COMPUTATIONAL INTELLIGENCE METHODS ON
BIOMEDICAL SIGNAL ANALYSIS AND DATA MINING
IN MEDICAL RECORDS

By
Liviu-Mihai Vladutu

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
UNIVERSITY OF PATRAS
PATRAS, GREECE
SEPTEMBER 2004

Interdepartamental Program of Postgraduate Studies in
BIOMEDICAL ENGINEERING
ΕΠΙΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

1. χ. Α. Μπεζερίδανος, Καθηγητής, Τμήμα Ιατρικής, Πανεπιστήμιο Πατρών
2. χ. Γ. Νικηφοροδής, Καθηγητής, Τμήμα Ιατρικής, Πανεπιστήμιο Πατρών
3. χ. Ν. Παλληραφόρος, Καθηγητής, Τμήμα Ιατρικής, Πανεπιστήμιο Πατρών
4. χ. Τ. Μπουντής, Καθηγητής Τμήμα Μαθηματικών, Πανεπιστήμιο Πατρών
5. χ. Μ. Βραχάτης, Καθηγητής, Τμήμα Μαθηματικών, Πανεπιστήμιο Πατρών
6. χ. Ν. Φακωτός, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών
7. χ. Δ. Αλεξόπουλος, Καθηγητής, Τμήμα Ιατρικής, Πανεπιστήμιο Πατρών

ΤΡΙΜΕΛΗΣ ΣΥΜΒΟΥΛΕΥΤΙΚΗ ΕΠΙΤΡΟΠΗ

1. χ. Α. Μπεζερίδανος, Καθηγητής, Τμήμα Ιατρικής, Πανεπιστήμιο Πατρών, Πρόεδρος
2. χ. Μ. Βραχάτης, Καθηγητής, Τμήμα Μαθηματικών, Πανεπιστήμιο Πατρών
3. χ. Ν. Φακωτός, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών
To my parents, Dumitru and Maria
Table of Contents

Table of Contents
Abstract
Περίληψη
Acknowledgments
Introduction
  0.1 Objectives and Achievements
  0.2 Original Contributions
  0.3 Organization of the Thesis
1 Wavelets in Signal Processing
  1.1 Dyadic Wavelet Transform (DWT)
  1.1.1 Series Expansion of Signals: Multiresolution Decomposition onto Orthonormal Wavelet Bases
  1.1.2 Pyramidal Implementation of Dyadic Wavelet Transform
  1.2 Daubechies Family of Regular Filters and Wavelets
  1.3 Wavelets Shrinkage
  1.3.1 General Principles of Wavelets Shrinkage
  1.3.2 Hard and Soft Thresholding
  1.4 Application: Denoising of Principal Components time series
2 The Self-Organizing Map
  2.1 Introduction
  2.2 Basic SOM
  2.3 Learning Vector Quantization
6.5 Ischemic episodes detection using NetSOM ........................................... 97
   6.5.1 Results ......................................................................................... 97
   6.5.2 Conclusions ............................................................................... 100

7 Support Vector Machines - introduction and application within sNet-SOM 105
   7.1 Support Vector Machines - SVM ....................................................... 105
      7.1.1 Fundamentals of Statistical Learning Theory ................................. 105
      7.1.2 Support Vector Machines ............................................................ 108
      7.1.3 Model Order Selection ................................................................. 111
   7.2 The supervised Network Self-Organizing Map: sNet-SOM ................. 113
      7.2.1 The Classification Partition SOM (CP-SOM) ............................... 115
      7.2.2 The Supervised Expert Network ................................................... 122
   7.3 Applications of sNet-SOM ................................................................. 125

8 Conclusions and Future Work Directions .................................................. 129

Bibliography ......................................................................................... 131
Abstract

This thesis is centered around the development and application of computationally effective solutions based on artificial neural networks (ANN) for biomedical signal analysis and data mining in medical records. It is important to recall that the ultimate goal of this work in the field of Biomedical Engineering is to provide the clinician with the best possible information needed to make an accurate diagnosis (in our case of myocardial ischemia) and to propose advanced mathematical models for recovering the complex dependencies between the variables of a physical process from a set of perturbed observations.

After describing some of the types of ANN mainly used in this work, we start designing a model for pattern classification, by constructing several local models, for neighborhoods of the state space. For this task, we use the novel $k$-windows clustering algorithm, to automatically detect neighborhoods in the state space. This algorithm, with a slight modification (unsupervised $k$-windows algorithm) has the ability to endogenously determine the number of clusters present in the dataset during the clustering process. We used this method together with the other 2 mentioned below (NetSOM and sNet-SOM) for the problem of ischemia detection.

Next, we propose the utilization of a statistically extracted distance measure in the context of Generalized Radial Basis Function (GRBF) networks. The main properties of the GRBF networks are retained in a new metric space, called Statistical Distance Metric (SDM). The regularization potential of these networks can be realized with this type of distance. Furthermore, the recent engineering of neural networks offers effective solutions for learning smooth functionals that lie on high dimensional spaces. We tested this solution with an application from bioinformatics, one example from data mining of commercial databases and finally with some examples using medical databases from the UCI (University

We continue by establishing the network self-organizing map (NetSOM) model, which attempts to generalise the regularisation and ordering potential of the basic SOM from the space of vectors to the space of approximating functions. It becomes a device for the ordering of local experts (i.e. independent neural networks) over its lattice of neurons and for their selection and co-ordination.

Finally, an alternative to NetSOM is proposed, which uses unsupervised ordering based on Self-organizing maps (SOM) for the "simple" regions and for the "difficult" ones a two-stage learning process. There are two differences resulted from the comparison with the previous model (NetSOM), one is that we replaced a fixed-size of the SOM with a dynamically expanded map and second, the supervised learning was based this time on Radial Basis Functions (RBF) Networks and Support Vector Machines (SVM). There are two fields in which this tool (called sNet-SOM) was used, namely: ischemia detection and Data Mining.
ΠΕΡΙΛΗΨΗ

Η παρούσα διδακτορική διατριβή είναι επικεντρωμένη γύρω από την ανάπτυξη και εφαρμογή, με χαμηλές υπολογιστικές απαιτήσεις, βασισμένες σε Τεχνητά Neurωνικά Δίκτυα, για την Ανάλυση Βιοϊατρικών σημάτων και Data Mining σε Ιατρικά Δεδομένα.

Απότερος σκοπός της παρούσης διατριβής στον τομέα της Βιοϊατρικής Τεχνολογίας είναι να παρέχει στους ιατρούς με την καλύτερη δυνατή πληροφόρηση για να κάνουν μια ακριβή διάγνωση (στην περίπτωση του ισχαιμικού μυοκαρδίου) και να προτείνει αναπτυγμένα μαθηματικά μοντέλα για να ανακάλυψε πολύπλοκες εξαρτήσεις μεταξύ των μεταβλητών μιας φυσικής διεργασίας από ένα σύνολο διαφορετικών παρατηρήσεων.

Μετά την περιγραφή μερικών από τους βασικούς τύπους τεχνητών Neurωνικών Δίκτυων που χρησιμοποιούνται στην παρούσα διατριβή, εμείς αρχίζαμε να σχεδιάζουμε ένα μοντέλο για ταξινόμηση προτύπων κατασκευάζοντας πολλά τοπικά μοντέλα γειτονικά με τον παρόντα χώρο. Για αυτό το σκοπό εμείς χρησιμοποιούμε το αλγόριθμο για clustering k-windows για να ανιχνεύσει αυτόματα γειτονικές στον παρόντα χώρο. Αυτός ο αλγόριθμος με μια ελαφριά τροποποίηση έχει την ικανότητα να καθορίζει ενδογενώς την παρουσία του αριθμού των clusters στο σύνολο των δεδομένων κατά την διάρκεια της διαδικασίας του clustering. Όταν η διαδικασία του clustering ολοκληρώνεται ένα εκπαιδευμένο Εμπροσθοπροφοδοστούμενο Neurωνικό Δίκτυο δρα ως ο τοπικός προβλέπτης για κάθε cluster.

