Studies in Computational Intelligence 322

Bruno Baruque
Emilio Corchado

Fusion Methods for Unsupervised Learning Ensembles

Springer

Studies in Computational Intelligence, Volume 322

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 299. Vassil Sgurev, Mincho Hadjiski, and Janusz Kacprzyk (Eds.)
Intelligent Systems: From Theory to Practice, 2010
ISBN 978-3-642-13427-2

Vol. 300. Baoding Liu (Ed.)
Uncertainty Theory, 2010
ISBN 978-3-642-13958-1

Vol. 301. Giuliano Armano, Marco de Gemmis, Giovanni Semeraro, and Eloisa Vargiu (Eds.)
Intelligent Information Access, 2010
ISBN 978-3-642-13999-4

Vol. 302. Bijaya Ketan Panigrahi, Ajith Abraham, and Swagatam Das (Eds.)
Computational Intelligence in Power Engineering, 2010
ISBN 978-3-642-14012-9

Vol. 303. Joachim Diederich, Cengiz Gunay, and James M. Hogan
Recruitment Learning, 2010
ISBN 978-3-642-14027-3

Vol. 304. Anthony Finn and Lakhmi C. Jain (Eds.)
Innovations in Defence Support Systems, 2010
ISBN 978-3-642-14083-9

Vol. 305. Stefania Montani and Lakhmi C. Jain (Eds.)
Successful Case-Based Reasoning Applications-1, 2010
ISBN 978-3-642-14077-8

Vol. 306. Tru Hoang Cao
Conceptual Graphs and Fuzzy Logic, 2010
ISBN 978-3-642-14086-0

Vol. 307. Anupam Shukla, Ritu Tiwari, and Rahul Kala
Towards Hybrid and Adaptive Computing, 2010
ISBN 978-3-642-14343-4

Vol. 308. Roger Nkambou, Jacqueline Bourdeau, and Riichiro Mizoguchi (Eds.)
Advances in Intelligent Tutoring Systems, 2010
ISBN 978-3-642-14362-5

Vol. 309. Isabelle Bichindaritz, Lakhmi C. Jain, Sachin Vaidya, and Ashlesha Jain (Eds.)
Computational Intelligence in Healthcare 4, 2010
ISBN 978-3-642-14463-9

Vol. 310. Dipti Srinivasan and Lakhmi C. Jain (Eds.)
Innovations in Multi-Agent Systems and Applications – 1, 2010
ISBN 978-3-642-14434-9

Vol. 311. Juan D. Velásquez and Lakhmi C. Jain (Eds.)
Advanced Techniques in Web Intelligence, 2010
ISBN 978-3-642-14460-8

Vol. 312. Patricia Melin, Janusz Kacprzyk, and Witold Pedrycz (Eds.)
Soft Computing for Recognition based on Biometrics, 2010
ISBN 978-3-642-15110-1

Vol. 313. Imre J. Rudas, János Fodor, and Janusz Kacprzyk (Eds.)
Computational Intelligence in Engineering, 2010
ISBN 978-3-642-15219-1

Vol. 314. Lorenzo Magnani, Walter Carnielli, and Claudio Pizzi (Eds.)
Model-Based Reasoning in Science and Technology, 2010
ISBN 978-3-642-15222-1

Vol. 315. Mohammad Essaaidi, Michele Malgeri, and Costin Badica (Eds.)
Intelligent Distributed Computing IV, 2010
ISBN 978-3-642-15210-8

Vol. 316. Philipp Wolfrum
Information Routing, Correspondence Finding, and Object Recognition in the Brain, 2010
ISBN 978-3-642-15253-5

Vol. 317. Roger Lee (Ed.)
Computer and Information Science 2010
ISBN 978-3-642-15404-1

Vol. 318. Oscar Castillo, Janusz Kacprzyk, and Witold Pedrycz (Eds.)
Soft Computing for Intelligent Control and Mobile Robotics, 2010
ISBN 978-3-642-15533-8

Vol. 319. Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, Tokuro Matsuo, and Hirofumi Yamaki (Eds.)
Innovations in Agent-Based Complex Automated Negotiations, 2010
ISBN 978-3-642-15611-3

Vol. 320. xxx

Vol. 321. Dimitri Plemenos and Georgios Miaoulis (Eds.)
Intelligent Computer Graphics 2010
ISBN 978-3-642-15689-2

Vol. 322. Bruno Baruque and Emilio Corchado
Fusion Methods for Unsupervised Learning Ensembles, 2010
ISBN 978-3-642-16204-6

Bruno Baruque and Emilio Corchado

Fusion Methods for Unsupervised Learning Ensembles

Springer

Dr. Bruno Baruque
Departamento de Ingeniería Civil
Escuela Politécnica Superior
Universidad de Burgos
Avda. Cantabria, s/n
09006 Burgos, Spain
E-mail: bbaruque@ubu.es

Dr. Emilio Corchado
Departamento de Informática y Automática
Facultad de Ciencias
Universidad de Salamanca
Plaza de la Merced, s/n
37008 Salamanca
Spain
E-mail: escorchado@usal.es

ISBN 978-3-642-16204-6 e-ISBN 978-3-642-16205-3

DOI 10.1007/978-3-642-16205-3

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2010936510

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Abstract

The application of a "committee of experts" or ensemble learning to artificial neural networks that apply unsupervised learning techniques is widely considered to enhance the effectiveness of such networks greatly. This book examines in one of its chapters the potential of the ensemble meta-algorithm by describing and testing a technique based on the combination of ensembles and statistical PCA that is able to determine the presence of outliers in high-dimensional data sets and to minimize outlier effects in the final results. After that, it presents its central contribution, which consists on an algorithm for the ensemble fusion of topology-preserving maps, referred to as Weighted Voting Superposition (WeVoS), which has been devised to improve data exploration by 2-D visualization over multi-dimensional data sets. This generic algorithm is applied in combination with several other models taken from the family of topology preserving maps, such as the SOM, ViSOM, SIM and Max-SIM. A range of quality measures for topology preserving maps that are proposed in the literature are used to validate and compare WeVoS with other algorithms. The experimental results demonstrate that, in the majority of cases, the WeVoS algorithm outperforms earlier map-fusion methods and the simpler versions of the algorithm with which it is compared. All the algorithms are tested in different artificial data sets and in several of the most common machine-learning data sets in order to corroborate their theoretical properties. Moreover, a real-life case-study taken from the food industry demonstrates the practical benefits of their applications to more complex problems.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

This book conducts a review of research into the application of "committee of experts" or ensemble learning techniques to artificial neural networks that employ unsupervised learning for dimensionality reduction tasks and especially, for visualizing multi-dimensional data sets. In addition, it sets out a method of assessing and identifying data-set outliers by means of ensemble and statistical analysis.

1.1 Background

Data Mining (DM) (Frawley et al, 1992) is the process of sorting through large amounts of data and picking out relevant information. It is usually employed by business intelligence organizations, and financial analysts, although its use is increasingly prevalent in various scientific disciplines to extract information from the enormous data sets generated by modern experimental and observational methods. It has been described as "the non-trivial extraction of implicit, previously unknown, and potentially useful information from data" and "the science of extracting useful information from large data sets or databases". One of the many techniques used for this relevant method of information extraction is data visualization.

An Artificial Neural Network (ANN) is a software simulation that typically emulates certain features of real neural networks found in animal brains. A branch of Artificial Intelligence (AI), ANNs consist of connectionist systems that have different applications based on their neural architecture. They can be used for pattern recognition, information compression and dimensionality reduction, clustering, classification, and visualization, among others. Some of these tasks, which will be discussed in greater detail in Chapter 2, also come under the umbrella of data mining, a field in which ANN tools are also considered useful. This book examines ANNs that implement unsupervised learning algorithms, where unsupervised is taken to mean that the networks will not be trained with a data set that includes pre-labelled data. This

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 1–4.
springerlink.com

process models similar ones found in young animals: all animals must learn to identify structure in their environment in an unsupervised manner.

Data projection or visualization, which facilitates the analysis of internal data set structures for the human expert, figures prominently among the applications of these unsupervised artificial neural networks. This can be achieved by data projection over more informative axes or by generating maps that represent the inner structure of data sets. Techniques such as Hebbian learning can be used for the first type of data visualization, and Self-Organizing Maps (SOM) are probably the most widely used technique for the second type.

Topology-preserving maps consist of algorithms that visualize and interpret large high-dimensional data sets, which means that the topology-preserving map is a useful tool for data mining by visual inspection. Typical applications are visualization of process states or financial results by representing instances of central dependency in the map data.

A major criticism of artificial neural networks in general is that their algorithms are rather unstable. One of the most common procedures to overcome instability in supervised learning algorithms is the use of the ensemble learning schema. In the field of AI, ensemble learning (Polikar, 2006) 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 performance -classification, prediction, function approximation, etc.- of a model, or to reduce the likelihood of a poorly adapted model being selected. An overview of these ideas and techniques is included in Chapter 3. It is important to note that the main ensemble learning algorithms and their most common applications are in the field of supervised learning.

1.2 Contributions

A preliminary study is performed to justify the concept of combining ensemble learning and unsupervised learning, which involves a method for adapting one of the most simple ensemble algorithms, known as Bagging (Breiman, 1996), to Principal Component Analysis (PCA) (Hotelling, 1933). The model has a dual application: firstly it can serve to indicate the presence of outliers in an analyzed data set; secondly, it can improve principal component (PC) calculation by approximating the PCs to those that would be obtained if no outliers were present in the data set.

Although this is not strictly an unsupervised learning technique, it can in one sense be compared with unsupervised learning; as whenever a data set includes outliers, it is not possible to determine whether the data projection directions are greatly influenced by the presence of those outliers or whether the influence of those outliers is negligible. The results confirm the usefulness of this approach both for synthetic and for real data sets. Chapter 3 details the implementation of the model and the tests that were performed.

The central contribution of this book lies in the field of ensemble learning for unsupervised algorithms; more specifically, it concerns the family of algorithms used for topology-preserving mapping. Certain algorithms have recently been proposed to obtain an ensemble of artificial neural networks and to fuse then into a final map, with the aim of improving on the results obtained by a single map. Two such algorithms -Fusion by Euclidean Distance and Fusion by Voronoi Polygon Similarity-, initially conceived for data classification and topology learning, respectively, are studied and discussed in Chapter 5. They provide useful solutions to the problems for which they were originally intended, although certain functional issues need to be considered for data visualization. In the same chapter, an initial approach to fusion algorithms for topographic-preserving mapping -Superposition- is presented, which aims to improve the data visualization features of the previously proposed models. This algorithm imposes a more restricted training on each of the maps in the ensemble, enabling the final fused map to be calculated in a simpler way, which is done by calculating the centroids of the neuron weights at each position on all the maps.

An upgrade of this same algorithm, called Weighted Voting Superposition (WeVoS) is described, which generates better visualization results than previous algorithms in the resulting fused maps. It is based on obtaining a performance or "quality" measure for the maps that compose the ensemble. Subsequently, instead of simply calculating the centroids of the neurons, which amounts to simple averaging, a weight is assigned to each neuron that is proportional to the "quality" of the neuron, in a similar way to the weighted voting scheme for combining ensemble outputs. This has the effect of selecting the best possible position for each neuron, from the set of possibilities that are available on the map ensemble. Moreover, the neighbourhood function is used when performing the fusion, which is a decisive feature for the visualization of the correct structure in a data set. The WeVoS model and its capabilities are subjected to a thorough study, which comprises various quality measures, tests with different data sets, and comparisons with similar algorithms. Different data sets were used to test WeVoS and to compare its results with those of the SOM, ViSOM, SIM and Max-SIM.

Finally, it is shown that WeVoS is able to improve the visualization performance of single topology-preserving models and the ensemble fusion algorithms previously presented in the literature.

1.3 Organization

This book is structured as follows: in Chapter 2, the main ideas underlying the human learning are outlined. Likewise, the implementation, using various computing techniques, of various sub-processes that involve ANNs and their different learning strategies are described, paying special attention to topology-preserving maps. Chapter 3 presents a summary of the most interesting ideas that relate to ensemble learning: theoretical explanations,

different types of ensemble training and different techniques to combine the results of their components. A combination of ensemble learning incorporating a statistical technique to discover interesting projections for a data set, and to identify the presence of outliers in the data set is presented and discussed in Chapter 4. Moreover, it presents and discusses the experimental results for this algorithm, and a real life case study to which was applied. Chapter 5 centres on combination techniques for ensembles of topology-preserving maps, which are known as fusion techniques. It includes a detailed study of two previously presented algorithms and proposes a preliminary version of a novel algorithm. Moreover, it sets out the experiments, results and comparisons of all these endeavours in combination with different topology-preserving algorithms. Chapter 6 presents the main contribution of the book, consisting of an improved version of the fusion algorithm presented in the previous chapter, entitled WeVoS. It details a complete study of its performance using various data sets, which includes a real life case-study. Finally, Chapter 7 presents the conclusions of the book and possible lines of future work. Appendix A contains background information on the real-life case study taken from the food industry.

Chapter 2
Modelling Human Learning: Artificial Neural Networks

2.1 The Human Learning Process

The autonomatic nervous system conducts stimuli from sensory receptors to the brain and the spinal cord, which conducts these 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 Figure 2.1 and the peripheral nervous system -the nerves that carry impulses to and from the central nervous system- (Britannica, 2009).

This book concentrates on functional aspects of one of the main parts of this system: the human brain.

Our intention is to describe the biological foundations of modern-day information processing which seeks to emulate the functioning of the human brain to obtain results that other approaches could only achieve with great difficult. Four main functions may be considered in relation to the performance of the human brain:

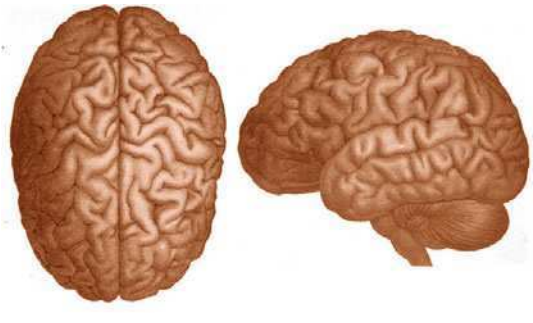


Fig. 2.1 Depiction of the human brain
Figure taken from (Carus, 1905)

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 5–29.
springerlink.com

1. *Sense.* The brain has a set of sensory faculties -sight, hearing, smell, taste, touch- which relay information to it on events in the outside world. It integrates information from these different senses and creates 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 an 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.
2. *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 is relayed to the brain for processing. Sensory information from both the external and the internal environment enters the central nervous system at the level of the brain stem and mid-brain. As this primary sensory input comes in, it is matched against previously stored patterns of activation and if unknown, or if associated with previous threats, the brain will activate a set of responses that are designed to help promote survival. As the brain cannot possibly create a unique neural imprint or pattern of change to store every constituent element of an experience, it stores "template" patterns based upon the first set of organisational 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 reflect these tiny differences.
3. *Store.* As the brain processes incoming information, it has the capacity to store elements of these incoming signals. It owes its ability to create memories to changes in its neurons and neural systems from one "homeostatic" state to another. Neurons undergo molecular changes that respond to a set of stimuli-induced -e.g., sensations- alterations in activity.
4. *Act.* Finally, the brain mediates and controls the actions of the human body. By regulating and directing the actions of the neuromuscular, autonomic nervous, endocrine and immune systems, the brain is able to control 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 -far beyond the reach any existing mathematical model of a complex system-.

Although interesting advances have certainly been made in artificially emulating all four functions, the scope of this work will centre on emulating 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 Biological Neuron

Ramon y Cajal (Ramon y Cajal, 1889) 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 specialised 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. The scheme representing a biological neuron is showed in Figure 2.2.

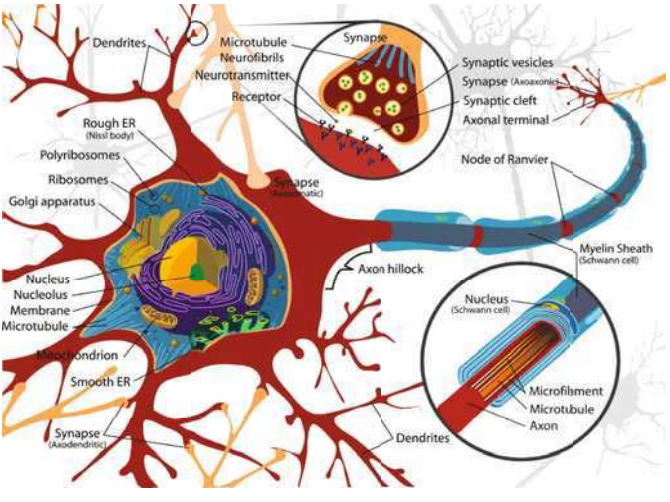


Fig. 2.2 Diagram of a biological neuron
Figure taken from (Ruiz, 2007)

- Dendrites are spindly 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 them into a series of all-or-no electrical potentials that propagate along the axon.
- The axon is a long, thin projection of the neuron through which action potentials are propagated to other neurons, often over a considerable distance. Most of the axon in the majority neurons is covered by a myelin sheath, which speeds up 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 junction 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 next neuron's dendrite by occupying specialised sites on its membrane. This is where the chemical signal from one neuron is converted into an electrical signal in the next one.

2.2 Artificial Neural Networks

Certain Artificial Neural Network (ANN) concepts used in this book are presented in this section, and the different learning algorithms in ANNs are explained. We will see that some of them are based on the biological networks presented in Section 2.1. Ever since their inception, studies on ANNs have been motivated by an understanding that the brain computes in an entirely different way from the conventional digital computer.

The brain makes up for the relatively slow operation of its neurons (nerve cells) by having a truly staggering number of , 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, which makes it an enormously complex structure.

The brain is a highly complex, non-linear, parallel computer (information-processing system). It has the capability of organising 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. recognising a familiar face embedded in an unfamiliar scene- in something of the order of 100-200 ms while a conventional computer would take much longer to perform a similar task.

An ANN 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 inter-neuron connection strengths, known as synaptic weights, are employed to store this knowledge. The rule which modify the synaptic weights of a network in an orderly way to reach a desired objective is referred to as a learning algorithm.

2.2.1 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 is defined by characterising a learning problem, rather than by characterising learning methods. This type of learning is concerned with how an agent ought to take actions in an environment so as 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.

The elements required in supervised learning are:

- the input
- the network's internal dynamics
- an evaluation of the target.

Supervised learning algorithms use these three elements to set the weights of the model -in short, to train the model- before using it for its final purpose. Typically, ANNs using supervised learning use error descent to modify weights.

On the other hand, for unsupervised learning only the first two elements are required. No external mechanism is used to check on the weight setting mechanism.

2.2.2 Reinforcement Learning

Reinforcement learning (Sutton and Barto, 2005) involves learning how to map situations to actions so as to maximise 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 behaviour 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.

2.2.3 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 (Kosko, 1987) 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: auto-associative and hetero-associative. Auto-associative memories recall the same pattern y as the input x, that is $x = y$. In hetero-associative memories the recall pattern is different from the input, $x \neq y$. Clearly, an association between patterns is stored in this case. Auto-associative 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 the target outputs of which are known. Thus the training set must already include the answer to the problem that is presented to the network.

For auto-associative supervised networks, the input data is presented to the input neurons; it is propagated 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.

2.2.4 Unsupervised Learning

Human beings seem 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 methods that are more plausible from a biological perspective than those using error descent methods. For example, such algorithms involve local processing at each synapse and it is not necessary for global information to pass through the network. So an unsupervised network must self-organise with respect to its internal parameters, without external prompting, and to do so, it must react to some aspect of the input data. Typically this will be either redundancy in the input data or clusters in the data; i.e. there must be some structure in the data to which it can respond.

There are two major methods of unsupervised learning:

- Hebbian learning.
- Competitive learning.

which are described in the following two sections (2.3 and 2.5).

2.3 Hebbian Learning

Some of the methods discussed in this book are based on Hebbian learning. Hebbian learning is named after Donald Hebb (Hebb, 1949) 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 (Haykin, 1999) 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 could be interpreted as the weight between an input neuron and an output neuron, which is greatly 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 favours a strong reaction: if the weights between inputs and outputs are already large -and so an input will have a strong effect on the outputs-, the eventuality that the weights will increase is also large. The architecture of the feed-forward network is shown in Figure 2.3.

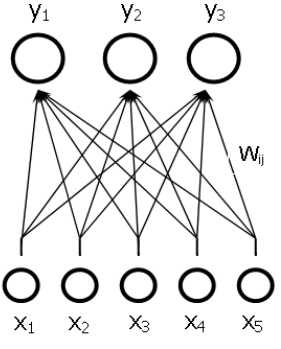


Fig. 2.3 Basic architecture of a feed-forward ANN

In mathematical terms, if we consider the simplest feed-forward neural network which has a set of input neurons with associated input vector, $\mathbf{x}$, and a set of output neurons with associated output vector, $\mathbf{y}$, we have the expression:

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

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

The Hebbian learning rule is defined by:

$$\triangle W_{ij} = \eta(x_j y_i) \tag{2.2}$$

where η is the learning rate parameter. In other words, the weight between each input and output neuron increases in proportion to the magnitude of the simultaneous firing of these neurons. Now, we can introduce the value of $\mathbf{y}$ into the learning rule, calculated by the feed-forward step of the activity to get:

$$\triangle W_{ij} = \eta x_j \sum_k W_{ik} x_k = \eta \sum_k W_{ik} x_k x_j \tag{2.3}$$

The statistical properties of the learning rule are emphasised in the last expression. This may now be seen how the learning rule depends on the correlation between different parts of the inputs data's vector components.

The basic rule, as it stands, gives us a positive feedback rule which has an associated difficulty due to lack of stability: if the i^{th} input and j^{th} output neurons tend to fire strongly together, the weight between them will tend to grow more strongly; if the weight grows strongly, the j^{th}output will fire more strongly the next time the i^{th} input neuron fires, which will lead to a higher value for the rate of change of the weights. So it is necessary to take preventive measures in order to curtail unrestricted growth of the weights. These preventive measures can include:

1. Bounding the weights i.e. defining a range of values $[w_{min}, w_{max}]$ within which the weights must remain.
2. Normalising the weights after each update, which ensures that the maximum value of the weight for each output neuron is equal to 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.

2.4 Hebbian Learning and Statistics

This section outlines several statistical methods and related neural algorithms and coding techniques. It is not meant as an exhaustive list, but to list methods that are used in this project. Firstly, the well-known Principal Component Analysis (PCA) technique (Pearson, 1901; Hotelling, 1933) is described followed by an explanation of three neural implementations of this method.

2.4.1 Principal Component Analysis

This section explains the nature of PCA by referring to several definitions of its various aspects, and discusses its strong and weak points.

PCA originated in work by (Pearson, 1901), and independently by (Hotelling, 1933) to describe the variations 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 Figure 2.4. 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 it is possible to characterise the data with a few variables, it does not follow that it is possible to assign an interpretation to these new variables.

An important problem when analysing high-dimensional data is the identification of patterns that cut across dimensional boundaries. Such patterns may become visible if the basis of the space is changed, however an *a priori* decision as to which basis will reveal most patterns requires prior knowledge of the unknown patterns. PCA offers a potential solution to this problem.

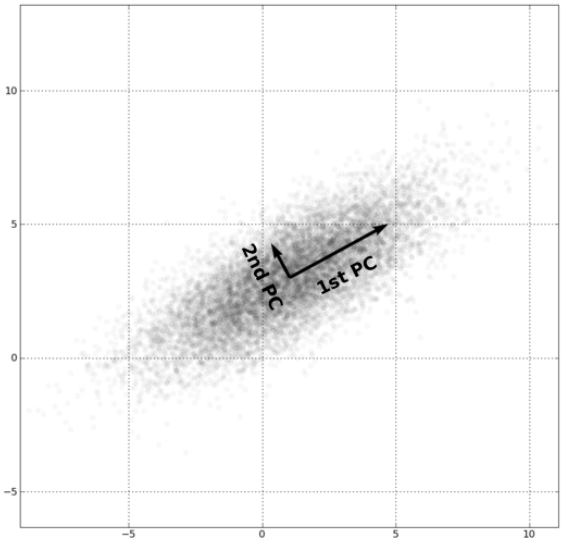


Fig. 2.4 PCA of a multivariate Gaussian distribution centred at [1,3]

It aims to find the orthogonal basis that maximises the data's variance for a given basis dimensionality. The usual method is to find the direction which accounts for most of the data's variance -the first basis vector (the first Principal Component direction)- and then to find the direction that accounts for most of the remaining variance -the second basis vector- and so on. Projecting data onto the Principal Component directions, we perform a dimensionality reduction that is accompanied by the retention of as much variance (or information) in the data as possible. This dimensionality reduction can be considered of great use in data mining applications, as the processing of low dimensional data is always considered less draining on computational resources.

The basis vectors of this new co-ordinate system can be shown to be the eigenvectors of the covariance matrix of the data set 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.

Using the analysis proposed by (Bishop, 1995), it is possible to describe this operation as mapping vectors $\mathbf{x^d}$ in an N-dimensional space onto vectors in an M-dimensional space $(x_1 ... x_N)$, where $M \leq N$. x may be represented as a linear combination of a set of N orthonormal vectors W_i:

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

The vectors satisfy the orthonormality relation:

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

where δ is the Kronecker delta.

Making use of the Eq. 2.5, the coefficients y_i may be given by:

$$y_i = W_i^T x \tag{2.6}$$

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

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

Consider the whole data set of D vectors $\mathbf{x^d}$, where $d = 1, \ldots, D$. It is necessary to make the best choice of and so that values obtained by Eq. 2.6

give the best approximation of Eq. 2.7 over the whole data set. The vector $\mathbf{x^d}$ has an error due to the dimensionality reduction:

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

The best approximation can be defined as that which minimises the sum of the squares of the errors over the whole data set

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

Calculating the derivative of with respect to and set it to zero, then:

$$bi = \frac{1}{D} \sum_{d=1}^{D} y_i^d = W_i^T \bar{x} \quad \forall i \in M+1 ... N \tag{2.10}$$

where the sample mean vector $\bar{\mathbf{x}}$ is defined as:

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

Now, the sum of squares error can be written as:

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

where Σ is the sample covariance matrix of the set of vectors and is given by:

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

Then, minimising E_M with respect to the choice of W_i, it can be shown (Bishop, 1995) that the minimum occurs when the basis vectors satisfy

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

So, those W_i are the eigenvectors of the covariance matrix. If the covariance matrix is real and symmetric, it can be proved to have orthogonal eigenvectors, which is assumed. Substituting the 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 expressed as:

$$E_M = \frac{1}{2} \sum_{i=1}^{M} \lambda i \tag{2.15}$$

Then the minimum error is obtained by choosing the $(N-M)$ smallest eigenvalues, and their corresponding eigenvectors, as the ones to discard. These $\mathbf{y}$'s are usually called the principal components.

2.4.2 Oja's Models

Oja made significant contributions (Oja, 1982, 1989) during the resurgence of research into ANNs 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. The way in which Oja's models deal with this problem in a way that also gives them important information extraction properties.

2.4.2.1 Oja's Weighted Subspace Algorithm

Oja developed his previous work (Oja, 1992) in (Oja, 1989) to now identify the actual principal components using the Weighted Subspace Algorithm. It recognised the importance of introducing asymmetry into the weight decay process in order to force weights to converge towards the Principal Components. The model is defined by the equations:

$$y_i = \sum_{i=1}^{N} W_{ij} x_j \tag{2.16}$$

where a Hebb-type rule with decay modifies the weights according to

$$\triangle W_{ij} = \eta y_i (x_j - \Theta_i \sum_{k=1}^{M} y_k W_{kj}) \tag{2.17}$$

Ensuring that $\Theta_1 < \Theta_2 < \Theta_3 ...$ allows the neuron whose weight decays proportionally to Θ_1 -i.e. whose weight decays least quickly- to learn the principal values of the correlation in the input data. As a consequence, this neuron will respond maximally to directions that run 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 of the outputs in the network. It can be shown that the weight vectors will converge to the principal eigenvectors in the order of their eigenvalues.

2.4.3 Negative Feedback Network

The Negative Feedback Network (Fyfe, 1993) is introduced in this section and reference will be made to it in subsequent sections.

Consider an N-dimensional input vector, $\mathbf{x}$, and a M-dimensional output vector, $\mathbf{y}$, with W_{ij} being the weight linking j^{th} input to i^{th} output, 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 fed forward via weights from the input neurons (the y-values) to the output neurons (the x-values) where a linear summation is performed to give the activation of the output neuron. This can be expressed as:

$$y_j = \sum_{j=1}^{N} W_{ij} y_j, \ \forall i \tag{2.18}$$

The activation is fed back through the same weights and subtracted from the inputs -where the inhibition takes place-:

$$e_j = x_j - \sum W_{ij} y_i, \ \forall j \tag{2.19}$$

After that simple Hebbian learning is performed between input and outputs:

$$\Delta W_{ij} = \eta e_j y_i \tag{2.20}$$

The effect of the negative feedback is to stabilise network learning. Thus, it is not necessary to normalise or clip the weights to achieve convergence towards a stable solution. Note that this algorithm is clearly equivalent to Oja's Subspace Algorithm (Oja, 1989) since

$$\Delta W_{ij} = \eta e_j y_i = \eta (x_j - \sum_k W_{kj} y_k) y_i \tag{2.21}$$

This network is capable of finding the principal components of the input data (Charles and Fyfe, 1998) in an equivalent way to Oja's Subspace algorithm (Oja, 1989), and so the weights will not find the Principal Components but a basis of the Subspace spanned by these components. The architecture of the typical ANN using negative feedback is shown in Figure 2.5. The difference with the feed-forward model (Figure 2.3) is that the network's inputs ($\mathbf{x}$) can be modified.

The network uses simple Hebbian learning to enable the weights to converge in order to extract the maximum information content from the input data. Writing the algorithm in this way provides a model of the process which allows different versions and algorithms to be devised such as the Maximum Likelihood Hebbian learning rule (Corchado et al, 2004), based on an explicit view of the residual $(x - Wy)$ which is never independently calculated when, for example, using 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.

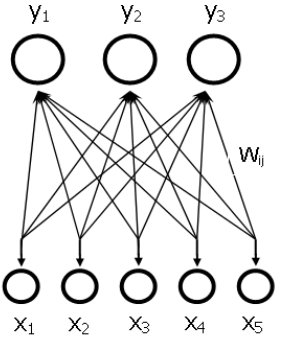


Fig. 2.5 Basic architecture of a negative feedback network

2.5 Competitive Learning

In this kind of learning, the output neurons of a neural network compete among themselves to become the active (firing) neuron. This also mirrors the reality of what takes place in the brain, in that there are finite resources for learning and so one neuron's gain is another neuron's loss. This is the biggest difference with Hebbian learning in which several output neurons may be simultaneously active; 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. Rumelhart and Zipser (Rumelhart and Zipser, 1985) claim that there are three basic elements to a competitive learning rule:

1. A set of neurons that are identical 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 mechanism that permits these 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 any one 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.

2.5.1 The Self-Organizing Map

The Self-Organizing Map (SOM) (Kohonen et al, 1977; Kohonen, 1995) is the most widely used of all the neural network models referred to as

Topology-Preserving Maps. All these models share the same objective: to generate a low dimensional representation of the training samples while preserving the topological properties of the input space. On account of this characteristic, their main use is for the visualisation and clustering of data. In most cases, high-dimensional data is analysed, thus any data-mining techniques that use the data space are 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 Figure 2.6).

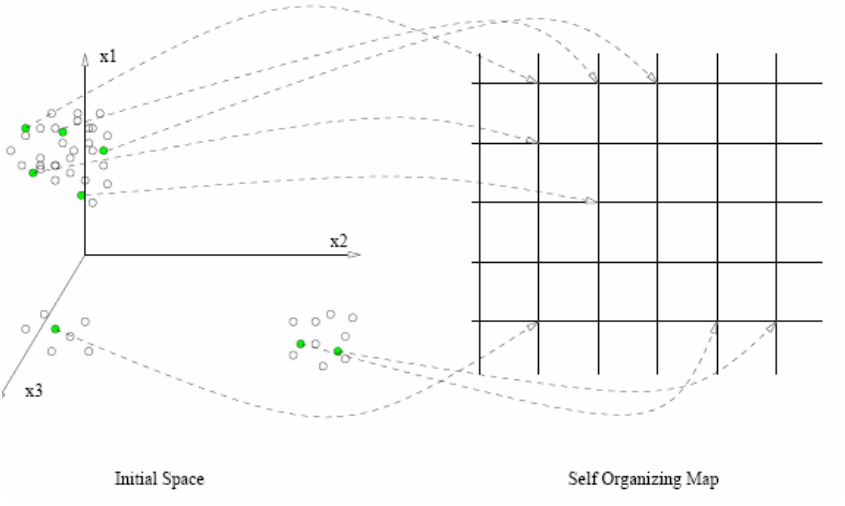


Fig. 2.6 Conceptual diagram of a 2D-SOM representation of 3D data set

Each unit j has an associated D-dimensional characteristics vector $w_j = [w_{j1}, ..., w_{jd}]$. 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 nearby objects in topological terms -map units and neurons- remain close to each other.

The learning process, by which the network neurons adapt to the data is an iterative process. Several operations have to be performed at each training step t. The first operation is to select an entry at random from the data set that is analysed, consisting of a d-dimensional vector x_i, which 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 the

winning neuron, which will be called the Best Matching Unit (BMU). This situation is expressed mathematically in Eq. 2.22:

$$w_v = arg\, min_k \{\|x_i - w_k(t)\|\} \tag{2.22}$$

Subsequently, the characteristics vectors for the BMU and its neighbourhood are "moved" towards the presented input, to reinforce the similarity between the BMU -and its neighbourhood- and the inputs. Thus, a neuron specialises in recognising input patterns that are similar to the one that is presented. The most characteristic element of SOM learning is the use of a neighbourhood function, which enables the BMU to update its vector to the input and also enables neighbouring neurons to update in direct proportion to their distance from the BMU in the map lattice. This neuron update process in the SOM is presented graphically in Figure 2.7.

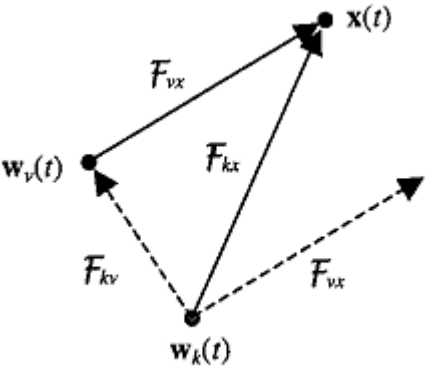


Fig. 2.7 Updating of the characteristics vectors of SOM neurons
Figure taken from (Yin, 2002b)

Neuron updating in the SOM can be represented as:

$$w_k(t+1) = w_k(t) + \alpha(t)\eta(v,k,t)(x(t) - w_k(t)) \tag{2.23}$$

where, x denotes the network input, w_k the characteristics vector of each neuron; α, is the learning rate of the algorithm; and $\eta(v,k,t)$ is the neighbourhood function, in which v represents the position of the winning neuron (BMU) in the lattice, and k the positions of the neurons in its neighbourhood. The most common function is a Gaussian function centred on the position of the BMU; although other functions, such as the difference of Gaussians, are also widely used.

