0827	1.866±0.0853	1.954±0.0946	1.883±0.0760	1.851±0.0896	1.738±0.0777

	288	1.827±0.0721	1.679±0.0702	1.676±0.0748	1.859±0.0642	1.979±0.0992	1.828±0.0719	1.838±0.0837	1.713±0.0627
	358	1.831±0.0650	1.685±0.0633	1.675±0.0679	1.861±0.0616	1.911±0.0635	1.826±0.0809	1.830±0.0584	1.705±0.0587
9	462	0.467±0.0206	0.317±0.0187	0.310±0.0217	0.524±0.0290	0.628±0.0732	0.456±0.0190	0.510±0.0270	0.542±0.0229
	924	0.474±0.0337	0.338±0.0322	0.323±0.0266	0.542±0.0355	0.621±0.0468	0.468±0.0323	0.537±0.0341	0.557±0.0341
	1386	0.484±0.0172	0.330±0.0124	0.319±0.0123	0.551±0.0142	0.683±0.0727	0.471±0.0161	0.536±0.0190	0.558±0.0187
	1848	0.472±0.0175	0.320±0.0263	0.305±0.0197	0.539±0.0203	0.647±0.0631	0.467±0.0187	0.531±0.0203	0.548±0.0210
	2310	0.472±0.0164	0.320±0.0172	0.304±0.0122	0.541±0.0151	0.670±0.0686	0.460±0.0191	0.525±0.0143	0.549±0.0153
10	1287	0.593±0.0272	0.523±0.0220	0.514±0.0195	0.650±0.0216	0.674±0.0221	0.588±0.0299	0.619±0.0265	0.596±0.0335
	2574	0.595±0.0286	0.518±0.0191	0.505±0.0221	0.651±0.0238	0.674±0.0212	0.592±0.0218	0.629±0.0324	0.599±0.0244
	3861	0.597±0.0163	0.514±0.0135	0.503±0.0098	0.658±0.0180	0.694±0.0179	0.593±0.0222	0.629±0.0151	0.600±0.0199
	5148	0.596±0.0168	0.516±0.0091	0.499±0.0100	0.654±0.0113	0.687±0.0168	0.593±0.0167	0.622±0.0142	0.584±0.0166
	6435	0.600±0.0127	0.516±0.0081	0.497±0.0092	0.655±0.0119	0.688±0.0103	0.594±0.0103	0.624±0.0113	0.587±0.0094
11	297	0.277±0.0268	0.255±0.0201	0.255±0.0277	0.311±0.0311	0.345±0.0273	0.276±0.0325	0.286±0.0269	0.281±0.0269
	594	0.283±0.0225	0.247±0.0157	0.246±0.0149	0.334±0.0318	0.364±0.0287	0.274±0.0238	0.295±0.0266	0.287±0.0251
	891	0.274±0.0159	0.241±0.0122	0.238±0.0093	0.317±0.0243	0.356±0.0346	0.274±0.0150	0.291±0.0211	0.282±0.0183
	1188	0.276±0.0181	0.246±0.0158	0.240±0.0144	0.317±0.0248	0.356±0.0287	0.270±0.0191	0.290±0.0249	0.282±0.0204
	1484	0.278±0.0124	0.246±0.0108	0.239±0.0118	0.320±0.0131	0.349±0.0222	0.272±0.0109	0.293±0.0117	0.288±0.0120
12	2199	0.719±0.0198	0.679±0.0216	0.667±0.0175	0.888±0.0172	1.097±0.1305	0.702±0.0218	0.907±0.0238	0.916±0.0195
	4398	0.695±0.0129	0.667±0.0147	0.651±0.0094	0.872±0.0140	1.032±0.1071	0.683±0.0140	0.890±0.0121	0.893±0.0150
	6597	0.705±0.0103	0.665±0.0149	0.653±0.0146	0.877±0.0061	1.080±0.1605	0.693±0.0129	0.896±0.0096	0.904±0.0096