Εν συνεχεία, προτείνουμε τη χρήση εξαγόμενης στατιστικής μετρητικής απόστασης, μέσα στο γενικότερο πλαίσιο των δικτύων (GRBF). Οι κύριες λειτουργίες των GRBF (Generalized Radial Basis Functions) δικτύων διατηρούνται στο καινούργιο μετρητικό χώρο. Η δυναμική κανονικοποίηση αυτών των δικτύων μπορεί να πραγματοποιηθεί με αυτό τον τύπο αποστάσεων. Επιπλέον η πρόσφατη τεχνολογία των NN (Neural Networks) προσφέρει αποτελεσματικές λύσεις για τη μάθηση ομαλών συναρτήσεων που
βρίσκεται σε υψηλούς διαστατικούς χώρους. Δοκιμάσαμε αυτή τη λύση σε εφαρμογή
βιοπληροφορικής, μία από εμπορικές βάσεις δεδομένων και τέλος με μερικά
παραδείγματα χρησιμοποιώντας βάσεις δεδομένων από το UCI (University of California
at Irvine) από το ιατρικό πεδίο.

Συνεχίζοντας, καθιδρύουμε το δίκτυο NetSOM (network Self-Organizing Map),
που προσπαθεί να γενικεύσει (generalize) την κανονικοποίηση (regularization) και να
dώσει δυναμικές εντολές (ordering) του βασικού SOM από το διανυσματικό χώρο στο
χώρο των προσεγγιστικών συναρτήσεων. Αποτελεί μια εντολοδόχο διαδικασία για τους
tοπικούς ειδικούς πάνω από το πλέγμα των νευρώνων και για την επιλογή και το
συντονισμό τους.

Τέλος, αναλύεται μια εναλλακτική λύση του NetSOM, που χρησιμοποιεί μια
εκπαιδευμένης εντολές βασισμένες στο SOMs για τις “απλές” περιοχές και για τις
“δύσκολες” μια διαδικασία μάθησης 2-επιπέδων. Υπάρχουν 2 διαφορές στα
αποτελέσματα από την σύγκριση με το προηγούμενο μοντέλο (NetSOM), η πρώτη είναι
ότι αντικαταστήσαμε (we replaced) a fixed-size των SOM με ένα πιο δυναμικό ταίριασμα
(mapping) και η δεύτερη, η εκπαιδευόμενη εκμάθηση βασίστηκε αυτή τη φορά στην
RBF και στις μηχανές υποστήριξης διανυσμάτων (SVM). Αυτό το εργαλείο
χρησιμοποιήθηκε στην αναγνώριση των ισχαιμών και εξόρυξη δεδομένων από βάσεις
dεδομένων.
Acknowledgments

Several people have influenced the work of this thesis. The most important are: my advisor, Professor Anastasios Bezerianos, who first introduced me to Biosignal Processing, and Stergios Papadimitriou who paved the way of my first application of Computational Intelligence within a biomedical research group.

I am also thankful to my second and third advisors, Professors Michael Vrahatis (great theoretician and a friend) and Nikos Fakotakis for their guidance through the years of my Ph.D.

I am indebted to Professor Emeritus Basil Proimos, Professor George Nikiforidis, Professor Nicholas Pallikarakis (from University of Patras, Greece) and Professor Mihai Tarata (University of Medicine and Pharmacy from Craiova, Romania) for guiding me to the European Course on Medical Physics and Biomedical Engineering (that I followed as a preliminary step for my M.Sc.) within the framework of ”Erasmus” project and for creating wonderful conditions there.

My work has benefitted from conversation with a number of people, including the following: Alin Achim and Mihai Popescu - thank you for your friendship and help and also for all the good and bad times we had together; many thanks to Ovidiu Zainea and Adrian Cimponeriu (for the support they have given me in order to get the flavor of Linux). I am indebted to distinguished professors like: Iuliana Coravu, Dumitru Busneag, Oliviu Gherman, Jak Marcovic, Mircea Tatulescu and Oleg Cernian, all of whom taught and advised me during my high school and undergraduate studies, and helped me shape the mathematical and engineering basis of my career.

Thanks to all the people who have used the software or the theory described in author’s articles, contributed with comments, ideas or criticism related to it and included those publications to their reference list; a certain number of those are listed in the Chapter ”Conclusions”.

It is a pleasure to acknowledge several organizations and institutions which facilitated or sponsored my research. The work in this thesis was supported by IKY, (Greek State Scholarship Foundation)- grant 373/1997, Greek GSRT and ESF (European Social Fund)
with the program PENED (ED 146).

There is certainly a larger list of people that contributed to this work by sending at my request theses or reprints, but I cannot certainly avoid enumerating these: Ingrid Daubechies (Princeton), Judea Pearl (UCLA - University of California in Los Angeles), Simon Haykin (McMaster University, Canada) - I cannot be thankful enough to these two.

Finally, I would like to express my gratitude for my family with love: my mother, who invested so much of herself into providing me with all the best opportunities and my father who taught me about motivation perseverance, and vision. This dissertation is dedicated to them.

Patras, Greece
July, 2004

Liviu-Mihai Vladutu
Introduction

0.1 Objectives and Achievements

The aim of the work presented in this thesis was the development of computationally effective neural network solutions for biomedical signal analysis and data mining in medical records. The central element of this thesis is artificial neural network (ANN), which is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. We are going to describe in the first part of the thesis the use of ANN in Data Mining, also known as knowledge-discovery in databases (KDD) and which is the practice of automatically searching large stores of data for patterns. The process of partitioning large sets of patterns into disjoint and homogeneous clusters is fundamental in knowledge acquisition. In the literature it is called Clustering and it is applied in various fields including data mining, statistical data analysis, compression and vector quantization. The first part of the thesis is dealing with data mining in medical records, using a new unsupervised Clustering algorithm, called k-windows.

The distance between patterns in addition to providing information about the proximity of patterns should also obey some mathematical criteria in order to be applicable. Traditional distances are inadequate to access the differences between symbolic patterns. Our work continued with the utilization of a statistically extracted distance measure for Generalized Radial Basis Function (GRBF) networks. When the features are symbolic (as usually happens in bioinformatics and in commercial data mining applications) the utilization of the traditional distance metrics yields inadequate performance. In order to be able to obtain an effective description of the distances between patterns with symbolic feature
values we have adapted statistical distance measures within the context of the GRBF, which
have a form initially developed for nearest neighbor schemes. The statistical distance mea-
sure computes the distance between two values for a feature by accounting for the overall
similarity of classification over all the instances of the training set. This method extracts
with a statistical approach a separate matrix for each feature from the training set. Each
such matrix defines the distances between all possible values of a given feature.

The main properties of these networks are retained in the new metric space. Espe-
sially, their regularization potential can be realized with this type of distance. However, the
examples of the training set for applications involving symbolic patterns are not all of the
same importance and reliability. Therefore, the construction of effective decision boundaries
should consider the numerous exceptions to the general motifs of classification that are fre-
quently encountered in data mining applications. We support that heuristic Instance Based
Learning (IBL) training approaches can uncover information within the uneven structure of
the training set. This information is exploited for the estimation of an adequate subset of
the training patterns serving as RBF centers and for the estimation of effective parameter
settings for those centers.

The second part of the thesis is dealing with design of complex ANN architectures
for Myocardial Ischemia (MI) detection. MI is the most common cause of death in the
industrialized countries and, as a consequence, its early diagnosis and treatment is of great
importance. That’s why one of the main applications for biomedical signal processing that
we tackled is myocardial ischemia detection, based on recognition of ST-T segment changes,
[8], or the problem of detection of the ischemia episodes, by an intelligent design of a window
that we fed at the inputs of sNet-SOM, see  [58].

We faced the following problem of beats classification: where the state space for many
complex pattern classification problems consists of regions that lie near class separation
boundaries it is required the construction of complex discriminants while for the rest regions
the classification task is significantly simpler.

In accordance to that, two models were designed: NetSOM and the Supervised Network
Self-Organizing Map (sNet-SOM) model.
NetSOM is from the family of hierarchical ANN, also known as committee machines. It uses a fixed size internal SOM, that is optimized for the task of separating subspaces. Its task is to perform an effective state space partitioning and to map a subspace to each of its neurons. Classification decisions are performed by NetSOM only in the cases of unambiguously subspaces where an entropy-based criterion is based to quantify the ambiguity. The local experts (from the family of Multi-layer perceptrons and Radial Basis Function Networks) are supervised neural models optimized for a particular subspace created for a NetSOM neuron, and they (the local experts) handle the ambiguous region. The drawback of NetSOM was that the size of the internal SOM was designed on the base of a try-and-error method.