The characteristic Gaussian function may be defined as:

$$\eta(v,k,t) = e^{\frac{\|_{w_v - k}\|}{2\sigma^2(t)}} \tag{2.24}$$

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

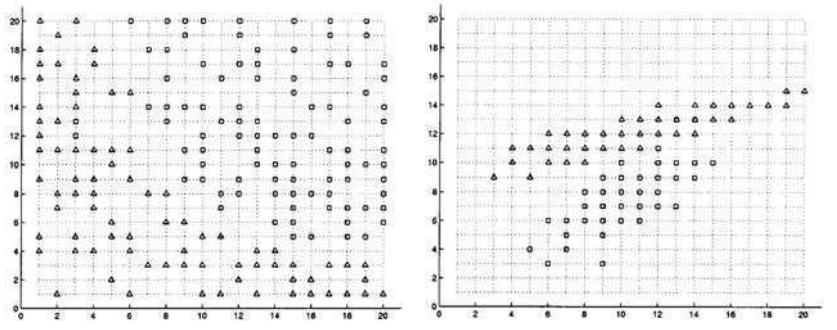
Furthermore, an additional step can be performed: neighbourhood learning of randomly chosen neurons can be reinforced by using those neuron's weights as the input for a small percentage of the updating times -e.g., 10% iterations-, which avoids having to have empty spaces in the map where neurons have not reacted to any data. As a result of the leaning 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 the input space are preserved in U to the highest possible degree. By taking account of the BMU as well as its neighbouring neurons (2.23) when updating; nearby neurons gradually specialise to represent similar inputs, and the representations become ordered on the map lattice. This topological ordering of data into a 2D map is one of the main features of the SOM algorithm.

2.5.2 The Visually Induced SOM

Neighbourhood learning is adopted in the SOM to retain the topological order of the neurons in the map. Thus, the map can be used to show the relative relationships between data points. However, the SOM does not directly show the inter-neuron distances on the map. For visualisation, the SOM requires the assistance of a colouring scheme to print the inter-neuron distances, which therefore allows the clusters and boundaries to 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 neighbouring neurons should be approximately the same and the distances between a neuron and its furthest neighbouring neurons should increase proportionally and regularly according to the structure of the map grid. If so, the positions of the neurons can serve as grades for measuring the distance of any mapped points. The map will then appear as a smooth, graded mesh embedded in the data space, onto which the data points are mapped and the inter-point distances are approximately preserved.

Like the SOM, the ViSOM (Yin, 2002b,a) projects high-dimensional data in an unsupervised manner, but it constrains the lateral contraction force and hence regularises 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. Figure 2.8 presents the difference in the representation obtained by a SOM and a Vi-SOM trained over the same data set. In this case the well-konwn Iris data set is represented by a SOM and a ViSOM, both of size 20x20.



(a) SOM data visualisation **(b)** ViSOM data visualisation

Fig. 2.8 Comparison of the representation of the Iris data set by a SOM and a ViSOM

Figure taken from (Yin, 2002b)

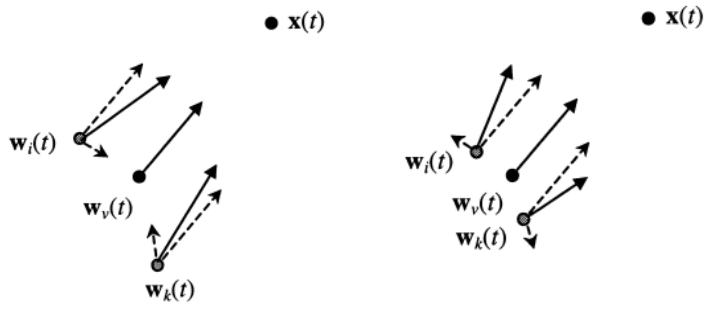
Like the SOM, the ViSOM uses a neuronal grid structure and they both basically use the same training algorithm. The difference between both algorithms lies in the way the weights of its composing units are updated. The steps of this algorithm can be summarised as follows:

At time step t, an input $\mathbf{x(t)}$ is drawn randomly from the data set or data space. A winning neuron can be found on the basis of its distance to the input, using the same expression as the SOM (Eq. 2.22). Then, in the SOM algorithm, the weights of the neurons in a neighbourhood of the winner are updated by Eq. 2.23 (see also Fig.2.6). The second term in this equation $(x(t) - w_k(t))$, which can be considered as the *updating force*, can be decomposed into two different forces:

$$\mathcal{F}_{kx} \equiv x(t) - w_k(t) = [x(t) - w_v(t)] + [w_v(t) - w_k(t)] \equiv \mathcal{F}_{vx} + \mathcal{F}_{kv} \quad (2.25)$$

The first force, $\mathcal{F}_{vx}$, represents the updating force from the winner v to the input $\mathbf{x}$, which is the same as that used by the winner in (Eq. 2.23). It adapts the neurons toward the input in a direction that is orthogonal to the tangent plane of the winner. While the second force, $\mathcal{F}_{kv}$, is a lateral or contraction force bringing neuron k to the winner v. It is this contraction force that brings neurons in the neighbourhood towards 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 by regularising the distance between a neighbouring neuron to the winner. These forces are graphically represented in Figure 2.9.

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



(a) Contraction force between the BMU and two neighbouring neurons

(b) Expansion force between the BMU and two neighbouring neurons

Fig. 2.9 Contraction or Expansion force for the updating of the ViSOM neurons Figure taken from (Yin, 2002a)

The scale of the force is controlled by the normalised 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 as:

$$w_k(t+1) = w_k(t) + \alpha(t)\eta(v,k,t)\left[(x(t) - w_v(t)) + (w_v(t) - w_k(t))\frac{d_{vk} - \triangle_{vk}\lambda}{\triangle_{vk}\lambda}\right] \tag{2.26}$$

where, d_{vk} and $\triangle_{vk}$ are the distances between neurons 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 neighbouring nodes- reflected in the input space.

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.

2.5.3 The Scale Invariant Map

The Scale Invariant Map (SIM) (Fyfe, 1996) is a regular array of nodes arranged on a lattice, similar to a Self-Organising Map (SOM) (Kohonen, 1988), although it uses a training method based on a negative feedback network. A neighbourhood function and competitive learning are used in the same way as with the SOM. The input data is fed forward to the outputs in the usual way. Following its selection, 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 fed back 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 neighbourhood of the winner.

Training on a SOM relies on iteratively selecting a winner stimulated by the inputs, and updating the weights. With the SIM, the weights of the winning node are fed back as inhibition to the inputs -similar to the process detailed in Section 2.4.3-, and simple Hebbian learning is then used to update the weights of all nodes in the neighbourhood of the winner.

$$e = x - W_c y_c, \ (y_c = 1) \tag{2.27}$$

$$\triangle W_i = h_{ci} \eta e, \ \forall i \in N_c \tag{2.28}$$

This has the effect of updating all weight vectors in parallel to the vector $x - W_c$, which is made clear by rewriting Eq. 2.28 as:

$$\triangle W_i = h_{ci} \eta (x - W_c), \ \forall i \in N_c \tag{2.29}$$

In the negative feedback from a winning neuron to the input data, there are large number of residuals which are relatively small in magnitude and a much larger number of residuals which are much greater in magnitude. This is more likely to be approximated by an exponential distribution than by a Gaussian distribution.

The final effect of this type of feedback learning is that a pie-slice of data is actually won by each neuron. This algorithm is called a scale-invariant feature map because it ignores the magnitude of each input vector, and responds solely to the relative proportions of the magnitudes of each input vector element. The way the SIM matches the data in a "pie-slice" manner, is shown in Figure 2.10.

It is well known that, given similar input data to those used above, a one-dimensional self-organizing map (SOM) will self-organize to spread itself over the square to minimise the expected distance between the code points and

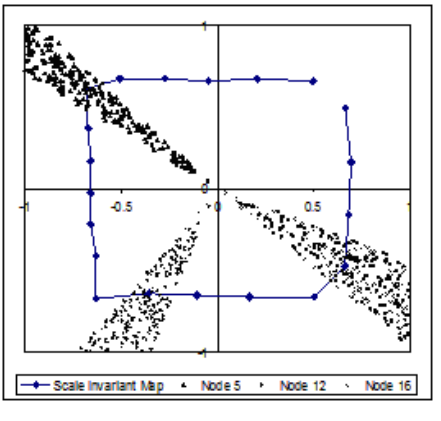


Fig. 2.10 A SIM trained on uniformly distributed data

the points of the square. However, if this features map increases the learning rate, an interesting effect comes into play: the mapping winds round 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 Figure 2.11. Note that those weights which continue to occupy space within the outer ring of neurons do not win any competitions. They can be thought of as backups for those neurons which are winning competitions: a substitute is ready to replace one of the winners should it fail.

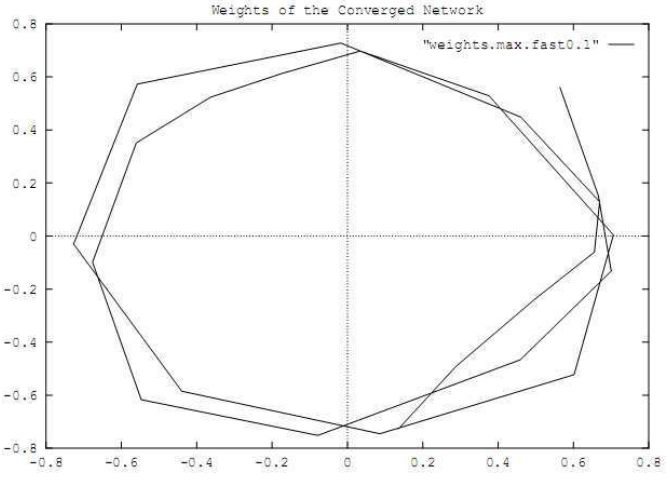


Fig. 2.11 Results for the SIM when the learning rate is increased

2.5.3.1 The Maximum Likelihood Scale Invariant Map

Exploratory Projection Pursuit (EPP) (Friedman and Tukey, 1974; Friedman, 1987) is a more recent statistical method than PCA aimed at solving the difficult problem of identifying structure in high dimensional data. It does this by projecting the data onto a low dimensional subspace in which we search for its structure by eye. However not all projections will reveal the structure of the data equally well. An index measures the degree of "interest" of a given projection, and the data is then represented in terms of projections that maximise that index.

"Interesting" structures are usually defined in terms of near-Gaussian distributions, as most projections of high-dimensional data onto arbitrary lines through most multi-dimensional data yield this type of distribution (Diaconis and Freedman, 1984). Therefore to identify "interesting" features in the data, it is possible to look for those directions onto which the data-projections are as far from the Gaussian as possible. (Corchado et al, 2004)

presents a neural method of performing EPP from a probabilistic perspective called "Maximum likelihood".

A formula is derived to replace the classical Hebbian rule in an ANN (Section 2.3) to perform this kind of EPP:

$$\triangle W_{ij} = \eta \cdot y_i sign(e_j)|e_j|^{p-1} \tag{2.30}$$

Applying the Maximum Likelihood Hebbian Learning (MLHL) to work in the SIM algorithm is a very straightforward procedure.

It can be done by simply modifying the weight update of all nodes in the neighbourhood of the winner, which is expressed as in Eq. 2.31.

$$\triangle W_i = h_{ci} \cdot \eta \cdot sign(e - W_c)|e - W_c|^{p-1}, \quad \forall i \in Nc \tag{2.31}$$

By giving different values to the parameter p (Corchado and Fyfe, 2002a), the learning rule is optimal for different probability density functions of the residuals. h_{ci} is the neighbourhood function as in the case of the SOM and N_c is the number of output neurons. Finally, η represents the learning rate.

This new algorithm is called Maximum Likelihood Scale Invariant Map -or Max-SIM for short- (Corchado and Fyfe, 2002b) and will be used in following chapters.

2.5.4 Assessing Quality of Training of Topology Preserving Models

In the case of supervised learning, having established the collection of desired outputs for the ANN inputs usually implies that a certain deviation from the ANN's learning state and the desired output can be calculated quite easily. When dealing with unsupervised learning, the process only depends on the inputs and the dynamics of the learning rule; which makes it much more difficult to determine the accuracy of the training in relation to the input data set. Such a measure would be relevant, not only for a 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 is no general canonical measure to determine the quality of the training of unsupervised learning algorithms, which also includes Self-Organizing Maps.

In the case of the SOM, several quality measures have been proposed in the literature (Polani, 2003; Pozlbauer, 2004), which examine different characteristics of the model. For a better understanding of the results presented in this book, a list of the most interesting measures accompanied by an explanation of each one is included in the following subsections.

2.5.4.1 Quantization Error

Quantization Error is related to all forms of vector quantization and clustering algorithms. Thus, this measure completely disregards map topology and alignment. It requires a test data set to be calculated. Quantization error is computed by determining the average distance of the test data set entries to the cluster centroids by which they are represented. In case of the SOM, the cluster centroids are the characteristics vectors. Its mathematical expression is:

$$E_Q = \frac{1}{|D|} \sum_{x_i \in D} \|x_i - w_i\|^2 \tag{2.32}$$

where $|D|$ is the number of entries in the data set used D, and w_i the BMU for each presented input x_i.

2.5.4.2 Topographic Error

Topographic Error (Kiviluoto, 1996) is the most simple of the topology-preservation measures. A data set is also needed to calculate this measure. For all data samples, the respective best and second-best matching units -1^{st} BMU and 2^{nd} BMU- are determined. If these are not adjacent on the map lattice, this is considered an error. The total error is then normalized to a range from 0 to 1, where 0 means perfect topology preservation. Its mathematical expression is as follows:

$$E_T = \frac{1}{|D|} \sum_{x_i \in D} u(x_i) \tag{2.33}$$

where $|D|$ is the number of entries in the data set used D, and the value of function u is either 1, if the first and second BMU for input entry $\mathbf{x_i}$ are adjacent in the map grid, or 0 in any other case.

Usually, a single value is returned that quantifies this property. It is however possible to decompose the Topographic Error such that they can be visualized on a map lattice. This can be done, for example, by raising the error for a unit every time it is selected as the 1^{st} BMU by a data sample, and the 2^{nd} BMU is not adjacent in the output space.

2.5.4.3 Topographic Product

One of the main uses of non-linear mapping methods is for visualization of high-dimensional data. In such visualizations, it is crucial that data proximities are trustworthy. This measure (Bauer and Pawelzik, 1992) attempts to determine the degree to which that is true; i.e. if two data samples are represented as nearby (or related) on the map, then to what extent is the same also true for the input space.

Computation of the Topographic Product only involves the map's characteristics vectors. Its three-part mathematical expression -which are called P_3 and P in the original publication- is as follows:

$$P_3(j,k) = \left(\prod_{l=1}^{k} Q_2(j,l) Q_2(j,l) \right)^{1/k} \tag{2.34}$$

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} log(P_3(j,k)) \tag{2.35}$$

In which, $Q_1(j,k) = \frac{d^V(w_j, w_{n_k^A(j)})}{d^V(w_j, w_{n_k^V(j)})}$ and $Q_2(j,k) = \frac{d^A(w_j, w_{n_k^A(j)})}{d^V(w_j, w_{n_k^V(j)})}$ considering $n_k^A(j)$ the k^{th} nearest neighbour of the neuron j with distances measured in the output space, and $n_k^V(j)$ as the k^{th} nearest neighbour of the neuron j with distances measured in the input space, between w_j and $w_{n_k^V(j)}$.

2.5.4.4 Distortion

As explained in (Lampinen and Oja, 1992), there is a function that the algorithm optimizes, when using a constant radius for the neighbourhood function of the learning phase of a SOM. This function, called Distortion measure, can be used to measure the overall topology preservation of a map in a more detailed way than Topographic Error. Also a test data set is needed for its calculation.

$$E_D = \sum_{x_i \in D} \sum_{w_k \in W} \eta(v_i, k) \left\| x_i - w_k \right\|^2 \tag{2.36}$$

where $\mathbf{x}_i$ represents each entry of the data set;$\eta(v_i, k)$ represents the neighbourhood function between the BMU and every other neuron in the map; (v_i the position of the BMU corresponding to $\mathbf{x}_i$; k the position of each other neuron in the network; and $\mathbf{w_k}$ being each of the characteristics vector that compose the network (W). Further discussion can be found in (Vesanto et al, 2003).

2.5.4.5 Goodness of Map

This measure, described in (Kaski and Lagus, 1996) combines two of the previous error measures: the square quantization error and the topographic error. It takes into account both the distance between the input and the BMU and the distance between the first BMU and the second BMU on the shortest path between both neurons along the grid map, calculated by only taking into account only neurons that are direct neighbours in the map for each input; measuring both the continuity of the mapping from the data set

to the map grid, and the accuracy of the map in representing the set. The mathematical expression of this measure for each data entry is shown below:

$$d(x_i) = \|x_i - v_i\| + min \sum_{k=0}^{|K_{v'_i}|-1} \left\| w_{I_{i(k)}} - w_{I_{i(k+1)}} \right\| \tag{2.37}$$

where, v_i and v'_i represent the weights of the first BMU and the second BMU respectively, corresponding to data entry x_i. $I_{i(k)}$ and $I_{i(k+1)}$ represent indexes of the k^{th} and the k^{th+1} neurons along the minimum path from v_i to v'_i, both neurons being direct neighbours in the map grid. According to that definition $w_{I_i(0)} = v_i$; which is to say, the first neuron in the path is the first BMU for data entry $\mathbf{x}_i$ and $w_{I_i(k_{v'_i})} = v'_i$, which is to say, the last neuron in the path that corresponds to the second BMU for data entry $\mathbf{x}_i$. The final goodness of the map is defined as the average of the values obtained by Eq. 2.37 for all data entries in the test data set.

2.6 Conclusions

This section has introduced the basic concepts of ANNs, concentrating particularly on unsupervised networks, which will be investigated in greater depth further on this book, and has remarked on the main differences with the supervised ones. Special emphasis is placed on Hebbian Learning and Competitive Learning; the two types of unsupervised learning on which this book is centred. Accordingly, Oja's model and negative feedback algorithms are discussed, which are algorithms that make use of the first type of learning, and the Self-Organizing and its derivations, which are the main representatives of the second type of learning.

Chapter 3
The Committee of Experts Approach: Ensemble Learning

3.1 The Ensemble Meta-algorithm

The main concept behind ensemble learning model is the simple intuitive idea of a committee of experts working together to solve a problem.

In all likelihood, when dealing with a complicated problem, a group of experts with varied experience in the same area will have a higher probability of reaching a satisfactory solution than a single expert. All members contribute their own experience and initiatives and the group as a whole can choose to uphold or to reject a new idea on its own merits.

In the field of 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 performance -classification, prediction, function approximation, etc.- of a model, or reduce the likelihood of an unfortunate selection of a poor one.

3.1.1 The Classification Problem

At its inception, the ensemble meta-algorithm was created to improve the capabilities of existing models for data classification. In this section, the problem is formally presented, to provide a fuller understanding of ensembles and their use.

Statistical classification is a procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items -referred to as traits, variables, characters,...etc.- and based on a training set of previously labelled items.

In a typical classification algorithm supervised training, the data set consists of a series of vectors of features (denoted x), each of which is considered as a data entry.

Each entry has also an associated class label (y), so each entry belongs to a separate class within the data set. It is assumed that there exits some

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 31–47.
springerlink.com

underlying function f, such that $y = f(x)$ for each data entry (x, y). The goal of the classification algorithm is to find a good approximation of h to f, which can be applied to assign labels (to classify) new input data entries that were not previously presented to the classifier.

As pointed out by (Dietterich, 2000), classification algorithms that output a single hypobook suffer from three problems that can be partially overcome by the use of ensemble methods:

- *Statistical problem:* this problem arises when the hypobook space that the algorithm is trying to learn is too large for it to be correctly represented by the amount of data entries available for training. In those cases, there might be a series of very different hypobook that obtain the same accuracy on the training data, which turns the choice of training data into a random decision. There is a risk that the chosen hypobook might work well with available data, but that its accuracy might suffer when classifying future data. An algorithm suffering from this problem it is said to have high 'variance'.
- *Computational problem:* this problem appears when it can not be guaranteed that the learning algorithm will find the best hypobook within the hypobook space. In ANNs or the so called decision trees (Breiman et al, 1984), finding the hypobook that best fits the data is computationally too expensive, so heuristics are used instead. These heuristics can get trapped by local minima and hence fail to find the best hypobook. Algorithms that suffer from this problem are sometimes described as having high 'computational variance'.
- *Representation problem:* this last problem arises when the hypobook space does not contain a hypobook that is a good approximation of the real underlying function f . Algorithms suffering from this problem are said to have high 'bias'.

3.1.2 Ensemble General Concepts

As can be inferred from Section 3.1.1, building a highly accurate classification rule is certainly a difficult task. On the other hand, it is hardly a difficult task to come up with very rough rules of thumb that are only moderately accurate. Ensemble Learning, is based on the observation that finding many rough "loose" rules for classification can be a lot easier than finding a single, highly accurate prediction rule. The ensemble approach begins with a method or algorithm to discover the rough rules of thumb. The ensemble meta-algorithm repeatedly applies this "weak" or "base" learning algorithm, feeding it each time with a different subset of the training examples. Each time it is called, the base learning algorithm generates a new weak classification rule. After repeated rounds the boosting algorithm must combine these weak rules into a single prediction rule that will hopefully be much more accurate than any one of the weak rules.

Ensemble algorithms are probably an effective method of producing a very accurate classification rule by combining rough and moderately inaccurate rules of thumb.

The ensemble meta-algorithm intuitively helps to reduce the problems discussed in Section 3.1.1. The problem of being unable to generalize well when presented with new data, due to having chosen the wrong hypobook (statistical problem) appears less serious when several different hypobook are used. The problem of the learning algorithm stuck in local minima when using heuristics (computational problem) is softened by the use of several models trained in a similar, although not in an exactly equal way, so that each of them will face different local minima, as they are trained with different data sets. And finally, it appears easier to obtain a final hypobook to solve a complex problem by dividing the problem into several simple ones and tackling each of them individually (representational problem). In a final stage, the hypotheses of each classifier may be combined to obtain the final solution.

There is considerable evidence to suggest that the use of ensembles can lead to an improvement in the performance of single models in classification or regression tasks (Breiman, 1996; Freund and Schapire, 1996; Schapire, 1990; Kuncheva et al, 2002). The underlying reason for increased reliability through the use of ensembles is that different classification algorithms will show different patterns of generalization. In other words, each one will commit different errors when dealing with the same data set; hopefully only one will commit an error at any one time. Thus, in the combined output of the ensemble, individual errors will be compensated by correct responses from the rest of the ensemble members. More formal explanations of the way ensembles can improve performance may be found in (Sharkey and Sharkey, 1997; Heskes, 1997; Ruta and Gabrys, 2002; Kuncheva, 2004). As is widely accepted, in (Geman et al, 1992) this improvement potential is explained in terms of two concepts called 'bias' and 'variance'. Simplifying the explanations in that work, variance can be considered as representing the extent to which the output of a classifier is sensitive to the data set in which it is trained; in other words, the extent to which, when trained with one data set as its input, the classifier will yield outputs that are as accurate as those it would generate were it trained with another data set. On the other hand, bias refers to the capacity of the classifier to generalize correctly when presented with a test set that is different from the one that is used for its training, as an average over the set of possible training sets.

The strength of the ensemble meta-algorithm is its potential to achieve a compromise between the desired result of both a small variance and a small bias; as a trade off between fitting the data too closely (high variance) and not taking data into account at all (high bias). It is generally accepted (Bishop, 1995) that the effect of combining a set of classifiers, each one trained on a different subset of a general data set, represents a decrease in variance, without affecting the bias. Thus, the best strategy would be to concentrate on obtaining a series of classifiers with low bias in their own subsets and then combine

them to compensate for the variance that arises when calculating the final output. An important element is the effective combination of the classifiers, which relies in part on the presence of a certain variance in the components of the ensemble that is generally referred as 'diversity'. Obviously, there would be no advantage in having an ensemble of classifiers generalizing in exactly the same way. What is needed is a set of classifiers, each of which generalize the problem well, leading to a small amount of errors. The key point being that the other components do not share the error which a classifier commits, so that even though not all of the classifiers obtain the correct answer, a majority of them do. In other words, the set of classifiers in the ensemble must exhibit a certain degree of diversity as a group. A much more thorough study of diversity properties can be found in (Sharkey and Sharkey, 1997).

Ways of achieving an advantageous diversity when constructing an ensemble is the main point in these kinds of meta-algorithms, some of which are discussed below.

3.2 Commonly Used Ensemble Models

When constructing an ensemble there are two main approaches to how each of the classifiers or components of the ensemble are going to be trained to solve the problem at hand:

- ▷ Independent Training: in which each of the classifiers is trained without any knowledge of how the other components of the ensemble are trained. The most widely known algorithm of this type is known as the Bagging ensemble (Breiman, 1996) (see 3.2.1).
- ▷ Coordinated Training: in which one of the classifiers composing the ensemble is trained, taking account of how previous classifiers were trained to modify the way in which the current one performs its training, so it can overcome possible weaknesses of the other components. This is the idea behind Boosting algorithms (Schapire, 1990; Freund and Schapire, 1996) (see 3.2.2).

3.2.1 Bagging

'Boostrap aggregating' or 'bagging' for short, is one of the earliest, most intuitive and perhaps the simplest ensemble-based algorithms, the performance of which is surprisingly good (Breiman, 1996). Having a data set composed of N data entries, in each iteration of the meta-algorithm, a lower number of N' entries are randomly and uniformly selected, with replacement, from the N entries composing the original data set and grouped into what will be considered a new data set. In that way, it is possible to obtain several re-sampled data sets in which various entries from the original data set will appear -one or many

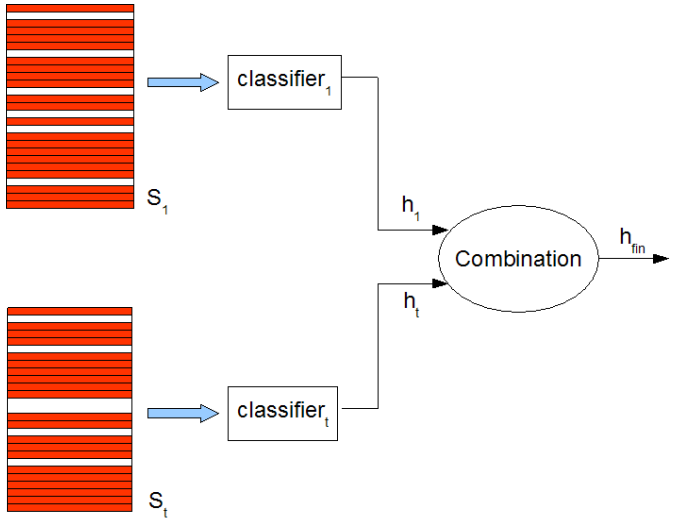


Fig. 3.1 Schematic diagram of the Bagging process

times- and other entries will not appear at all. A classifier will be trained employing only one of those sub-sets. This classifier may be considered as an unspecified learning algorithm, usually called the weak learning algorithm, which is denoted generically as *"WeakLearn"*. The training algorithm for the classifier may be considered quite unstable, which implies that minor changes in the training can lead to quite different results. This procedure will yield an ensemble of classifiers with sufficient diversity of hypobook. The schematic diagram depicting this process is shown in Figure 3.1.

In a more formal way, this idea can be explained as follows: L is the learning data set, consisting of data with the form (x_n, y_n), $n = 1 \ldots N$, where y_n represents the entry's class. Let us assume that there is a learning algorithm to train a classifier $h(x, S)$ that approximates the desired hypobook $y = h(x, S)$.

Now, let us suppose a sequence of different learning sets $\{S_t\}$, each consisting of N' independent observations from the same distribution. The idea is to obtain a better classifier from those data sets than the one obtained by a single learning set, by using the sequence of classifiers trained over the data sets $\{h(x, S_t)\}$.

If y is a numerical value, the most straightforward approximation would be to calculate the average of $\{h(x, S_t)\}$ over t. If y is a class $j \in \{1 \ldots J\}$, then one method of aggregating the $h(x, S_t)$ is by voting. If $N_j = nr\,\{k\varphi(x, L_k) = j\}$ then this can be done by taking $h_{fin}(x) = argmax_j N_j$ that is, the j for which the maximum is N_j.

Usually only one data set S is available, without further replications. In this case, a replication of the process leading to the $h_{fin}(x)$ can be achieved

Algorithm 1. Bagging

Input: number of classifiers required: M
Output: final hypobook

```
1: Select a training set S = ⟨(x_1, y_1) ... (x_m, y_m)⟩
2: procedure TRAIN ENSEMBLE(S_1...S_m)
3:     for t = 1 to T do
4:         Construct a training set S_t ▷ sampling N' items randomly from S
5:         Call WeakLearn, with training set S_t
6:         Get back hypobook h_t: X → Y
7:     end for
8: end procedure
```

For any new entry **x**, the bagged output will be calculated as:

▷ $h_{fin}(x) = \frac{1}{M}\sum_t h_t(x)$ if the expected output is a numerical prediction
▷ $h_{fin}(x) = argmax_j N_j$ if the output is a discrete class.

by repeatedly obtaining several bootstrap samples of the original data set S and placing them in separated data sets $\{S_t\}$.

The pseudo-code detailing how the bagging meta-algorithm works is presented in Algorithm 1.

The mathematical explanation of why this meta-algorithm works is the following:

If we denote that over many independent replicates of the training set L, ϕ predicts class label y for input x with a relative frequency of $Q(y|x) = P(\phi(x, L) = y)$, then the overall probability that the classifier classifies the input x with the correct label is:

$$r = \int \left[\sum_y Q(y|x)P(y|x)\right] P_X(dx) \tag{3.1}$$

in which $P(y|x)$ is the probability that input x generates class y and $P_X(dx)$ the x probability distribution.

For the aggregated classifier, which output is calculated as a vote $\phi_A(x) = argmax_i Q(i|x)$, the probability of a correct classification of x is:

$$\sum_y I(argmax_i Q(i|x) = y)P(y|x) \tag{3.2}$$

where $I(\cdot)$ is the indicator function.

Call ϕ order-correct at the input x, if when input x results in class y more often than in any other class, then ϕ also yields class y more often than any other class. If we call C the set of all inputs where ϕ is order-correct, we arrive at the following expression for the correct classification probability of ϕ_A:

$$rA = \int_{x \in C} max_y P(x|y) P_X(dx) + \int_{x \in C'} \left[\sum_y I(\phi_A(x) = y) P(x|j) \right] P_X(x) \tag{3.3}$$

Even if ϕ is order-correct at x, its correct classification rate can be far from optimal, but ϕ_A is optimal. If a predictor is good in the sense that it is order-correct for most inputs x, then aggregation can transform it into a near-optimal predictor.

Conversely, poor predictors can be transformed into worse ones. There is another obvious limitation of bagging. For some data sets, it may happen that $\varphi(x, L)$ is close to the accuracy limits that the data may attain. Hence, no amount of bagging will ever improve accuracy. Bagging unstable classifiers usually improves them. Bagging stable classifiers is not a good idea.

3.2.2 Boosting

As pointed out above, all the classifiers in the Bagging algorithm (Section 3.2.1) ensemble are trained independently by manipulating the data set entries, or more formally by generating a different distribution over the training examples. Then unweighted voting between all the classifiers determines which ensemble output label will be used when classifying a new sample.

A natural evolution of this idea would be to gather information about how a distribution works when presented to a classifier and then to choose the next distribution of data so it can improve the overall classification accuracy of the ensemble, by making it "expert" in certain classification areas where the other classifiers are not performing so well.

As explained in Section 3.1.2, an ensemble would perform better when a correct balance in the diversity of its components is achieved. Therefore, boosting meta-algorithms are generally meant to place the greatest weight on the examples that are most frequently misclassified by the preceding weak rules; thereby forcing the base learner to focus its attention on the "hardest" examples. The desired diversity will be achieved by the fact that different data distributions are used, so classifiers will be trained in different ways. The main point of the boosting algorithm is that, unlike bagging, these differences in their behaviour will centre precisely on the areas where data is more difficult to classify. Hence, having different hypobook for classifying "difficult" entries will increase the probability of correctly classifying those data entries.

Intuitively, taking a weighted majority over many hypotheses, all of which were trained on different samples taken from the same training set, has the effect of reducing the random variability of the combined hypobook.

One of the first and the simplest boosting algorithms, as proposed by (Schapire, 1990), illustrates the way in which boosting works. Each iteration of boosting creates three weak classifiers: the first classifier C_1 is trained with a random subset of the available training data. The training data subset for the second classifier C_2 is chosen as the most informative subset, given C_1.

Specifically, C_2 is trained on a training data only half of which is correctly classified by C_1, and the other half is misclassified. The third classifier C_3 is trained with instances on which C_1 and C_2 disagree. The three classifiers are combined through a three-way majority vote, which means that, intuitively speaking, there is an increased probability that, when classifying a new sample, at least 2 of the 3 classifiers should agree in the correct class, so the ensemble will reach the correct classification through its voting process.

3.2.2.1 AdaBoost

Among the great variety of boosting algorithms, the most widely known might well be the "Adaptative Boosting" or "AdaBoost" proposed by Freund and Schapire in (Freund and Schapire, 1996).

This new algorithm is very nearly as efficient as the boost-by-majority explained in Section 3.2.2. However, unlike boost-by-majority, the accuracy of the final hypobook produced by the new algorithm depends on the accuracy of all the hypotheses returned by the weak classifiers composing the ensemble. It is therefore able to exploit the power of the "weak" learning algorithm more fully.

Also, this new algorithm gives a clean method for handling real-valued hypotheses which are often produced by neural networks and other learning algorithms.

In (Freund and Schapire, 1996) two different variants of the algorithm are denoted AdaBoost.M1 and AdaBoost.M2. The two versions are equivalent for binary classification problems and differ only in the way that they handle problems with more than two classes.

The main process is the same for both variants: as with the Bagging algorithm, the input of the boosting algorithm takes a training set of N examples $S = \langle (x_1, y_1) \dots (x_n, y_n) \rangle$ in which $\mathbf{x_i}$ is an instance drawn from a space X and represented in some way -typically, a vector of attribute values- , and $y_i \in Y$ is the class label associated with $\mathbf{x_i}$. The boosting algorithm calls on the services of *WeakLearn* repeatedly in a series of rounds. On each iteration t, the booster also provides WeakLearn with a distribution D_t over a training set S_t. In response, *WeakLearn* computes a classifier or hypobook h_t: $X \rightarrow Y$ which should misclassify a non-trivial fraction of the training examples, relative to D_t . Thus, the weak learner's goal is to find a hypobook h_t which minimizes the (training) error $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y]$. Note that this error is measured with respect to the distribution D_t that was provided to the weak learner. This process continues for T iterations, and, at last, the booster combines the weak hypotheses $h_1 \dots h_T$ into a single final hypobook h_{fin}.

The difference between both variants of the algorithm lies in the way in which D_t is computed in each iteration and the way the final hypobook h_{fin} calculates its outputs. The basic procedure for AdaBoost is detailed in Algorithm 2.

AdaBoost.M1: (Freund and Schapire, 1996) The initial distribution D_1 is uniform over S so $D_1(i) = 1/m$ for all i . To compute distribution D_{t+1} from

Algorithm 2. AdaBoost General Algorithm

Input: sequence of m examples $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1 \dots k\}$
weak learning algorithm *WeakLearn*
number of iterations T
Output: final hypobook

Initialize: $D_1(i) = 1/m$ for all i
Select a training set $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$
procedure TRAIN ENSEMBLE($S_1 ... S_m$)
    for $t = 1$ to T **do**
        Construct a training set S_t ▷ sampling N' items randomly from S
        Call WeakLearn, with training set S_t and distribution D_t
        Get back hypobook h_t: $X \rightarrow Y$
        Calculate the error of h_t: ϵ_t
        Set $\beta = \epsilon_t / (1 - \epsilon_t)$
        Update distribution D_t
    end for
end procedure
Output: the final hypobook:

$$h_{fin}(x)$$

D_t and the last weak hypobook h_t, has to be multiplied by the weight of example i has to be multiplied by some number $\beta_t \in [0, 1)$ if h_t classifies x_i correctly, and otherwise the weight is left unchanged. The weights are then re-normalized by dividing them by the normalization constant Z_t, chosen so that D_{t+1} would be a distribution, as shown in Equation 3.4:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & if\ h_t(x_i) = y_i \\ 1 & otherwise \end{cases} \tag{3.4}$$

In effect, "easy" examples that are correctly classified by many of the previous weak hypotheses get lower weights, and "hard" examples which often tend to be misclassified get higher weights. Thus, AdaBoost focuses most weight on the examples which appear to be hardest for WeakLearn.