	8796	0.702±0.0099	0.669±0.0125	0.651±0.0040	0.880±0.0073	1.084±0.1445	0.689±0.0130	0.901±0.0069	0.903±0.0077
	10992	0.699±0.0109	0.662±0.0081	0.649±0.0124	0.881±0.0068	1.073±0.1346	0.689±0.0056	0.899±0.0065	0.904±0.0096
13	1124	2.470±0.0891	2.406±0.0658	2.390±0.0886	2.712±0.0712	3.239±0.1038	2.456±0.0834	2.814±0.0665	2.714±0.0742
	2248	2.444±0.0340	2.347±0.0402	2.311±0.0376	2.693±0.0374	3.230±0.0561	2.445±0.0339	2.807±0.0380	2.691±0.0361
	3372	2.462±0.0278	2.340±0.0336	2.305±0.0185	2.691±0.0211	3.171±0.0785	2.447±0.0271	2.800±0.0185	2.696±0.0274
	4496	2.460±0.0324	2.331±0.0319	2.311±0.0270	2.705±0.0311	3.216±0.0826	2.447±0.0253	2.811±0.0329	2.711±0.0324
	5620	2.455±0.0238	2.337±0.0205	2.296±0.0174	2.702±0.0158	3.191±0.1283	2.454±0.0158	2.813±0.0197	2.707±0.0202
14	4000	0.526±0.0070	0.439±0.0076	0.425±0.0086	0.601±0.0081	0.716±0.0292	0.524±0.0090	0.566±0.0103	0.450±0.0046
	8000	0.527±0.0114	0.436±0.0054	0.423±0.0060	0.603±0.0048	0.735±0.0387	0.525±0.0068	0.568±0.0047	0.447±0.0050
	12000	0.527±0.0040	0.436±0.0037	0.421±0.0035	0.604±0.0042	0.729±0.0218	0.522±0.0071	0.568±0.0040	0.449±0.0053
	16000	0.530±0.0029	0.434±0.0047	0.420±0.0022	0.604±0.0050	0.738±0.0269	0.523±0.0067	0.567±0.0045	0.448±0.0020
	20000	0.530±0.0033	0.432±0.0034	0.420±0.0054	0.605±0.0035	0.715±0.0352	0.521±0.0051	0.567±0.0035	0.447±0.0037

Table 8.9 Average testing TE ± STD over the 14 benchmark datasets

Dataset	Nº samples	SIM	Beta-SIM	WeVoS-Beta-SIM	SOM	WeVoS-SOM	MLHL-SIM	ViSOM	GNG
1	69	0.239±0.1829	0.224±0.1579	0.417±0.2234	0.127±0.1420	0.149±0.1203	0.258±0.1316	0.039±0.0634	0.768±0.1341
	138	0.364±0.1955	0.311±0.1050	0.401±0.1315	0.057±0.0656	0.160±0.1298	0.378±0.1434	0.093±0.0827	0.805±0.1124
	207	0.280±0.1090	0.391±0.1368	0.357±0.0948	0.073±0.0714	0.204±0.1149	0.310±0.0887	0.072±0.0337	0.787±0.0873
	276	0.311±0.0615	0.363±0.1388	0.333±0.0792	0.087±0.0535	0.149±0.0434	0.312±0.0741	0.066±0.0597	0.744±0.1065
	345	0.319±0.0861	0.386±0.0780	0.391±0.1045	0.108±0.0524	0.140±0.0720	0.340±0.0634	0.055±0.0495	0.730±0.0589
2	20	0.200±0.2582	0.150±0.2415	0.150±0.2415	0.000±0.0000	0.000±0.0000	0.150±0.2415	0.350±0.3375	0.600±0.3162
	40	0.110±0.1912	0.143±0.1601	0.190±0.1404	0.053±0.1167	0.092±0.1493	0.000±0.0000	0.070±0.1135	0.600±0.3074
	60	0.098±0.1395	0.119±0.1431	0.133±0.0994	0.000±0.0000	0.149±0.1466	0.050±0.0805	0.166±0.1422	0.438±0.1767