The other designed model, the sNet-SOM utilizes unsupervised learning for classifying at the simple regions and supervised learning for the difficult ones in a two stage learning process. The unsupervised learning approach is based again on the Self-Organizing Map (SOM) of Kohonen while supervised learning is based this time on the Generalized Radial Basis Functions (GRBF) networks and on the Support Vector Machines (SVM’s). This time, the optimal size of the internal SOM (called CP-SOM) was determined automatically based on an dynamic expansion process. The performance of the sNet-SOM has been evaluated on synthetic data, on simulated data and on an ischemia detection application with data extracted from the European ST-T database. In all cases, the utilization of sNet-SOM with supervised learning based on both Radial Basis Functions and Support Vector Machines has improved the results significantly related to those obtained with the unsupervised SOM and has enhanced the scalability of the supervised learning schemes.

0.2 Original Contributions

The original contributions of the author to biomedical signals analysis according to the material presented in this thesis can be summarized as follows:

- The new modular approach, [80], proposed by us consisted in a clustering process based on unsupervised $k$-windows algorithm combined with the use of a local FNN
(feedforward NN) as a local predictor for each cluster determined in this fashion.

- A new approach for data mining of symbolic data was introduced, see [59], with twofold contribution: adaptation of a statistical distance metric (SDM) for symbolic features and a new Radial Basis Function Networks (RBFN) design, to exploit effectively the irregularity of the problem’s state space with the selection of the proper training examples as RBF centers.

- In [8] we describe an analysis for the optimum selection (based on energy analysis) of the number of Principal Components (PC) for the representation of ECG signals. We used multiresolution analysis (wavelets) for the denoising of the Principal Components time series. It is the first application of committee machines (hierarchical mixture of experts model) for ischemia detection; the supervised experts were well-known ANN (artificial neural networks) like RBF (Radial Basis Function) Networks and MLP (multi-layer perceptrons). The topology was built around a SOM (Kohonen map).

- It was also developed a semi-supervised classification ANN architecture based on a classification partition (CP) SOM by adding a dynamic expansion process controlled by entropy-based criterion to the original SOM algorithm. This algorithm proceeds until the total number of training patterns that are mapped to neurons with high entropy (and therefore with ambiguous classification) reduces to a size that can be handled easily by a supervised model. In my knowledge, it was the first application for biomedical applications of such a modern ANN architecture based on “dynamic expansion maps”. It was applied to myocardial ischemic episodes detection, [58], but also tested with synthetic data from a Lorenz chaotic system, [60].

The research presented in this thesis contributed also to the following papers published in Conference Proceedings: [90, 91, 92].
0.3 Organization of the Thesis

The dissertation is organized as follows:

Chapter 1 provides a review of the basic wavelet concepts that we use for our later developments and after, a description of two denoising methods for denoising of time series corresponding to Principal Components.

The complexity of the neural networks architectures involved in our research determined the ordering of the other chapters contained in the present thesis.

Chapter 2 starts with a general overview on the main concepts of Self-Organizing Maps, also known as Kohonen Maps, starting with the motivation and goals that Professor Teuvo Kohonen had in his mind when he designed this type of ANN. A certain number of analogies with the natural NN are presented together with the description of the processes that take place in neural networks. It continues with a lot of technical details of implementation and fine tuning for SOM resulted from the experience of using SOM in ischemia detection. The chapter ends with the description of a fine-tuning technique, known as Learning Vector Quantization (LVQ).

Chapter 3 considers first the description of the most used type of ANN, feedforward NN (FNN) and continues with the description of the Backpropagation learning algorithm. Emphasizing the weak points of classical FNN it’s explained the need of introducing the enhanced learning algorithms for FNN like RProp and AOBP (Adaptive Stepsize Algorithm for On-line Training).

Chapter 4 begins with a taxonomy of Data-mining techniques like the well-known $k$-means Clustering and introduces the new concept of $k$-windows Clustering, first introduced by Professor M. Vrahatis, that we have used when we have revisited the problem of ischemic episodes detection using the European ST-T Database. The conclusions present the classification of a FNN when trained with different types of learning paradigms like BP, AOBP, Rprop and it’s variants, one neural network for each cluster previously detected with $k$-windows algorithm.

Chapter 5 is dealing with a brief description of Generalized RBF (Radial Basis Function) Networks, known as some of the best approximators among ANN topologies. The work presented in this paragraph, proposes the utilization of a statistically extracted distance
measure in the context of Generalized Radial Basis Function (GRBF) networks. The main properties of the GRBF networks are retained in the new metric space. The regularization potential of these networks can be realized with this type of distance. Furthermore, the recent engineering of neural networks offers effective solutions for learning smooth functionals that lie on high dimensional spaces.

Chapter 6 presents first a description of committees machines (known also as hierarchical mixture of experts), and accordingly, proposes the network self-organising map (NetSOM) model as an enhancement to the Kohonen self-organised map (SOM) model. This model is capable of effectively decomposing complex large-scale pattern classification problems into a number of partitions, each of which is more manageable with a local classification device. The NetSOM becomes a device for the ordering of local experts (i.e. independent neural networks) over its lattice of neurons and for their selection and co-ordination. Each local expert is an independent neural network that is trained and activated under the control of the NetSOM. This method is evaluated with examples from the European ST-T database.

In Chapter 7 we addressed the problem of maximizing the performance of the detection capabilities of modern ANN topologies in case of difficult pattern classification tasks. It describes a semi-supervised learning, based on classic SOM which is modified with a dynamic expansion process controlled with an entropy based criterion, that allows the adaptive formation of the proper SOM structure. The second learning phase (the supervised training) has the objective of constructing better decision boundaries at the ambiguous regions. At this phase, a special supervised network is trained for the computationally reduced task of performing the classification only for the ambiguous regions of the state space. The utilization of sNet-SOM with supervised learning based on the RBF and support vector machines has resulted in an improved accuracy of solving difficult problems in data mining and classification problems.

Chapter 8 draws some general concluding remarks and outlines future research perspectives.
Chapter 1

Wavelets in Signal Processing

This chapter is intended to be an introduction to wavelet theory, the one that was used as an auxiliary means for signal representation and processing, see Chapters 4.3, 6.4, 7.3. One can easily find a lot of good books describing wavelets, like [18, 38, 44, 51] or articles at introductory level, describing the mathematical mechanisms [14, 82]. The Wavelet theory has become a powerful alternative for the analysis of the non-stationary signals since the classical Fourier transform gives the frequency contents of the signals without providing information about the time localization of the observed frequency components. The Wavelet Transform (WT) is characterized by a frequency response logarithmically scaled along the frequency axis, as opposed to the STFT, which uses a fixed window in time domain. Thus, it provides a good time resolution at high frequencies and a good frequency resolutions at low frequencies, being appropriate to discriminate transient high-frequency components closely located in time and long duration components closely spaced in frequency.

The important properties of the WT that make it suitable for signal and image processing applications are:

- Multiresolution - the WT offers a scale invariant representation
- Sparsity - the wavelet coefficients of medical signals / images are sparsely distributed.
- Speed - there are fast algorithms for efficient decomposition and reconstruction
- Edge detection - small wavelet coefficients correspond to homogenous areas, while large wavelet coefficients correspond to signal edges.
1.1 Dyadic Wavelet Transform (DWT)

1.1.1 Series Expansion of Signals: Multiresolution Decomposition onto Orthonormal Wavelet Bases

The focus of this section is directed to series expansion of continuous and discrete time sequences, related to the concept of multiresolution decomposition for one-dimensional signals. The general reasons for such expansions are linked to signal analysis, denoising, approximations and compression.

If $L^2(R)$ denote the Hilbert space of measurable, square-integrable one dimensional functions $f(x)$, the wavelet decomposition of multiresolution type makes use of a linear approximation operator $A_{2^j}$, which transform a function $f(x) \in L^2(R)$ into approximations at different resolution levels $2^j$. The operator $A_{2^j}$ is an orthogonal projection on the vector space $V_{2^j} \subseteq L^2(R)$ of all possible approximations at resolution $2^j$ of functions in $L^2(R)$. A multiresolution approximation of $L^2(R)$ is defined as any set of vector spaces $(V_{2^j})_{j \in \mathbb{Z}}$ which satisfies the following properties [38, 42]:

1) Causality property: The approximation of a signal at a resolution $2^{j+1}$ contains all the necessary information to compute the same signal at a smaller resolution $2^j$, i.e.

$$\forall j \in \mathbb{Z}, V_{2^j} \subset V_{2^{j+1}} \quad (1.1.1)$$

2) The spaces of approximated functions should be derived from one another by scaling each approximated function by the ratio of their resolution values, i.e.

$$\forall j \in \mathbb{Z}, f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}} \quad (1.1.2)$$

3) Discrete characterization: The approximation $A_{2^j} f(x)$ of a signal $f(x)$ can be characterized by $2^j$ samples per length unit, i.e.