The final hypobook h_{fin} is a weighted vote -i.e. a weighted linear threshold- of the weak hypotheses. That is, for a given instance x, h_{fin} outputs the label y that maximizes the sum of the weights of the weak hypotheses predicting that label. The hypobook weight is defined as:

$$h_{fin}(x) = argmax_{y \in Y} \sum_{i: h_t(x_i) = y_i} log \frac{1}{\beta_t} \tag{3.5}$$

so that greater weight is given to hypotheses with lower error.

AdaBoost.M2: (Freund and Schapire, 1996) The main disadvantage of AdaBoost.M1 is that it is unable to handle weak hypotheses with error greater than $1/2$. The expected error of a hypobook which randomly guesses the label is $1 - 1/k$, where k is the number of possible labels. Thus, the AdaBoost.M1 requirement for $k = 2$ is that the prediction is just slightly better than random guessing. However, when $k > 2$, the requirement of AdaBoost.M1 is much stronger than that, and might be hard to meet.

The second version of AdaBoost attempts to overcome this difficulty by extending communication between the boosting algorithm and the weak learner. Firstly, the weak learner is allowed to generate more expressive hypotheses the output of which is a vector in $[0, 1]^k$ rather than a single label in Y. Intuitively, they^{th} component of this vector represents a "degree of belief" that the correct label is y. The components with values close to either 1 or 0 respectively correspond to labels that are considered either plausible or implausible.

To measure the accuracy of this degree of belief, a more complex error measure is obviously needed. This pseudo-loss measure is computed with respect to a distribution over the set of all pairs of examples and incorrect labels. By manipulating this distribution, the boosting algorithm can focus the weak learner not only on hard-to-classify examples, but more specifically, on the incorrect labels that are hardest to discriminate.

More formally, a mislabel is a pair (i, y) where i is the index of a training example and y is an incorrect label associated with example i. Let B be the set of all mislabels: $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq yi\}$. A mislabel distribution is a distribution defined over the set of all mislabels. On each boosting iteration t , AdaBoost.M2 supplies the weak learner with a mislabel distribution D_t. In response, the weak learner computes a hypobook h_t, expressed as h_t: $X \times Y \rightarrow [0, 1]$.

Then, the pseudo-loss of hypobook h_t with respect to mislabel distribution D_t is calculated as:

$$\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y) \tag{3.6}$$

It is important to outline that the pseudo-loss is minimized when correct labels y_i are given values near to 1 and incorrect labels $y \neq y_i$ values near to 0. More details on this formula are provided in (Freund and Schapire, 1997).

The distribution of mislabels is updated to reflect the accuracy of the entries selected in that iteration are classified by the newly generated hypobook in a similar way to AdaBoost.M1:

$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \times \beta^{(1/2)(1+h_t(x_i,y_i)-h_t(x_i,y)} \tag{3.7}$$

In this case, the update of the distribution is not only based on whether the entry was correctly classified but also on a quantification for a particular

entry of how near or how far away the "degree of belief" was to the correct solution.

The weak learner's goal is to find a weak hypobook h_t with small pseudo-loss. Thus, some modification may be needed to use a standard "off-the-shelf" learning algorithms in this way, although this is often straightforward. After receiving h_t, the mislabel distribution is updated using a similar rule to the one used in AdaBoost.M1. For a given instance x, the output of the final hypobook h_{fin} is the label y that maximizes a weighted average of the weak hypobook values $h_t(x, y)$. Also, although weak hypothesesh_t are evaluated with respect to the pseudo-loss, the evaluation of the final hypobook h_{fin} uses the ordinary error measure.

$$h_{fin}(x) = argmax_{y \in Y} \sum_{t=1}^{T} \left(log \frac{1}{\beta_t} \right) h_t(x, y) \tag{3.8}$$

The experiments presented in (Freund and Schapire, 1997) prove that boosting significantly and uniformly outperforms bagging when the weak learning algorithm generates fairly simple classifiers.

3.2.3 Mixture of Experts

The mixture of experts is also a widely known paradigm for the combination of classifiers (Jacobs et al, 1991). The main difference with the previously discussed models is that this one is composed of two differentiated parts: the set of classifiers and a final *"gating mechanism"* that manages the combination of the outputs of the ensemble components.

Both the classifiers and the gating mechanism -which usually consists of an ANN that has been trained by using the expectation maximization (EM) algorithm (Dempster et al, 1977)- are trained at the same time over a given data set. Each of the ensemble components generates a particular hypobook with regard to the data. The gating mechanism is also trained using the same data set as the other components of the model. In this case, though, the aim of this training is to determine which weights will need to be given to each of the classifiers' outputs in relation to a particular input, in order to maximize the accuracy of the output of the final hypobook. In other words, the underlying idea of using the gating mechanism is to find out whether training data have been properly learnt by each of the classifiers.

Figure 3.2 illustrates the mixture of experts model. The outputs of these classifiers on their training data sets samples, along with the actual correct labels for those blocks constitute the training data set for the gating mechanism. By comparing the output of each one with the expected one, the gating mechanism can identify which of the classifier are experts in certain data regions.

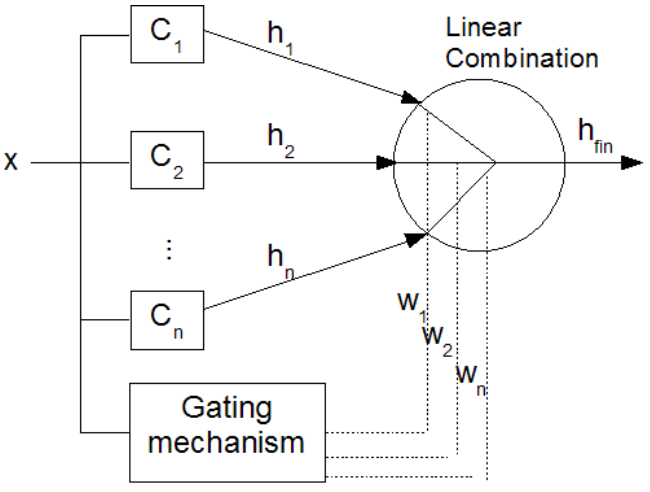


Fig. 3.2 Mixture of Experts architecture

Therefore, when the system is asked to classify a new entry, each classifier will generate an output according to its hypobook. The gating mechanism will determine the degree to which a classifier is considered an expert in the input space region where the new entry belongs and will accept the response of those classifiers with a higher weight in the voting over the final output.

The output of the whole system would be:

$$h_{fin} = \sum_{j=1}^{M} w_j h_j \tag{3.9}$$

in which, w_j is the weight that the gating mechanism gives to a particular classifier. For a single pattern (x, d) the error function to be minimised for the system would be: $e_{fin} = \frac{1}{2}(h_{fin} - d)^2$. Then, if using gradient descent for training the system, the weights given to each classifier would be calculated using the partial derivative:

$$\frac{\partial e_{mix}}{\partial g_i} = hi(\sum_{j=1}^{M} w_j h_j - d) \tag{3.10}$$

Mixtures of experts are particularly useful when different experts are trained on different parts of the feature space, or when heterogeneous sets of features are available to be used for a data-fusion problem. Several mixture-of-experts models can also be further combined to obtain a hierarchical mixture of experts (Jordan and Jacobs, 1994).

3.3 Combining Ensemble Results

The most common ensemble algorithms have been briefly overviewed in the previous section. An important aspect that might improve the ensemble results is hinted at in those algorithms. It is not only a matter of how each component is trained, but also of how the results of each component are combined to obtain the final output.

The combination of the models can be accomplished at any of these three levels:

1. In the input space, by a process known as Data Fusion.
2. In the architecture of the machines, by a process that we call Fusion.
3. In the output space, by a process known as Aggregation.

The first level is a complex subject that will not be directly addressed as it is considered beyond the scope of this book. The second level, on the other hand, represents the central contribution of this book, and is the subject of detailed analysis in subsequent chapters. The third level is briefly discussed in this section, as certain aspects bear some relation to the rest of the book.

According to (Polikar, 2006) there are generally two types of combinations: machine selection and machine fusion. In machine selection, each model is trained to become an expert in some local area of the total feature space, and the output is aggregated or selected according to its performance. In machine fusion, all the learners are trained over the entire feature space; the combination process involves merging the individual machine designs to obtain a single expert with a superior performance.

A very simplified overview of the combination methods used in the previously discussed algorithms -in the machine selection category- is presented in this section. For further information, a much more detailed classification is proposed and discussed in (Ruta and Gabrys, 2000).

3.3.1 Selection

The simplest way to obtain a better performance of an ensemble of classifiers is to estimate the performance of each one and to select the model that obtains better results, dropping the rest. This technique is also known in literature as 'bumping' (Tibshirani and Knight, 1999) meaning "boostraped umbrella of models parameters".

Bumping is said to work because the bagging procedure sometimes leaves out data points which have a strong pull on the classifier, prompting solutions which exhibit poor generalisation. Outliers for example are of little use when training a classier for good generalisation and so a classifier which is trained on data samples minus the outliers is likely to exhibit good generalisation.

This technique seems too naive, and in almost every case, is outperformed by the more sophisticated combination techniques. For example in (Petrakieva and Fyfe, 2003) a comparison of the technique with other more

complex ones in the field of topology-preserving maps thereby favoured the voting combinations discussed in Section 3.3.2. The same was true for (Heskes, 1997). Nevertheless, as it is also a much simpler technique when referring to its computational complexity, there can be cases where the simpler approach, which requires less computational power, can perform equally well.

In (Heskes, 1997) Heskes also proposes an algorithm called 'balancing' that is considered a compromise solution, half-way between bumping and the voting process in bagging, although for this simple classification, it should be included under the heading of voting combinations (Section 3.3.2).

3.3.2 Voting Combinations

In cases where the expected output is a discrete class, the simplest strategic combination for the final output of an ensemble is simple voting of its outputs, as for the original Bagging algorithm (Section 3.2.1). A very simple and widely used combination rule is presented as the final output, the most frequent label among the labels predicted by each member of the ensemble.

Bahler and Navarro (Bahler and Navarro, 2000) conducted a broad empirical study on the use of different combination rules. They found that when accuracy is approximately balanced across the estimators, majority voting performs as well as any more complex combination rule. When accuracy is imbalanced, majority voting tends to decrease in reliability, while more complex methods which take account of the individual performances, such as Bayesian methods (Ali, 1995), retain their performance.

3.3.3 Linear Combinations

In cases where the expected output is a number, rather than a discrete class, the most straightforward approach to the combination of the ensemble outputs consists in calculating the mean of the output values for all the components:

$$h_{fin} = \frac{1}{M} \sum_{i=1}^{M} f_i \tag{3.11}$$

When information about the performance of the classifiers is available, the linear weighted summation of the outputs will usually obtain more accurate results. This is the case of the AdaBoost algorithm (Freund and Schapire, 1996) (Section 3.2.2.1) and the Mixture of Experts (Jacobs et al, 1991) (Section 3.2.3).

$$h_{fin} = \sum_{i=1}^{M} w_i f_i \tag{3.12}$$

One of the first studies to consider combinations of different estimators -in this case for regression problems- was by Perrone (Perrone, 1993). It proposes

a method for determining the optimal weights to perform the combination that is based on a calculation of the correlation matrix. This matrix can not be calculated analytically, but can be approximated with a validation set. See also (Hashem, 1997) for more details.

Several other studies on classification problems have also been published. For example, (Tumer and Ghosh, 1996) provides a theoretical framework for analysing the simple averaging combination rule when the classifier outputs are estimations of the subsequent probabilities of each class. In (Fumera and Roli, 2001), this framework is extended to work with non-uniform weighting. This study provides an analytical way to calculate the corresponding weight of each classifier for classifiers with non-correlated estimator errors. It acknowledges that the analytical computation of optimal weights is a very difficult problem for classifiers with correlated estimation errors -the most likely situation- , there being no solution to this problem in the literature.

3.4 Ensembles of Artificial Neural Networks

As explained in Chapter 2, ANNs can be used as universal approximators. Due to this characteristic, among many of their applications, they can be used for pattern recognition, classification, regression analysis, prediction, function approximation,...etc.

One of the most interesting strengths of ANNs, and one of the main reasons why they are used for tasks that other analytical solutions find difficult to apply, lies in their ability to generalize beyond the data that has been used for their training. As also stated in Chapter 2, before an ANN can be used, it is trained over a sample set of the data with which it will subsequently be required to function. Its performance will be measured in terms of its ability to generalise beyond this sample, and to perform correctly when responding to examples that had never before been presented to it.

In this case, the problem is to obtain a sample of the data set that is sufficiently representative of the future problems that the ANN will be asked to solve. Unfortunately, for most real world problems that ANNs help to solve, this representative sample is impossible to determine. It is here where ensemble techniques can be of such help. When trained in a concrete data set, from which *non a priori* information is known, it would be quite reasonable to think that an ANN will on some occasions generalize incorrectly, it is also quite reasonable to think that, due to the benefits explained in Section 3.1.2, the ensemble meta-algorithms can be used to improve the performance of these models.

Generally, ANN algorithms are based on an iterative process in which data set entries are randomly presented to the network. The same learning algorithm that makes these models exhibit great generalization abilities, also makes them quite unstable. Making any changes to the training data set or to the learning parameters can lead to very different results. According

to (Sharkey and Sharkey, 1997) when a data set is presented to an ANN, its learning algorithm responds with an approximation of the function from which the data was extracted. A small change to this training data set can make the same ANN -even when using the same parameters for the learning phase- approximate a different function, thus changing the results. Another of the weaknesses of these types of iterative algorithms is that they are quite computationally intensive, often taking hundreds or thousands of iterations to complete the learning process.

The task of selecting the correct set of training parameters or data set is a very daunting task because of these characteristics. As explained, this leaves room for improvement through the use of ensemble algorithms, as achieving sufficient diversity for the components of the ensemble is a rather simple task. Usually, only a small variation in the data set is usually enough. These variations are achieved by means of quite simple operations such as data set re-sampling, non-linear transformations or pre-processing operations such as dimensionality reduction. Also, the task of training several rather approximate networks, instead of one very accurate one, can reduce the risk of using too many iterations when attempting to train a network that, in the end, has to be discarded, because its generalization capacity is not as good as expected.

3.4.1 Supervised ANNs

As explained earlier, the learning algorithm for the majority of ANNs used in combination with any kind of ensemble meta-algorithm, is quite prone to improve the results of single models, due to the inherent characteristics of its training. Furthermore, those ANNs that use the supervised learning type of algorithm, may potentially be used with the more sophisticated ensemble methods that can adapt ensemble construction during their training without much effort, such as AdaBoost (Section 3.2.2.1). Using supervised learning, an estimation of the strengths and weakness of the previously trained component can be determined at each step of the ensemble construction, so that this information is available for the construction of the next classifier.

The combination of supervised ANNs and ensembles has been explored in many previous studies (Perrone and Cooper, 1993; Jimenez, 1998; Maqsood et al, 2004) with quite interesting and promising results.

3.4.2 Unsupervised ANNs

There is nothing new in the application of ensembles with ANNs. A bibliography on ensemble use in supervised ANNs will contain many different models and applications, which have been proposed, tested and analyzed over the last 10 years. In contrast, the use of any kind of ensemble meta-algorithms

with ANNs that make use of unsupervised learning has hardly been explored in a comparable way.

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

In the case of the unsupervised ANNs, the main obstacle is the difficulty of determining the degree to which network performance deviates from the expected optimum. In many cases, this is because the expected performance remains undefined.

Despite this, many steps have already been taken towards improving the performance of unsupervised ANNs with the aid of ensembles. In (Giacinto and Roli, 2001), ensembles and unsupervised ANNs are used for the classification of images; in (Miskin, 2000), this combination is used for classification of multi-dimensional data; in (Greene et al, 2004), it is used for clustering in medical applications; and, in (Bennett et al, 2002) a novel semi-supervised learning model is based on a combination of unsupervised learning techniques.

3.5 Conclusions

In this chapter, state-of-the-art ensemble meta-algorithms have been presented. The chapter began with a brief introduction and went on to provide an explanation of their advantages and the problems they can solve. The most widely-known algorithms have been presented along with their most interesting characteristics. Finally, the use of this meta-algorithm with ANNs, which is central to this book, has been reviewed.

Chapter 4
Use of Ensembles for Outlier Overcoming

4.1 Introduction

A method for overcoming the influence of outlier samples in projection methods is described in this chapter, having been devised and tested as a proof of concept for the use of ensembles in combination with unsupervised learning.

The main goal of all visualization techniques is to bring to the user a deeper understanding of a multi-dimensional data set by generating some kind of graphical representation that can be easily inspected by the naked eye, enabling the viewer to rapidly focus on the most interesting groups or clusters of data. Projection methods are those based on the identification of "'interesting" directions in terms of any specific index and/or projection. These indexes or projections are, for example, based on the identification of directions that account for the largest variance of a data set, by using methods such as Principal Component Analysis (PCA) (see Section 2.4.1) (Pearson, 1901; Hotelling, 1933), or by looking for higher order statistics, such as the skweness or kurtosis -which is the case of Exploratory Projection Pursuit (EPP) (Friedman, 1987)- .

Having identified the interesting projections, the data is then projected onto a lower dimensional subspace in which it is possible to examine its structure visually, which normally involves plotting the projection onto two or three dimensions. The remaining dimensions are usually discarded as they relate mainly to a very small percentage of the information or the data set structure. In that way, the structure identified through a multi-variable data set may be easily analyzed with the naked eye.

As explained in the following section, the presence of outliers in the data is a common problem that greatly affects the results of these methods. The use of ensembles can potentially help to overcome this instability.

4.2 The Outlier Problem

Outliers are observations that lie at an abnormal distance from other values in a data set. In one sense, this definition leaves it up to the analyst -or a

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 49–66.
springerlink.com

consensus process- to decide what will be considered abnormal. The presence of outliers can be caused by a number of different reasons and usually indicates faulty data, erroneous procedures, or areas where a theory might be invalid. A number of these outlier cases appear in almost every medium-sized non-artificial data set, distorting its variance and hence hindering its analysis (Dixon, 1950).

As already explained, some projection methods and particularly PCA try to determine the directions of the highest second statistical order (variance) in the data set. In a simpler way, these high variance directions account for directions with the highest statistical dispersion, averaging the squared distance of their possible values from the expected values (mean).

The problem in this case is that the variance calculation is done globally by employing both statistical and connectionist models. Outlier samples in a data set can affect its direction of highest dispersion. Each data point that is situated at some distance from the majority of the other points in the data set can influence the final result, as it introduces a higher variance in comparison with the rest; even though it might be very small and even anecdotal or dispensable. A number of these outlier cases appear in almost every medium-sized non-artificial data set, distorting its variance and hence hindering its analysis.

The outlier situation is illustrated in Figure 4.1. Considering the same data set as in Chapter 2, the first two principal components (PCs) are quite clear without the presence of appreciable outlier data points. In Fig.4.1 they are outlined in continuous black. Now, this situation could change, if that data set contained outliers. By adding five outlier points (marked in Figure 4.1 with

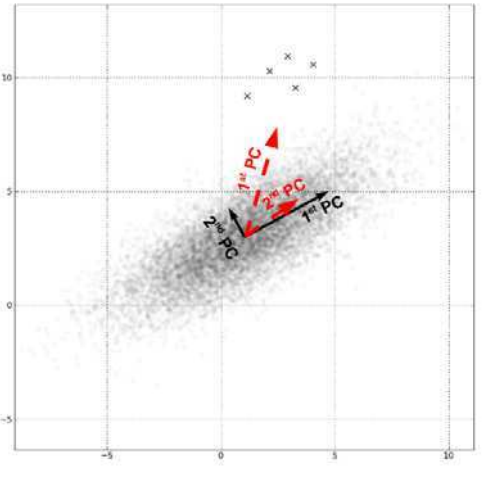


Fig. 4.1 Principal components of a data set with and without outliers

an X) the direction of the highest data dispersion -or, in statistical terms, variance- changes to those marked in Figure 4.1 with dashed red line. Were no outliers present at all, there would be no difference between the second PC and the first.

Obviously, the best axes for projecting this data set would be the ones detected with no outliers in the data set, as in those directions where the vast majority of the data set is spread out.

The problem is that the presence of outliers, although not a representative sample of the data set, shifts the variance in those directions, as the variance calculation only accounts for the variance and does not take account of the number of samples in any one direction.

The main purpose of this chapter is to present a method that attempts to solve this problem by using an ensemble meta algorithm. It is stated in (Kuncheva, 2004) that bagging is especially recommended when applied to unstable algorithms or learning methods. The main premise of the investigation detailed in this chapter is that, as an inherently unstable algorithm in the presence of outliers, PCA can be stabilized by the application of bagging.

4.3 The Re-sampling PCA Algorithm

This section presents the proposed solution to the outlier problem described in the previous one, devised during the work done for the present thesis. The technique used to detect and to overcome the presence of outliers in a multidimensional data set is based on statistical re-sampling theory. Calculation of the maximum variance over a data set is quite straightforward. It is not so simple to determine whether maximum variance is merely due to a few or very many samples. In other words, the problem can consist in determining whether that direction is really interesting in terms of how representative it is in relation to the whole data set. The algorithm proposed will be referred to as Re-PCA in the rest of the thesis.

4.3.1 Ensemble Construction

There is no objective function in the calculation of variance. The ensemble construction algorithm used in this study will therefore be the Bagging algorithm. As explained in Chapter 2 this algorithm trains all components independently; it does not need to know the performance of one to train another. To apply this technique, it is necessary to have several replications of the data set under analysis. Accordingly, several independent analyses can be performed over different data sets obtained from the same source. As so many different data sets are in many cases often unavailable, they are emulated by the use of re-sampling with replacement over the data set.

If the entire data set does not include elements that drastically alter their statistical properties -i.e. in this case, its second statistical moment: the variance- , the set of results obtained from the analysis of different subsets should be similar within a small margin. On the other hand, if few outliers that alter these statistical properties are included in the main data set, different results may be expected in terms of the directions of the principal components. As there are, by definition, very few outliers in relation to the size of the entire data set,it is easy to imagine that, when re-sampling the data, one of those infrequent outlier data points would only be included in a minority of the subsets. Intuitively speaking, it is also reasonable to expect that PCA performed on subsets containing outliers will be influenced more by the outliers, if the ratio of the outliers to the number of other data points is high.

The next step consists of performing an individual PCA analysis on each one of the t subsets obtained by re-sampling the original one. These calculations use the statistical method detailed in Chapter 2, Section 2.4.1. Theoretically, the analysis can also be performed by means of a connectionist model using Hebbian learning, although there are certain problems in doing so.

At the end of this stage, t different sets of principal components are available- all calculated over a slightly different data set- .

4.3.2 Results Combination

The combination of results to obtain the final directions -called eigenvectors- for the Principal Components are done by averaging the results of each of the statistical analyses composing the ensemble.

Each analysis presents its result for each of the required components, and a process of voting and averaging of the results is performed. Firstly, the sum of all eigenvectors is calculated. This will be considered as a tentative "average" of the directions.

Then, for the same component, the cross products of each eigenvector and each of the aforementioned sum vectors are calculated. Any overly deviated direction has to be identified -due to the inclusion of outliers in that particular subset of data- to complete the averaging of directions, in order to arrive at the final principal components by resolving the final direction of each component. The scalar product (the projection) of each one of the analyzed directions (eigenvectors) in reference to the previous calculated sum is calculated. In the following steps, the vectors which are further away -their scalar product is higher than a certain threshold- are disregarded.

Finally, the sum of the vectors is calculated again, which now corresponds to an average for the directions that were considered "similar", as they were no longer influenced by the more pronounced deviations. This process is described in the algorithmic pseudo-code in Algorithm 3.

Algorithm 3. PCA ensemble results combination

Input: A set of different vectors representing a set of principal components $V = \langle (v_{11}, v_{12}...v_{1n}), (v_{21}, v_{22}...v_{2n})...(v_{m1}, v_{m2}...v_{mn}) \rangle$
Output: Final principal components $V' = \langle (v'_1, v'_2...v'_n) \rangle$

1: **procedure** COMBINATION(V)
2: **for** $i = 0$ to m **do**
3: Calculate the sum of all vectors: $\overrightarrow{S}_i = \sum_{j=1}^{m} \mathbf{v}_j$
4: Calculate the scalar product of each vector and the sum: $P_j = \overrightarrow{S_i} \cdot \overrightarrow{v}_j$
5: Construct another set of vectors including only those that correspond to a scalar product lower than a set threshold (th): $\overrightarrow{v}_j \in V'_i \; if \; P_j < th$
6: Calculate the sum of V'_i: $\overrightarrow{S'_i} = \sum_{j=1}^{m} \mathbf{v}'_j$
7: **end for**
8: Output $\overrightarrow{S'}$
9: **end procedure**

As can be seen, the idea is firstly to identify the analyses which "proposed" a direction for a Principal Component that is quite different from the directions "proposed" by the other analyses composing the ensemble. It is assumed that in such cases the re-sampled data set will contain one or more outliers, thus, their results will not be taken into account. It is also very simple to take account of the analyses that were not included in the final combination and the differences between them and the others. This provides additional help when determining whether the data modifying the entire data set analysis can be regarded as outliers. If a low number of analyses contain non-frequent directions, their data may be considered rare in the data set, so they can be regarded as outliers. If the same non-frequent directions appear in a relatively high number of analyses, then those data can be considered to represent a significant portion of the data set and they therefore have to be taken into account in the analysis.

4.4 Experiments and Results

Three different data sets were analysed as a test of the method presented in Section 4.3.2. The first data set is artificial, the second is the well-known 'BUPA' Liver Disorders data set, extracted from the UCI Machine Learning Repository (Asuncion and Newman, 2007), and the third was extracted from an interesting case-study on the food industry.

4.4.1 Artificial Data Set

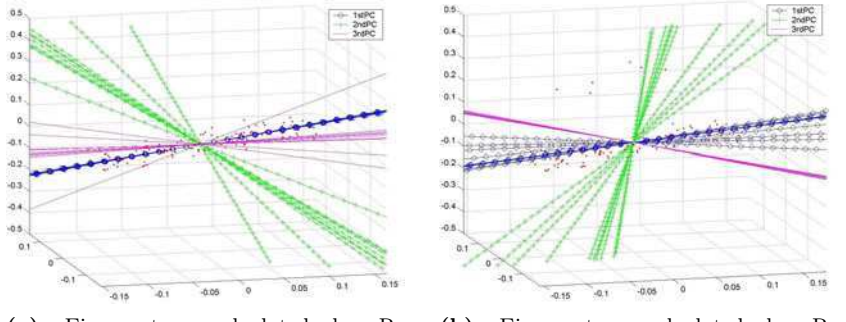
The artificial data set used in this series of experiments comprises one main point cloud, at some distance above which there are several points that are considered outliers. The main cloud is an elongated cluster which moves within the axis delimited by the line defined by points [1,1,0], [2,1.6,0], [3,2.2,0] and [4,2.8,0]. It is expected that 3 clear principal components will be obtained by employing this data set, as the variance of each direction differs in each case with respect to the other two. The outlier points are spread over the same axis, but are displaced 5 units above on the vertical axis. There are 118 points in the main cluster and 8 outliers.

In order to test various characteristics of the proposed Re-PCA algorithm with regard to different proportions of outliers in the other data points and various sizes of the data sets, experiments with 30, 50 and 100 randomly selected points were performed. In each case, the experiments were repeated 10 times for each of these cases and the comparative analysis is presented below.

Some of this experimental results can also be found in (Gabrys et al, 2006).

Experiment 1

In this set of experiments, 10 subsets of 100 randomly selected points from the entire data set (without replacement) were generated and PCA was performed on each one. Firstly, the above-described method was only applied to the data set that is formed of the main elongated data cluster -i.e. without the outliers-. The results of PCA obtained from those 10 subsets are represented in Figure 4.2. In that figure eigenvectors determining the direction of higher variance in the data set without outliers (Fig. 4.2a) and with outliers (Fig. 4.2b) are displayed for a data set composed of 100 samples.



(a) Eigenvectors calculated by Re-sampling PCA (excluding outliers)

(b) Eigenvectors calculated by Re-sampling PCA (including outliers)

Fig. 4.2 Eigenvectors determining the direction of higher variance in the data set with and without outliers (100 samples data set)

Examining Figure 4.2a it is easy to see that the Re-PCA method has found almost the same direction for the first principal component, as it was expected. The directions of the second and third principal components were slightly more dissimilar in the different tests, but they all still followed a consistent direction.

Figure 4.2b represents the results obtained by performing exactly the same experiment except for the inclusion of the 8 outliers in the sampled data set. As expected, these outliers destabilized the data set, making different PCAs behave in inconsistent ways and throwing up very different results where the analysis is made over subsets of the same data set. As can be seen, the distribution of the directions corresponding to the principal components, produced when outliers are taken into account, are more spread out than in Figure 4.2a -data without outliers- . Notably, there is less consensus in the component that has to represent the greatest amount of information -the1^{st} principal component- than when the outliers are not included. This means that the directions found in each case are rather dissimilar to the others. The final averaged directions are shown in Figure 4.3.

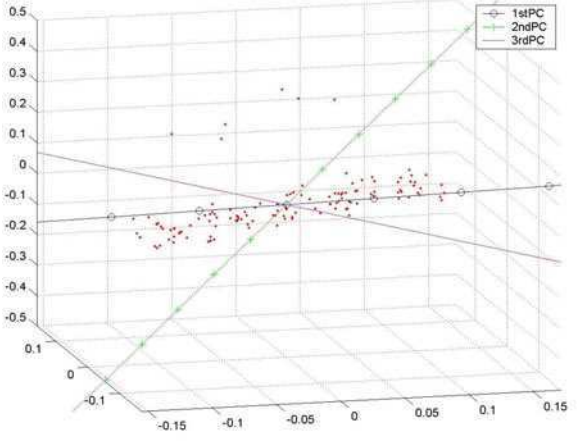


Fig. 4.3 Average for each of the principal components as a result of averaging the directions obtained by the Re-PCA method using 100 samples

The averaged directions in Figure 4.3 display a reasonably correct final consensus for the Principal Components to the naked eye, if compared with Figure 4.2. The first crosses the entire data set in the most elongated direction, the second captures the second direction in which the data set stretches -which coincides with the direction towards the outlier location- and is perpendicular to the first, and, finally, the third crosses the other two perpendicularly.

Experiment 2

In this case the same experiment is also performed 10 times for 50 randomly selected points from the entire data set (without replacement). It should be noted that the decrease in the number of samples included in each of the subsets analysed by Re-PCA has a destablizing effect on the experiments. The "fans" formed -in the case of the 50-point data set- by the directions corresponding to the three principal components in the ten tests are far more separated than those obtained in an analogous experiment based on 100 samples. The "percentage of information" that is represented by each of the principal components is shown in Table 4.1, including maximum and minimum information percentages from the 10 subsets.

Exclusion of the 50 data samples had a destabilizing effect on the individual PCAs, although the first and second principal components appear to run in an almost perpendicular direction to the other eight on two out of ten occasions (20%), indicating some instability which may be due to the presence of outliers in the data set. The second principal component is always very unstable because all the outliers are in its direction. Table 4.2 shows the "percentage of information" for each of the principal components, including the maximum and minimum percentage of information from the 10 subsets under analysis.

Table 4.1 Percentage of information captured by each of the principal components (selecting 50 points but excluding outliers)

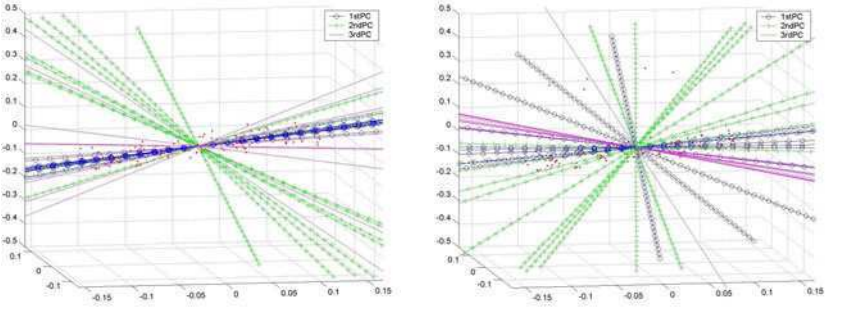
	Percentage of information captured	
Principal Component	max	min
First	72%	68%
Second	16%	14%
Third	14%	12%

Table 4.2 Percentage of information captured by each of the principal components (selecting 50 points and including outliers)

	Percentage of information captured	
Principal Component	max	min
First	70%	44%
Second	44%	15%
Third	13%	9%

Experiment 3

To test the stability of the presented model, PCA was performed over 10 subsets, this time each of only 30 points. The results for this data set are shown in Figures 4.4a and 4.4b. Figure 4.4a represents eigenvectors determining the direction of higher variance in the data set without outliers (Fig. 4.4a) and with outliers (Fig. 4.4b) are displayed for a data set composed of 100 samples.



(a) Eigenvectors calculated by Re-sampling PCA (excluding outliers)

(b) Eigenvectors calculated by Re sampling PCA (including outliers)

Fig. 4.4 Eigenvectors determining the direction of higher variance in the data set with and without outliers (30 samples data set)

The percentage of information -in the form of the explained variance- that is represented by each one of the principal components is detailed in Table 4.3, including the maximum and minimum percentages of information (variance) from the analysed 10 subsets.

As can be seen from the experiments described above, the greater the number of samples included in the analysis, the more stable the behaviour of each individual PCA. Compared with Figure 4.2, Figure 4.4 provides visual confirmation that the directions found using 100 points are more consistent

Table 4.3 Percentage of information captured by each of the principal components in the first part of the experiment (30 samples without outliers)

	Percentage of information captured	
Principal Component	max	min
First	72%	68%
Second	18%	14%
Third	14%	11%

than those using only 50 or 30 points. It can also be seen (Figure 4.2 and Figure 4.4) that the inclusion of outliers in the analysed data set introduces a substantial degree of instability, which yield "fans" that are more spread out -less consistent results- , or which even yield completely different directions for its principal components. This also serves as proof of the intuitive idea mentioned in Section 4.2: as whenever less data is included in the data set, the influence of the outliers on the variance analysis is greater, as the proportion of deviated samples with respect to non-deviated ones increases.

The percentage of information that is represented by each of the principal components in this case is detailed in Table 4.4, including maximum and minimum information percentages (variance) from the 10 subsets.

Table 4.4 Percentage of information captured by each of the principal components in the second part of the experiment (30 samples with outliers)

	Percentage of information captured	
Principal Component	max	min
First	69%	49%
Second	41%	17%
Third	13%	8%

The results presented in Table 4.4 are quite different from the results obtained without the inclusion of the outliers. It is clear from a comparison of both tables (Table 4.3 and Table 4.4), that the presence or the absence of outliers in a data set influences not only the direction of the largest variance, but also the relative difference between the maximum and the minimum values of the principal components. The amount of information associated with the first principal component is different depending on whether outliers are included in the analysed data. The amount of information detected by the first component (Table4.4) in the presence of the outliers is inferior to the amount it detects without outliers (Table 4.3). In this case, the amount of information represented by this second component (Table 4.4) is substantially higher than when outliers are not included (Table 4.3), which is due both to the shape of the artificial data set that is used and due to the fact that the outliers are situated in the direction associated with the second principal component.