	80	0.087±0.0799	0.190±0.1909	0.199±0.1774	0.013±0.0395	0.100±0.1294	0.174±0.1323	0.061±0.0645	0.417±0.1434
	100	0.081±0.0793	0.088±0.1082	0.150±0.0972	0.010±0.0316	0.010±0.0316	0.041±0.0711	0.068±0.0793	0.401±0.1517
3	9043	0.054±0.0133	0.118±0.0277	0.109±0.0483	0.027±0.0155	0.074±0.0350	0.057±0.0192	0.021±0.0081	0.323±0.0437
	18086	0.048±0.0135	0.116±0.0419	0.098±0.0174	0.020±0.0065	0.116±0.0740	0.066±0.0169	0.028±0.0060	0.293±0.0370
	27129	0.051±0.0219	0.106±0.0362	0.125±0.0305	0.025±0.0095	0.105±0.0782	0.060±0.0234	0.026±0.0121	0.312±0.0324
	36172	0.050±0.0187	0.123±0.0347	0.092±0.0239	0.023±0.0096	0.164±0.1018	0.054±0.0132	0.027±0.0109	0.302±0.0392
	45211	0.065±0.0317	0.123±0.0456	0.121±0.0382	0.022±0.0050	0.120±0.0632	0.057±0.0125	0.028±0.0100	0.316±0.0276
4	30	0.267±0.3443	0.242±0.3250	0.125±0.2265	0.158±0.2306	0.375±0.3711	0.308±0.3380	0.033±0.1054	0.533±0.3244
	60	0.241±0.2004	0.423±0.2518	0.177±0.1302	0.062±0.1048	0.146±0.1286	0.319±0.1823	0.000±0.0000	0.652±0.1935
	90	0.277±0.1686	0.335±0.1647	0.174±0.1129	0.138±0.1062	0.238±0.1438	0.220±0.1254	0.060±0.0868	0.586±0.1702
	120	0.209±0.0900	0.269±0.1457	0.245±0.1549	0.113±0.1002	0.149±0.0736	0.277±0.1622	0.091±0.0817	0.555±0.1944
	150	0.260±0.1456	0.333±0.1089	0.300±0.1305	0.153±0.1219	0.207±0.1153	0.160±0.1004	0.073±0.0734	0.587±0.0932
5	36	0.205±0.2577	0.375±0.2443	0.127±0.1705	0.020±0.0632	0.140±0.1818	0.248±0.2217	0.125±0.2265	0.618±0.1729
	72	0.211±0.1196	0.288±0.2056	0.357±0.2234	0.109±0.0827	0.152±0.1546	0.268±0.1742	0.041±0.0663	0.764±0.1157
	108	0.225±0.0851	0.378±0.0943	0.195±0.1107	0.081±0.0784	0.103±0.1185	0.213±0.0982	0.019±0.0403	0.661±0.1558
	144	0.231±0.0845	0.318±0.1290	0.257±0.1208	0.076±0.0832	0.161±0.0908	0.208±0.0892	0.021±0.0338	0.694±0.1519
	178	0.310±0.0748	0.395±0.1535	0.208±0.0897	0.074±0.0538	0.136±0.0868	0.269±0.0991	0.056±0.0741	0.725±0.1448
6	295	0.081±0.0322	0.199±0.0870	0.145±0.0620	0.030±0.0399	0.176±0.0726	0.085±0.0586	0.030±0.0247	0.580±0.0437
	590	0.059±0.0293	0.134±0.0564	0.105±0.0429	0.043±0.0259	0.142±0.0580	0.058±0.0379	0.024±0.0198	0.500±0.0329
	885	0.084±0.0249	0.123±0.0483	0.091±0.0521	0.025±0.0167	0.154±0.0879	0.089±0.0180	0.025±0.0175	0.539±0.0540