There exist an isomorphism $I$ from $V_1$ onto $l^2(\mathbb{Z})$

4) Translation of the approximation: When $f(x)$ is translated by a length proportional to $2^{-j}$, $A_{2^j} f(x)$ is translated by the same amount and is characterized by the same samples which have been translated, i.e.
1.1 Dyadic Wavelet Transform (DWT)

\[ \forall k \in \mathbb{Z}, A_1 f_k(x) = A_1 f(x - k) \]  
(1.1.3)

where \( f_k(x) = f(x - k) \).

5) Translation of the samples:

\[ I(A_1 f(x)) = (\alpha_i)_{i \in \mathbb{Z}} \iff I(A_1 f_k(x)) = (\alpha_{i-k})_{i \in \mathbb{Z}}. \]  
(1.1.4)

6) The approximated signal should converge to the original as the resolution increases to \( \infty \), i.e.

\[ \lim_{j \to +\infty} V_{2^j} = \bigcup_{j = -\infty}^{+\infty} V_{2^j} \]  
(1.1.5)

is dense in \( L^2(\mathbb{R}) \).

7) As the resolution decreases to zero, the approximated signal contains less and less information and converges to zero, i.e.

\[ \lim_{j \to -\infty} V_{2^j} = \bigcap_{j = -\infty}^{+\infty} V_{2^j} = \{0\} \]  
(1.1.6)

Stephane Mallat, [42] has shown that an orthonormal basis of \( V_{2^j} \) can be defined using only dilations and translations of an unique function \( \Phi(x) \). He proved that if \( (V_{2^j})_{j \in \mathbb{Z}} \) is a multiresolution approximation of \( L^2(\mathbb{R}) \), called a scaling function, such that if we set \( \Phi_{2^j}(x) = 2^j \Phi(2^j x) \), for \( j \in \mathbb{Z} \), (the dilation of \( \Phi(x) \) by \( 2^j \)), then \( (\sqrt{2^{-j}} \Phi_{2^j}(x - 2^{-j} n))_{n \in \mathbb{Z}} \) is an orthonormal basis of \( V_{2^j} \). Thus, the approximation of the signal \( f(x) \) at the resolution \( 2^j \), \( A_{2^j} f(x) \), can be characterized by a set of inner products, which are denoted by:

\[ A_{2^j}^d f = \left\{ \langle f(u), \Phi_{2^j}(u - 2^{-j} n) \rangle \right\}_{n \in \mathbb{Z}} \]  
(1.1.7)

\( A_{2^j}^d \) is called a discrete approximation of \( f(x) \) at the resolution \( 2^j \). The regularity condition requires that the scaling function \( \Phi(x) \) must be continuously differentiable and the asymptotic decay of \( \Phi(x) \) and \( \Phi'(x) \) at infinity must satisfy the conditions:

\[ |\Phi(x)| = O(x^{-2}) \quad \text{and} \quad |\Phi'(x)| = O(x^{-2}) \]
In the same article as above, Mallat, [42], gives a practical characterization of the Fourier transform of a scaling function. If $H$ is a discrete filter with impulse response $h(n) = \langle \Phi_{2^{-1}}(u), \Phi(u - n) \rangle$, then the Fourier series $H(\omega)$ satisfies the following two properties:

$$|H(0)| = 1, h(n) = O(n^{-2}), n \to \infty$$

(1.1.8)

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1$$

(1.1.9)

Conversely, if $H(\omega)$ is a Fourier series satisfying 1.1.8 and 1.1.9 and such that:

$$|H(\omega)| \neq 0, \omega \in \left[0, \frac{\pi}{2}\right],$$

(1.1.10)

then the function defined by:

$$\Phi(\omega) = \prod_{p=1}^{+\infty} H(2^{-p}\omega)$$

(1.1.11)

is the Fourier transform of a scaling function. Thus, the smoothness class of $\Phi(\omega)$, and its asymptotic decay at infinity can be estimated from the properties of $H(\omega)$, see [17]. The filters that satisfy property 1.1.9 are called conjugate filters.

The *multiresolution decomposition* is based on the difference of information available at 2 successive resolutions $2^j$ and $2^{j+1}$, which is called *detail signal* at the resolution $2^j$. One can show that the detail signal at the resolution $2^j$ is given by the orthogonal projection of the original signal on the orthogonal complement of $V_{2^j}$ in $V_{2^{j+1}}$. Let $O_{2^j}$ be this orthogonal complement, i.e. $O_{2^j}$ is orthogonal to $V_{2^j}$ and $O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}$. Mallat, [42] showed that an orthogonal basis of $O_{2^j}$ can be built by scaling and translating a function $\Psi(x)$ which is called an orthogonal wavelet. In this thesis, we have used exclusively only orthogonal wavelets. Specifically, let $\Psi(x)$ be a function whose Fourier transform is given by

$$\Psi(x) = G(\frac{x}{2})\Phi(\frac{x}{2}), \text{ with } G(\omega) = e^{-i\omega H(\omega + \pi)}$$

If $\Psi_{2^j}(x) = 2^j\Psi(2^jx)$ denote the dilation of $\Psi$ by $2^j$, then $(\sqrt{2^{-j}}\Psi_{2^j}(x - 2^{-j}n)_{(n,j) \in \mathbb{Z}^2})$ is an orthonormal basis of $L^2(R)$. As a consequence, if $P_{O_{2^j}}$ is the orthogonal projection on the vector space $O_{2^j}$, then we can write this operator as:

$$P_{O_{2^j}} f(x) = 2^{-j} \sum_{n = -\infty}^{+\infty} < f(u), \Psi_{2^j}(u - 2^{-j}n) > \cdot \Psi_{2^j}(x - 2^{-j}n)$$

(1.1.12)
1.1 Dyadic Wavelet Transform (DWT)

$P_{O_{2^j}}$ yields to the detail of $f(x)$ at the resolution $2^j$, and it is characterized by the set of inner products:

$$D_{2^j} f = \langle f(u), \Psi_{2^j}(u - 2^{-j}n) \rangle_{n \in \mathbb{Z}}$$  \hspace{1cm} (1.1.13)

$D_{2^j} f$ is called the discrete detail signal at the resolution $2^j$. It can be proved that each of these inner products is equal to the convolution of $f(x)$ with $\Psi_{2^j}(-x)$ evaluated at $2^{-j}n$, i.e.:

$$\langle f(u), \Psi_{2^j}(u - 2^{-j}n) \rangle = (f(u) \ast \Psi_{2^j}(-u))(2^{-j}n)$$  \hspace{1cm} (1.1.14)

Furthermore, equations 1.1.13 and 1.1.14 show that the discrete detail signal at the resolution $2^j$ is equal to a uniform sampling of

$$(f(u) \ast \Psi_{2^j}(-u)(x))$$

at rate $2^j$:

$$D_{2^j} f = ((f(u) \ast \Psi_{2^j}(-u))(2^{-j}n))_{n \in \mathbb{Z}}.$$  \hspace{1cm} (1.1.15)

It can be proved by mathematical induction that for any $J > 0$, the original discrete signal $A_1^d f$ measured at resolution 1 can be represented by:

$$(A_{2^{-j}}^d, (D_{2^j} f)_{-J \leq j \leq -1})$$  \hspace{1cm} (1.1.16)

This set of discrete signals is called an orthogonal wavelet representation, and consists of the reference signal at a coarse resolution $A_{2^{-j}}^d f$ and the detail signals at the resolution $2^j$ for $-J \leq j \leq -1$. It can be interpreted as a decomposition of the signal in a set of independent frequency channels.

1.1.2 Pyramidal Implementation of Dyadic Wavelet Transform

I shall try to briefly describe now the iterative algorithm described by Mallat [42] for computation of the discrete approximations. In practice we are dealing with signals at finite resolution. For normalization purpose, let’s suppose that this resolution is equal to 1, i.e.
$A^d_1 f$ is the discrete approximation that is measured. The causality principle, 1.1.1 asserts that from it we can compute all the discrete approximations $A^d_{2j} f$ for $j < 0$. Let $(V_{2^j})_{j \in \mathbb{Z}}$ be a multiresolution approximation and let $\Phi(x)$ be the correspondent scaling function. Let say that $\tilde{H}$ be the mirror filter which has an impulse response equal to $h(-n)$; I have introduced this filter $H$ in the previous subsection. Thus, we can obtain:

$$\langle f(u), \Phi_{2j}(u - 2^{-j}n) \rangle = \sum_{k=-\infty}^{+\infty} \tilde{h}(2n - k) \cdot \langle f(u), \Phi_{2j+1}(u - 2^{-j-1}k) \rangle$$  \hspace{1cm} (1.1.17)

The equation 1.1.17 shows that $A^d_{2j} f$ can be calculated by a convolution between $A^d_{2j+1} f$ with $\tilde{H}$ and keeping every other sample of the output. All the discrete approximations $A^d_j f$, for $j < 0$ can thus be computed from $A^d_1 f$ by iterating this process. This operation is called a pyramidal transform and the algorithm is illustrated by the upper part from figure 1.1, where $D \downarrow$ is the binary decimation operator.