The use of PCA ensembles in such cases is particularly useful, as in 70% of cases where "the true" principal component is found, it is representative of the majority selection, and then stability is enhanced by averaging the eigenvectors from the majority (70%) of similar principal directions. Moreover, in 3 out of 10 cases (30%), the method finds that the first and second principal components have opposing directions to the majority ones.

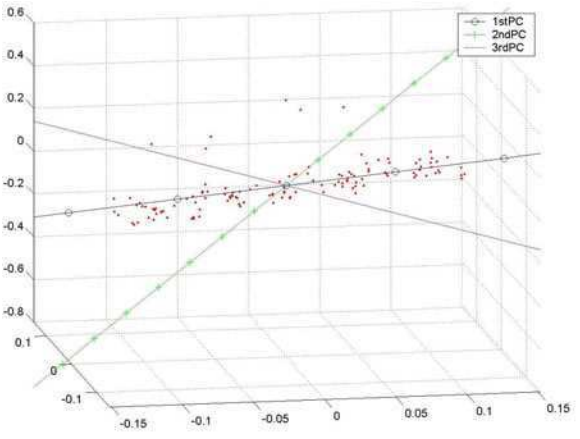


Fig. 4.5 Average for each of the principal components as a result of averaging the directions obtained by the Re-PCA method using 30 samples

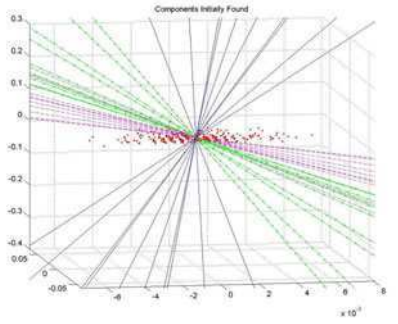
Looking at Figure 4.4b, it can be seen that the first principal component runs in an almost horizontal direction on 7 occasions, while it runs along a diagonal from the bottom-right to the upper-left corner on the other 3 occasions. These three deviated directions will not be taken into account when calculating the averages, as the majority cluster consists of the 7 cases where the first Principal Component appears in the horizontal direction.

As explained above, calculating the average directions (Figure 4.5) yields approximately the same main directions established by the three principal components in the experiment using 120 points. This can be considered an empirical proof of the robustness of the proposed Re-PCA method.

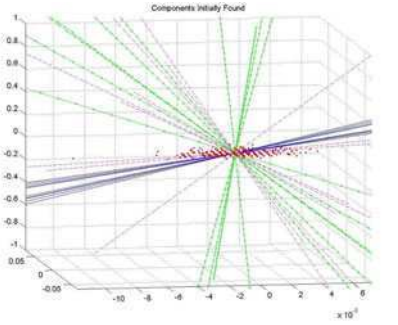
4.4.2 Real Life Data Set: Liver Disorder Data Set

The proposed method was tested with a real data set called 'BUPA' obtained from the UCI online repository (Asuncion and Newman, 2007). Applying the method described in this study to the complete data set with over 200 points in each sampling test -2/3 of the whole data set- gives the result shown in Figure 4.6b. Figure 4.6b shows the application of the same model to the data set without outliers.

The samples considered as outliers after inspection by the naked eye are designated by their order of appearance in the data set. Outliers in class 1 (21): 190, 317, 316, 182, 205, 335, 345, 343, 189, 312, 344, 175, 168, 183, 25, 172, 311, 167, 326, 148, 261. Outliers in class 2 (20): 85, 36, 134, 233, 331, 300, 179, 323, 342, 111, 115, 77, 186, 252, 294, 139, 307, 224, 286, 157.



(a) Directions corresponding to 200 points over the 'BUPA' data set (excluding outliers)



(b) Directions corresponding to 200 points over the 'BUPA' data set (including outliers)

Fig. 4.6 Directions calculated by the Re-PCA ensemble over the 'BUPA' data set

In Figure 4.6b, the directions corresponding to the first principal component are very tight, although those corresponding to the second and third are too spread out -several directions are even almost perpendicular to others corresponding to the same principal component- . The direction of the first component is tight due to the outliers -laid out in that particular direction- . The opposition of the first and the second principal components are due, once again, to the instability that those outliers bring to the whole data set. On the contrary, after analyzing Figure 4.6b, it may be seen that, although the directions of the first PC are much more spread out than in Figure 4.6b, they do not run in opposing directions -there is a fair degree of consensus over the direction- and the second and third components run in a significantly tighter direction.

4.4.3 Real Life Data Set: Food Industry Application

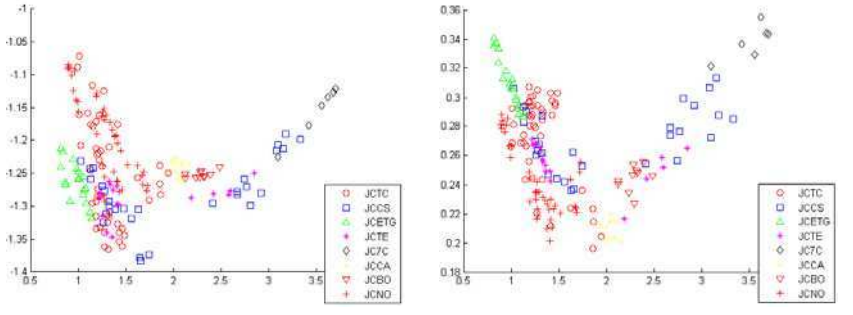
The data set used in this set of experiments is taken from a case study on the food industry. It consists of data obtained from the analysis of certain chemical properties of samples cut from different types of Spanish cured hams. As this same data set is used in further experiments, full details are included in Appendix A.

In this case, Ham data set 1 was used, composed of 176 samples of ham, each made up of 18 different variables. Some of the results from this experiment may also be found in (Baruque et al, 2006).

Experiment 1

Performing a single PCA analysis over the original data and representing the data on the axes obtained in the analysis gives a quite interesting result. In

the resulting projection (Figure 4.7a), the samples of the highest quality hams -JC7C, JCCS and partially JCTE- can be seen to the right of the image; the altered parts are all grouped together in the center of the image, and the standard and the low quality hams are situated to the left of the image -JCNO, JCTC- . Even the fact that some of the high-quality ham samples -JCCS and JCTE- were more rancid than others is reflected in the image, where several of them appear mixed up in the group of standard quality types. The fact that those samples are precisely the more rancid ones is easily verifiable by attaching an identifier to each of the points. It can be clearly seen in the projection of the data over the first and second principal components, as well as in the first and third components as shown in Figure 4.7.



(a) Projection over the 1^{st} and 2^{nd} Principal Components

(b) Projection over the 1^{st} and 3^{rd} Principal Components

Fig. 4.7 The ham data set projected over the principal components obtained form a single statistical PCA (without outliers)

Table 4.5 shows the percentage of information captured by each one of the first three principal components. This information can be compared with the same information in following experiments, where the inclusion of a few outliers varies the whole analysis.

Table 4.5 Percentage of information captured by each of the principal components in the first experiment (176 samples without outliers). In this case the simple PCA is applied

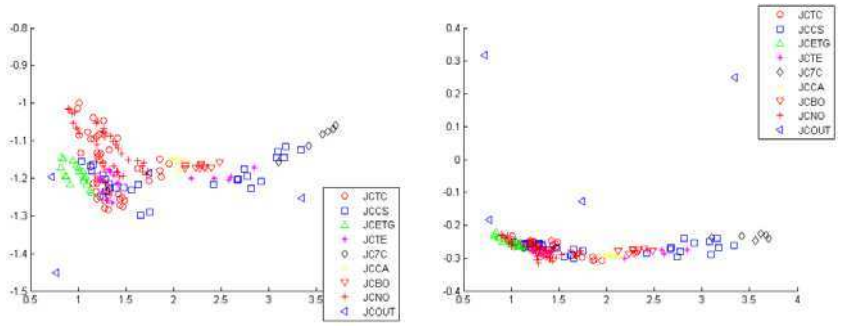
Principal Component	Percentage of information captured
First	86.58%
Second	8.47%
Third	4.66%

Performing a Re-PCA analysis over this data set does not reveal further noticeable information. As no outlier points are included in the data set all the independent tests composing the Re-PCA give almost the same result, so the average of their results coincides almost completely with the result of a simple PCA.

Experiment 2

Four outlier measures were added to the original data set for this experiment, in order to observe the effect that the inclusion of outliers has on the PCA analysis. Performing a simple PCA analysis and projecting the data on the two axes determined by the principal components, in exactly the same way as in the previous experiments, gave the results shown in Figure 4.8. In this figure the outliers are called JCOUT to differentiate them from the other samples.

It may be seen that although the first and second components are affected by the inclusion of outliers, the groups described in Experiment 1 can still



(a) Projection over the 1^{st} and 2^{nd} Principal Components

(b) Projection over the 1^{st} and 3^{rd} Principal Components

Fig. 4.8 The ham data set projected over the principal components obtained from a single statistical PCA (including 4 outliers)

Table 4.6 Percentage of information captured by each of the principal components in the second experiment (176 samples including outliers). In this case the simple PCA is applied

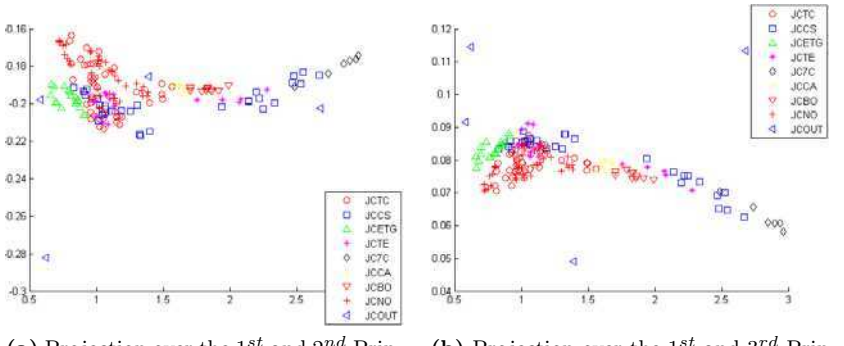
Principal Component	Percentage of information captured
First	83.78%
Second	8.4%
Third	7.8%

be distinguished. On the contrary, the third principal component found is completely different, as the projection of the data set over the first and third principal components no longer shows those groups.

Regarding Table 4.6 it may be seen that the percentage of information captured by the first principal component is lower in this case, while that percentage rises for the third PC. This means that the first component is no longer able to capture as much information as before -as the variance of the data set has been altered- and the third component is the "one in charge" of dealing with the information that was previously captured by the first one. This provides empirical confirmation that the presence of very few outliers can alter the results of the PCA analysis in a significant way.

Experiment 3

In this experiment, the Re-PCA was applied to the data set including outliers. 80 samples from the whole data set were randomly selected (without replacement) for each PCA analysis. Ten different analyses were performed and averaged to obtain the results shown in Figure 4.9.



(a) Projection over the 1^{st} and 2^{nd} Principal Components

(b) Projection over the 1^{st} and 3^{rd} Principal Components

Fig. 4.9 The ham data set projected over the principal components obtained from a Re-PCA of 80 samples (including 4 outliers)

Inspecting Figure 4.9a it is quite clear that the first and second principal components are able to display almost the same information as in Experiment 1, which was the experiment done without outliers in the data set. An inspection of Figure 4.9b (corresponding to 1^{st} and 3^{rd} principal components) reveals a slightly clearer structure.

At first sight, this might be interpreted as a very good improvement on Experiment 2, but checking Table 4.7 it can be seen that the process is still too unstable, as the single PCA analyses that compose the Re-PCA outperform the single PCA on some occasions -87% of information for the 1^{st} PC- but

Table 4.7 Percentage of information captured by each of the principal components in the third experiment (80 samples including outliers). In this case the Re-PCA is applied

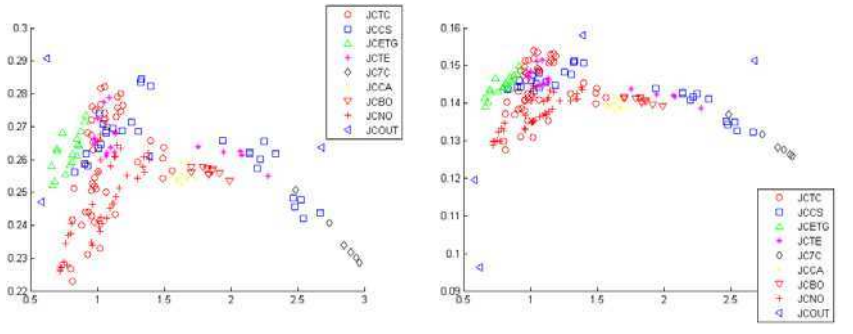
	Percentage of information captured	
Principal Component	max	min
First	87.1%	79.9%
Second	11%	8.5%
Third	9.2%	4.3%

very poorly in the other -79.9% of information for the 1^{st} PC- . This can be explained by the fact that less than the half of data set is used for each of the analyses.

Experiment 4

On this occasion the procedure followed in Experiment 3 was repeated, but this time 120 randomly selected samples were included from the whole data set in each of the ten analyses performed over the Re-PCA process. The results of projecting the data set over the three main principal components in this experiment are shown in Figure 4.10.

The results in this case are similar to those obtained in Experiment 1, taking into account that the outliers also have to be represented -so the images get a bit distorted- . The group information described in Experiment 1 is clearly distinguishable on both the axes formed by the 1^{st} and 2^{nd} and



(a) Projection over the 1^{st} and 2^{nd} Principal Components

(b) Projection over the 1^{st} and 3^{rd} Principal Components

Fig. 4.10 The ham data set projected over the principal components obtained from a Re-PCA of 120 samples (including 4 outliers)

Table 4.8 Percentage of information captured by each of the principal components in the first part of the experiment (120 samples, including outliers). In this case the Re-PCA is applied

	Percentage of information captured	
Principal Component	max	min
First	86.5%	82.5%
Second	9.4%	8%
Third	7.9%	4.6%

by the 1^{st} and 3^{rd} principal components. In Table 4.8, the difference between the information percentages captured by each of the principal components is displayed. As may be verified, the difference is much lower now, which shows that in this experiment the directions identified in each of the ten tests coincide to a much greater higher. This means that the directions of maximum variance found in this experiment are more consistent than those found in Experiment 3.

4.4.4 ANNs Approach

As explained in Chapter 2, PCA analysis can also be performed using an ANN with feed-forward Hebbian Learning. This type of algorithm, when appropriately trained, i.e. when it is made to converge towards a solution, can find the directions of principal components to a degree of accuracy that is comparable to pure statistical calculations. Moreover, as it is considered an unstable algorithm -based on connection dynamics- , the possibility of improving ANN dimensionality reduction is an interesting prospect.

Chapter 3 describes why potential improvements to the ensemble lie in the possibility of constructing a set of classifiers or an analysis that strikes the right balance between diversity that is often either too high or too low. The problem in this case is that the algorithm performing the analysis is too unstable to work well as a straightforward ensemble. The neuronal PCA can offer quite dissimilar results, due to the random nature of its iterative learning algorithm; even when trained with the same set of parameters and the same data set. So, while the algorithm converges towards its solution, which may be as correct as that obtained by statistical analysis -capturing the same percentage of information- , the directions do not have to coincide with the others, nor do they even have to coincide with the results of two different analyses by exactly the same learning algorithm.

Therefore, in the case of the PCA calculated by an ANN with the proposed algorithm, it is impossible to distinguish between the case where no outliers are present in the training data set and the case where they are present, as the

directions obtained by the Re-PCA exhibit the same inconsistencies in both cases. This is the reason why the Re-PCA is only used in these experiments in combination with the statistical PCA analysis.

4.5 Conclusions

This chapter has described an initial approach to the use of ensembles for improving the results of a single statistical model. This adaptation is done without the aid of an objective function. Instead, a way of calculating a consensus from the components of an ensemble is devised. The technique has demonstrated its usefulness in overcoming the distortion effect that the "noise" of outlier samples can have on data-set dimensionality reduction. It is true that the models in this case can not be considered unsupervised learning models, as they are obtained by mathematical calculations. The intention to apply the same technique to an unsupervised learning technique was not possible, due to the high instability of the model. Nevertheless this is a step forward in the study of how ensemble meta-algorithms may improve unsupervised learning models for projection-based data visualization.

Chapter 5
Ensembles of Topology Preserving Maps

5.1 Introduction

In the case of projection methods based on unsupervised learning, the problem of instability makes it difficult to devise an easy way of combining their results into a final solution. This is mainly because it is a global type of analysis, which means that very little information can be obtained once it has been completed.

This chapter centres on different methods of enhancing the use of ensembles with another visualisation-centred unsupervised learning algorithms: the topology-preserving map family.

5.2 Problem Statement

As happens with many ANN algorithms, topology-preserving maps -such as the SOM, ViSOM, SIM or Max-SIM (see Chapter 2)- are quite unstable (Bishop, 1995). The presence or absence of a little data can modify the behaviour of the whole model in very different ways. When dealing with unsupervised learning models, this problem becomes even more apparent, as there is no clear way to make a quantitative assessment of its performance or the degree to which two different executions of the same algorithm differ.

The ensemble model can be regarded as a possible solution to overcome this problem. The same principle, which was used in Chapter 4 in the ensemble meta-algorithm to overcome the outlier problem in projection methods, can be used here to improve the performance of this family of algorithms. It is also true that this type of learning is rather robust against outliers in the training data set, but ensembles can be used to boost other key features of the topology-preserving maps, such as pattern recognition or data representation capabilities. The SOM is regarded as a vector quantization method for space approximation and tessellation, which can be used to faithfully approximate statistical distributions in a nonparametric, model-free fashion. SOM learning

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 67–94.
springerlink.com

is also efficient, effective, and suitable for high-dimensional processing. Thus, in both statistical and computational terms, SOM is a promising scheme for complex data fusion and joint modelling.

The characteristic competitive learning of these models is also a random iterative process, but in this case, results are not obtained globally for the whole data space, but instead different sets of neurons are trained over different regions of the data space, leaving more options open to combine the results of different maps.

If a SOM does not correctly represent a portion of the data space, perhaps because it is dedicating more neurons to represent another, it is reasonable to believe that a different SOM trained over slightly different data samples belonging to the same portion of space may obtain better results, at least for that portion of space. In that case, ensemble fusion or combination algorithms can potentially obtain a final map that is as clear as possible and that represents the data set characteristics detected by each of the individual maps that compose the ensemble.

5.3 Topology-Preserving Map Combination Models

Topology-preserving maps -especially the most widely known: the SOM- have been used for information fusion applications in several previous works.

The first proposed model, which included various cooperating SOMs to outperform what could be achieved with only one map, should be the Hierarchical SOM proposed by Luttrell in (Luttrell, 1989). Usually a 'hierarchical SOM' refers to a tree of maps in which the higher maps act as a pre-processing stage for the lower ones. As the hierarchy is traversed upwards, the information becomes more and more abstract. He pointed out that although adding extra layers to a vector quantifier yields a higher Distortion in reconstruction, it also effectively reduces the complexity of the task. Another advantage is that different kinds of representations are available at different levels of the hierarchy. Later on, a multi-layer version of that algorithm was proposed by Lampinen and Oja (Lampinen and Oja, 1992).

In (Cho, 2000), Cho uses a bagged ensemble of a modification of the SOM -called 'Structure-Adaptative SOM'- to classify handwritten numbers. The interesting part of this work is its use of two different multi-map levels. The first one is called the Structure-Adaptative SOM, which can "split" neurons of the original map into new sub-maps when the representation detail of that neuron is insufficient. Secondly, a bagged ensemble of the proposed Structure-Adaptative SOM is used for classification by majority voting. In (Petrakieva and Fyfe, 2003) Petrakieva and Fyfe use a 1-D SOM ensemble for data classification purposes. Also, in (Jiang and Zhou, 2004), Jiang and Zhou use a SOM ensemble for image segmentation by clustering image pixels. In this last case, a final calculation step is needed to combine the different clustering results they obtain.

A factor which groups together the above-described works is that they all use the SOM algorithm as a pattern recognition and/or classification tool. It is therefore easy to employ any of the combination algorithms detailed in Chapter 3 (Section 3.3) or similar ones to obtain the final outputs.

5.3.1 Previously Proposed Models for SOM Ensemble Summarization

When using the SOM as an analysis tool, the combination of results is easy, as there is no real need for human interpretation of the intermediate results, due to the fact that in the majority of cases the final result is the interesting result. The problem with these types of combinations arises when the desired output is not a numeric prediction or a class membership degree, but the plain 2-D representation of a multi-dimensional data set for easy inspection by humans. Representing all the networks in a simple image is only useful when dealing with 1-D maps -as seen in (Petrakieva and Fyfe, 2003)- but gets too messy when visualising 2-D maps.

In that case, the expected output value would not be yielded by the ANN, but by the map generated by the ANN. A much more complex fusion algorithm is needed as it is necessary to devise a system that can calculate a final topology-preserving map that somehow summarises the best aspects of the maps composing the ensemble.

To summarise this situation, the desired outcome of the ensemble combination process is a single map that unites the improved performance and stability of the use of ensembles with the simplicity and clarity of multi-dimensional data, all of which may be inspected on a single map.

Various works (Georgakis et al, 2005; Saavedra et al, 2007) have developed map summarizations to obtain a final map, two of which are detailed below.

5.3.1.1 Map Fusion by Euclidean Distance

In (Georgakis et al, 2005) Georgakis et al. use a SOM ensemble for document clustering. They propose a fusion of an ensemble of maps by firstly, aligning the neurons and then by calculating the centroid of the vectors corresponding to neurons aligned towards the same position. These centroids are calculated as the simple sum of all vectors, as can be seen in Eq. 5.1.

$$w_c = \frac{1}{|W_k|} \sum_{w_i \in W_k} w_i \tag{5.1}$$

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.

When aligning the networks, one should make a partition of the $L \times D$ neurons into$L$ disjoint clusters. There are two constraints, though, both of which have to be met at the same time:

- ▷ Each cluster will contain only D neurons.
- ▷ Each cluster will contain only one neuron from each of the D networks.

The schematic diagram of the alignment of the neurons in the different networks is presented in Figure 5.1.

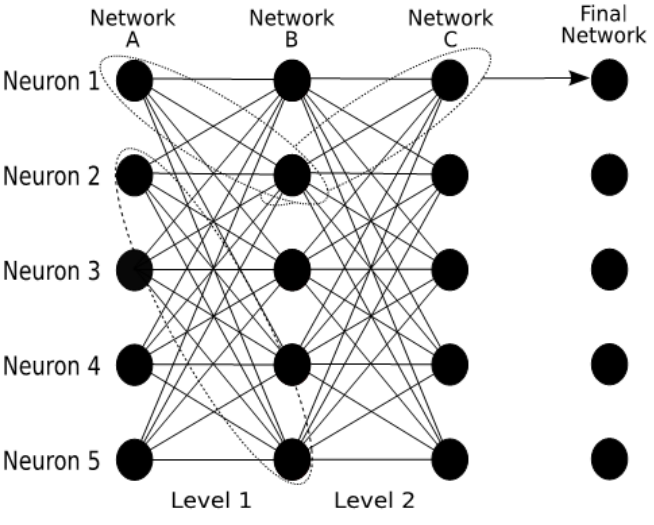


Fig. 5.1 Alignment of three networks and their subsequent merging into one final map

Figure taken from (Georgakis et al, 2005)

The algorithm can be implemented by using dynamic programming to deal with its high computational complexity. The fused map is firstly initialised by using only two maps of the ensemble to calculate the fusion. Then the same calculation is performed again between the resultant fused map and another one of the maps composing the ensemble. This is repeated until all the maps of the ensemble have been included in the calculation of the fused map. A pseudo-code of how this algorithm works can be found in Algorithm 4.

Algorithm 4. Map Fusion by Euclidean Distance

Input: Set of trained topology-preserving maps: $M_1 ... M_n$
Output: A final fused map: M_{fus}

1: Select a training set $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$
2: train several networks by using the bagging (re-sampling with replacement) meta-algorithm : M_n
3: **procedure** FUSION(M_n)
4: initialise M_{fus} with the weight vectors of the first map: $M_{fus} \leftarrow M_1$
5: **for all** $M \in M_n$ **do**
6: **for all** $w'_i \in M_{fus}$ **do**
7: calculate Eucl. Dist. between w'_i and ALL neurons of map M_i ▷ let w^* be the closest neuron in map M_i to the one selected in M_{fus}
8: $w^* \leftarrow argmin_i \left(ED(w'_i, w_i)\right)$
9: $w_c \leftarrow w'_i + w^*/2$ ▷ applying Eq. 5.1 to two neurons
10: $w'_i \leftarrow w_c$ ▷ replace w_i by the centroid (w_c)
11: **end for**
12: **end for**
13: **end procedure**

Finally, in addition to calculating the characteristic vectors of the neurons in the fused map, the set feature vectors corresponding to each neuron should also be updated. These sets are the unions of the sets corresponding to the clusters of neurons formed in the previous step. Let f_{ij} denote frequency of the j^{th} input in the set $\mathcal{L}_{ifinal}$. If the frequency is close to the value$D$ then more neurons in the constituent networks have that particular neuron assigned to them during the training phase. Therefore, the more frequent an input is, the higher its importance to the particular neuron of the final map. In other words, the inputs assigned to a particular neuron are ordered in descending order of appearance.

It can be inferred by analysing the pseudo-code that the computational complexity of the algorithm is of $O((M-1)N^2)$, in which M is the number of maps in the ensemble, and N the number of neurons in each map.

5.3.1.2 Map Fusion by Voronoi Polygons Similarity

In (Saavedra et al, 2007) Saavedra *et al.* proposed a SOM fusion model that aims 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 neighbourhood relation is given by the edges that measure 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 the degree to which the data partition represented by two different neurons overlaps -and therefore, the degree to which they can may considered similar- , a binary vector of the same size as the data set used to train the network is associated with each map neuron. This vector contains a '1' in the data position that was recognised by the neuron and '0' in the data position that was not recognised. It will serve to compute the dissimilarity between two neurons by using Eq. 5.4.

$$ds(b_r, b_q) = \frac{\sum XOR(b_r, b_q)}{\sum OR(b_r, b_q)} \tag{5.2}$$

Let r and q be the neurons whose dissimilarity will be determined and $\mathbf{b_r}$ and $\mathbf{b_q}$ the binary vector relating to each neuron with the data sample that it recognises.

This vector also serves to calculate the usage of each of the neurons, which is 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. Those sets must satisfy the following criteria:

$$\begin{cases} ds(b_r, b_q) < \theta_f & \forall r, q \in Ws_n \\ ds(b_r, b_q) > \theta_f & \forall r, q \notin Ws_n \end{cases} \tag{5.3}$$

where θ_f is a connection threshold that is introduced as a parameter of the algorithm.

The described sets become neurons in the final map, by calculating the centroids of its composing neurons (Eq. 5.1). Finally, to reconstruct the map lattice; the connections between the neurons in the fused map must be re-calculated. Neurons obtained from fusing clusters that are similar enough, are considered neighbours in the final map. This may be expressed as: .

$$\underset{b_r \in Ws_k, b_q \in Ws_l}{min} ds(b_r, b_q) < \theta_c \tag{5.4}$$

where Ws_k and Ws_l are two different sets of neurons considered similar enough to be fused and θ_c is a given connection threshold.

The complete pseudo-code for the proposed fusion algorithm is presented in Algorithm 5.

Visual examples of how this fusion algorithm behaves on artificial data sets can be found in Figure 5.2.

As may be appreciated from the examples, this SOM Fusion algorithm preserves the topology of the input space by effectively locating the prototypes and relating the neighbouring nodes, which improves the performance of the individual SOM and the resulting lattice represents the topology of the data.

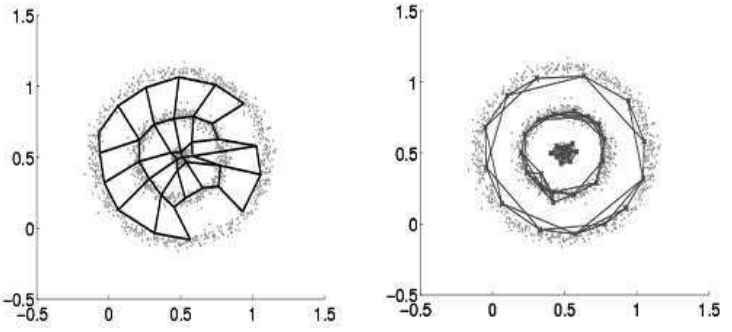
Algorithm 5. Map Fusion by Voronoi Polygon Similarity

Input: Set of trained topology-preserving maps: $M_1 ... M_n$,
usage threshold: θ_u , fusion threshold: θ_f, connection threshold: θ_c
Output: A final fused map: M_{fus}

1: Select a training set $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$
2: train several networks by using the bagging (re-sampling with replacement) meta-algorithm : M_n
3: let θ_u, θ_f and θ_c be the usage, fusion and connection thresholds respectively
4: **procedure** FUSION($M_1 ... M_n$)
5: **for all** $M_i \in M_n$ **do** ▷ for all maps in the ensemble
6: **for all** $w_j \in W_i$ **do** ▷ for all neurons in each map
▷ accept neurons with a recognition rate higher than a given threshold
7: $W_{fus} \leftarrow w_i \; if \; \sum_i b_r(i) > \theta_u$
8: **end for**
9: **end for**
10: **for all** $w_i \in W_{fus}$ **do**
11: calculate dissimilarity between w_i and ALL neurons in W_{fus} (Eq. 5.2)
12: $D_i \leftarrow ds(w_i, w_k) \forall w_k \in W_{fus}$
13: **end for**
14: group into different sub-sets (Ws_n) the neurons that satisfy the conditions of Eq. 5.3
15: **for all** Ws_n **do**
16: calculate the centroid (w_c) of the set by using Eq. 5.1
17: add the centroid to the set of nodes of the final map (W^*_{fus})
18: **end for**
19: **for all** $w_r \in W^*_{fus}$ **do** ▷ for all neurons in the fused map
20: Connect w_r with any other neuron in W^*_{fus} , if they satisfy Eq. 5.4
21: **end for**
22: **end procedure**

The algorithm's computational complexity is also quadratic in this case.

All neurons considered for each map must be compared by using the following expression to calculate their dissimilarity: $O(N'^2)$, in whichN' is the total number of neurons of all maps that have recognition rate higher than θ_u. The same complexity also applies to the calculation of which neurons must be grouped, through:$O(N'^2)$. Likewise, when calculating the final dissimilarity between the fused neurons: $O(C^2)$, in which C is the final number of neurons in the map.



(a) Single SOM

(b) Fusion by Voronoi Polygon Similarity

Fig. 5.2 The figures show the topology approximation of the SOM and the SOM Fusion by Similarity algorithm to the "doughnut" artificial data set
Figure taken from (Saavedra et al, 2007)

5.3.2 Novel Proposed Model: Superposition

Both of the previously presented models satisfy the condition of being able to generate a single final map by fusing an ensemble of SOMs. In their respective works they are used and tested to fuse SOMs, although they are equally applicable to other topology-preserving map models.

As pointed out, both also had different objectives in mind for the final fused map. Fusion by Euclidean Distance was originally considered a tool for clustering and pattern recognition. Fusion by Similarity of Voronoi Polygons is mainly used for learning data topology. Accordingly, several characteristics of the topology-preserving maps are omitted in these algorithms. This is especially so when referring to one of their most defining characteristics: their ability to represent high-dimensional data as a 2-D map, preserving the inner structure of the data set.

The earlier models aim to calculate the best characteristics vector for each neuron in the fused map from the characteristics vector of the neurons in the ensemble maps, but they do not take the neighbourhood of each neuron into account. This results in poor topology preservation in the final fused map. Even in the case of the second algorithm, the reconstruction of a 2-D map is impossible. Many neurons can be disregarded, due to their low recognition rate, resulting in blank spaces in the grid of the map. Additionally neighbouring relationships between the final neurons are recalculated, forming a new shape that will not necessarily resemble the shape of a grid.

In order to avoid those problems and to obtain a truthful representation of a data set under study an alternative algorithm for topology-preserving map fusion is presented. It consists of "superposing" the maps formed by the networks composing the ensemble into a final map, by means of a neuron-by-neuron comparison. Note that the weights of each network in the ensemble

are initialised in a way that makes the neurons in the same position of two -or more- networks comparable. To do so, it is important to make the maps as similar as possible; the bagging algorithm being given the task of including sufficient variance for the ensemble to outperform the single model. To ensure as much similarity as possible between maps, all of them are trained using the same parameters and each of the ensemble components is initialised with the state in which the training of the previous was finished. Also, when each map finishes its training, a measure of quality selected among those presented in Chapter 2 is calculated. If the measure does not reach a certain limit, the map is discarded and a new one is trained to replace it. The pseudo-code for this fusion algorithm is included in Algorithm 6.

Algorithm 6. Map Fusion by Superposition

Input: Set of trained topology-preserving maps: $M_1 ... M_n$
Output: A final fused map: M_{fus}

1: Select a training set $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$
2: train several networks by using the bagging (re-sampling with replacement) meta-algorithm: M_n
3: **procedure** FUSION($M_1 ... M_n$)
4: **for all** $w_j \in W_i$ **do**
5: $w_i^* \leftarrow \sum_{k=1}^{n} w_i^k / n$ ▷ centroid of the units in the same position in all maps
6: $b_i^* \leftarrow OR(b_i^k)$ ▷ Frequency at recognising a class: sum of frequencies of the neurons selected for each class.
7: **end for**
8: **end procedure**

Additionally there is a one last step that can be performed. As described, the algorithm has an additional interesting effect: the same data samples of the data set are said to be recognised by two or more different units. This method differs from competitive learning with which the maps are usually trained. It is due to the way the data that activated a neuron as the BMU is computed in step 5 of the Algorithm 6. If the same data sample is recognised by two -or more- neurons in different positions of two different ensemble maps, the two neurons in that position in the final map will be marked as the BMUs for that sample. The implications of this situation are discussed in subsequent sections.

Furthermore, the option exists of a final re-labelling step in the algorithm, in which the training data set is presented to the map again to decide, once and for all, which of the neurons in the final map is really the BMU for each sample; the final result yields a regular topology-preserving map. At the experimental stage, the map obtained through the application of Algorithm 6 is referred to as 'Superposition', while the map obtained after applying this final re-labelling step is referred to as 'Superposition + Re-labelling'.

The computational complexity in this case is $O(M \times N)$, in which M is the number of maps in the ensemble, and N the number of neurons in each map. As in most cases $M \ll N$, this algorithm consumes less computational resources than the two previously described fusion, as it is less complex to calculate.

5.3.3 Discussion of the Fusion Models

As detailed in the subsequent sections -Section 5.4- the idea of having an ensemble of topology-preserving maps as classifiers is perfectly valid. Examples show that performance is increased by the use of the bagging meta-algorithm. It is reasonable to suggest that the primary intention of the ensemble or committee of experts idea was to improve the classification capabilities of the so called "weak learners".

Nevertheless, it should also be added that classification was not intended as the main feature of these kinds of algorithms. It should not, therefore, be very difficult to identify models that yield better results.

The concept of a unique model summarising the characteristics of several other diverse ones seems more suitable for one of the most characteristic features of this kind of models: data inspection by the naked eye. The single map has numerous advantages for human experts who would otherwise have to compare several maps to establish the most interesting characteristics of data sets. This unique map idea can also be easily adapted to well-known SOM architectures that would be more complicated to implement with an ensemble of maps such as the Hierarchical SOMs.