	1180	0.081±0.0302	0.124±0.0254	0.105±0.0454	0.036±0.0164	0.123±0.0785	0.101±0.0436	0.043±0.0177	0.546±0.0371
	1473	0.081±0.0437	0.155±0.0669	0.081±0.0353	0.039±0.0199	0.176±0.0693	0.075±0.0179	0.037±0.0167	0.508±0.0344
7	346	0.127±0.0570	0.286±0.0815	0.208±0.0407	0.046±0.0281	0.312±0.0952	0.116±0.0522	0.072±0.0543	0.681±0.0613
	692	0.132±0.0392	0.327±0.0594	0.192±0.0633	0.052±0.0268	0.207±0.0633	0.134±0.0244	0.049±0.0316	0.687±0.0386
	1038	0.128±0.0322	0.314±0.0612	0.194±0.0684	0.050±0.0256	0.223±0.0388	0.133±0.0194	0.059±0.0202	0.672±0.0498
	1384	0.156±0.0443	0.289±0.0420	0.174±0.0338	0.042±0.0122	0.202±0.0496	0.123±0.0294	0.064±0.0267	0.670±0.0458
	1728	0.133±0.0309	0.306±0.0425	0.177±0.0293	0.054±0.0244	0.198±0.0346	0.124±0.0255	0.052±0.0160	0.686±0.0286
8	72	0.082±0.0979	0.219±0.1702	0.161±0.1943	0.095±0.1111	0.098±0.1170	0.111±0.1119	0.113±0.1123	0.557±0.2346
	144	0.132±0.1237	0.167±0.0991	0.166±0.0738	0.077±0.0653	0.115±0.0961	0.131±0.0826	0.026±0.0453	0.569±0.1583
	216	0.230±0.1105	0.183±0.0808	0.125±0.0848	0.060±0.0507	0.148±0.0939	0.150±0.0914	0.032±0.0306	0.455±0.1053
	288	0.107±0.0496	0.166±0.0832	0.239±0.0712	0.045±0.0232	0.145±0.0598	0.133±0.1113	0.034±0.0363	0.456±0.1382

	358	0.163±0.0798	0.168±0.0649	0.182±0.0774	0.056±0.0460	0.097±0.0673	0.154±0.0729	0.028±0.0229	0.436±0.0924
	462	0.143±0.0751	0.283±0.0583	0.236±0.0671	0.084±0.0276	0.113±0.0513	0.148±0.0980	0.061±0.0433	0.292±0.0906
	924	0.125±0.0674	0.285±0.0359	0.263±0.0586	0.053±0.0202	0.104±0.0398	0.177±0.0794	0.047±0.0215	0.318±0.0581
9	1386	0.146±0.0474	0.263±0.0554	0.269±0.0525	0.066±0.0330	0.115±0.0381	0.181±0.0774	0.062±0.0240	0.284±0.0408
	1848	0.130±0.0421	0.268±0.0420	0.256±0.0498	0.078±0.0229	0.121±0.0632	0.143±0.0743	0.061±0.0213	0.318±0.0481
	2310	0.120±0.0460	0.257±0.0400	0.228±0.0446	0.069±0.0122	0.102±0.0314	0.119±0.0447	0.046±0.0165	0.287±0.0384
	1287	0.215±0.0873	0.308±0.0815	0.225±0.0727	0.069±0.0241	0.068±0.0239	0.219±0.0745	0.046±0.0208	0.493±0.0584
	2574	0.188±0.0954	0.294±0.0655	0.233±0.0622	0.075±0.0186	0.071±0.0188	0.194±0.0725	0.042±0.0126	0.492±0.0299
10	3861	0.191±0.0446	0.303±0.0444	0.249±0.0593	0.087±0.0256	0.083±0.0176	0.193±0.0597	0.034±0.0089	0.520±0.0376
	5148	0.228±0.1048	0.298±0.0506	0.223±0.0411	0.078±0.0153	0.068±0.0181	0.164±0.0451	0.041±0.0089	0.500±0.0320