![Diagram](image1.png)

Figure 1.1: (a) Block-diagram of the pyramidal decomposition algorithm used for DWT computation. (b) Block-diagram of the pyramidal reconstruction algorithm

If the original signal has a finite number of samples, $A^d_1 f = (\alpha_n)_{1 \leq n \leq N}$, then each
discrete signal $A_d^j f(j < 0)$ has $2^j N$ samples (to avoid border problems when computing $A_d^j f$) the signal $A_d^j f$ is symmetrized with respect to $n = 0$ and $n = N$).

With the same derivation steps it can be shown that $D_{2^j} f$ can be calculated by a convolution between $A_{2^j+1} f$ with a discrete filter $G$. For any $n \in \mathbb{Z}$, the function $\Psi_{2^j}(x - 2^{-j}n)$ is a member of $O_{2^j} \subset V_{2^j+1}$ and it can be expanded in an orthonormal basis of $V_{2^j+1}$.

Let $G$ be the discrete filter having the impulse response:

$$ g(n) = \langle \Psi_{2^{-j}}(u), \Phi(u - n) \rangle $$ (1.1.18)

and $\overline{G}$ be the symmetric filter given by: $\overline{g}(n) = g(-n)$. Thus it can be shown [42] that:

$$ \left\langle f(u), \Psi_{2^j}(u - 2^{-j}n) \right\rangle = \sum_{k=-\infty}^{+\infty} \overline{g}(2n - k) \left\langle f(u), \Phi_{2^{j+1}}(u - 2^{-j-1}k) \right\rangle $$ (1.1.19)

Equation 1.1.19 shows that we can compute the detail signal $D_{2^j} f$ by convolving $A_{2^j+1} f$ with the filter $\overline{G}$ and retaining every other sample of the output. The orthogonal wavelet representation of a discrete signal $A_d^j f$ can therefore be computed by successively decomposing $A_{2^j+1} f$ into $A_{2^j} f$ and $D_{2^j} f$ for $-J \leq j \leq -1$. This algorithm is illustrated by the block diagram shown in Figure 1.1.

There is a relation that links the impulse response of filter $G$ to the impulse response of the filter $H$, which is:

$$ g(n) = (-1)^{1-n} h(1 - n) $$ (1.1.20)

In signal processing $G$ and $H$ are called quadrature mirror filters [77]. If the original signal has $N$ samples, then the discrete signals $D_{2^j} f$ and $A_{2^j} f$ have $2^j N$ samples each.

Thus, a wavelet representation $(A_{2^j-1}^j, f, (D_{2^j} f)_{-J \leq j \leq -1})$ has the same total number of samples as the original approximated signal $A_d^j f$ (this occurs due to the orthogonality of the representation).

As previously shown, the wavelet representation is complete. It can be proved that the original discrete signal can also be reconstructed with a pyramid transform. Briefly, $A_{2^j+1} f$ can be reconstructed by putting zeroes between each sample of $A_{2^j} f$ and $D_{2^j} f$ and convolving the resulting signals with the filters $H$ and $G$, respectively. A similar process process
can be found in the reconstruction algorithm of Burt and Adelson from their Laplacian pyramid, [12]. Figure 1.1 b) illustrates this algorithm.

1.2 Daubechies Family of Regular Filters and Wavelets

In Section 1.1.1 it has been shown that we can construct orthonormal families of functions where each function is related to a single prototype wavelet through shifting and scaling. This construction is a direct continuous-time approach based on the concept of multiresolution analysis. A different, indirect approach starts from discrete-time filters, which can be iterated and, under certain conditions, leads to continuous time prototype wavelet and the corresponding derived orthonormal families of functions. This construction pioneered by Daubechies [17] provides very practical wavelet decomposition schemes, implementable with the pyramidal algorithms described in Section 1.1.2 and based on finite-length discrete time filters. The method of construction can be demonstrated by starting from a M-channel orthogonal filter bank as illustrated in figure 1.2.

![Diagram of M-channels filter bank]

**Figure 1.2:** M-channels filter bank with analysis filters $H_0...H_{M-1}$ and synthesis filters $F_0...F_{M-1}$.

The discrete time-domain low-pass and high-pass analysis filters are denoted by $h[n]$
Figure 1.3: Discrete dyadic wavelet transform from the iteration of a 2-channel filter bank and \( g[n] \), while the synthesis filters are denoted by \( \tilde{h}[n] \) and \( \tilde{g}[n] \), respectively. It should be mentioned that orthogonality imposes that the impulse response of the analysis filters are the time-reversed versions of the synthesis filters \([38]\). Considering that the filter bank is iterated on one of the 2 branches with the low-pass filter as shown in figure 1.3, the two equivalent filters after \( i \) steps can be expressed in the \( z \)-domain using the fact that filtering with \( \tilde{H}(z) \) followed by upsampling by 2 is equivalent to upsampling by 2 followed by filtering with \( \tilde{H}(z^2) \), as follows:

\[
\tilde{H}^{(i)}(z) = \prod_{k=0}^{i-1} \tilde{H}(z^{2^k}), \quad (1.2.1)
\]

\[
\tilde{G}^{(i)}(z) = \tilde{G}(z^{2^{i-1}}) \prod_{k=0}^{i-2} \tilde{G}(z^{2^k}), \quad i = 1, 2, \ldots \quad (1.2.2)
\]

The discrete-time iterated filters \( \tilde{h}^{(i)}[n] \) and \( \tilde{g}^{(i)}[n] \) are associated with the continuous-time functions \( \varphi^{(i)}(t) \), \( \psi^{(i)}(t) \) as follows:

\[
\varphi^{(i)}(t) = 2^{i/2} \tilde{h}^{(i)}[n], \quad \frac{n}{2^i} \leq \frac{n + 1}{2^i} \quad (1.2.3)
\]

\[
\psi^{(i)}(t) = 2^{i/2} \tilde{g}^{(i)}[n], \quad \frac{n}{2^i} \leq \frac{n + 1}{2^i} \quad (1.2.4)
\]

We divided the elementary interval by \( \frac{1}{2^i} \) in order to ensure that the associated continuous function remains compactly supported despite the fact that the length of the equivalent discrete-time filters increases with each iteration. The factor \( 2^{i} \) that multiplies the iterated discrete-time filters is necessary to preserve the \( L^2 \) norm between the discrete and continuous-time cases. Figure 1.4 illustrates the graphical function corresponding to the
first four iterations of a length-4 Daubechies’ filter, indicating the piecewise constant approximation and the halving of the interval.

The Smith-Barnwell condition expresses the perfect reconstruction together with orthonormality and is:

\[ |M(e^{j\omega})|^2 + |M(e^{j(\omega + \pi)})|^2 = 1 \]  

(1.2.5)

where \( M(e^{j\omega}) = \tilde{H}(e^{j\omega})/\sqrt{2} \) is normalized such as \( M(1) = 1 \) and \( M(\pi) = 0 \)

Figure 1.4: Graphical scaling functions corresponding to the first four iteration of the orthonormal 4-tap Daubechies filter.

A discrete-time filter is called *regular* if it converges through the iteration scheme to a scaling function and to a wavelet with some degree of regularity as piecewise smooth, continuous or derivable. We imposed the following condition on \( M(e^{j\omega}) \) for regularity.
related purpose:

\[ M(e^{j\omega}) = \left[ \frac{1}{2} (1 + e^{j\omega}) \right]^N R(e^{j\omega}), \quad N \geq 1 \]  

(1.2.6)

i.e. \( M(e^{j\omega}) \) must have \( N \) zeroes at \( \omega = \pi \).