The differences between the three fusion models presented in this chapter are dependent on the final intention for which they where devised. Fusion by Euclidean Distance was originally intended to improve clustering and classification, which is why it stresses re-calculating the position of the neurons and especially the samples recognised by each neuron. This has the same effect as explained in the case of superposition: two different neurons are supposedly detected as the BMU for the same data sample. The effect of this can be seen in some of the experimental results included in Section 5.4. The final map surface in Fusion and Superposition is covered by a much higher number of labelled neurons than in a single map, which will all be able to classify data when fired. This is an advantage when classifying data, but can be considered a disadvantage, in that it detracts from the representation of the data set, when that is the desired final output.

Also, this fusion completely disregards the neighbourhood of neurons in the map. Thus, the result is a final map with poor topology preservation, due to twists in the map's grid and disordered neurons; what makes it a poor tool for visualisation. Finding the actual closest unit to another one using Euclidean distance would be a NP-complete problem. It would simply take too long to compute in large maps, which is why the algorithm works with an approximation to identify the units to be fused. This also means that the

map is easily calculated in a batch process, when all the composing maps have finished their training; although this makes it quite unusable for an on-line process.

Fusion by Voronoi Polygons Similarity was especially devised for recognition of data topology, but not for representing that topology in a map. The map's grid is the actual desired representation. In that sense, the final fusion obtained with this algorithm is closer to Neural Gas (Martinez and Schulten, 1991) than to the SOM. The final map consists of a series of neurons connected as a graph, with no defined number of neighbours for each neuron. Usually the number of neurons for that map is lower than for its component maps, as it omits non-responding units.

The final fused map obtains a very low quantization error, which also makes it quite useful for pattern recognition tasks, but which means that it is impossible to obtain a 2-D map representation of the data structure.

Finally, the Superposition algorithm tries to avoid the pitfalls of those previous models. The idea is to impose stricter restrictions when training the different maps composing the ensemble, by using the state of the previous one to initialise the next one. Thus, it is reasonable to consider that, in each of the composing maps, the most similar neuron to the neuron in the final map will be in the same position in all maps. The neighbourhood relationships are therefore completely preserved by using this approach. The union of the sets of samples recognised by the corresponding neuron in each map is used for storing data samples recognised by each neuron. In that way, the classification advantages of Fusion by Distance can apply.

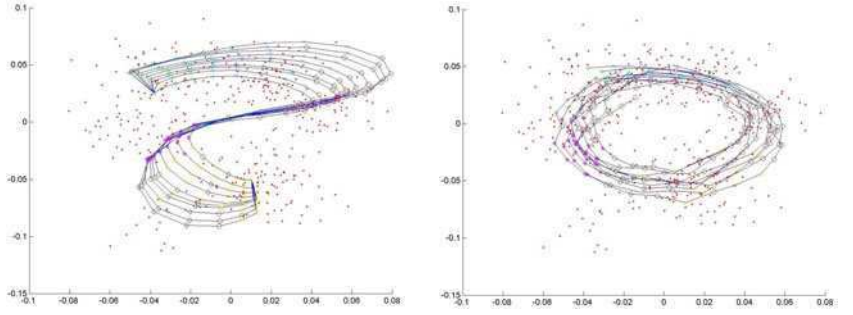
Having a final map with the same structure as a single map also enables it to be used in more complicated architectures, such as hierarchical SOMs or as an on-line learner, as new data is easily incorporated in the ensemble and pushed up to the Superposed map.

5.4 Experiments and Results

5.4.1 Comparison between Single Model and Ensemble as Classifiers

The first experiment was designed to confirm the adaptability of the concepts relating to improvements in classification results presented in (Petrakieva and Fyfe, 2003), in this case for another topology-preserving map architecture. In the case of this experiment, the tests were performed for comparison of the SOM and Max-SIM models -which are described in Chapter 2- .

A radial data set was generated in order to test the performance of the three fusion models in a data set where it supposedly will make a clear difference. It was composed of six normal 2-dimension distributions laid out radially, the centres of which were situated at points [3,2], [1,4], [-2,4], [-3,1],



(a) Ensemble of 10 SOMs trained on the bagged data

(b) Ensemble of 10 Max-SIMs trained on the bagged data

Fig. 5.3 Comparison of the SOM and Max-SIM bagged models when trained on a radial data set

[-2,-4] and [1,-2] respectively. The number of samples corresponding to each distribution was as follows: 50, 100, 70, 50, 20 and 100. A visual representation of the results is shown in Figure5.3.

The SOM networks of the experiment were trained following the same steps detailed in (Petrakieva and Fyfe, 2003). The weights of the neurons of the maps were initialised following the first principal component of the data set. In the case of the Max-SIMs, neurons were subjected to radial initialization.

Table 5.1 shows the classification results when the Max-SIMs were trained over the radial data set described above. The results were obtained from a 10-fold cross validation -minimum, maximum and average accuracy- testing runs are shown in the table. In this case the data set did not contain any outliers.

Table 5.2 shows the classification results when the data set includes several outliers in the outward part of the radial set. The results are obtained from a 10-fold cross validation -minimum, maximum and average accuracy- from 10-fold cross-validation testing runs are shown in the table. In this case, including 20 outlier samples.

Table 5.1 Classification accuracy of three different models applied to the radial data set (without outliers)

	Accuracy of the model (without outliers)		
	min	max	average
Single Max-SIM	81.28%	86.15 %	83.58%
Ensemble Max-SIM	86.15%	88.2%	87.02%
Ensemble SOM	80.76 %	86.41 %	83.11%

Table 5.2 Classification accuracy of three different models applied to the radial data set (including 20 outliers)

	Accuracy of the model (including outliers)		
	min	max	average
Single Max-SIM	75.3%	83.1%	79.6%
Ensemble Max-SIM	82.1%	86.3%	84.1%
Ensemble SOM	79.2%	85.6%	82.8%

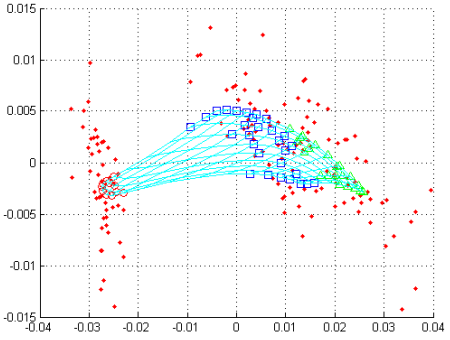
As expected, the Max-SIM ensemble model obtains better results than the single Max-SIM and the SOM ensemble models, with and without outliers in the data set, as can be seen in both Table 5.1 and Table 5.2. The results confirm those obtained by Petrakieva and Fyfe (Petrakieva and Fyfe, 2003). This was clearly expected as the results are based on classical bagging theory. The experiment relies on the bagged ensemble of several classifiers to obtain a final result by majority voting. In this sense, the Max-SIM functions like any other classifier, such as classification trees, Bayes classifiers or perceptron trees. These experiments confirm that the advantage of using ensemble methods are also applicable to topology-preserving maps.

The added robustness of the ensemble model versus the single map against outliers is confirmed. Comparing both Tables (5.1 and 5.2), it can be seen that the inclusion of the outliers degrades the results of all models. The difference is that the average classification results for the cross-validation are almost 3% in the case of the single Max-SIM, but they are only 2% in the case of the Max-SIM ensembles, and only 1% in the case of the SOM ensembles.

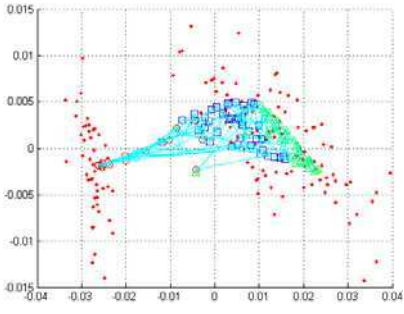
5.4.2 Comparison between Fusion by Distance and Fusion by Similarity Algorithms

The next experiment is intended to generate a comparison between the two SOM fusion algorithms that were previously proposed in the literature: Fusion by Distance and Fusion by Similarity. Further information about the experiments presented in this section is available in (Baruque et al, 2007).

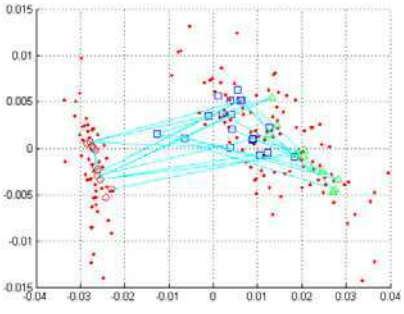
The first notable point is the structural difference between the maps generated by the two models. The clearest way to compare these differences visually is by representing them in the input space of the data with which they are trained. In our case, Figure 5.4 depicts three maps represented over the well-known Iris data set (Asuncion and Newman, 2007). The maps were trained over the complete data set -4 dimensions- but to represent them on a 2-D image, the first 2 Principal Components of the data set were calculated and both data and map's neurons weights were projected over those two components.



(a) Single SOM



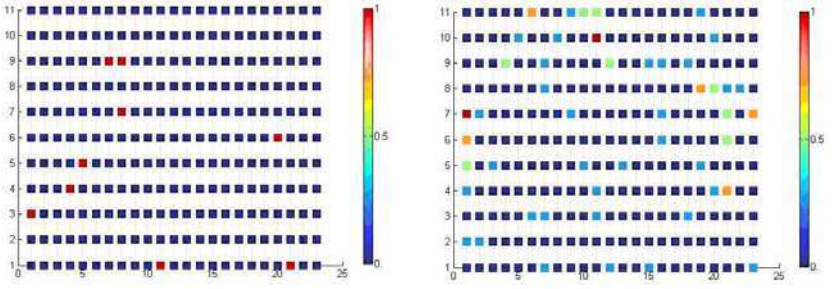
(b) Fusion by Euclidean Distance



(c) Fusion by Voronoi Polygon Similarity

Fig. 5.4 2-D representation of the compared model's grid represented over the Iris data set

Looking briefly at at Figure 5.4, there are several obvious differences. The single map (Figure 5.4a) is presented as a clear grid, which attempts to cover as much of the data space as possible, to enable it to represent all data in the final map. Fusion by Euclidean Distance (Figure 5.4b) is extended in the same way, but in this case the grid is not so clear, as it includes many twists and the neurons appear disordered. This is due to the way it is computed, which favours the position of single neurons to approximate data, but leaves aside the neighbourhood of the neuron. Fusion by Voronoi Polygon Similarity (Figure 5.4c) also presents a completely different structure. Although it is also expanded to reach the whole data set, in this case, not even a grid is preserved. The number of neurons in this third map is lower than in the other two models, as neurons that do not react to any data are removed from the final map. There are several neurons positioned on each side of the data set, but there are no neurons in the gap between the two groups. Moreover, the way in which the neurons are connected is no longer a grid with a fixed number of neighbours for a particular neuron. Instead, the neighbouring relationships depend on the Voronoi polygon covered by each neuron. This means that there can be neurons with a high number of neighbours or isolated neurons



(a) Topographic Error for each neuron in a single SOM

(b) Topographic Error for each neuron in Fusion by Euclidean Distance

Fig. 5.5 Topographic Error calculated over the single SOM and the Fusion by Euclidean Distance

with no neighbours at all. The structure of this map (Figure 5.4c) therefore resembles a graph more than a grid of interconnected units.

These same differences are confirmed in a more analytical way by means of topographic error analysis -explained in Chapter 2-. The results of the Topographic Error calculated over the single SOM in Figure 5.4a and over the Fusion by Euclidean distance in Figure 5.4b, are shown in Figure 5.5.

The higher topographic error of Fusion by distance is graphically represented in Figure 5.5. Neurons coloured in red (high error) are localized in the SOM map (5.5a) where the gap between two groups of the data set is situated. In Fusion by Distance (5.5b) these red/orange marked neurons appear all over the map, without any clear structure, illustrating the way in which, for no particular reason, many more neurons generally display this kind of error in relation to the data set.

Further analytical results are provided by a comparison between the two fusion algorithms used in combination with two different topology-preserving models. The comparison includes the two fusion algorithms applied to two different topology-preserving algorithms -the SOM and the ViSOM – in order to verify whether the different algorithms share similar characteristics. This comparison is based on two different quality measures for the maps -Quantization Error and Distortion –, each of them presented in Table 5.3 and Table 5.4. Three widely known data sets held in the UCI repository (Asuncion and Newman, 2007) -Iris, Wisconsin Breast Cancer and Wine- were used for the tests.

It may be seen from Table 5.3 that the Mean Quantization Error (MQE) of Fusion by Distance is, by a small margin, lower than the single model in almost all cases for both models. This was expected, as the fusion is intended to reduce the distance between neurons and data samples by re-adapting the neurons. In the case of Fusion by Voronoi Similarity, the MQE values are much lower than the single model, because of the lower number of neurons

Table 5.3 Comparison of the two topology-preserving models (SOM, ViSOM) using an ensemble of 10 maps to calculate the MQE for the following: the average of all 10 maps, the 10-map Fusion using the Euclidean distance algorithm, and the 10-map Fusion using the Voronoi similarity algorithm. The results of the table are the average of five different tests performed using cross-validation

	SOM			ViSOM		
	Single	Fus. Eucl. Dist.	Fus. Voronoi Simil.	Single	Fus. Eucl. Dist.	Fus. Voronoi Simil.
Iris	0.19	0.20	0.14	0.18	0.17	0.13
Cancer	1.95	1.93	1.16	1.74	1.54	1.23
Wine	9.91	10.40	4.42	9.40	9.13	4.06

Table 5.4 Comparison of the Distortion of the Fus. by Euclidean Dsitande and Fus. by Voronoi Similarity of an ensemble of SOM and an esnemble of ViSOM

	SOM			ViSOM		
	Single	Fus. Eucl. Dist.	Fus. Voronoi Simil.	Single	Fus. Eucl. Dist.	Fus. Voronoi Simil.
Iris	1.35	1.50	2.12	1.45	1.59	2.33
Cancer	19.03	25.12	43.52	15.46	19.23	41.98
Wine	69.12	71.50	60.93	65.82	55.81	62.81

in the final map in comparison with the other two models. These values will always be lower when less neurons contribute to the general error.

The Distortion measure in Table 5.4 is an error measure that shows how each map preserves the topology of the data set used for its training. Table 5.4 shows a comparison of the two topology-preserving models -SOM, ViSOM- using a 10-map ensemble by calculating Distortion for: the average of all 10 maps, the Fusion of the 10 maps using the Euclidean distance algorithm, and the fusion of the maps using the Voronoi similarity algorithm. The results of the table are the average of five different tests performed using cross-validation.

As shown in Figures 5.4 and 5.5, Fusion by distance obtains a higher error than the single model in almost every case, which was expected. Although the results of this fusion algorithm outperform the single model when used to classify new data, the final fused map does not preserve the relationship between its neurons. Fusion by Voronoi Similarity yields similar results to the

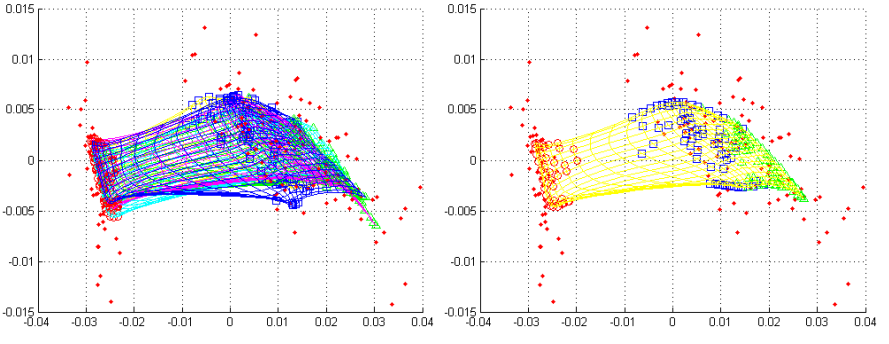
other fusion -exhibiting higher Distortion- , except in the Wine data set. In the case of this data set, which is the most complex of the three, the density of edge-connecting neurons is higher than in the Iris data set, as there are fewer "dead" neurons. In that case, Distortion will decrease, as the number of edges for a neuron is usually higher than in a single map.

When comparing two variants of the topology-preserving algorithms, it can be inferred that the ViSOM obtains better results than the SOM in the three data sets both for the MQE (Table 5.3) and the Distortion measures (Table 5.4), albeit by a small margin. This is due to its procedure of updating inter-neuron weights, as it forces the inter-neuron distances on the map to adapt to the inter-data distances of the input space; improving overall adaptation to the data set.

5.4.3 Comparison between Fusion by Distance and Superposition Algorithms

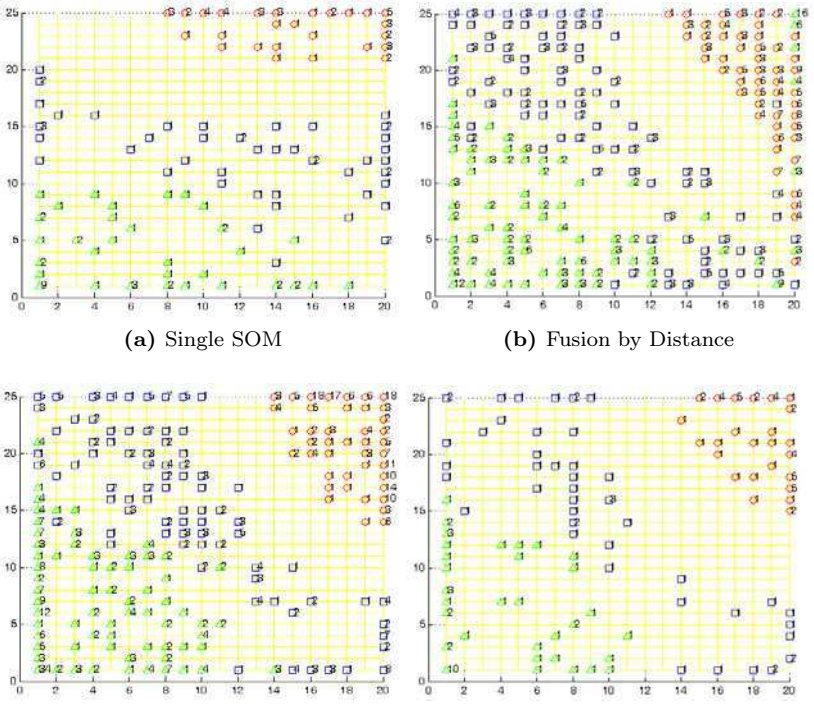
The following experiment aims to test the differences in the behaviour of the Fusion by Distance algorithm and the proposed Superposition algorithm. As in the previous experiment, the first thing to notice would be the structure of their networks. A representation of the map in the input space that is projected over the Iris data set is displayed in Figure 5.6. The figure shows an ensemble of five SOMs trained over the Iris data set -each one represented in one colour- and the final Superposition calculated over that ensemble. Both the data set and the maps are represented over the first 2 Principal Components of the data set.

The first thing that meets the eye in Figure 5.6 is that, with this fusion algorithm, a much smoother grid was obtained than in the previous test. The algorithm imposes a stricter initialisation of the map than the previous two.



(a) Ensemble of five SOMs **(b)** Superposition calculated over five SOMs

Fig. 5.6 Ensemble of 5 SOMs trained over the Iris data set and the final Superposition from that ensemble



(a) Single SOM **(b)** Fusion by Distance

(c) Superposition **(d)** Superposition + Re-Labelling

Fig. 5.7 The Iris data set as represented by Single SOM, Fusion by Distance, Superposition and Superposition + Re-Labelling

These ensembles consider neurons that are located at the same position as direct candidates to be fused, so there is no need to search the entire map for a suitable neuron, which is a much better way of preserving the neighbourhood of the map.

The final objective of the map fusion is depicted in Figure 5.7. It is the Iris data set as represented by the four models under comparison: Single SOM, Fusion by Distance, Superposition and Superposition + Re-Labelling. Summaries were obtained from the same ensemble of five SOMs, trained using the Bagging meta-algorithm, all maps exactly with the same parameters. The bagged SOMs composing the ensemble were trained by re-sampling 120 samples each.

Although all four representations retain similar structure, there are several interesting differences. In the case of Fusion by Distance (Figure 5.7b) and Superposition (Figure 5.7c), there is a greater density of neurons that recognise data. This is due to the effect of having more than one BMU in the fusion stage, which to some extent explains the improvements to classification that were originally reported for Fusion by Distance. However, it should

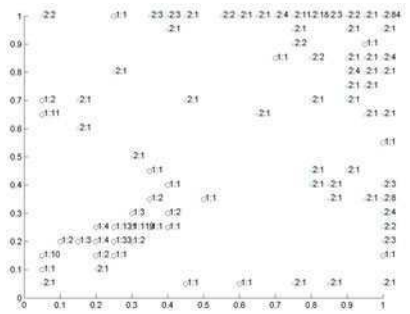
also be noted that several neurons appear disordered in the case of Fusion by Distance (Figure 5.7b). For example, two classes are mixed at the top right of the image, and the separation between the two linear groups is not very clear at the bottom right. Superposition (Figure 5.7c) yields a similar result in relation to the density of the reacting units, but with a more ordered lay out. The outlined defects in the representation mentioned for Fusion by Distance are not present in this map. Finally, Superposition + Re-Labelling (Figure 5.7d) have the effect, once again, of obtaining a lower number of units, in a similar way to the single SOM (Figure 5.7a), although the separation between groups appears to be more clearly represented in the first than in the second. As previously intended, the most suitable map depends on the tasks it is intended to perform. For data visualisation, it can be argued that Superposition + Re-Labelling creates a more truthfully representation of the data set on which it was trained. For data classification it might be more useful to have extra classifying units, as in Superposition -or even Fusion by Distance- .

For further analytical proof of the classification capabilities of the different models, a 5-fold cross-validation over the Iris data set is included in Table5.5. This table includes the classification results for the 4 single maps -the single map and three of the fusion algorithms- plus the classification rate achieved by the bagged ensemble of maps when it is not fused, for the case of the SOM (10x10 and 20x20) and the ViSOM (10x10 and 20x20). The results of the table are the average of five different tests performed using cross-validation.

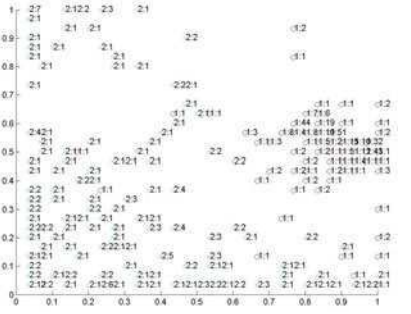
Table 5.5 Classification accuracy of the different models obtained from a SOM and ViSOM ensemble for the Iris data set

	Best Single Map	Bagged Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (10x10)	78%	94%	92%	73%	75%
SOM (20x20)	50%	81%	62%	59%	58%
ViSOM (20x20)	82%	92%	78%	74%	77%
ViSOM (30x30)	74%	83%	82%	70%	71%

From the analysis of Table 5.5 some conclusions can be inferred. First of all, as expected, the best classifier is always the bagged ensemble for all the models, which is precisely its objective. An ensemble of classifiers always appears to outperform a single model. Also, when comparing classification



(a) Visualisation of the SOM ensemble using the Superposition+Re-Labelling method



(b) Visualisation of the ViSOM ensemble using the Superposition+Re-Labelling method

Fig. 5.8 Comparative representation of the Cancer data set using SOM and ViSOM and Superposition+Re-Labelling

accuracy, Superposition outperforms Superposition + Re-Labelling. As explained before, this seems to be due to the higher amount of classifying neurons in the first model. Finally, Fusion by Distance seems to be a good solution to improve performance when the single map fails to obtain very good results -in the case of the SOM (20x20)- , but in all cases it is outperformed by Superposition. This is because both use the same process for preparing their units prior to classification, but Superposition preserves the neighbourhood of the map more clearly.

A similar study to the previous one is presented in this case for the Wisconsin Breast Cancer data set, also obtained from (Asuncion and Newman, 2007). Superposition and Superposition+Re-Labelling of the data set are shown in Figure 5.8: in a comparative representation of the Cancer data set using two different topology-preserving models (SOM and ViSOM), but the same Bagging parameters for the training of the ensemble and the same algorithm for the fusion of maps (Superposition+Re-Labelling).

Similar differences hinted at the previous comparison may be observed here.

Almost the same may be said for the experiments performed on this data set (Table 5.6), wich show the classification accuracy of the different models obtained from a SOM and ViSOM ensemble on the Cancer data set. The result of the table is the average of the five cross-validation tests. The only difference is that in this case, fusion by Fusion Distance outperforms Superposition on several occasions -SOM (20x20), SOM (30x30), ViSOM(30x30)- . The Cancer data set is a binary class problem that contains many more samples, and one class is especially denser than the other. As it is a completely different data set- and specially with binary class -is not surprising that in some cases one algorithm outperforms the other, without any of them being clearly superior.

Table 5.6 Classification accuracy of the different models obtained from a SOM and ViSOM ensemble on the Cancer data set

	Best Single Map	Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (10x10)	92%	96%	95%	93%	93%
SOM (20x20)	77%	95%	74%	87%	85%
SOM (30x30)	69%	92%	70%	82%	76%
ViSOM (20x20)	94%	96%	96%	94%	94%
ViSOM (30x30)	94%	97%	94%	95%	95%
ViSOM (40x40)	91%	96%	94%	92%	92%

5.4.4 Comparison between Bagging and Boosting as Ensemble Training Algorithm

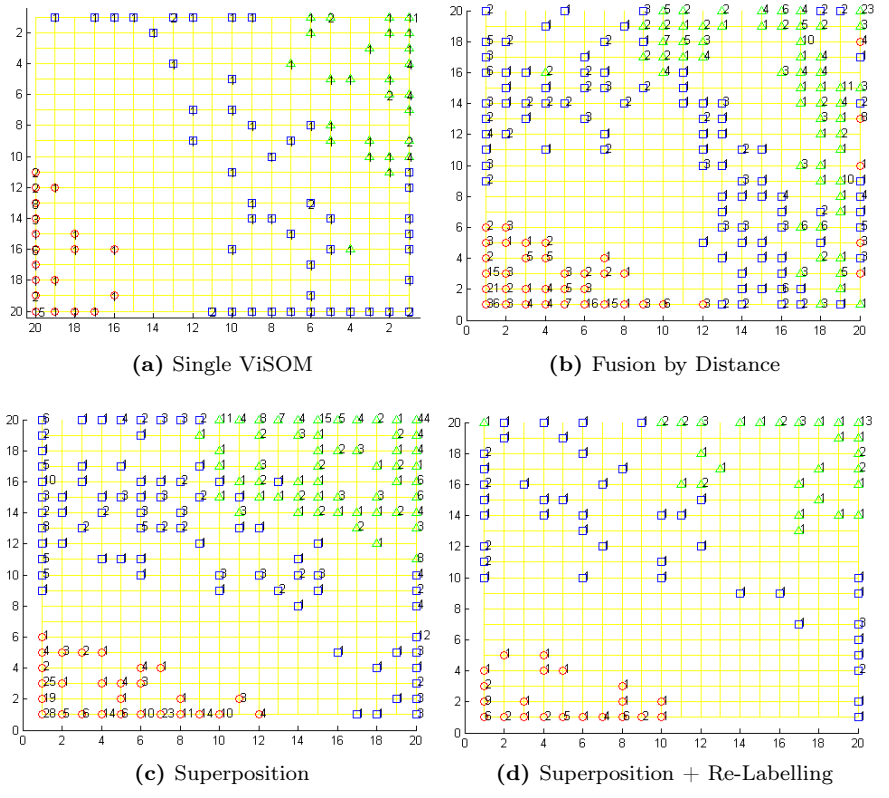
This experiment 5.7 aims to determine whether the way the ensemble is trained has a direct and noticeable effect, both on the ensemble and on the ensuing fusion map calculated from it. This comparison will be performed between two of the most spread algorithms for ensemble construction: Bagging and AdaBoosting. Further experiments regarding this issue can also be found in (Corchado et al, 2007).

As in previous experiments a visual representation of the results is shown in Figure 5.9. All maps were trained using the same parameters. The ensemble was constructed using the AdaBoost.M2 algorithm (Freund and Schapire, 1996). Its has 5 maps, each trained with 120 samples from the Iris data set.

Visually, the difference between using one algorithm or another is hardly significant. No special features can be distinguished by the naked eye, when comparing Figure 5.9 with Figure 5.8, which was trained using the Bagging algorithm. The use of different algorithms for training the ensemble hardly has a drastic effect on the visualization results.

With regard to the classification results, the four tables shown (Table 5.7 to Table 5.10) present the accuracy results obtained for two different data sets by the two different algorithms for ensemble training -Bagging and AdaBoost-. All results are the average obtained form a 10-fold cross validation.

Table 5.7 and Table 5.8 show the classification differences when using Bagging and AdaBoosting for ensemble construction with the Iris data set. The results appear to contradict the idea that the AdaBoost would further improve the ensemble classification results. On the basis of average improvements in accuracy in both tables, for each of the ensemble models over the



(a) Single ViSOM **(b)** Fusion by Distance

(c) Superposition **(d)** Superposition + Re-Labelling

Fig. 5.9 Maps obtained by a single ViSOM and the three Fusion algorithms over the Iris data set. The ensemble was trained using the AdaBoost.M2 algorithm

Table 5.7 Percentage of correct recognition of samples in the Iris data set training the ensemble with Bagging algorithm

	Best Single Map	Ensemble	Super-position	Superp. + Re-labelling	Fusion by Dis-tance
SOM (5x5)	94%	97.33%	84%	94%	94.6%
SOM (10x10)	73.3%	97.3%	80.6%	82%	83.3%
SOM (20x20)	48.6%	90.6%	66.6%	69.3%	58%
ViSOM (10x10)	93.3%	92.6%	61.3%	87.3%	94%
ViSOM (20x20)	87.3%	97.3%	80.6%	89.3%	94%

Table 5.8 Percentage of correct recognition of samples in the Iris data set training the ensemble with AdaBoost.M2 algorithm

	Best Single Map	Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (5x5)	95.3%	97.3%	77.3%	90.6%	96.0%
SOM (10x10)	75.3%	95.3%	80.6%	83.3%	83.3%
SOM (20x20)	58%	90%	71.3%	71.3%	61.3%
ViSOM (10x10)	92%	94.6%	79.3%	97.3%	90.6%
ViSOM (20x20)	86.6%	94.6%	86.6%	88.6%	88.6%

Table 5.9 Percentage of correct recognition of samples in the Wisconsin Breast Cancer data set training the ensemble with Bagging algorithm

	Best Single Map	Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (5x5)	97%	97%	89.4%	96.1%	96.2%
SOM (10x10)	93%	96%	93.4%	93.3%	94.7%
SOM (20x20)	77.6%	96%	83.4%	90%	82.4%
ViSOM (10x10)	94.1%	97%	96.1%	93.6%	94.4%
ViSOM (20x20)	80.9%	95.3%	92.3%	83%	84%
ViSOM (30x30)	77.3%	93.3%	91.7%	78.3%	79.6%

best single map, it may be concluded that, except in the case of the Superposition, the use of AdaBoost does not improve the classification accuracy of the ensemble. Although the AdaBoost ensemble does improve the classification results with respect to the single map, it does not outperform the improvements of the Bagging ensemble.

Table 5.9 and Table 5.10 show the classification differences when using Bagging and AdaBoosting for ensemble construction with the Wisconsin Breast Cancer data set. In this case, having calculated the average improvements of the ensemble-based methods, the results favour the AdaBoost over the Bagging algorithm for the ensemble training.

It should be noted that as the two data sets have a different number of classes, a different variant of the AdaBoost was used in each case. The variants were chosen according to the authors' recommendations on the number

Table 5.10 Percentage of correct recognition of samples in the Wisconsin Breast Cancer data set training the ensemble with AdaBoost.M1 algorithm

	Best Single Map	Ensemble	Super-position	Superp. + Re-labelling	Fusion by Dis-tance
SOM (5x5)	96%	95.8%	93.8%	96%	96.7%
SOM (10x10)	91.6%	96.4%	94.3%	92.8%	94.5%
SOM (20x20)	79.6%	96.6%	94.4%	86.1%	89.3%
ViSOM (10x10)	85.4%	96.9%	93.8%	94%	93.6%
ViSOM (20x20)	84.9%	96.3%	95.1%	86.8%	87.7%
ViSOM (30x30)	77.9%	95.1%	93.3%	83.8%	84.5%

of classes to distinguish in the data set. As the Iris data set contains three classes, the AdaBoost.M2 algorithm is considered the most appropriate one; whereas the AdaBoost.M1 was selected for the binary classification of the Cancer data set. It is also important to note that these differences in the algorithms should theoretically obtain better results if used with the appropriate data set.

The cause of these contradictory results would appear to lie in the difficulty of obtaining a good balance over the diversity of the ensemble. As explained in Chapter 3, for a really useful ensemble, it is important to generate an ensemble of single models that have sufficient diversity to produce results that complement each other. In the case of the fusion of topology-preserving maps, this balance should be observed even more strictly. When using an ensemble of maps for classification, a certain degree of diversity is advantageous. If the intention is to fuse them into a final map, this diversity should not exceed a certain limit. If the trained maps are too dissimilar, the risk is that the maps that are to be fused might exhibit very different characteristics, so much so that they will neither be clearly nor correctly represented in a single, uniform map. As pointed out before, when training the maps in the ensemble, it is important to try to do so with the most similar characteristics as possible, with the exception of the data sub-set. In this case, the difference in how the data set is selected clearly influences the training of each map, such that the training of the ensemble has an added degree of freedom in which the single maps can differentiate. As the results suggest, this is not necessarily a bad result, but it is definitively a point to be taken into account when planning ensemble training, especially in cases where the nature of the data set requires a very low or a very high degree of diversity.

5.4.5 Food Industry Application

The final set of experiments involved an industrial case study: the application of topology-preserving maps for visualisation and assessment to the classification of a data set of Spanish cured hams -See Appendix A for a detailed description-.

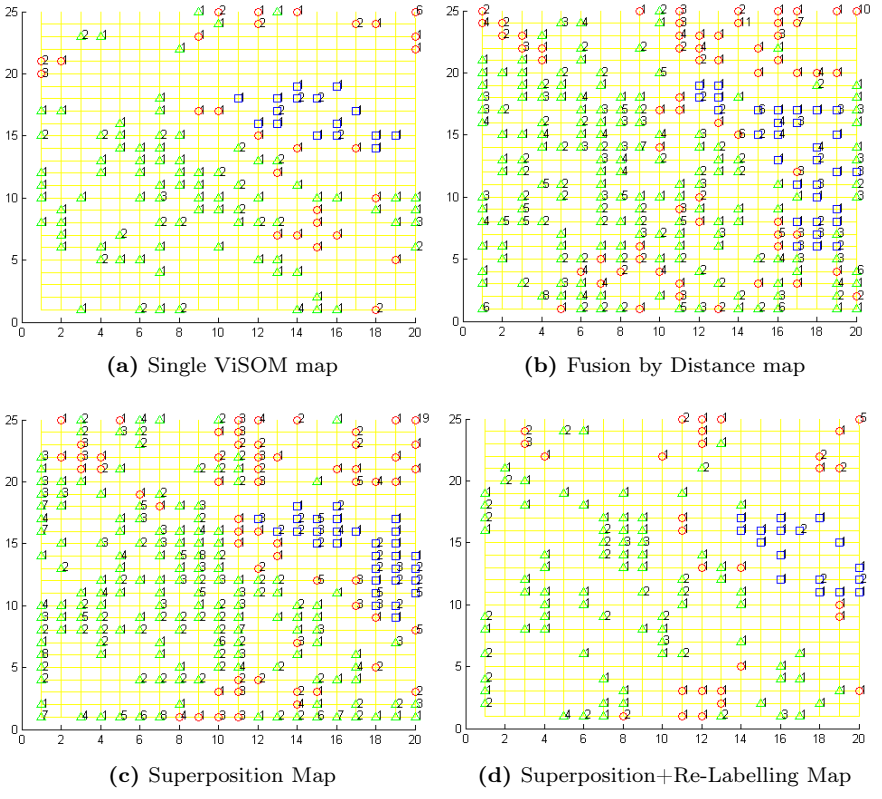
As explained in the description of the data set, the main objective is to obtain a meaningful representation of the data set, so that an intuitive idea of the quality of the ham samples can be presented as a visual image. The analysis is also intended to assess the classification of new samples by human experts. To do so, the intention is to represent the new samples in the map so experts can have a clear idea of the location of the new samples in relation to the data that they have previously studied in a very intuitive way.