	6435	0.213±0.0810	0.295±0.0639	0.218±0.0477	0.082±0.0191	0.083±0.0257	0.173±0.0759	0.042±0.0084	0.492±0.0345
	297	0.253±0.0686	0.307±0.0544	0.225±0.0768	0.060±0.0302	0.162±0.0770	0.232±0.0959	0.054±0.0324	0.542±0.0945
	594	0.239±0.0507	0.289±0.0627	0.216±0.0325	0.076±0.0252	0.165±0.0642	0.263±0.0550	0.032±0.0243	0.490±0.0684
11	891	0.305±0.0572	0.317±0.0442	0.220±0.0434	0.079±0.0287	0.162±0.0593	0.280±0.0680	0.048±0.0145	0.509±0.0608
	1188	0.257±0.0616	0.320±0.0418	0.219±0.0620	0.084±0.0327	0.141±0.0525	0.316±0.0561	0.064±0.0271	0.482±0.0369
	1484	0.314±0.0253	0.284±0.0333	0.214±0.0387	0.071±0.0258	0.115±0.0383	0.291±0.0373	0.046±0.0172	0.485±0.0573
	2199	0.094±0.0203	0.143±0.0244	0.101±0.0266	0.054±0.0140	0.141±0.0696	0.100±0.0289	0.048±0.0116	0.309±0.0349
	4398	0.102±0.0260	0.152±0.0238	0.094±0.0157	0.050±0.0139	0.103±0.0479	0.101±0.0112	0.038±0.0083	0.301±0.0397
12	6597	0.091±0.0189	0.131±0.0138	0.086±0.0131	0.051±0.0079	0.134±0.0982	0.110±0.0228	0.043±0.0116	0.294±0.0239
	8796	0.105±0.0319	0.137±0.0144	0.091±0.0189	0.048±0.0079	0.128±0.0955	0.098±0.0239	0.040±0.0076	0.302±0.0212
	10992	0.089±0.0171	0.132±0.0142	0.081±0.0145	0.052±0.0086	0.123±0.0631	0.106±0.0141	0.044±0.0098	0.300±0.0223
	1124	0.060±0.0225	0.115±0.0309	0.072±0.0293	0.047±0.0147	0.213±0.0600	0.079±0.0393	0.031±0.0178	0.302±0.0348
	2248	0.060±0.0105	0.101±0.0245	0.061±0.0279	0.036±0.0134	0.215±0.0447	0.053±0.0111	0.037±0.0161	0.284±0.0294
13	3372	0.059±0.0158	0.092±0.0127	0.059±0.0188	0.040±0.0123	0.135±0.0448	0.062±0.0157	0.029±0.0105	0.292±0.0277
	4496	0.056±0.0107	0.099±0.0122	0.056±0.0145	0.040±0.0102	0.173±0.0300	0.060±0.0123	0.026±0.0086	0.297±0.0277
	5620	0.058±0.0152	0.102±0.0154	0.059±0.0138	0.038±0.0059	0.165±0.0549	0.062±0.0134	0.031±0.0076	0.286±0.0166
	4000	0.203±0.0233	0.257±0.0254	0.221±0.0187	0.060±0.0141	0.183±0.0418	0.193±0.0266	0.087±0.0170	0.468±0.0317
	8000	0.213±0.0181	0.240±0.0143	0.206±0.0204	0.062±0.0105	0.193±0.0553	0.197±0.0166	0.074±0.0144	0.471±0.0270
14	12000	0.206±0.0135	0.241±0.0206	0.206±0.0207	0.075±0.0084	0.172±0.0191	0.212±0.0192	0.075±0.0096	0.477±0.0124
	16000	0.213±0.0113	0.249±0.0107	0.202±0.0135	0.071±0.0085	0.181±0.0316	0.213±0.0186	0.069±0.0085	0.480±0.0157
	20000	0.202±0.0152	0.242±0.0171	0.199±0.0132	0.071±0.0101	0.169±0.0459	0.199±0.0152	0.069±0.0101	0.470±0.0123