Hence, we can write \(|M(e^{j\omega})|^2\) as:

\[ |M(e^{j\omega})|^2 = [\cos^2 \frac{\omega}{2}]^N |R(e^{j\omega})|^2. \]  

(1.2.7)

Since \(|R(e^{j\omega})|^2 = R(e^{j\omega}) \cdot R^*(e^{j\omega}) = R(e^{j\omega}) \cdot R(-e^{-j\omega})\) it can be expressed as a polynomial in \( \cos^2 \frac{\omega}{2} \). We can introduce now 2 notations (shorthands): \( y = \cos^2 \frac{\omega}{2} \) and \( P(1-y) = |R(e^{j\omega})|^2 \). So, I can rewrite the equation 1.2.5 as:

\[ y^N P(1-y) + (1-y)^N P(y) = 1, \quad P(y) \geq 0, \quad y \in [0,1] \]  

(1.2.8)

I suppose that exist a polynomial \( P(y) \) that satisfies the previous condition and moreover,

\[ \sup_{\omega} |R(e^{j\omega})| = \sup_{y \in [0,1]} |P(y)|^{\frac{1}{2}} < 2^{N-1}, \]  

(1.2.9)

then there exist an orthonormal basis associated with \( \tilde{H}(e^{j\omega}) \) since the iterated filter will converge to a continuous scaling function from which a wavelet basis can be obtained as shown in Section, 1.1.1. I. Daubechies, [17, 18] has shown that any polynomial \( P \) solving the equation 1.2.8 it has the form:

\[ P(y) = \sum_{j=0}^{N-1} \left( \begin{array}{c} N - 1 + j \\ j \end{array} \right) y^j + y^N Q(y), \]

where \( Q \) is an antisymmetric polynomial. Furthermore Daubechies constructed filters of minimum order, i.e. \( Q = 0 \), called \textit{maximally flat filters} (they have a maximum number of zeros at \( \omega = \pi \). The \( R \) is derived from \( P \) using spectral factorization [38]. Figure 1.5 illustrates the iterated graphical scaling function and wavelet for \( N=8 \) (the twelfth iteration is plotted). The corresponding filters coefficients are also illustrated (the filter coefficients for different values of \( N \) are tabulated in [17, 38].
Figure 1.5: Iterated graphical scaling function and wavelet for N=8 (the twelfth iteration is plotted) and the corresponding Daubechies’ decomposition and reconstruction filters coefficients.
1.3 Wavelets Shrinkage

1.3.1 General Principles of Wavelets Shrinkage

The joint time-frequency analysis effected by the WT provides natural settings for statistical applications, which include estimation, filtering and compression. Particularly, a considerable effort has been recently directed to develop asymptotically minimax methods based on the orthogonal wavelet transform in order to recover signals from noisy data, the greatest contribution being of D. Donoho [19, 20]. Donoho & Johnstone start from the following additive model of a discrete signal \( g \) and noise \( \epsilon \):

\[
\mathbf{f} = \mathbf{g} + \mathbf{\epsilon}
\]

(1.3.1)

Specifically, the signal \( g \) that should be recovered is a deterministic signal. The vector \( f \) is the recorded signal, while the noise \( \epsilon \) is a vector of independent and identically distributed (i.i.d.) random variables with distribution \( N(0, \sigma^2) \).

1.3.2 Hard and Soft Thresholding

One of the most popular approach for signal denoising is wavelet thresholding, due to its simplicity. We decompose now the noisy signal, in an orthogonal basis, \( \mathbb{B} = \{b_m\}_{0 \leq m \leq N} \):

\[
\langle f, b_m \rangle = \langle g, b_m \rangle + \langle \epsilon, b_m \rangle
\]

Since, as I said, \( \epsilon \) is Gaussian white, the inner products:

\[
\langle \epsilon, b_m \rangle = \sum_{n=0}^{N-1} \epsilon[n]b_m^*[n]
\]

are independent Gaussian random variables of variance \( N(0, \sigma^2) \). We are trying now to compute an estimator \( \bar{Y} \) with a ”diagonal” attenuation of the decomposition coefficients:

\[
\bar{Y} = \sum_{m=0}^{N-1} < f, b_m > \theta[m]b_m
\]

(1.3.2)

Since \( f = g + \epsilon \) and \( \mathbb{E}\{\epsilon, b_m^2\} = \sigma^2 \), we derive that:

\[
\mathbb{E}\{< g, b_m > - < f, b_m > \theta[m]\}^2 = | < g, b_m > \theta[m]^2 + \sigma[m]^2.
\]

(1.3.3)
This error is minimum for:

$$\theta[m] = \frac{|<g,b_m>|^2}{|<g,b_m>|^2 + \sigma^2}$$

(1.3.4)

**Ideal Coefficient Selection**

In practice we cannot implement an *ideal attenuation* since we don’t know the values $<g,b_m>$. A simple estimator is obtained by restricting $\theta[m] \in \{0,1\}$. This means that $\tilde{Y}$ is computed by selecting a subset of the noisy coefficients $<f,b_m>$. The error 1.3.3 is minimized by:

$$\theta[m] = \begin{cases} 1, & \text{if } |<g,b_m>|^2 \geq \sigma^2 \\ 0, & \text{if } |<g,b_m>|^2 < \sigma^2 \end{cases}$$

(1.3.5)

**Thresholding estimators**

Since as we said we can’t implement the ideal coefficient selection, we tried the approach of Donoho & Johnstone, which proved that the performance of the thresholding estimator is close to ideal coefficient selections and attenuations. We study the diagonal estimators $\tilde{Y}$ of each $g$ that attenuate independently each coefficient $<f,b_m>$ with some nonlinear function $\theta(x)$:

$$\tilde{Y} = \sum_{m=0}^{N-1} \theta(<f,b_m>)b_m$$

(1.3.6)

A hard thresholding estimator is implemented with:

$$\theta(x) = \theta_h(x) = \begin{cases} x, & \text{if } |x| > T \\ 0, & \text{if } |x| \leq T \end{cases}$$

(1.3.7)

A soft thresholding estimator $\tilde{Y}$ is computed from 1.3.6 with the soft-thresholding function:

$$\theta(x) = \theta_s(x) = \begin{cases} x - T, & \text{if } x \geq T \\ x + T, & \text{if } x \leq -T \\ 0, & \text{if } |x| \leq T \end{cases}$$

(1.3.8)

We have tried both thresholding for the domain of principal components coefficients [8] and the soft-thresholding method outperformed hard-thresholding method, at a reasonable computational cost.
1.4 Application: Denoising of Principal Components time series

We shall focus now on a very interesting application, that requested the use of wavelet analysis above presented. In the time series representation of the principal components (PC’s), see 6.4, the ischemic episodes appear as peaks. Following the extraction of principal components a noise reduction approach is used to improve these coefficients. The utilization of an advanced wavelet denoising technique has improved the classification results. The selected noise reduction approach relies on the possibility that we have to modify the properties of the ECG signal by processing its Wavelet Transform (WT) modulus maxima and to reconstruct the corresponding function [45, 46, 56, 57]. The WT modulus maxima carry the most important signal information that corresponds to the signal singularities. It is important to find the location of signal singularities and distinguish them from noisy components. This is accomplished by computing the Lipschitz regularity exponents [46] that are used for the characterization of singularities. If the Lipschitz regularity exponent is positive, the amplitude of the WT modulus maxima should increase when the scale increases. On the contrary, negative Lipschitz regularity maxima correspond to decreasing amplitudes as the scale increases [43]. Since noise creates always singularities with negative Lipschitz regularity exponents [45] (in contrast to ECG signal singularities), we can discriminate the modulus maxima created by the noise from those produced by the signal, by analyzing the evolution of their amplitude across scales. The adopted denoising algorithm detects modulus maxima with amplitude that decreases when the scale increases, indicating that the corresponding singularities have negative Lipschitz regularity exponent. These maxima are removed since they are dominated by noise components. The remaining maxima reconstruct successfully the denoised ECG signal with the inverse WT. The separation of noise components from the singularities of the ECG signal becomes sometimes problematic. This is due to the fact that the regions of high variability in the ECG signal (most notably the QRS peak) produce Wavelet Maxima with complicated evolution across scales. The simple rule of extracting those maxima that increase toward smaller scales is not always appropriate since it can destroy information on ECG structures. We found that it is difficult to
state explicit rules that separate the ECG Wavelet Maxima from the noise ones. Therefore, we have trained a Radial Basis Function [7, 26] neural network to accomplish this classification task by learning the dynamics of the evolution of the Wavelet Transform Maxima of the ECG signal. The demonstrated performance by the neural network is superior to the applicability of simple non-linear filtering rules. The utilization of Wavelet Denoising in the domain of Principal Component coefficients has resulted in a significant improvement of the classification performance. We have also used other methods for denoising, including a method based on soft thresholding [20]. We have chosen five levels of wavelet decomposition and Daubechies-type wavelets. This latter method is faster than the previously described being more amenable to an online monitoring application.
Figure 1.7: Plot of denoised time series obtained from recording e0103 with wavelet soft-thresholding technique
In figure 1.6 we include a representation of the PC time series before the denoising; after the denoising are represented in 1.7. For the PC extraction we used the e0103 record from the European ST-T Database, the V4 channel.
Chapter 2

The Self-Organizing Map

2.1 Introduction

The Self-Organizing Map is a special class of unsupervised artificial neural networks (NN); in fact Self-organizing maps (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduce the dimensions of data through the use of self-organizing neural networks. The problem that data visualization attempts to solve is that humans simply cannot visualize high dimensional data as is, so techniques are created to help us understand this high dimensional data.