Figure 5.10 shows the results of applying the different models to a simplified version of Ham data set 1 described in Appendix A. The data set is exactly the same, although for reasons of clarity the samples are labelled with a simpler classification. Only three classes are used: 'unspoilt', represented both by number '1' and green triangles; 'acid/rancid' represented both by number '3' and red circles and 'tainted', represented both by number '2' and blue squares. The four maps are obtained from VISOM models with a size of 30x30. The ensemble fusion maps have been calculated by using 10 bagged models.

Analysis and visualisation are quite promising in Figure 5.10. In all figures it can be seen that the 'tainted' samples -represented as blue squares- appear in a compact group to the middle-right of the images. In the single SOM (Figure 5.10a) this group appears quite isolated from the rest, except on the bottom-left part of the group, where it is quite near a group of normal samples. Superposition (Figure 5.10c) also presents this group as isolated from the rest. In this case the ordering of data is even clearer as all neurons surrounding the 'tainted' area are neurons that recognise 'acid/rancid' smells. Superposition + Re-Labelling (Figure 5.10d) presents the same isolated 'tainted' area, along with more clear groupings among the 'unspoilt' general cloud. Observing Figure 5.11, which shows a more expressive classification of this data over the same map showed in Figure 5.10d, it can be confirmed that those are meaningful groups, as they mainly group together samples of similar quality.

Classification accuracy is also measured for this experiment (Tables 5.11 and 5.12). Unlike the previous experiment, on this occasion two different ways of training the ensemble are compared. One ensemble has been trained using the Bagging algorithm, while the AdaBoost algorithm was used to train the other. As explained in Chapter 3, this second algorithm uses the classification information of the previously trained classifier in an attempt to improve the accuracy of the next one.

The results are presented in Table 5.11 and Table 5.12. All measures are the average of using a 10 fold cross-validation over the Ham data set. The



(a) Single ViSOM map **(b)** Fusion by Distance map

(c) Superposition Map **(d)** Superposition+Re-Labelling Map

Fig. 5.10 Visual Comparison of the Single ViSOM and the three presented Fusion algorithms using the ham data set

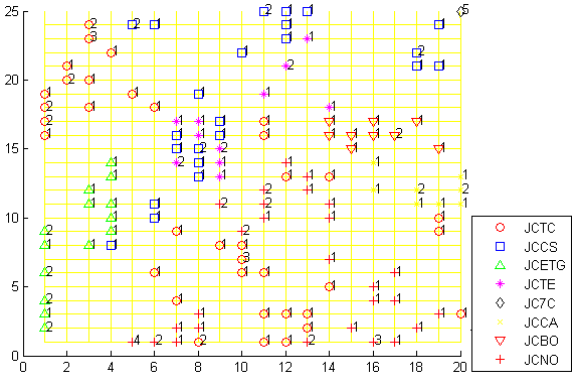


Fig. 5.11 Same Superposition+Re-Labelling as previously shown with more detailed labelling of neurons

Table 5.11 Model classification accuracy over the Ham data set training ensembles with the Bagging meta-algorithm

	Best Single Map	Bagged Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (10x10)	68.4%	84.1%	73%	69%	78.4%
SOM (20x20)	49.5%	78.7%	62.2%	57.7%	59.3%
SOM (30x30)	40.8%	67.2%	49.1%	52.6%	52.1%
ViSOM (20x20)	78%	90%	87.1%	83.9%	79.7%
ViSOM (30x30)	69.1%	82.8%	81.5%	61.7%	60.4%
ViSOM (40x40)	62.5%	86.9%	52.7%	70.4%	71.3%

Table 5.12 Model classification accuracy over the Ham data set training ensembles with the AdaBoost.M2

	Best Single Map	Bagged Ensemble	Super-position	Superp. + Re-labelling	Fusion by Distance
SOM (10x10)	70.2%	82.9%	71.7%	73.4%	78%
SOM (20x20)	47.8%	78.3%	62.6%	61.7%	62.6%
SOM (30x30)	38%	66.7%	58.7%	49.4%	51.7%
ViSOM (20x20)	80.9%	91.1%	81.1%	82%	84.6%
ViSOM (30x30)	68%	90%	72.1%	70.9%	75.5%
ViSOM (40x40)	64.9%	85.6%	75.8%	66.3%	71%

ensembles are all composed of 10 networks, altough trained using a different ensemble meta-algorithm. The performance is very similar in all cases and the map size is correct, so few tests were performed with the more dubious parameters in order to ascertain whether the ensembles could contribute any significant improvement.

Regarding the automated classification of samples, the ViSOM outperformed the SOM in almost all cases, when training the ensembles with the Bagging and with the AdaBoosting algorithms. This behaviour was expected as the ViSOM is devised to adapt to the data set in a more accurate way.

Comparing Tables 5.11 and Table 5.12, it can be inferred that the use of the AdaBoost algorithm has a noticeable effect on the classification accuracy

of the generated ensemble. Calculating the average increase in accuracy of the ensemble over the best single map -which is calculated by subtracting the value of the first column from the second one and averaging the results-, it can be seen that that difference was hardly noticeable: being 20.2% for the bagging algorithm, and 20.8% for the AdaBoost algorithm. In the case of the fusion algorithms the average improvement for Fusion by Distance is 5.5% when using the Bagging algorithm and 8.93% when using the AdaBoost, which is almost a 5% improvement in accuracy.

Regarding the different ensemble fusion algorithms, the results are comparable to those obtained in previous experiments. The accuracy of the ensemble of maps is much higher than its equivalent single and combined networks. The three fusion models are much more dependent on individual performances, so they obtain much lower classification accuracy than the ensemble, but still achieve significantly better classification rates than the single model. AdaBoost obtains better results than Bagging in most cases, although the difference is not very significant.

5.5 Conclusions

This chapter has presented ensemble fusion as a viable option for the combination of ensembles of topology-preserving models. The aim of these kinds of algorithms is to obtain a single map that can outperform the results obtained by the single map method, but which still retains the simplicity of handling a map instead of an ensemble. This is particularly useful for data visualisation of the maps.

Three different algorithms for map fusion have been described and discussed. Two of them had already been presented in earlier works and the third is presented in this thesis as an algorithm that overcomes certain problems observed with the previous ones. Each has its own advantages and disadvantages, as they were all designed to enhance different characteristics of the maps.

Several data sets have been used to test and compare all three of them, in order to understand their different capabilities more fully.

Chapter 6
A Novel Fusion Algorithm for Topology-Preserving Maps

6.1 Introduction

As observed in Chapter 5, few methods have been proposed to fuse the components of an ensemble of topology-preserving maps into a final map that summarizes the information of the different composing maps(Georgakis et al, 2005; Saavedra et al, 2007). Nevertheless, as previously explained, there are multiple advantages to this approach.

Seeking further improvements to the results, two different approaches have been followed and are presented in the rest of this chapter. Firstly, an algorithm combining two of the previously proposed models -Fusion by Distance and Fusion by Similarity- is presented in Section 6.2.

Subsequently, an improved version of the Superposition algorithm is presented in Section 6.3.1.

6.2 Fusion by Ordered Similarity

In an attempt to overcome the different problems detected in the previously proposed map-fusion models, a combination -Fusion by Voronoi Polygon Similarity and Fusion by Euclidean Distance- was devised and is included in the comparative study of the different models.

The Fusion by Voronoi Polygon Similarity algorithm can obtain a very good adaptation of the map neurons to the data set, with a very low quantization error. This characteristic can be very useful when the objective of the training is to learn and to represent the topology of a 2-D data set. However, it is not a great help when trying to represent the inner structure of a multi-dimensional data set in a 2-D map, as a lot of neighbouring information between neurons is disregarded. Neurons that do not recognize any data entry are left out of the final map, and their positions are replaced by blank spaces. One possible way to improve this situation is to use one of the two previously discussed criteria for determining the two closest neurons to

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 95–122.
springerlink.com

fuse, depending on the situation. The concept is to calculate the fusion on a neuron-by-neuron basis again, as described in Chapter 5, Sections 5.3.1.1 and 5.3.1.2.

This variation consists in relying on two different strategies to find suitable neurons to fuse. Firstly, a search is made for the most similar neuron in the Voronoi Similarity Polygon (see Eq. 5.2) in order to identify a suitable neuron to fuse in a map from the ensemble. If the neuron in the fused map does not recognize any data entry, then the search for the nearest neuron by Euclidean Distance is performed. The fusion of neurons is done as in all other fusion methods, by calculating its centroid. Thus, the blank spaces that the Voronoi similarity fusion method left, are filled with neurons that do not recognize any data; but they preserve the neighbouring neurons and therefore, the topology. The approach in question is defined and tested for the first time in this doctoral book. This criterium will be referred to henceforth as 'Ordered Similarity'.

The computational complexity of this algorithm is quadratic; as the other two fusion methods are on which it is based (see Chapter 5). Firstly, a calculation of the dissimilarity between all neurons in all the maps must be performed, which is $O(N^2)$, in which, N is the total number of all neurons in all the maps. Then the calculation of dissimilarities between all neurons must be determined, in which, once again, the complexity is expressed as $O(N^2)$. Finally, calculation of dissimilarities between fused neurons, to determine the new neighbourhood has a complexity of $O(C^2)$, in which C is the number of fused neurons.

6.3 The Weighted Voting Superposition Algorithm

As discussed in earlier chapters, previously presented fusion algorithms have been devised with different purposes in mind, such as data classification or topology learning. This entails several weaknesses when those fusions are used for data visualization; a function that was not foreseen by the authors of the aforementioned fusion algorithms.

This book has previously presented and discussed an initial fusion algorithm called Superposition in the preceding Chapter 5, in an attempt to obtain a suitable model for the purposes of data visualization; one of the most characteristic features of topology-preserving maps.

In this section a novel algorithm for performing this task is presented. It is based on the Superposition algorithm also proposed in Chapter 5. The intention behind this algorithm is to improve on previously presented models that relate to data visualization; one of the most characteristic features of topology-preserving maps.

6.3.1 WeVoS Algorithm

Simplifying things, the idea of Superposition is to consider each individual map neuron as a final result of the competitive learning process. Therefore, finding the arithmetic average of its weights vector could be seen as classic averaging of the final results and could therefore be performed, for example, by the original Bagging algorithm.

In line with the idea of combining the ensemble results, another very common technique for the combination of results for the components of the ensemble is weighted voting. Weighted voting is based on *a priori* knowledge of a component's performance, so that its vote will have greater or lesser weight than others depending on how well it performs.

Although topology-preserving maps make use of unsupervised learning, there are many measures (Lampinen and Oja, 1992; Kaski and Lagus, 1996; Kiviluoto, 1996) to assess the quality of the map that is generated. Some of those measures have been presented in detail in Chapter 2 and are briefly revised in Section 6.4.3. Many of them can be decomposed to obtain a measure of the quality -or error- of each individual unit of the map. Using this measure, there is a way to determine the quality level of each of the components in a set of neurons or, in other words, a criterium to determine the degree to which the training of each neuron is either better or worse than the other neurons in a given data set.

Applying this idea to the Superposition algorithm, the weights of the neuron in the same position in each ensemble map are taken into consideration when calculating the weights of a specific neuron in the fused map. In a similar way to the weighted combination of results in the ensembles, all of them enter into the calculatation of the final weights of the fused neuron, but the weight of each one will be proportional to its quality. This informed decision on the quality of each neuron is expected to improve on the results obtained by the other fusion algorithms.

Many of the aforementioned measures are in fact error calculations, and many of those same measures are calculated with the training data set. The number of recognized samples is also taken into account, in order to distinguish between neurons that have a low error because they are activated by only a few samples and those that despite being activated by many data samples obtain a low error rate.

The mathematical expression for neuron vote weighting is expressed in Eq. 6.1.

$$V_{p,m} = \frac{\sum b_{p,m}}{\sum_{i=1}^{M} b_{p,i}} \cdot \frac{\sum q_{p,m}}{\sum_{i=1}^{M} q_{p,i}} \tag{6.1}$$

The algorithm making use of this weighted voting for the final arrangement or tuning of the fused map neurons is called Weighted Voting Superposition (WeVoS).

Briefly, the WeVoS algorithm functions in the following way: first of all an ensemble of maps is trained. Then, the chosen quality/error measure is

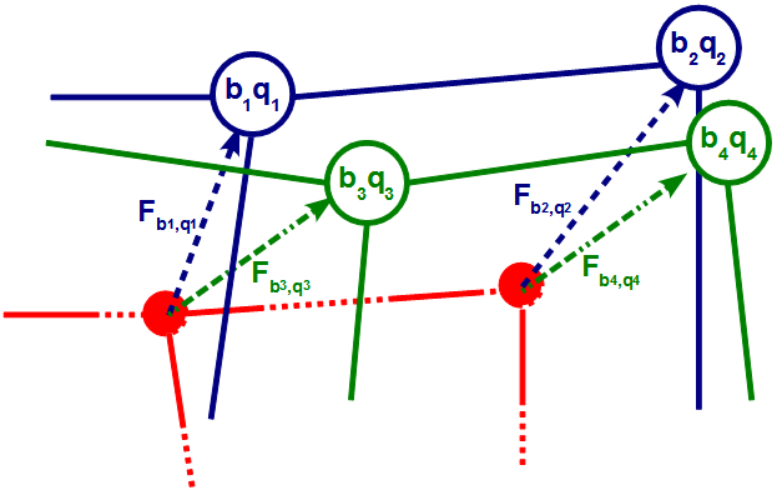


Fig. 6.1 Schematic diagram of the weighted voting in WeVoS

calculated for each of the neurons in all the ensemble maps. The fused map is initialized by calculating the centroids of the neurons in the same position of all the maps, that is, by calculating the superposition of the ensemble. For each of the neurons in the fused map, the average neuron quality as well as the number of total samples recognized in that position for the ensemble maps are calculated. The weight of the vote for each neuron can be calculated with this information by using Eq. 6.1. 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. The learning rate in each case will be the weight of the vote for that neuron. A schematic diagram of this situation is depicted in Figure 6.1, and a detailed description of the algorithm can be found in Algorithm 7.

The computational complexity of this algorithm, although higher than in the case of the Superposition algorithm described in Chapter 5 (Section 5.3.2), is lower than the computational complexity of the other fusion algorithms with which it has been compared. Firstly, the quality of each neuron must be calculated, which in the majority of the quality measures presented so far also depends on the data set used for training. So the computational complexity of that first stage is $O(M \times (N + D))$, where M is the number of maps, N is the number of neurons on each map and D the size of the data set. Then, a calculation of the new position for the fused neuron is calculated taking the weights of the neurons in the other maps into account, but only in the same position, which has a complexity of $O(M \times N)$. This is much less complex than the quadratic expressions in the previously described algorithms, as it is expected that $M \ll N$.

Algorithm 7. Weighted Voting Summarization algorithm

Input: Set of trained topology-preserving maps: $M_1 ... M_n$, training data set: S

Output: A final fused map: M_{fus}

1: Select a training set $S = \langle (x_1, y_1) \dots (x_m, y_m) \rangle$
2: train several networks by using the bagging meta-algorithm : $M_1 ... M_n$
3: **procedure** WEVOS($M_1 ... M_n$)
4: **for all** map $M_i \in M_n$ **do**
5: calculate the quality/error measure chosen for ALL neurons in the map
6: **end for**
▷ These two values are used in Eq. 6.1
7: calculate an accumulated total of the quality/error for each position $Q(p)$
8: calculate recognition rate for each position $B(p)$.
9: **for all** unit position p in M_i **do**
10: initialize the fused map (M_{fus}) by calculating the centroid (w_c) of the neurons of all maps in that position (p) Eq. 5.1
11: **end for**
12: **for all** map $M_i \in M_n$ **do**
13: **for all** unit position p in M_i **do**
14: calculate the vote weight (V_{p,M_i}) using Eq. 6.1.
15: feed the weights vector of neuron w_p into the fused map (M_{fus}) as if it was an input to the network.
The weight of the vote (V_{p,M_i}) is used as the learning rate (α).
The position of that neuron (p) is considered as the position of the BMU (v). ▷ This causes the neuron of the fused map (w_p^*) to approximate the neuron of the composing ensemble ($w_{p,m}$) according to the quality of its adaptation.
16: **end for**
17: **end for**
18: **end procedure**

6.3.2 Discussion

As upheld by the results, the Fusion by Ordered Similarity algorithm improves the results of the other two algorithms on which it is based, with regard to those aspects in which they each perform worse. For example, it enables the fused map to retain a certain topology, which Fusion by Similarity does not do, and it reduces the Distortion that in some cases arises with Fusion by Distance. Its problem is that it is unable to outperform any of the best characteristics of the other models. This situation results in a fusion model that is not really suitable for any of the proposed tasks.

The WeVoS algorithm appears to be a more interesting algorithm. It relies on the way the ensemble composing maps are trained in order to indirectly force commonality between maps, it does not need to search for available units to fuse all over the map, which results in lower complexity. It uses the ensemble algorithm to initialize, and, especially, to assess neuronal positions, rather than to calculate a completely different position for them, as the other two fusion algorithms do. It also preserves one of the most characteristic features of the topology-preserving maps by taking the neighbouring function of this kind of maps into account. As shown by the results in Section 6.5, this algorithm represents a useful improvement on topology-preserving map analysis of different data sets. Based on the ensemble model, it is also recommended for data sets with very few or highly dispersed entries.

The weighted voting scheme for fine tuning or adapting the final weights of the neurons of the map creates a framework that could include many variations. For example, instead of using a single measure for calculating this weight, a linear combination of more than one could be useful. Interesting variations of the algorithm proposed in this book could be based on adaptations of how the other fusion algorithms work. Euclidean distance or Voronoi polygon similarity calculations could be included in the weighting of the vote without too much difficulty. This could help to construct an even more informative decision about the final inter-neuronal weights of the maps, as it would add further information on whether the weights of a specific neuron deviate excessively from the other neurons in that position, and on whether their inclusion in the voting would negatively affect the final fused neuron.

Although these are interesting ideas to explore, it is difficult to define the degree to which the performance of the final map will either be improved by these modifications -which would also increase the computational complexity- or will remain unchanged. The inclusion of too many criteria for the adaptation of the final neurons also risks increasing the instability of the final calculations step, resulting in poorer results.

6.4 Application of WeVoS to Different Models

The WeVoS algorithm is a fusion algorithm for ensembles of topology-preserving maps. Its only initial condition is that the maps composing the ensemble should be reasonably similar. Several experiments were conducted to test the effects of employing the WeVoS algorithm in combination with different models to analyse a number of data sets. In the experiments that applied the WeVoS to different models, a map was initialised with the final weights of the previously trained map. The WeVoS algorithm neither constrains the algorithm used to train the maps, nor the algorithm used to train the ensemble. A brief overview of the different possible combinations is included in this section.

6.4.1 Topology-Preserving Models

The WeVoS algorithm is generally applicable to many members of the topology-preserving-map family of algorithms. The experiments to assess its capabilities were performed over the following algorithms.

Among the family of topology-preserving maps, the best known is the Self-Organizing Map (SOM) (Kohonen, 1995). The main application of this algorithm is to perform dimensionality reduction that enables it to present multi-dimensional data sets as 2-D maps that provide a simple visualization of the data set. The characteristic feature of these maps is that they are topographically ordered, so data samples that are similar in the input multi-dimensional space, will appear close in the final map. In its initial form, the SOM was a general purpose algorithm, applicable to any data set.

One interesting modification of the SOM is the Visualization Induced SOM (ViSOM) (Yin, 2002b) algorithm. This modification of the SOM is designed to improve its data visualization characteristics. It modifies the updating of neuron weights in the training of the map in such a way that they preserve not only the neighbouring relationships between the data they represent, but also the distance between such data. In this way, with just one quick glance at the map, the human operator can gain an intuitive idea, of which samples are similar and of the degree to which that is true. The ViSOM is also a general purpose algorithm, applicable to any data set.

A further modification of the SOM is the Scale-Invariant map or SIM (Fyfe, 1996). This ANN also yields a map as an output. The special feature of this algorithm is that the updating of its neurons is done by using the classic Hebbian learning rule (Hebb, 1949) in the neighbourhood of the winner. Instead of competitive learning, a negative feedback learning approach is used. The result of this type of update is a characteristic circular or spherical shape of this map. Rather than having a polygonal shape, the data space to which each neuron responds, as in the SOM, can be considered a "pie-slice" of the data due to its form. Thus, it is reacting to data regardless of the scale of the map. Due to this characteristic, the map is especially recommendable for use with circular or spherical shaped data sets.

Yet another modification of the SIM is the Maximum Likelihood SIM (or Max-SIM) (Corchado and Fyfe, 2002b), which is based on applying a modification of the Hebbian learning rule for training the neurons -called Maximum Likelihood (Corchado et al, 2004)- that enables the SIM to be adjusted for different data distributions.

6.4.2 Ensemble Models

As a fusion algorithm, the WeVoS algorithm can be applied to any ensemble of topology-preserving maps regardless of the algorithm with which they are trained. Nevertheless, the way the ensemble was trained will, of course, have a decisive effect on the final result.

Two different ensemble training algorithms were used in the experiments: Bagging (Breiman, 1996) and AdaBoost (Freund and Schapire, 1996). The former is one of the simplest and most widely used ensemble algorithms. It consists of training the same model -any machine learning algorithm- in different data sets extracted all from the same source. Usually, these several data sets are emulated by repeated bootstrapped re-samplings over the data set under analysis. Each of these re-sampling steps is done in a completely independent way to the other steps.

AdaBoost is based on the same idea, but tries to improve the learning of the subsequent models -initially classifiers- by obtaining information on the performance of the previously trained ones. Once a classifier has finished its training, its performance is evaluated in the training data set. This enables the AdaBoost algorithm to find samples for which the trained classifier performance is worse, and it will place special emphasis on those samples when training the next model. In that way, classifiers will be trained to perform well in specific areas where others are not performing as well, so they will better complement each other when working as an ensemble.

6.4.3 Quality Measures

Several quality measures for topology-preserving maps were used to compare the results of WeVoS with other similar algorithms. As previously explained -see Chapter 2- no agreed measure exists that determines the overall quality of a map. Instead, many measures have been proposed in the literature, each of which concentrate on a different feature of the SOM family.

The experiments shown in Section 6.5 featured several of those measures:

Distortion (Lampinen and Oja, 1992): can be used to calculate the overall topology preservation of a map in an accurate way. It is one of the main features that WeVoS is trying to improve, which reflects its importance to this line of research.

Goodness of Adaptation (Kaski and Lagus, 1996): measuring both the continuity of the mapping from the data set to the map grid and the accuracy of the map that represents the data set, this measure is of interest because it is the most general of all those used in the experiments.

Topographic Error (Kiviluoto, 1996)*:* one of the simplest topology-preservation measures, this measure is included in the experiments for the sake of completeness.

Mean Quantization Error: Quantization Error is related to all forms of vector quantization and clustering algorithms. It completely disregards map topology and alignment and, once again, is included in the experiments for the sake of completeness.

Classification Error: Although topology-preserving maps were not initially designed as classifiers, their pattern-matching abilities can to some extent classify new samples. This measure is calculated in few experiments, as it is not directly related to the quality of map representations. It is included

among the experiments because the ensemble meta-algorithms were originally intended to increase the classification accuracy of a single model.

6.5 Experiments and Results

6.5.1 Comparison of Fusion Algorithms over a 1-D SOM

The experiment consisted of training an ensemble of SOMs several times over the same data set, reducing the number of elements in the data set each time. Some outliers were also included in the sample. The objective is to verify the degree to which the maps obtained by the fusion algorithms are affected by the reduction in the number of samples with which they are trained. For each step in the experiment, a 5-fold cross-validation was performed. The data set was a 2-D horseshoe shaped one similar to that used in (Kaski and Lagus, 1996). It has 570 entries in the first step and of 163 entries in the last one.

The Distortion measure obtained in each of the steps for each of the models compared is presented in Figure 6.2.

It may be seen from Figure 6.2 that when the number of samples available for training the algorithm decreases, the Distortion of the entire model obviously increases. The most interesting feature of the figure is the way in which the inclusion of less data in the analysis changes the Distortion. It may be easily observed that WeVoS is the model with the lowest increase in the Distortion, followed by the single SOM and by the other two fusion algorithms.

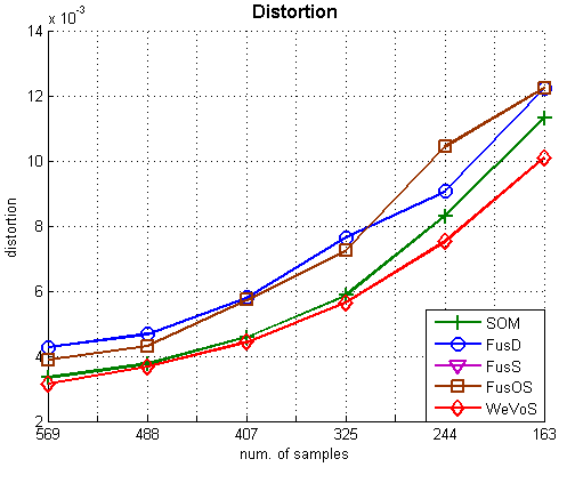


Fig. 6.2 Distortion measured over four different fusion algorithms for a 1-D ensemble of SOMs

This is proof that the use of ensemble meta-algorithms can lead to additional stability when combined with rather unstable models such as the SOM.

6.5.2 Comparison of Fusion Algorithms over the 2-D SOM

The first experiment drew a comparison of the four map fusion algorithms -Fusion by Distance, Fusion by Voronoi Polygons Similarity, Fusion by Ordered Similarity and WeVoS- when functioning in combination with the standard SOM algorithm. three of the most commonly used data sets from the UCI repository (Asuncion and Newman, 2007) were used in this comparative study: the Iris, the Wisconsin Breast Cancer and the Wine data sets.

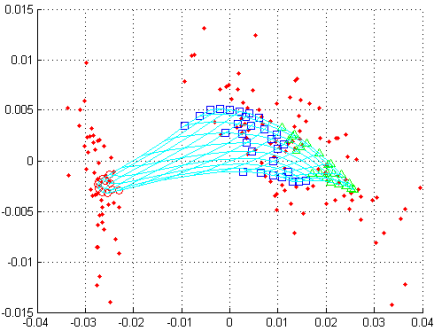
6.5.2.1 Iris Data Set

As in other experiments Figure, 6.3 depicts differences in the map grid structure of the maps under comparison for the single SOM and the four different fusion algorithms. The projections were obtained by calculating the first two Principal Components of the Iris data set and projecting all the data over those two axes. The ensemble used for this visual comparison is shown in Chapter 5 (see Figure 5.4). It is an ensemble of 5 SOMs, all trained with exactly the same parameters over the Iris data set using the bagging meta-algorithm. All three measures have errors, so the closer the result is to 0.0 the better it may be considered.

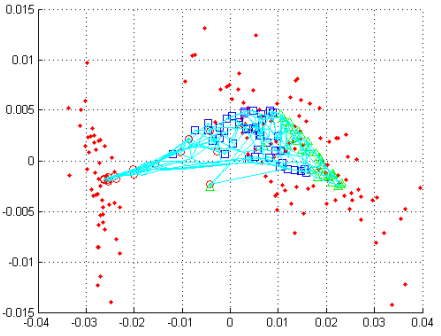
Figures for the SOM (Figure 6.3a), Fusion by Distance (Figure 6.3b) and Fusion by Similarity (Figure 6.3d) are the same as presented in Chapter 5 (Figure 5.4). Fusion by Ordered Similarity (Figure 6.3c) obtains a similar result to Fusion by Distortion, which at first glance can hardly be considered of interest for the purpose of data set visualization for the reasons that have previously been explained. On the other hand, WeVoS (Figure 6.3e) can be seen to generate a clear grid over the data set. Compared with the single SOM, it is more clearly spread out over the data set, especially in the group to the left of each image, which therefore means that the data is more sparsely displayed in the output space of the 2D map.

More analytical results are included in Figure 6.4, which show a comparison between the four models previously compared using several of the most widely used quality measures. The charts represent the evolution of the quality measure with respect to the four models (Y-axis) and the increasing number of maps used in the ensemble (X-axis).

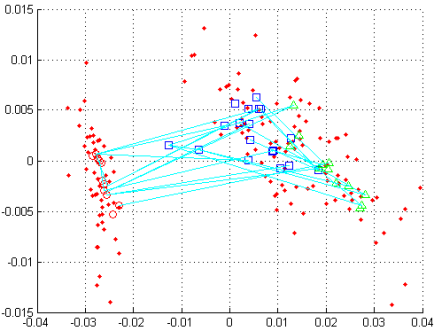
Regarding quantization error (Figure 6.4a), all the ensemble methods obtained higher errors than the single model, except for Fusion by Similarity. This situation was expected, as none of the summarization algorithms were intended to obtain a lower quantization error. On the contrary, one of the



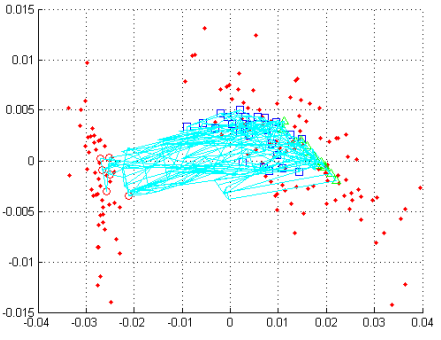
(a) Single SOM



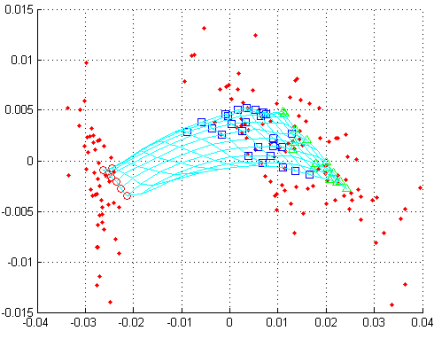
(b) Fusion by Euclidean Distance



(c) Fusion by Voronoi Polygon Similarity



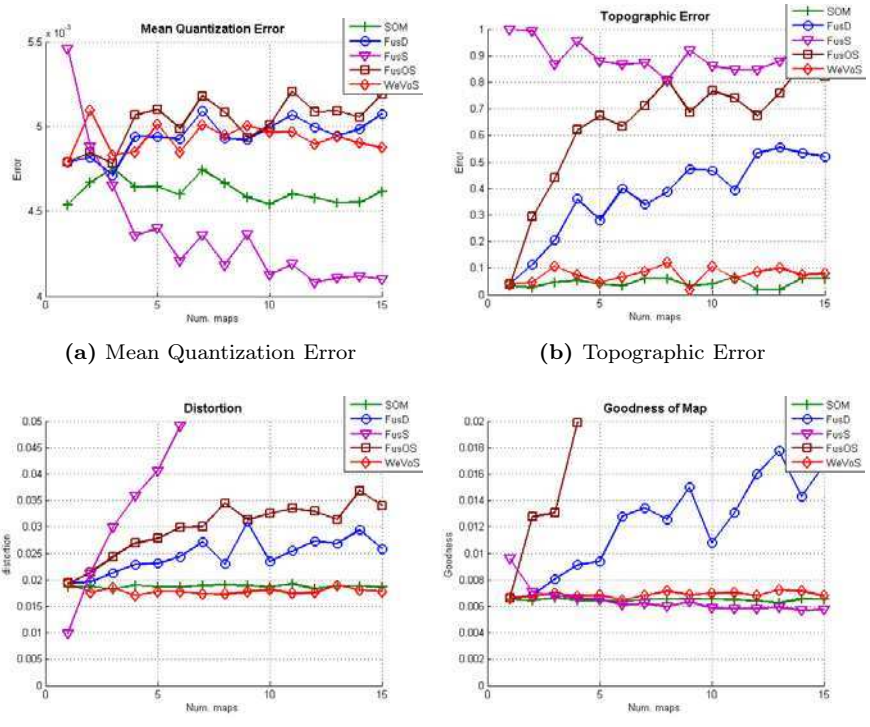
(d) Fusion by Ordered Similarity



(e) WeVoS-SOM

Fig. 6.3 Visual comparison of the five models -four ensemble fusion models and the single model- discussed in the book

ideas behind the algorithms is to reduce the potential overfitting of a single map (Bishop, 1995; Tetko et al, 1995), whether for obtaining better classification accuracy or for a better visual representation of the data set. Hence, an overly low quantization.



(a) Mean Quantization Error

(b) Topographic Error

(c) Distortion

(d) Goodness of Adaptation

Fig. 6.4 The 4 quality measures obtained from the 4 different summarization algorithms and the single SOM, trained on the Iris data set

The case of Fusion by Similarity is different, because neurons with a low activation rate are removed from the final fused map which will have less neurons than the other maps. Therefore, the presence of fewer neurons will mean less error, as fewer neurons will contribute to the total error obtained by the map. The mean is calculated from the data set input and not from the neurons on the map, so the mean obtained with fewer neurons will be lower than other maps with more neurons. Concentrating on measures directly related to visualization, it can generally be observed that the ensemble fusion methods, with the exception of WeVoS, perform worse than the single map.

The three previous fusion algorithms obtained a far higher error for Topographic Error (Figure 6.4b) than the single SOM. The WeVoS obtained a similar error to the SOM, although in most cases it was a bit higher. It is worth noting that this measure is one of the oldest and the most simplistic measures for topographic ordering and it is not widely regarded as a very trustful one.

The results of the following measure -Distortion (Figure 6.4c)- appear to contradict the previous one. In this case, it can be observed that WeVoS consistently presented a lower Distortion measure than the single SOM, which in turn has a lower error than the other three fusion algorithms. In this case the advantage of using the WeVoS algorithm appears to justify the added effort of constructing an ensemble.

Finally, in the case of the Goodness of Adaptation (Figure 6.4d), which as previously stated is a measure that combines the quantization and topographic characteristics of the SOM, both the WeVoS algorithm and the single SOM obtained similar results, although once again the single SOM obtained a slightly lower error. In conclusion, although WeVoS can obtain a map with less Distortion than the single SOM, it records a higher data quantization error than the SOM, as can be seen from Figure 6.4a. This is also the reason why Fusion by Similarity obtained a value that was even lower than the single SOM for Goodness of Adaptation.