In mathematical regression as we know, some simple function is usually fitted to the distribution of the sample values corresponding to the input data.

For the SOM, the "nonparametric regression" involves fitting a number of ordered discrete reference vectors to the input space. More details about the reference vectors are presented in subchapter containing the Voronoi Tessellation description.

In order to approximate the continuous function the reference vectors are here made to define the nodes of a kind of hypothetical "elastic network", whereby the topological order characteristic of this mapping, and a certain degree of regularity of the neighboring reference vectors ensue from their local interactions, reflecting a kind of "elasticity". The various neurons from the map develop into specific decoders or detectors of their respective signal domains in the input space. These decoders are formed onto the net in a meaningful order, as if some feature co-ordinate system were defined over the network. The basic principle of the computational maps, please see also [32] for further details, it is called
the principle of topographic map function, which may be stated as it follows: The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature of data drawn from the input space. The SOM are based on competitive learning; the output neurons compete among themselves to be activated or fired, with the result that only one output neuron, or one neuron per group, is on at any one time. An output neuron that wins the competition is called winner-takes-all neuron or simply a winning neuron.

One way of inducing a winner-takes-all competition among output neurons is to use lateral inhibitory connections (i.e. negative feedback paths) between them. This original idea originates from F. Rosenblatt, see also [68].

The principal goal of a self-organizing map (SOM) is to transform an incoming signal of high dimensionality into a one- or two-dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. So SOMs accomplish two things, they reduce dimensions and display similarities.

In the figure below 2.1 I shall represent the schematic diagram of a two-dimensional lattice of neurons commonly used as the discrete map:

As one can easily see, each neuron in the lattice is fully connected to all the source nodes, here in the figure, there are 3 (in the application of ischemia detection there are 5 source nodes, being 5 principal components) neurons in the input layer. This network represent in fact a feedforward topology of a neural network with a single computational layer, having all the neurons arranged in rows and columns. In our implementation we arrived to a layer composed of 10 rows and 10 columns, after experiments. Of course there are some criteria that I followed for the optimum size of a SOM:

* if the network is too small it can not learn all the information from the input space so it does not have good classification performance (it has only a partial ”image” of the input space);

* if the dimension of the SOM is too large, it is called ”overfitted” - the network not only extracts the useful information from the input space but also ”learn the noise” embedded in the signal we analyze.

There is also, as I mentioned, an other class of SOM, that of one-dimensional lattice, which is a special case of the lattice depicted in Figure 2.1. In this particular case, there is only one layer consisting of one row and one column. Each input pattern that is presented
Figure 2.1: Two-dimensional (4 rows by 4 columns) lattice of neurons
to the network it consists typically of a localized region or "spot" of activity against a quiet background.

Generally there are two great classes of self-organized feature maps:

- Willshaw-von der Malsburg’s model;
- Kohonen model.

The second one it is preferred in the literature of speciality due to the following reasons:

- it is more general than the Willshaw-von der Malsburg’s model; captures the essential features of the computational maps remaining computationally tractable;

- it does not necessary try to explain neurobiological details as Willshaw-von der Malsburg’s model does.

These are the reasons for us to choose the Kohonen model for implementation, see also [32], a seminal work for the field of SOM. The implemented algorithm starts with the formation of the map by initializing the synaptic weights in the network; for this purpose we assigned small random values; we did so, in order not to impose a prior order in the map. After the initialization of the map there are three major processes involved in the evolution of the network and these are:

1) **Competition** For each input pattern (from the training set) the 100 neurons in the map compute the values of a discriminant function. This discriminant function provides the basis for the competition among the neurons. This discrimination function is presented in 2.2. The particular neuron that has the larger value for the discriminant function it is declared winner of the competition. It can be finally one neuron that wins the competition or a group of neurons.

2) **Cooperation** This winning neuron determines the spatial location of a topological neighborhood of excited neurons, thereby providing the basis for a cooperation among such neighboring neurons. At the beginning of the training phase, we start with a larger value for the Nearby (describing the neighborhood of the winning neurons) and then this neighborhood decrease in time (it shrinks). It is essential to have initially a large enough value of the nearby value; otherwise, the map will not be ordered globally and we will get various types of mosaic-like tessellation, between which the ordering direction changes continuously.
3) Synaptic adaptation This last process enables the excited neurons to increase their individual values of the discriminant function in relation to the patterns presented at the input, through suitable adjustments applied to their synaptic weights. The adjustments are made in such a way to enhance the response of the winning neuron to a similar input pattern.

2.2 Basic SOM

In this section I present the processes involved in the formation of SOM, the phases of the adaptive process and in the end I’ll discuss about k-Nearest Neighbor and a summary of SOM properties.

Processes involved in the formation of SOM

A. Competitive Process

If we denote by \( m \) the dimensionality of the input space (in our case \( m=5 \)) an input pattern (vector), selected randomly may be denoted by:

\[
x = [x_1, x_2, \ldots, x_m]^T.
\] (2.2.1)

The synaptic weights vector of each neuron in the network has the same dimension as the input space. This vector can be denoted as it follows:

\[
w_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T, \ j = 1, 2, \ldots, l
\] (2.2.2)

where \( l \) is the total number of neurons in the network (in our case \( l=100 \)). Our goal is, as mentioned previously, to find the best match of the input vector \( \mathbf{x} \) with the synaptic weights vector, \( \mathbf{w}_j \); for this, the inner product \( \mathbf{w}_j^T \mathbf{x} \) is computed for all neurons in the network and select the largest. This is the discriminant function to select the winner. This assumes that we apply the same threshold to all the neurons. Generally, for all types of neural networks,
the threshold is the negative of the bias. In this way, by selecting the neuron (with the largest inner product) we determine the location where the topological neighborhood of excited neurons is to be centered. Now remains the problem of the \textit{winner’s neighborhood}: precisely, which should be the criterion to identify the neuron which best matches \( \mathbf{x} \), the input pattern. If we say that \( i(x) \) is the index that best matches \( \mathbf{x} \), we may determine \( i(x) \) by applying the condition:

\[
i(x) = \arg \min_j \| x - w_j \|, j = 1, 2, \ldots, l \tag{2.2.3}
\]

where \( \| \| \) represents the Euclidean distance. The above equation, 2.2.3 sums the essence of the competition process among neurons. Again there is a problem, since there are also other types of distances, (not only the Euclidean distance which is used in the above equation). There is an as called ”city distance” (or ”Manhattan distance” after other authors); after this type of criterion, finding the best matches should be:

\[
i(x) = \arg \min_j (|x - w_j|), j = 1, 2, \ldots, l \tag{2.2.4}
\]

It proved that in our case the ”MANHATTAN” type distance produced better results for the classification. We focus our attention on \( i(x) \), since we can in this way identify the neuron that \textit{best-matching or winning} neuron for the input vector \( \mathbf{x} \). Equations 2.2.3, or 2.2.4 lead us to the following concluding statement:

A continuous input space of activation patterns is mapped onto a discrete output space of neurons by a process of competition among the neurons in the network.

### B. Cooperative Process

Until now we found a method which determines the center of a topological neighborhood of cooperating neurons. The problem that we have to face is: how to determine the topological neighborhood that is correct from the neurobiological point of view? There is strong evidence in nature that exists neurobiological evidence for \textit{lateral interaction} among a set of
excited neurons. A neuron that is fired tends to excite the neurons that are located in it’s immediate neighborhood than those which are located farther away from it. Simulations performed during many years by a great number of researchers have convincingly shown that the best self-organizing results are obtained if the following two partial processes are implemented in their purest form:

1) Decoding of that $m_i$, denoted by $m_c$, ("winner") that has the best match with $x$.  
2) Adaptive improvement of the match in the neighborhood of neurons centered around the "winner", as described at A) -Competitive process.