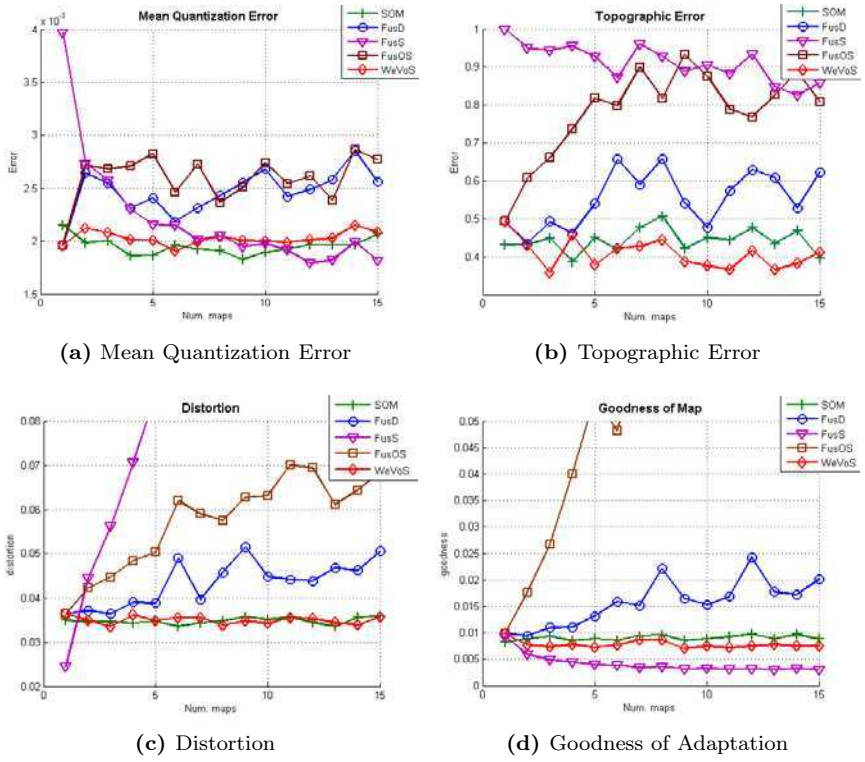
As an overall observation, it is also worth pointing out that the WeVoS obtains quite stable results, which vary slightly depending on the number of maps; in contrast with the other three fusion algorithms, which yield high variations in their results, as in many cases the error increases as the number of maps increase. Obviously the single SOM obtains stable results, as they are always measured from the training of a single map with the same parameters only in slightly different sets of data inputs.

6.5.2.2 Additional Experiments

The same experiment was repeated for the Wine and Wisconsin Breast Cancer data sets, to check if the results presented for the Iris data set were consistent and to investigate the degree to which it may be generalized to other situations. The results are respectively presented in Figures 6.5 and 6.6.

Mean Quantization Error (Figure 6.5a) yielded a slightly different result. Nevertheless, Fusion by Distance and by Ordered Similarity obtained far higher errors than the others, but in this case Fusion by Similarity did not clearly outperform the other fusion methods and only obtained similar results when the size of the ensemble was 5 maps or more. Moreover, WeVoS and the single SOM performed equally well and obtained very similar results.

Regarding Topographic Error (Figure 6.5b), the WeVoS algorithm clearly outperformed the other models, as its Topographic Error was lower than that of the single SOM and much lower than those of the other ensemble fusion methods. On the other hand, the single SOM clearly outperformed the WeVoS algorithm in the case of the Distortion measure, in the (Figure 6.5c) although the former still performed much better than the other ensemble fusion methods. Once again in this case, unlike in the previous experiment, Goodness of Adaptation (Figure 6.5d) performed better in the WeVoS algorithm than in the SOM, and these two results were far better than those of Fusion by Distance and Fusion by Ordered Similarity.



(a) Mean Quantization Error

(b) Topographic Error

(c) Distortion

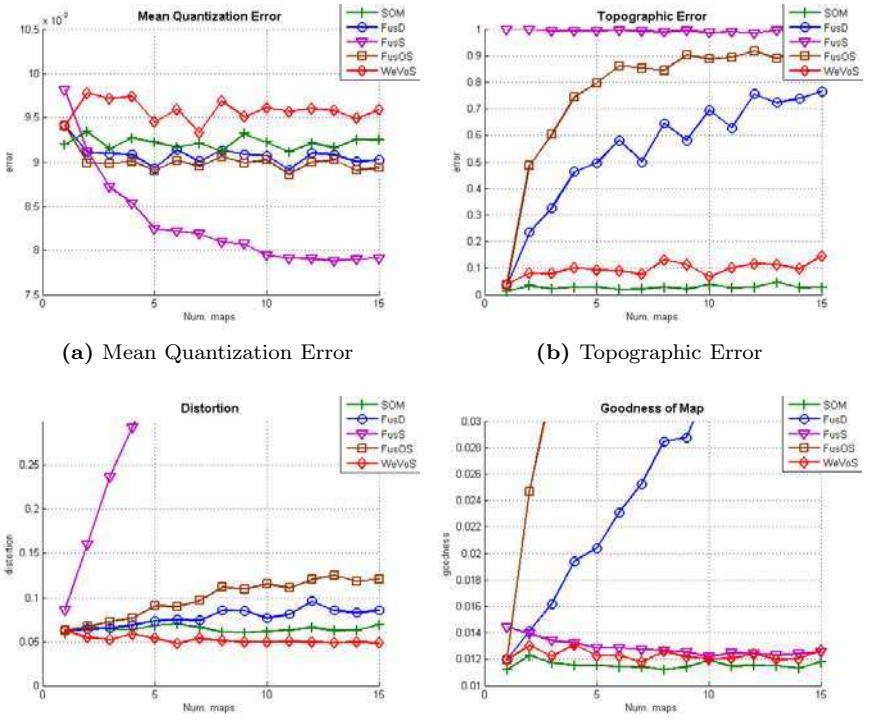
(d) Goodness of Adaptation

Fig. 6.5 The 4 quality measures obtained from the 4 different summarization algorithms and the single SOM, trained on the Wine data set

Moreover, Fusion by Similarity appears to obtain better results than any other model, but as previously explained, the quantization characteristic of this map means that it will always obtain better results, as it contains fewer neurons to add to the error total. The results of Fusion by Similarity differ when compared with the other algorithms as the Goodness of Adaptation measure involves quantization.

Finally, the same experiment was repeated for the Wisconsin Breast Cancer data set and the results are presented in Figure 6.6.

In this case, the results were also consistent with those presented previously. With the exception of quantization error, almost all the same remarks regarding the previous experiments are also applicable to this one. In this case, Quantization error (Figure 6.6a) obtained contradictory results to the two previous experiments. The summarization methods, except for WeVoS, obtained lower error measures than the single SOM. This is probably due to the nature of this particular data set, which is far less sparse than the other two; favouring algorithms that disperse their neurons throughout the input space.



(a) Mean Quantization Error **(b)** Topographic Error

(c) Distortion **(d)** Goodness of Adaptation

Fig. 6.6 The 4 quality measures obtained from the 4 different summarization algorithms and the single SOM, trained on the Wisconsin Breast Cancer data set

In this experiment, Topographic and Distortion errors (Figure 6.6b and Figure 6.6c) were similar to the first one -using the Iris data set- . While Topographic Error is lower for the single SOM, WeVoS is the fusion method that obtains the lowest Distortion. The other fusion algorithms obtain a higher error than these two in both measures.

Finally, in relation to the Goodness of Adaptation (Figure 6.6d), once again the single SOM and WeVoS performed in comparable ways, though the single SOM obtained a slightly lower error. Fusion by Similarity had a slightly higher error than WeVoS in this case, instead of obtaining a lower error than the SOM, as it did in the first case.

6.5.2.3 Overall Remarks

As seen in the analytical results, it can be inferred that the usefulness of the ensemble meta-algorithms, and WeVoS summarization in particular depend to a certain degree on the "sparseness" of the data set under analysis. A "sparse" data set means a data set with a high number of dimensions,

but a relative low number of entries. This "sparsity" is intuitively related to learning complexity and to the extraction of correct patterns from that data. Nevertheless, there are several consistent results that are worth noting.

The WeVoS-SOM algorithm did not obtain lower Quantization error than the Single SOM in any of the experiments, because this model focuses on the visualization of structure more than on the vector quantization capabilities of the maps it obtains. The results for Topographic Error were not conclusive, and were only clearly different in one experiment for the WeVoS-SOM and for the Single SOM. This was to some extent predictable, as this measure is widely considered a very simplistic one. Interestingly, the Distortion measure is the most important measure in this study as it relates more than any other to the visualization feature of topology-preserving maps. It represents the topographical ordering of the maps and was clearly and consistently lower for the WeVoS-SOM than for the single SOM in 2 out of 3 experiments.

The experiment in which that was not true involved the more complicated Wine data set. This data set has the highest number of dimensions (13) and relatively few entries (178) to analyse. Compared with the Iris (only 4 dimensions and 150 entries) and the Cancer (9 dimensions, but 683 entries), the nature of the Wine data set makes its analysis and visualization a more difficult task. On the other hand, Goodness of Adaptation -a measure combining quantization and topographic preservation measurement -yielded rather complementary results to the Distortion measure. This measure was generally quite high for the previously devised ensemble models (Fusion by Distance, Fusion by Similarity) and their combination (Fusion by Ordered Similarity) when compared with the WeVoS-SOM and the Single SOM, which obtained similarly low values. In 2 of the 3 experiments -Iris and Cancer- , the single SOM obtained better results, due mainly to the weaker quantization capabilities of the WeVoS-SOM. But in the third experiment involving the most complex data set in this study -Wine- , the WeVoS-SOM model obtained consistently better results than the Single SOM.

The results in this case are due to a combination of the characteristic low Distortion measure of the WeVoS-SOM and the low Quantization error that it obtained in this particular experiment. As expected, the advantages of using the ensemble meta-algorithm, and particularly the WeVoS-SOM, became increasingly evident in this situation as the data set under study became more complex.

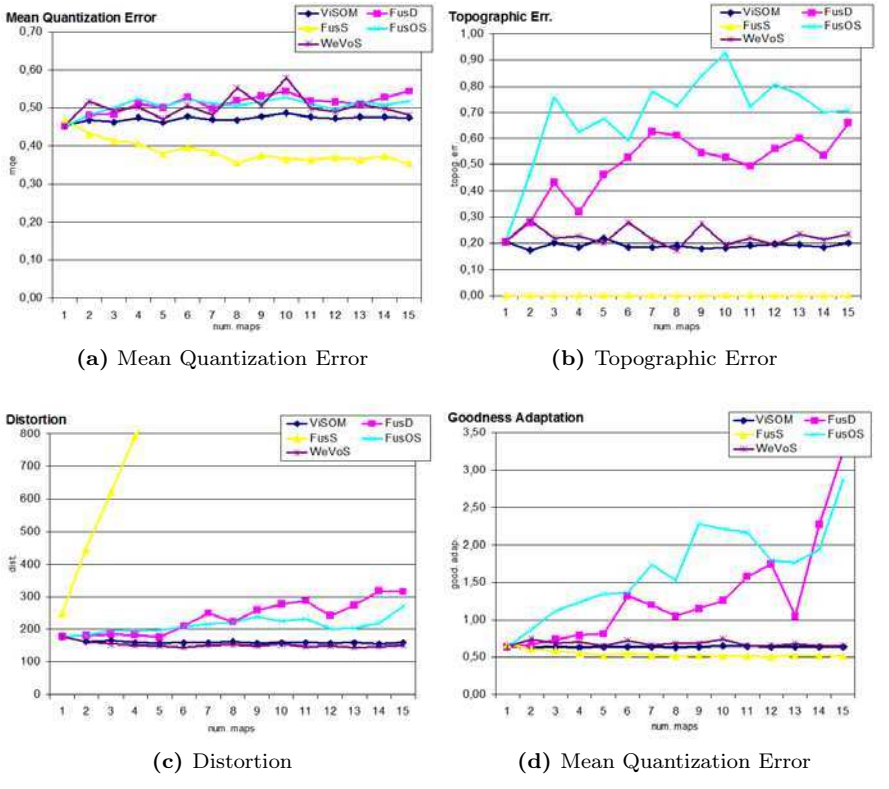
Ordered Similarity, the other fusion method presented in this book, is not really able to prove its worth, obtaining in almost every case higher errors than WeVoS. This algorithm was conceived as an attempt to overcome the problems of two previous fusion methods -Fusion by Distance and Fusion by Similarity- ; its main problem appears to be that it is unable to outperform their best aspects. Despite obtaining better results for neighbourhood preservation than Fusion by Similarity (see Topographic Error and Distortion measures), it never outperformed Fusion by Distance in those same measures. Furthermore, it was unable to outperform the best aspects of Fusion by

Similarity -Quantization error- . Thus, the model obtained the worst results for Goodness of Adaptation.

6.5.3 Comparison of Fusion Algorithms over the ViSOM

A second series of tests were performed to check the degree to which a change in the topology-preserving algorithm used for training the maps would impact on the results generated by the different summarization models.

As expected, the results were very similar to those of the SOM, so, to avoid undue repetition, the explanations are briefly outlined. Only Fusion by Similarity obtained a lower Quantization Error (Figure6.7a) than the average of the single map, which was predictable, as Fusion by Similarity contains fewer units than the other maps. This means that the measure does not directly reflect a better behaviour than the others. Both Fusion by Distance



(a) Mean Quantization Error

(b) Topographic Error

(c) Distortion

(d) Mean Quantization Error

Fig. 6.7 The 4 quality measures obtained from the 4 different summarization algorithms and the single ViSOM, trained on the Wine data set

and WeVoS obtained higher error values than the single map. The summarization models aim to obtain a representation that holds true for the inner structure of the data set as a whole, but less so for data representation in the Euclidean space.

With regard to the topology-preservation measures -Topographic Error in Figure 6.7b and Distortion in Figure 6.7c- , WeVoS-ViSOM clearly achieved a considerably lower error than the other models. Regarding Topographic Error (Figure 6.7b), the measures obtained were lower in most cases than those obtained by the single map. The WeVoS model consistently obtained the lowest error for the Distortion measure (Figure 6.7c).

The Goodness of Adaptation measure (Figure 6.7d) of the model that is presented, which is a blend of the other quality measures, once again yielded predictable results. When increasing the number of maps of the ensemble, the adaptation quality of the map thus obtained diminished in the case of Fusion by Distance, but remained stable for both Fusion by Similarity and WeVoS. The only fusion method with a error lower than that achieved by the single model was Fusion by Similarity, which is due to the very special way in which it is computed. It should not be forgotten that this algorithm yields maps with a lower number of units, thereby affecting the calculation of measures related to the quantization error of the whole map.

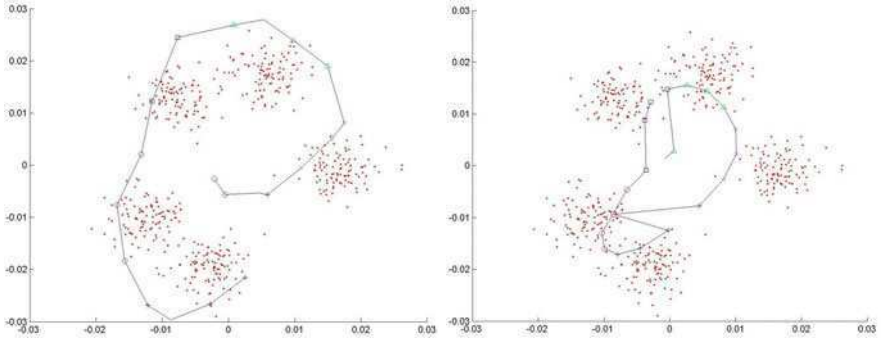
In summary, it is worth noting that although the ViSOM is usually a slightly more unstable algorithm than the SOM -as two different update forces take part in the neuron updating process- the results for the four measures are very close to those obtained by the regular SOM algorithm.

6.5.4 Comparison of Fusion Algorithms over the SIM and Max-SIM

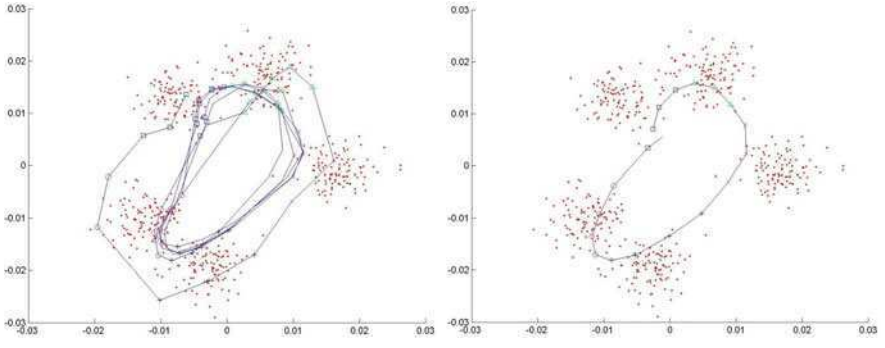
A final additional test was performed to verify whether using the different models in conjunction with the SIM and Max-SIM variants of the topological-preserving maps would yield similar results to those obtained with the previously tested map models.

In this case (Figure 6.8), all models use a circular shape for their adaptation to the data set. The resultant 1-D networks are shown embedded in the 2-D data set. This was expected, as this circular shape is the most characteristic feature of the SIM algorithm. The single SIM (Figure 6.9a) adapts to the data set correctly, but using a slightly too open map. The Fusion by Distance (Figure 6.8b) suffers from an expected problem: several twists appear on its structure. For classification problems that does not seem to be an important issue, but when dealing with visual inspection this approach has the problem of not always preserving strictly the topology of the network.

Concerning the Fusion by Similarity (Figure 6.8c) it can easily be seen that, although the shape is correct, there are too many unnecessary units. This is due to the fact that, with such a big data set, is impossible for a 1-D network to cover the data space correctly, so very few units are related with



(a) Single SIM **(b)** Fusion by Euclidean Distance



(c) Fusion by Voronoi Polygon Similarity **(d)** Weighted Voting Superposition

Fig. 6.8 The single SIM and the three summarizations for the same 6-network ensemble, trained over the circular data set, employing the SIM learning Algorithm

Voronoi polygons that overlap because they recognized mainly the same data entries.

The WeVoS-SIM (Figure 6.8d) obtains an oval shape without any twist or dead neurons, providing the best results. Results yielded by the applications of the WeVoS-SIM are very similar to those obtained by the WeVoS-MaxSIM. This second experiment's corresponding figures are not showed due to the limit of space. Regarding the different summarization models, it can be observed how single models, adapt to the data set correctly, but that their results can be improved by the use of the ensemble meta-algorithms. Among these summarization methods, the WeVoS obtains a simple network, without major twists and obtains a result more adapted to the data set shape than the single model.

In this case (Figure 6.8), all the models have a circular shape for their adaptation to the data set. This was expected, as this circular shape is the most characteristic feature of the SIM algorithm. The single SIM (Figure 6.9a) correctly adapts itself to the data set, but it uses a map that is too open.

Fusion by Distance (Figure 6.8b) suffers from a predictable problem: several twists appear in its structure. This does not seem to be an important issue for classification problems, but the problem of this approach when dealing with visual inspection is that it does not always strictly preserve the topology of the network.

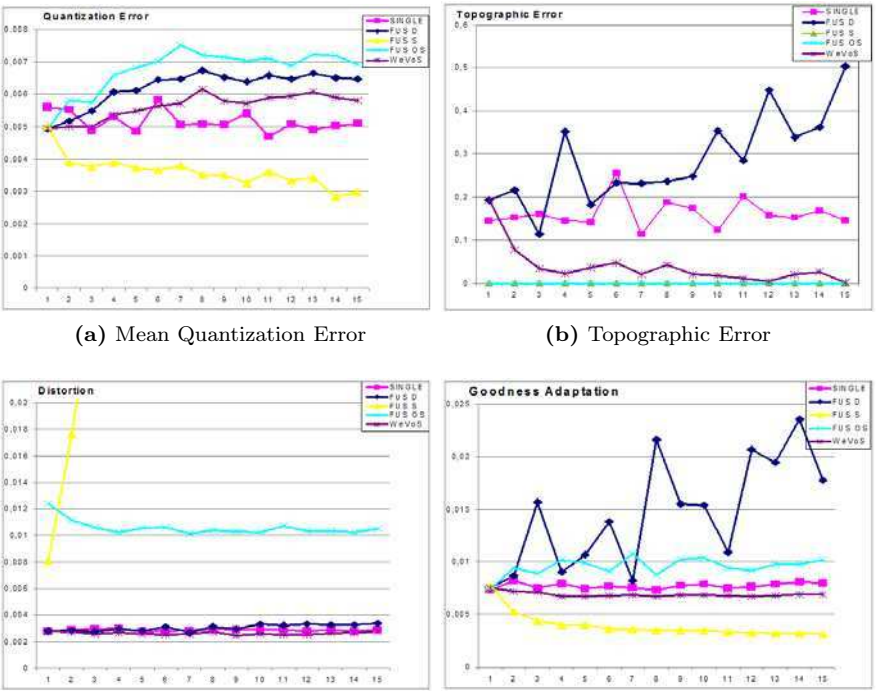
With regard to the Fusion by Similarity (Figure 6.8c), it can easily be seen that, although the shape is correct, there are too many unnecessary units. This is due to the fact that it is impossible for a 1-D network to cover the data space correctly with such a large data set, so very few units are related with overlapping Voronoi polygons because they mainly recognize the same data entries.

The best results were generated by the WeVoS-SIM (Figure 6.8d), which obtained an oval shape without any twists or dead neurons. The results yielded by the applications of the WeVoS-SIM are very similar to those obtained by the WeVoS-MaxSIM. The figures on this second experiment are not shown due to limitations on space. As for the different summarization models, it can be seen how single models adapt to the data set correctly, although their results could be improved by the use of the ensemble meta-algorithms. Among these summarization methods, WeVoS obtains a simple network, without major twists and obtains a result that is more adapted to the data set shape than the single model.

The comparison of quality obtained by each of the four fusion algorithms, calculated over the same ensemble of maps is shown for each measure in each figure (Figure 6.9 and Figure 6.10). They represent variations in the measures when the number of maps included in the summary are increased from 1 to 15. The X-axis represents the number of maps, while the Y-axis represents the measure. As expected, according to the measures on topographic ordering, WeVoS obtained better results than other models both for the SIM and the Max-SIM. The exception to this is Fusion by Similarity, the results of which were altered and were not directly comparable, as the number of units they contained were different from the rest. The other three models behave in a more consistent way, and the WeVoS model obtained the lowest error rates. As already made clear, the meta-algorithm's main purposes are preservation of neighbourhood and topographic ordering. In confirmation of these results,again with the exception of Fusion by Similarity, WeVoS obtained better results than any other model for Goodness of Adaptation of the map.

6.5.5 Comparison of Fusion Algorithms When Combined with Boosting

The objective of the following experiment was to test the degree to which the meta-algorithm used for the training of the ensemble would impact on the final results of the fusion algorithms. Some results of similar experiments can be also found in (Baruque et al, 2009).



(a) Mean Quantization Error **(b)** Topographic Error

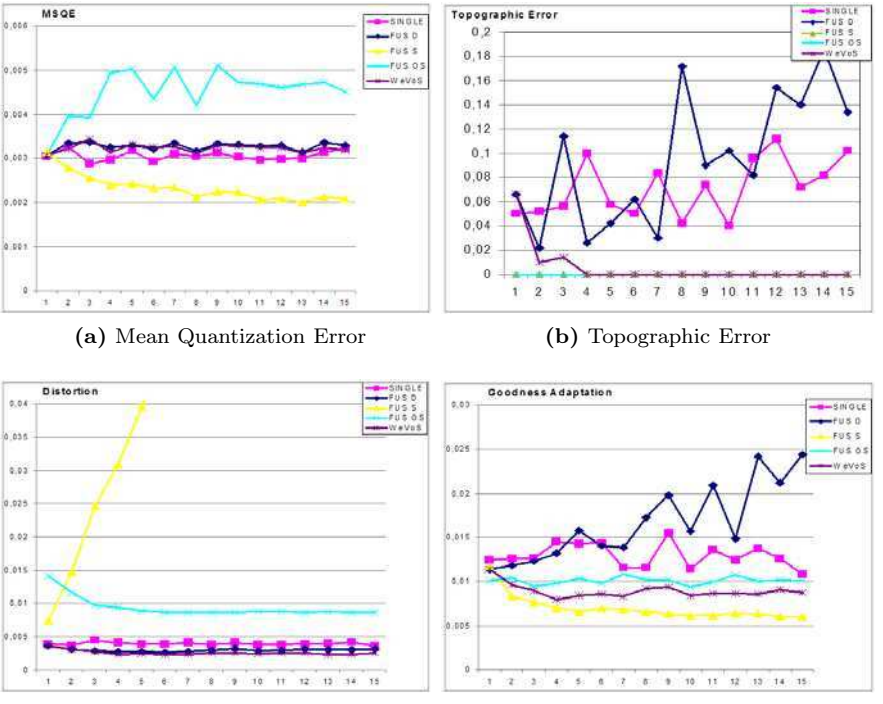
(c) Distortion **(d)** Goodness of Adaptation

Fig. 6.9 The 4 quality measures obtained from the 4 different summarization algorithms and the single SIM, trained on the artificial circular data set

Figure 6.11 shows the results of having applied a different ensemble algorithm to the same data set with the same topology-preserving algorithm. Both AdaBoost.M1 and AdaBoost.M2 were tested for the sake of comparison, taking into account that the first algorithm can also be applied to multi-class data sets, although the second one is understood to be more suitable. All the SOMs in the ensembles that are shown were trained using the same parameters.

Figure 6.11a displays the map obtained by a single SOM. It contains three different classes, one of them -class one, represented by circles- clearly separated from the other two. Figure 6.11b represents the summary obtained by the WeVoS algorithm over an ensemble trained with the bagging meta-algorithm. In this case, a smooth map is obtained as all data set entries are considered of equal importance in all iterations. It is worth noting that the classes are displayed in a more compact way than in the single SOM. Class 1 (circles) appears further away from class 2 (squares) and classes 2 and 3 (triangles) display greater horizontal separation at the top of the image, although this separation is not so clear to the middle-left of the figure.

Figure 6.11c represents the map obtained from an ensemble trained on AdaBoost.M1 algorithm. As this algorithm concentrates on difficult to classify



(a) Mean Quantization Error **(b)** Topographic Error

(c) Distortion **(d)** Goodness of Adaptation

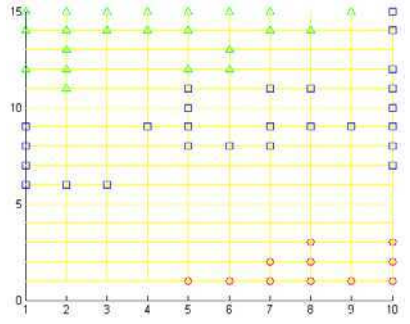
Fig. 6.10 The 4 quality measures obtained from the 4 different summarization algorithms and the single Max-SIM, trained on the artificial circular data set

classes, only one neuron is used in the final map to represent class 1, which is obviously the easiest of the three to distinguish. This may be considered a desirable or not-so-desirable result, depending on the use of the final network. In the case where the map is meant to be used for classification purposes, it may well be considered more suitable than the single map, although the contrary would be true, were it intended for visualization purposes.

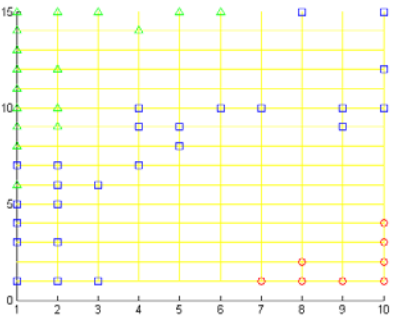
Finally, the result of using AdaBoost.M2 is shown in Figure 6.11d. As this algorithm uses a finer granularity for classification than the previous version, it again represents class 1 in greater detail than the AdaBoost.M1 (Figure 6.11c), but it shows slightly more compact groups than the single algorithm (Figure 6.11a) and a clearer separation of groups than the Bagging algorithm (Figure 6.11b).

In Figure 6.12, the quality measures obtained from the ensemble algorithms were measured from their corresponding WeVoS summarizations, while the data on the single models were obtained directly from those models.

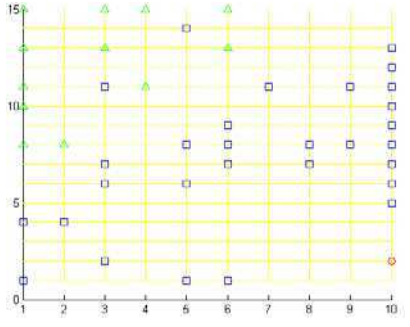
Figure 6.12a represents the classification error of the different variants under study. Obviously, the maps obtained through ensemble algorithms



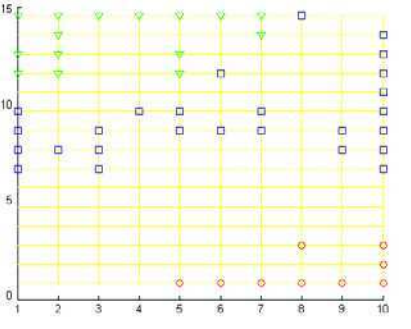
(a) Single SOM



(b) WeVoS from an ensemble trained with the Bagging algorithm



(c) WeVoS from an ensemble trained with the AdaBoost.M1 algorithm

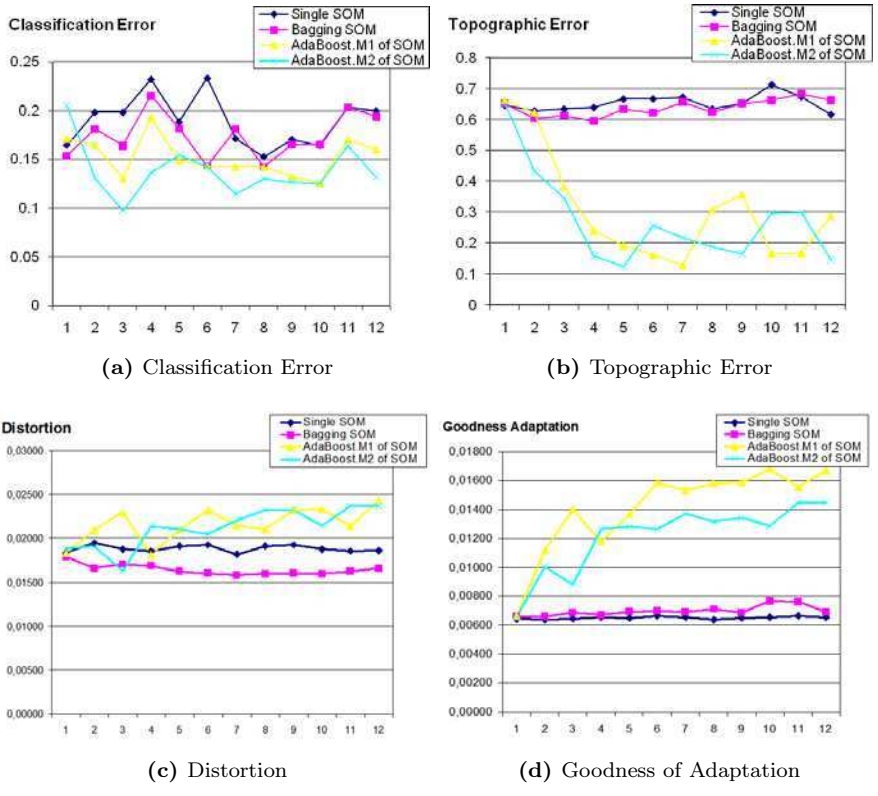


(d) WeVoS from an ensemble trained with the AdaBoost.M2 algorithm

Fig. 6.11 Different maps obtained by training the ensemble of SOM maps, using a different meta-algorithm -Bagging or AdaBoost- and finally applying the WeVoS algorithm

outperformed the single model as they were designed to do so. Figure 6.12b represents the topographic ordering of the final map. Again, ensemble models obtain lower errors than the single one, especially the two variants of AdaBoost. Figure 6.12c represents a more detailed measure of the topological ordering of the maps and shows a different situation than Figure 6.12b for the AdaBoost algorithm. This points to an overfitting problem.

Finally, Figure 6.12d represents a measure that combines quantization and Distortion errors. In this case, the results for the two variants of AdaBoost were predictable, as the algorithm tries to concentrate not on the whole data set, but on the samples within it that are the most difficult to classify, which increases its distance with respect to other samples. The bagging algorithm obtained a lower error, although a bit higher than the simple model. This is due to the nature of the WeVoS algorithm, which pays more attention to the topology preservation of the summary than to the quantization of the



(a) Classification Error **(b)** Topographic Error

(c) Distortion **(d)** Goodness of Adaptation

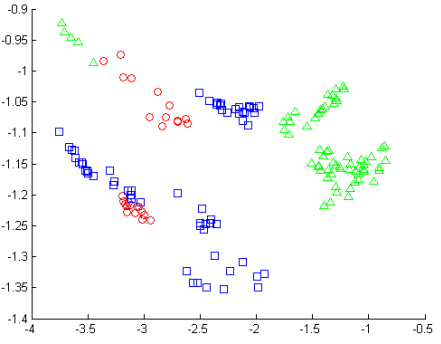
Fig. 6.12 Results of all 4 quality measures applied to the 3 different ensemble training algorithms -single, Bagging, AdaBoost- and the single SOM, trained using the Iris data set

model, the former being one of the most characteristic features of the family of models under study.

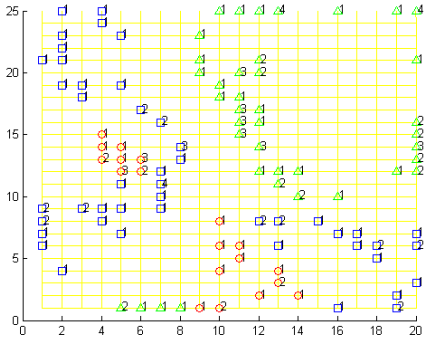
6.5.6 Food Industry Application

Finally, the real-life case study presented in Appendix A is used, in this case, to test and compare the models' behaviour in a practical situation. In this case the data set used is the Ham data set 2. Results regarding very similar experiments can be found in (Baruque et al, 2008).

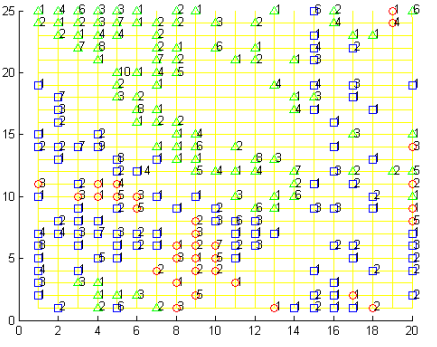
Regarding the visual inspection of the maps obtained in Figure 6.13, the projection of the data set over its two first Principal Components -obtained by a conventional PCA analysis- is shown alongside three maps obtained by training over the same data set.



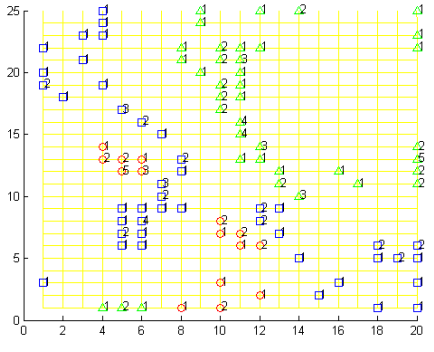
(a) Ham data set projection over the first 2 Principal Components



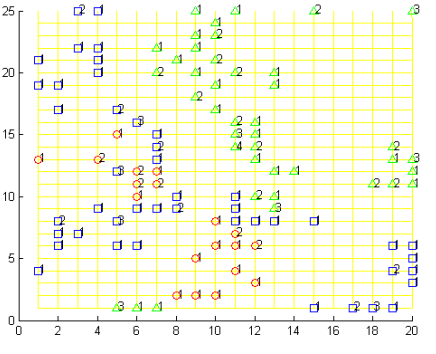
(b) Single SOM



(c) Fusion by Euclidean Distance



(d) Superposition + Re-Labelling



(e) Weighted Voting Superposition

Fig. 6.13 Visual comparison between a PCA analysis of the Ham data set, a single SOM and the three fusion algorithms performed over the same ensemble of 5 SOM; trained on the Ham data set

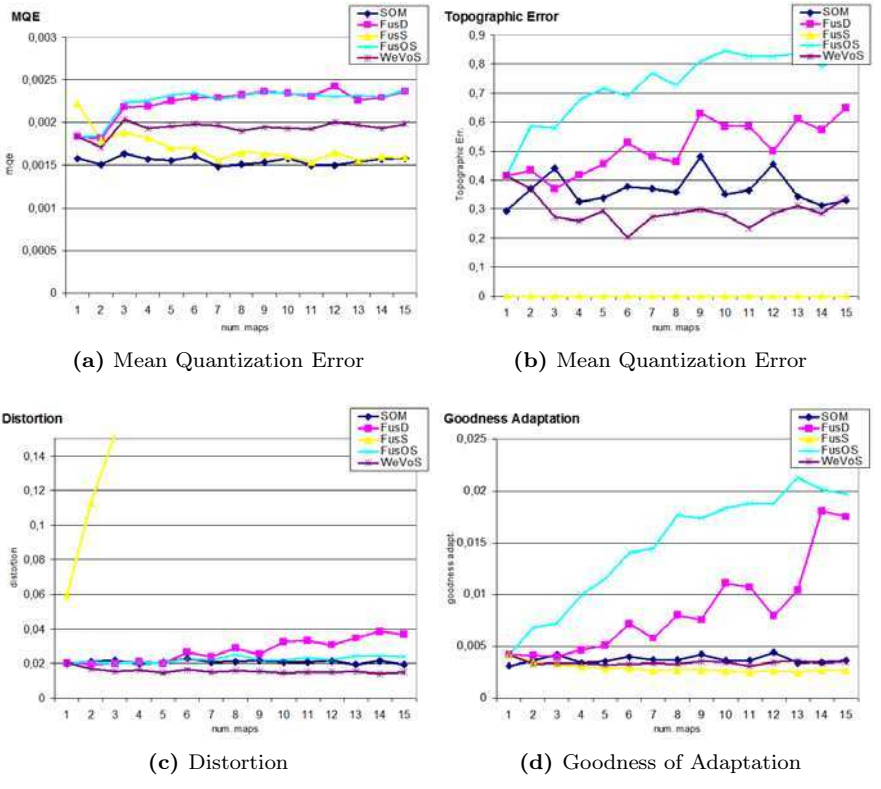
It may be observed from Figure6.13a, that the data set is clearly grouped. Most of the unspoilt samples are situated in a compact group on the right of the image (triangles), while tainted samples are represented in 2 main groups to the left of the image (squares). The rancid/acidic samples (circles), even if considered unspoilt at some point, are in the process of spoiling, so that is their natural position. In Figure 6.13a they are clearly shown as separate from the group of the normal samples and they lie within the group of definitively tainted samples. The exception is a group of samples situated on the top left of the image. These samples are considered to be from a high-quality ham. Usually this kind of ham, having been cured for more time than the other hams, has similar qualities to the rancid samples. In this case, that seems to be the reason why it is separated from the main unspoilt group, but also from the definitively acid/rancid samples.

This same organization can be observed, even after the samples were rotated, in the maps representing the data set. In Figure 6.13b, which depicts the map obtained by a single SOM, the unspoilt samples appear in a group, separated from the rest, in this case, to the upper right side of the image. There is also a group of a few unspoilt samples separated from the rest. The separate group of spoilt samples appears in two different subgroups: the acid/rancid samples grouped with the tainted ones, as in the PCA analysis.

As regards the differences between the models, it is easy to observe that the Fusion by Distance 6.13c obtained a much more distorted map than the other models. Superposition + Re-Labelling 6.13d obtained a clearer image, similar to that of the SOM, although the tainted and acid/rancid groups that appear from the middle-left to middle top of the image are more compact and defined in this map. Finally, the WeVoS-SOM has a general structure that is very similar to the other models. The difference in this case is that more neurons recognize samples in the same groups, so the groups are clearer to the viewer.

The next step in the study was the training of single maps and ensembles of a different number of maps over the same subset of samples, in order to compare their characteristics. This was done using a standard 5-fold cross-validation technique in order to be able to use the whole data set for the experiments. Each measure represented the average of the measures obtained by each of the maps trained with 4-folds and tested over the other remaining fold.

Figure 6.14 shows several measurements obtained from three models compared in the study, all of which are error measurements in different areas of representation of the data set. As may be observed, in the last three measurements (Figure 6.14b to Figure 6.14d), not only did the ensemble summarization methods (WeVoS) obtain a lower error, but they were also shown to be more stable, and were not dependent on any specific execution of the training. As expected, the only exception to this was the MQE, because it is a measurement of how far or how close the samples are from the unit that represents them, whereas the WeVoS meta-algorithm improves the visual



(a) Mean Quantization Error **(b)** Mean Quantization Error

(c) Distortion **(d)** Goodness of Adaptation

Fig. 6.14 The 4 quality measures obtained from the 3 different ensemble training algorithms and the single SOM, trained on the Ham data set

representation of the data set. The remaining measurements denote the accuracy of the representation in relation to the topological organization of the map.

6.6 Conclusions

This chapter has presented a study of the capabilities of a proposed method for the summarization of an ensemble of Self-Organizing Maps called Weighted Voting superposition (WeVoS), which aims to obtain the most faithful visual representation of a high-dimensional data set projected as a 2-D map. The combination of these algorithms is referred to as the WeVoS-SOM.

A complete study of its performance under several widely known real data sets, using a variety of analytical quality measures, has been performed and analysed. The main feature of this algorithm is a reliable visual representation of the data set by enhancing the topology preservation feature, which is one of the most important aspects that the original model that is intended to

improve. This characteristic is reflected by the Distortion and, to an extent, by the Goodness of Adaptation measures.

The WeVoS model proves its real usefulness when representing more complex data sets rather than simpler ones, where the added complexity of calculating an ensemble of maps is compensated by an improvement in the visualization results. As an added advantage, the computational complexity of the algorithm is lower than in other previously ensemble fusion algorithms. While previous models performed their calculations in relation to the entire extension of the maps to be fused, WeVoS performs calculations only on "homologous" neurons and their neighbourhoods in the respective maps of the ensemble; resulting in a less computationally extensive algorithm.

A novel modification of previous algorithms to carry out summarization of Self-Organizing Maps ensembles, called Fusion by Ordered Similarity, has also been presented, analysed and compared with the previous models on which it is based and with the WeVoS algorithm. Fusion by Ordered Similarity obtained better results than the two models on which it is based with respect to their worst respective facets, but it failed to outperform their best respective facets.

6.7 Future Work

During the work for this book some modifications of the WeVoS model were contemplated, although not implemented or tested. The concept behind those modifications is to integrate the idea of Euclidean Distance or Voronoi Polygon Similarity into the calculations of the Vote Weighting for the fused neurons.

Future work includes implementing and testing these variations with more topology-preserving models and with other interesting real data sets.

Chapter 7
Conclusions

7.1 Concluding Remarks

This thesis has explored the combination of ensemble learning in conjunction with unsupervised learning. It has also included some others, such as the combination of statistical PCA with ensemble learning. Its main contribution concerns topology-preserving maps and its principal objective is the improvement of the visualization capabilities of both statistical and neuronal methods for data visualization tasks.

The most significant contributions of the thesis are:

- A method for the assessment and the identification of outliers, and to an extent a method of dealing with their inclusion in a data set, through a combination of the Bagging algorithm and statistical PCA.
- An algorithm for obtaining the fusion of an ensemble of topology-preserving maps that is capable of improving the visualization achieved by the single map and other ensemble fusion algorithms.

Specifically, the first contribution is achieved by a repeated PCA analysis over a re-sampled data set and the averaging and the comparison of the components found for each of the re-sampled data sets. The tests shown in Chapter 4 prove the validity of the concept, as the proposed model is able to detect the presence of outliers in the data set by observing the spread of the representation of the directions found by the different analyses. This situation has been tested both for artificial and real data sets with good results in all cases.

Its second contribution is the development of an algorithm for generical fusion of topology preserving maps. In this work, a preliminary approach to this fusion is performed by an algorithm called Superposition, which is described in Chapter 5. Then, an improved version of that algorithm, called WeVoS, which is the central contribution of the thesis, is presented in Chapter 6. This algorithm calculates the quality measures in the literature for each neuron on the maps composing the ensemble. Once this is done, an informed decision

B. Baruque & E. Corchado: Fusion Meth. for Unsupervised Learn. Ensem., SCI 322, pp. 123–125.
springerlink.com

may be taken on the most advisable values for the final neuronal weights to be assigned to the neurons in the final map.

A complete study of the algorithm in combination with several topography preserving algorithms has been presented. For this study, the following data sets and ensemble combination algorithms were used in combination with the proposed algorithm: SOM, ViSOM, SIM and Max-SIM.

Some of the conclusions that may be drawn from these tests are:

- In general terms, the WeVoS algorithm works in a similar way for all topography preserving models regardless of the model that is used as a basis for the ensemble.
- The way the ensemble is trained has a direct effect on the final resultant map, so its selection is also a variable to take into account that depends on the task the fused map is meant to accomplish. For example, the Boosting algorithm obtains good results for classification tasks, but Bagging appears to obtain clearer results for visualization.
- The WeVoS algorithm proves its real usefulness when representing more complex data set rather than simpler ones, when the added complexity of calculating an ensemble of maps is compensated by an improvement in their visualization.

This weight of vote scheme opens up a framework where many other variations and options can be included.

Depending on the task to perform either one or the other measure can be used for calculation of the weights. For example, if the map is intended to be used for data exploration, measures such as topographic error or especially, Distortion would be more appropriate. If it is intended to perform data classification, the classification accuracy of each neuron might well be more advisable. Finally, instead of using a single measure for calculating this weight, a linear combination of more than one measure could be useful.

Although the use of ensemble algorithms has improved the stability of the model, it was observed that the effect of including too many vote-weighting measures adds a unstable component to the calculations that can degrade the performance of the fused map. No experimentation was undertaken in this thesis on that phenomenon, which is considered an interesting area for future research.

7.2 Future Research Work

As mentioned in previous sections, the weighted voting scheme for fine tunning or adapting the final weights of the neurons of the map creates a framework where many variations could be included. For example, instead of using a single measure for calculating this weight, a linear combination of more than one could be useful.

As described here, the algorithm only takes the performance of the neurons into account, and not their position in relation to the other neurons. Some

of the most interesting variations of the algorithm proposed in this thesis could be adaptations of the way in which other fusion algorithms work. It is not difficult to include Euclidean distance or Voronoi polygon similarity calculations into the weighting of the vote. This could serve to construct an even more informative decision of the final inter-neuronal weight of the maps, because this additional information would clarify whether the weighting of one specific neuron deviates from the others in that position, the inclusion of which would negatively affect the final fused neuron.

The inclusion of additional criteria in the weighting of the vote for the fusion of neurons has still to be formalised and tested.

Other future work includes implementing and testing these variations with more topology preserving models and with other interesting artificial and real data sets.

Appendix A
The Cured Ham Data Set

A.1 Sensory Analysis and Instruments

Sensory analysis is a method used to describe the sensations that humans perceive with their 5 senses when in contact with a product. Among the 5 senses, we can distinguish between physical -hearing, touch, sight- and chemical -olfaction, taste- senses. The sense of smell, also called olfaction, is the perception of odorant molecules either by direct inhalation or during mastication -retro nasal route- . The sense of taste consists of the perception of savours perceived by the tongue during tasting. Commonly, savours are classified into 5 categories: salt, sour, bitter, sweet, umami.

There exist 2 types of tests in sensory analysis:

- ▷ hedonic tests (I like it / I don't like it) which consist of consumer tests
- ▷ analytical tests (descriptive or discriminative) that are conducted by a trained sensory panel

For analytical tests, Electronic nose and tongue analyzers can also be used. These instruments have the specificity to analyze smell/taste in the same way as human nose and tongue: instead of separating and identifying the various chemical compounds, they measure a global smell/taste.

A.2 E-Nose Odour Recognition

Alpha MOS (AlfaMOS , 2008) has developed solutions linked to chemical senses. FOX Electronic Nose is a smell and Volatile Organic Compounds (VOC) analyzer. It is based on Metal Oxide Sensor detection system.

The odour recognition process may be summarized as:

1. The sample is heated for a given time to generate volatile compounds in the head space of the vial containing the sample.
2. The gas phase is transferred to a detection device which reacts to the presence of molecules.

3. The differences in sensor reactions are recorded using statistical calculation techniques to classify the odours.

This process, compared with the human sense of smell is presented in Figure A.1.

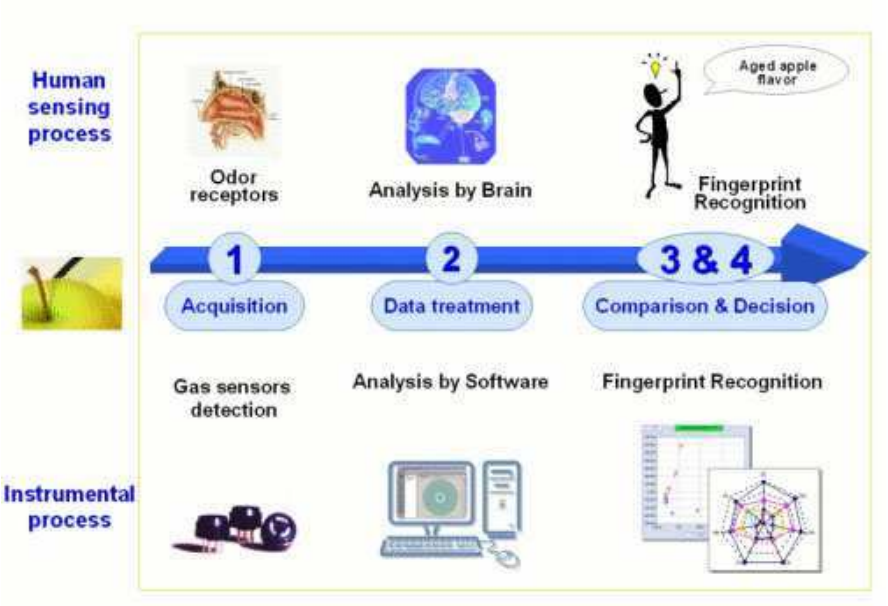


Fig. A.1 Human odour recognition process compared with E-Nose odour recognition process

Figure extracted from (AlfaMOS , 2008)

Fig. A.2 E-Nose αFOX 4000

Figure extracted from (AlfaMOS , 2008)

The readings taken by each sensor are separated and stored in a simple database for further study. In this study the analyses are performed using an E-Nose αFOX 4000 (Alpha M.O.S., Toulouse, France) with a sensor array of 18 metal oxide sensors. The E-Nose takes readings every 0.5 seconds, and has an acquisition time of 120 seconds and an acquisition delay of 600 seconds.

After having obtained the readings for each sample of cut ham taken from the 18-sensor array in the electronic nose, the highest reading from each sensor is stored in a database along with the corresponding results of the sensory evaluation by the professional human testers. Figure A.2 depicts this machine.

A.3 The Cured Ham Data Sets

A.3.1 Ham Data Set 1

The data set consists of measures obtained from several brands of seven different types of Spanish ham, available in the Spanish market. The types of ham selected for this objective -with their codification in this experiment between brackets- are the following: the fat of hams cured for the same period of time (JCTC), cured ham of superior quality (JCCS), ham cured under 'Traditional Speciality Guaranteed' or TSG (JCTSG), cured ham with protected 'Designation of Origin' (JCDO) from Teruel, a region in Spain (JCTE), ham that is cured over seven months (JC7C), cured ham with 2 differentiated defects, off-flavours or taints (JCCA and JCBO) and standard cured ham (JCNO). The commercial brands from which the samples were extracted are not taken into account in this study.

Moreover, a simpler version of the labelling of those samples is considered. in which the labels correspond to the level of quality that is detected. The classes are 'unspoilt', 'rancid/acid' and 'tainted'. The first corresponds to eateable samples ranging from high quality to standard ones; the second corresponds to samples that are detected with an acid/rancid smell but are still edible, and the third one corresponds to non-edible samples. Samples labelled as JCCA and JCBO in the first label, correspond to 'tainted' samples. The 'unspoilt' and 'rancid/acid' samples can correspond to any of the other procedence groups.

Several samples were cut from different parts of each type of ham and measurements were taken from each of these samples, using the E-Nose αFOX 4000 (Alfa MOS, Toulouse, France) with a sensor array of 18 metal oxide sensors. The final data set consisted of a total of 176 samples of ham, each composed of 18 different variables.

A.3.2 Ham Data Set 2

A second data set obtained from the same source was also used, but this time with much simpler labelling. In this case the sensor's data was obtained by

exactly the same process, but the labels were assigned by professional human testers according to their own perceptions; rather than by its procedence from an analysed sample.

Professional classifications are usually far more detailed, but were restricted in this initial study to three possible values: 'unspoilt', 'rancid/acid' and 'tainted'. Thus, our final data set consisted of readings taken from a total of 154 samples of ham, on which the readings were composed of 18 different variables measured over three possible categories.

A.4 Analysis of the Data Set

The main problem of any automated food taster is its subjectivity, which is also true for almost all human activities. No uniform, consensual and objective method exists of representing the reaction of a person to a stimulus. In the case of human tasters, their activity is expected to be as objective as possible, as their sole interest is to assess the quality of the food.

This assessment can influence a range of cases from those involving public health and spoilt or inedible food, to those that are less severe, but that are also of commercial importance, which influences the final price of the product. In the case of non-edible products, there are of course many more detailed standards with which products must comply before they are exposed to human consumption. Quality assessments of certain types of food are usually made by human expert tasters, which influence product price and prestige.

The idea behind the analysis of this data set is precisely to obtain a method to assess the way in which human experts determine, in the most objective way possible, the quality of a cured ham sample. As this analysis is directly related to how the human brain processes information from the senses, the neural network approach appears to be a very appropriate and interesting one.

Human expert tasters usually undergo a certain degree of training to become a recognized taster and possess more or less detailed knowledge of the chemical properties of the food. Nevertheless, the idea of presenting chemical analyses of the food samples does not appear sensible, as in this case it is rather counterintuitive and differs from the way in which the electronic nose functions -i.e. obtaining a global odour measure instead of analyzing chemical compounds- .

Therefore, in this approach, the analysis was performed by the E-Nose but the results were displayed for the human expert in a simple image that groups similar samples together, so that the expert can gain a straightforward idea of where a specific sample is situated in relation to its quality.

The human brain functions by acquiring the data through the senses and storing the interpreted sensations in similar brain regions so the become clustered and recognizable. In this way, it may be said that the AI process emulates the way in which the human brain functions in relation to sensory analysis. The aim is to present this clustering as relationships between stimulus in a simple image, so that the information is accessible and easily understood in the most immediate way possible.

Appendix B
Table of Experiments

This appendix summarizes all the experiments presented in the thesis, indicating in which Figure or Table the results of a determined experiment are showed.

B.1 Chapter 4

Data set	Samples / Outliers	Algorithm	Figures	Tables
Artificial	100 / no	Re-PCA	4.2a	
	100 / yes	Re-PCA	4.2b, 4.3	
	50 / no	Re-PCA		4.1
	50 / yes	Re-PCA		4.2
	30 / no	Re-PCA	4.4a	4.3
	30 / yes	Re-PCA	4.4b, 4.5	4.4
BUPA Liver Disorder	200 / no	Re-PCA	4.6a	
	200 / yes	Re-PCA	4.6b	
Spanish Ham	176 / no	PCA	4.7	4.5
	176 / yes	PCA	4.8	4.6
	80 / yes	Re-PCA	4.9	4.7
	120 /yes	Re-PCA	4.10	4.8

B.2 Chapter 5

Data set	Summarization	Map	Figures	Tables
Radial (artificial)	Bagging + Ensemble	SOM	5.3a	5.1, 5.2
		Max-SIM	5.3b	5.1, 5.2
Iris	single	SOM	5.4a, 5.5a, 5.7a	5.3, 5.4, 5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	Bagging + Ensemble	SOM	5.6a	5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	Bagging + Fus. Euc. Distance	SOM	5.4b, 5.5b, 5.7b	5.3, 5.4, 5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	Bagging + Fus. Similarity	SOM	5.4c	5.3, 5.4, 5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	Bagging + Superposition	SOM	5.6b, 5.7c	5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	Bagging + Superposition + Re-Labelling	SOM	5.7d	5.5, 5.7, 5.9
		ViSOM		5.5, 5.7, 5.9
	AdaBoost + single	SOM		5.8
		ViSOM	5.9a	5.8
	AdaBoost + Fus. Euc. Distance	SOM		5.8
		ViSOM	5.9b	5.8
	AdaBoost + Superposition	SOM		5.8
		ViSOM	5.9c	5.8

Data set	Summarization	Map	Figures	Tables
	AdaBoost + Superposition + Re-Labelling	SOM		5.8
		ViSOM	5.9d	5.8
Cancer	Bagging + Superposition + Re-Labelling	SOM	5.8a	5.6, 5.10
		ViSOM	5.8b	5.6, 5.10
Spanish Ham	Bagging + single	SOM		5.11
		ViSOM	5.10a	5.11
	Bagging + ensemble	SOM		5.11
		ViSOM		5.11
	Bagging + Fus. Euc. Distance	SOM		5.11
		ViSOM	5.10b	5.11
	Bagging + Fus. Similarity	SOM		5.11
		ViSOM		5.11
	Bagging + Superposition	SOM		5.11
		ViSOM	5.10c	5.11
	Bagging + Superposition + Re-Labelling	SOM		
		ViSOM	5.10d, 5.11	
	AdaBoost + single	SOM		5.12
		ViSOM		5.12
	AdaBoost + ensemble	SOM		5.12
		ViSOM		5.12

Data set	Summarization	Map	Figures	Tables
	AdaBoost + Fus. Euc. Distance	SOM		5.12
		ViSOM		5.12
	AdaBoost + Fus. Similarity	SOM		5.12
		ViSOM		5.12
	AdaBoost + Superposition	SOM		5.12
		ViSOM		5.12
	AdaBoost + Superposition + Re-Labelling	SOM		5.12
		ViSOM		5.12

B.3 Chapter 6

Data set	Summarization	Map	Figures
Horse shoe (artificial)	single	SOM	6.2
	Bagging + Fus. Distance	SOM	6.2
	Bagging + Fus. Similarity	SOM	6.2
	Bagging + Fus. Ord. Similarity	SOM	6.2
	Bagging + WeVoS	SOM	6.2
Radial (artificial)	single	SIM	6.8a, 6.9
		Max-SIM	6.10
	Bagging + Fus. Distance	SIM	6.8b, 6.9
		Max-SIM	6.10
	Bagging + Fus. Similarity	SIM	6.8c, 6.9

Data set	Summarization	Map	Figures
		Max-SIM	6.10
	Bagging + WeVoS	SIM	6.8d, 6.9
		Max-SIM	6.10
Iris	single	SOM	6.3a, 6.4, 6.11a, 6.12
	Bagging + Fus. Distance	SOM	6.3b, 6.4
	Bagging + Fus. Similarity	SOM	6.3c, 6.4
	Bagging + Fus. Ord. Similarity	SOM	6.3d, 6.4
	Bagging + WeVoS	SOM	6.3e, 6.4, 6.11b, 6.12
	AdaBoost.M1 + WeVoS	SOM	6.11c, 6.12
	AdaBoost.M2 + WeVoS	SOM	6.11d, 6.12
Wisconsin Breast Cancer	single	SOM	6.6
	Bagging + Fus. Distance	SOM	6.6
	Bagging + Fus. Similarity	SOM	6.6
	Bagging + Fus. Ord. Similarity	SOM	6.6
	Bagging + WeVoS	SOM	6.6
Wine	single	SOM	6.5
		ViSOM	6.7
	Bagging + Fus. Distance	SOM	6.5
		ViSOM	6.7
	Bagging + Fus. Similarity	SOM	6.5
		ViSOM	6.7

Data set	Summarization	Map	Figures
	Bagging + Fus. Ord. Similarity	SOM	6.5
		ViSOM	6.7
	Bagging + WeVoS	SOM	6.5
		ViSOM	6.7
Spanish Ham		PCA	6.13a
	single	SOM	6.13b, 6.14
	Bagging + Fus. Distance	SOM	6.13c, 6.14
	Bagging + Fus. Similarity	SOM	6.14
	Bagging + Fus. Ord. Similarity	SOM	6.14
	Bagging + Superposition + Re-labelling	SOM	6.13d
	Bagging + WeVoS	SOM	6.13e, 6.14

References

Alpha mos - smell, taste & chemical profiling (2008)

Ali, K.: A comparison of methods for learning and combining evidence from multiple models. technical report uci tr 95-47. Tech. rep., University of California, Irvine, Dept. of Information and Computer Sciences (1995)

Asuncion, A., Newman, D.J.: Uci machine learning repository (2007)

Bahler, D., Navarro, L.: Methods for combining heterogeneous sets of classifiers. In: 17th Natl. Conf. on Artificial Intelligence (AAAI), Workshop on New Research Problems for Machine Learning (2000)

Baruque, B., Gabrys, B., Corchado, E., Herrero, I., Rovira, J., González, J.: Outlier overcoming using re-sampling techniques. In: Diaz, F., Corchado, J.M., Fdez-Riverola, F. (eds.) 5th International Workshop on Practical Applications of Agents and Multiagent Systems (IWPAAMS 2006), Universidad de Valladolid. Servicio de imprenta, Segovia, Spain (2006)

Baruque, B., Corchado, E., Yin, H.: Visom ensembles for visualization and classification. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 235–243. Springer, Heidelberg (2007)

Baruque, B., Corchado, E., Rovira, J., Gonzalez, J.: Application of topology preserving ensembles for sensory assessment in the food industry. In: Fyfe, C., Kim, D., Lee, S.-Y., Yin, H. (eds.) IDEAL 2008. LNCS, vol. 5326, pp. 491–497. Springer, Heidelberg (2008)

Baruque, B., Corchado, E., Mata, A., Corchado, J.M.: Ensemble methods for boosting visualization models. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5517, pp. 165–173. Springer, Heidelberg (2009)

Bauer, H., Pawelzik, K.: Quantifying the neighborhood preservation of self-organizing feature maps (1992)

Bennett, K.P., Demiriz, A., Maclin, R.: Exploiting unlabeled data in ensemble methods. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 289–296 (2002)

Bishop, C.: Neural Networks for Pattern Recognition, Oxford (1995)

Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. WADSWORTH ADV. BOOK PROG (1984)

Britannica, E.: Human nervous system (2009)

Ramon y Cajal, S.: Conexión general de los elementos nerviosos. La Medicina Práctica (1889)
Carus, P.: The Soul of Man (1905)
Charles, D., Fyfe, C.: Modelling multiple cause structure using rectification constraints. Computation in Neural Systems 9, 167–182 (1998)
Cho, S.B.: Ensemble of structure-adaptive self-organizing maps for high performance classification. Information Sciences 123(1-2), 103–114 (2000)
Corchado, E., Fyfe, C.: Maximum likelihood topology preserving algorithms (2002a)
Corchado, E., Fyfe, C.: The scale invariant map and maximum likelihood hebbian learning (2002b)
Corchado, E., MacDonald, D., Fyfe, C.: Maximum and minimum likelihood hebbian learning for exploratory projection pursuit. Data Mining and Knowledge Discovery 8(3), 203–225 (2004)
Corchado, E., Baruque, B., Yin, H.: Boosting unsupervised competitive learning ensembles. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 339–348. Springer, Heidelberg (2007)
Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society Series B (Methodological) 39(1), 1–38 (1977)
Diaconis, P., Freedman, D.: Asymptotics of graphical projections. The Annals of Statistics 12(3), 793–815 (1984)
Dietterich, T.G.: Ensemble methods in machine learning. In: MCS 2000: Proceedings of the First International Workshop on Multiple Classifier Systems, pp. 1–15. Springer, London (2000)
Dixon, W.J.: Analysis of extreme values. The Annals of Mathematical Statistics 21(4), 488–506 (1950)
Frawley, W.J., Piatetsky-shapiro, G., Matheus, C.J.: Knowledge discovery in databases: an overview. AI Magazine 13(3) (1992)
Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)
Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55, 119–139 (1997)
Friedman, J.H.: Exploratory projection pursuit. Journal of the American Statistical Association 82(397), 249–266 (1987)
Friedman, J.H., Tukey, J.: A projection pursuit algorithm for exploratory data analysis. IEEE Transactions on Computers 23, 881–889 (1974)
Fumera, G., Roli, F.: Error rejection in linearly combined multiple classifiers. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 329–338. Springer, Heidelberg (2001)
Fyfe, C.: Pca properties of interneurons. In: Proceedings of International Conference on Artificial on Artificial Neural Networks - ICANN 1993, pp. 183–188 (1993)
Fyfe, C.: A scale-invariant feature map. Network: Computation in Neural Systems 7, 269–275 (1996)
Gabrys, B., Baruque, B., Corchado, E.: Outlier resistant pca ensembles. LNCS (LNAI), pp. 1434–1442. Springer, Berlin (2006)
Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation 4 (1992)

Georgakis, A., Li, H., Gordan, M.: An ensemble of som networks for document organization and retrieval. In: International Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2005) (2005)
Giacinto, G., Roli, F.: Design of effective neural network ensembles for image classification purposes. Image and Vision Computing 19(9-10), 699–707 (2001)
Greene, D., Tsymbal, A., Bolshakova, N., Cunningham, P.: Ensemble clustering in medical diagnostics. In: Proceedings of 17th IEEE Symposium on Computer-Based Medical Systems CBMS 2004 (2004)
Hashem, S.: Optimal linear combinations of neural networks. PhD thesis, Purdue University (1997)
Haykin, S.: Neural networks. Prentice Hall Upper Saddle River, NJ (1999)
Hebb, D.O.: The Organization of Behavior: A Neuropsychological Theory. Lawrence Erlbaum Associates (1949)
Heskes, T.: Balancing between bagging and bumping. In: Mozer, M.C., Jordan, M.I., Petsche, T. (eds.) Advances in Neural Information Processing Systems, Denver, Colorado, USA, pp. 466–472 (1997)
Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Education Psychology 24, 417–444 (1933)
Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3, 79–87 (1991)
Jiang, Y., Zhou, Z.H.: Som ensemble-based image segmentation. Neural Process Lett. 20(3), 171–178 (2004)
Jimenez, D.: Dynamically weighted ensemble neural networks for classification. In: The 1998 IEEE International Joint Conference on, Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence, vol. 1, pp. 753–756 (1998)
Jordan, M.J., Jacobs, R.A.: Hierarchical mixtures of experts and the em algorithm. Neural Computation 6(2), 181–214 (1994)
Kaski, S., Lagus, K.: Comparing self-organizing maps. In: Malsburg, Cvd., Seelen, Wv., Vorbruggen, J.C., Sendhoff, B. (eds.) Lecture Notes in Computer Science, Bochum, Germany, pp. 809–814. Springer, Berlin (1996)
Kiviluoto, K.: Topology preservation in self-organizing maps. In: IEEE International Conference on Neural Networks (ICNN 1996), vol. 1, pp. 294–299 (1996)
Kohonen, T.: An introduction to neural computing. Neural Networks 1(1), 3–16 (1988)
Kohonen, T.: Self-Organizing Maps, vol. 30. Springer, Berlin (1995)
Kohonen, T., Lehtio, P., Rovamo, J., Hyvarinen, J., Bry, K., Vainio, L.: A principle of neural associative memory. Neuroscience 2(6), 1065–1076 (1977)
Kosko, B.: Constructing an associative memory. BYTE 12(10), 137–144 (1987) issn = 0360-5280
Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
Kuncheva, L.I., Skurichina, M., Duin, R.P.W.: An experimental study on diversity for bagging and boosting with linear classifiers. Information Fusion 3(4), 245–258 (2002)
Lampinen, J., Oja, E.: Clustering properties of hierarchical self-organizing maps. Journal of Mathematical Imaging and Vision 2, 261–272 (1992)
Luttrell, S.P.: Hierarchical self-organizing networks. In: Proceedings of the ICANN 1989 (1989)

Maqsood, I., Khan, M.R., Abraham, A.: An ensemble of neural networks for weather forecasting. Neural Computing and Applications 13(2), 112–122 (2004)
Martinez, T.M., Schulten, K.J.: A neural-gas network learns topologies. Artificial Neural Networks, 397–402 (1991)
Miskin, J.W.: Ensemble learning for independent component analysis. Tech. rep., in Advances in Independent Component Analysis (2000)
Oja, E.: Simplified neuron model as a principal component analyzer. Journal of Mathematical Biology 15(3), 267–273 (1982)
Oja, E.: Neural networks, principal components, and subspaces. Int. J. Neural Syst. 1(1), 61–68 (1989)
Oja, E.: Principal components, minor components, and linear neural networks. Neural Networks 5(6), 927–935 (1992)
Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2(6), 559–572 (1901)
Perrone, M.: Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization. PhD thesis, Brown University, Institute for Brain and Neural Systems (1993)
Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks, pp. 126–142. Chapman and Hall, Boca Raton (1993)
Petrakieva, L., Fyfe, C.: Bagging and bumping self organising maps. Computing and Information Systems Journal (2003)
Polani, D.: Measures for the organization of self-organizing maps. In: Seiffert, U., Jain, L.C. (eds.) Self-organizing Neural Networks: Recent Advances and Applications (Studies in Fuzziness and Soft Computing), vol. 16, pp. 13–44. Physica-Verlag, Heidelberg (2003)
Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006)
Pozlbauer, G.: Survey and comparison of quality measures for self-organizing maps. In: Rauber, J.P., Polzlbauer, G., Andreas (eds.) Fifth Workshop on Data Analysis (WDA 2004), pp. 67–82. Elfa Academic Press, London (2004)
Ruiz, M.: Diagram of a typical myelinated vertebrate motoneuron (2007), `http://en.wikipedia.org/wiki/Neuron`
Rumelhart, D., Zipser, D.: Feature discovery by competitive learning. Cognitive Science 9, 75–112 (1985)
Ruta, D., Gabrys, B.: An overview of classifier fusion methods. Computing and Information Systems 7(1), 1–10 (2000)
Ruta, D., Gabrys, B.: A theoretical analysis of the limits of majority voting errors for multiple classifier systems. Pattern Analysis and Applications 5(4), 333–350 (2002)
Saavedra, C., Salas, R., Moreno, S., Allende, H.: Fusion of self organizing maps. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 227–234. Springer, Heidelberg (2007)
Schapire, R.E.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
Sharkey, A., Sharkey, N.: Combining diverse neural nets. Knowledge Engineering Review 12(3), 1–17 (1997)
Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press Cambridge, Massachusetts (2005)

Tetko, I.V., Livingstone, D.J., Luikov, A.I.: Comparison of overfitting and overtraining. Journal of Chemical Information and Computer Sciences 35(5), 826–833 (1995)

Tibshirani, R., Knight, K.: Model search by bootstrap "bumping". Journal of Computational and Graphical Statistics 8(4), 671–686 (1999), published by: American Statistical Association, Institute of Mathematical Statistics, and Interface Foundation of America (1999)

Tumer, K., Ghosh, J.: Error correlation and error reduction in ensemble classifiers. Connection Science 8(3-4), 385–403 (1996)

Vesanto, J., Sulkava, M., Hollmen, J.: On the decomposition of the self-organizing map distortion measure. In: Proceedings of the Workshop on Self-Organizing Maps (WSOM 2003), pp. 11–16 (2003)

Yin, H.: Data visualisation and manifold mapping using the visom. Neural Networks 15(8-9), 1005–1016 (2002a)

Yin, H.: Visom - a novel method for multivariate data projection and structure visualization. IEEE Transactions on Neural Networks 13(1), 237–243 (2002b)