As described in [33] by T. Kohonen, the following type of control of neighborhood as proposed by the author represents a new direction in neural modelling: the "winner neuron" directly modulates the synaptic plasticity in the lateral direction. According to this, for modelling the physiological SOM process we need to define two separate interaction kernels as in the Figure 2.2, below:

The kernel represented in Figure 2.2 b) is what interest us now, which defines how local activity determines the learning rate in it’s neighborhood. This kernel is nonnegative and usually in the literature, see also [26, Haykin’s book], is taken as a Gaussian. Since is an important, the neighborhood function is once again represented below, with more details:
Figure 2.3: A typical neighborhood function

In the caption above, \( h_{j,i}(x) \) is the neighborhood, \( d_{j,i}(x) \) is the lateral distance from the winning, \( \sigma \) is the "effective width". One can easily see that:

a) The amplitude of \( h_{j,i}(x) \) (the neighborhood function) decays smoothly (from the maximum point, also center of symmetry, defined below) with the increasing lateral distance \( d_{j,i}(x) \) from the winning neuron \( x_i \) and the excited neuron \( x_j \); also is decaying to 0 when \( d_{j,i}(x) \to \infty \); this is a necessary condition for convergence;

b) \( h_{j,i}(x) \) it has a maximum value (1.0) at the origin \( (d_{j,i}(x) = 0) \);

c) The topological amplitude it is symmetric about the maximum point defined above at b); A typical choice (as we did as well) for this \( h_{j,i}(x) \) is a Gaussian function, which has the expression given below:

\[
h_{j,i}(x) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right) \tag{2.2.5}
\]

This expression has an important feature: it is *translation invariant*, so it does not
depend on the location of the winning neuron. The parameter $\sigma$, which it is called "the effective width" of the topological neighborhood as is illustrated in the above Figure 2.3. In order to have a cooperation that holds the topological neighborhood $h_{j,i}(x)$ must be dependent on the lateral distance $d_{j,i}(x)$ between the winning neuron $x_i$ and the excited neuron $x_j$ and not on some distance measure in the original input space. This is what is reflected in equation 2.2.5. In the case of an one dimensional lattice, $d_{j,i}(x)$ is given by an integer given by $|j - i|$ and in the case of a two-dimensional lattice, by the equation 2.2.6:

$$d_{j,i}^2 = \| r_j - r_i \|^2$$  \hspace{1cm} (2.2.6)

where $r_j$ describes the position of the excited neuron and $r_i$ describes the discrete position of the winning neuron $i$; as I mentioned before, both of these two are measured in the discrete output space.

*Neighborhood shrinkage* In time the topological neighborhood of a self-organized map shrinks that is, $h_{j,i}(x)$ decreases with time. This is an other feature of SOM, that is made possible by making the width $\sigma$ of the topological neighborhood dependent after an exponential law on the discrete time law on the discrete time (see also [54, 65]) as in equation 2.2.7 below:

$$\sigma(n) = \sigma_0 \exp(-\frac{n}{\tau_1}), \quad n = 0, 1, \ldots$$  \hspace{1cm} (2.2.7)

so $n$ is the discrete time and $\tau_1$ is a *time constant*. We have the "width" $\sigma$ depending on the discrete time $n$, and so equation 2.2.5 becomes:

$$h_{j,i}(x)(n) = \exp(-\frac{d_{j,i}^2}{2\sigma^2(n)})$$  \hspace{1cm} (2.2.8)

It can be easily seen from the above expression that if $n$ increases, the topological neighborhood shrinks (it decreases with time) at an exponential rate. Subsequently we will call $h_{j,i}(x)(n)$ *neighborhood function*, where $n$ is the discrete time with integer values. Other interesting approach about this neighborhood function shrinkage is presented by Luttrell
in [37]: initially a wide value is selected for $h_{ji}(x)(n)$ in order to correlate the directions of weight variation for a large number of neurons from the network. I selected a nearby of 6 (for a lattice of 10 rows by 10 columns- so even greater than half the diameter of the SOM dimension); in time, during the learning phase this nearby decreases to 0, so we will have finally just one winner (WTA- winner-takes-all approach). As the width of $h_{ji}(x)(n)$ is decreased, in the same time the number of neurons whose update directions are correlated is also decreased.

C. The adaptive process The last process of the formation of a self-organized feature map is the adaptive process. It is already known (we analyzed at the self-organized maps) that the base principle is required to have the change of the synaptic weights vector $w_j$ in relation with the input vector $x$. This is called the Hebb’s postulate of learning.

But, if we have the rule of this change only in one direction, this will lead to a saturation. It is therefore mandatory to introduce an as called forgetting term $g(y_j)w_j$ and so, the equation of the change of synaptic weights becomes:

$$
\Delta w_j = \eta y_j x - g(y_j)w_j
$$

In 2.2.9 the first term from the right part of the equation represents the Hebbian learning term and the second, as discussed the forgetting term. Also $\eta$ in 2.2.9 represent the learning rate parameter, a very important variable to set and control in order to have a proper behavior of the SOM (and of a neural network in general).

In the unsupervised-learning phase we started with a small value of the learning rate ($0.1$) and we ended with $\eta = 0.01$. The supervised learning phase (learning vector quantization) requires even less values for the initial learning rate value, usually $10^{-4}$ and ends with $\eta = 10^{-6}$ since it is a fine-tuning phase, when from the beginning, the reference vectors are placed and they are only slightly moved, according to the input patterns.

The only requirement to impose for the forgetting term function $g(y_j)$ is that, if we
develop it in a Taylor series, the constant term is 0 for a zero response; that is, in a symbolic form:

\[ g(y_j) = 0, \text{ for } y_j = 0 \]  

(2.2.10)

Let’s say that we choose a linear function and set the response of the network to be equal to the neighborhood function, that is:

\[ y_j = h_{(j,i)}(x); \]  

(2.2.11)

Then, in a straightforward way, replacing equations 2.2.10 and 2.2.11 in 2.2.9 we get the following equation, which is also analyzed in reference works previously mentioned ([26, 33, 65]).

\[ \Delta w_j = \eta h_{j,i}(x)(x - w_j) \]  

(2.2.12)

Introducing a discrete-time formalism, that is to analyze the time evolution of the synaptic weight vector at discrete time values \( (w_j(n) ) \), then the value of this vector at time \( n+1 \) is given by the following recurrent formula:

\[ w_j(n+1) = w_j(n) + \eta(n)h_{j,i}(x)(n)(x - w_j(n)) \]  

(2.2.13)

This is an important formula, which is applied to all the neurons in the lattice that are inside the topological neighborhood of the winning neuron \( i \). It has the effect of moving the synaptic weight vector \( w_j \) toward the input vector \( x \). Upon repetitive presentations of the training data, the synaptic weight vector tends to follow the distribution of the input vectors due to the neighborhood updating. The learning rate \( \eta(n) \) has a time-varying evolution. For the SOM phase, it starts with a relatively large value (we selected \( \eta(0) = 0.5 \), as indicated in [26] ) and this value decrease in time, with the increasing \( n \). Other authors (see also [33]) indicate even larger values for the initial value of the learning rate (1 or even greater). The requirement of decreasing in time can be fulfilled by choosing an exponential decay for \( \eta(n) \).
as indicated in the formula 2.2.14, or a linear decay rule (we have chosen a linear decay).

\[ \eta(n) = \eta(0) \exp\left(-\frac{n}{\tau_2}\right) \]  

where \( \tau_2 \) is an other time constant of the SOM algorithm.

**The two phases of the adaptive process: ordering and convergence**

The network starts from a status of complete disorder, which is indicated in the Figure 2.4, and the SOM algorithm gradually, in time, leads to an organized representation of activation patterns drawn from the input space. The adaptation of the synaptic weights, according to the formula 2.2.13 can be decomposed into two phases: an ordering or self-organizing phase followed by the convergence phase. We can describe these two phases as it follows:

1) **Self-organizing or ordering phase** it is the coarse organization of the network, in which the topological ordering of the weight vector takes place. Usually has about 1000 iterations of the SOM algorithm (iterations that are also called epochs) or even more. In this phase, here below we present the way the parameters are varying:

* the learning-rate parameter \( \eta(n) \) starts with a value close to 0.1 (after some authors, see also [33] even greater) then it decreases gradually, slowly, but remain above 0.01. So, we can formalize according with the notations introduced as it follows:
  - \( \eta(0) = 0.1; \tau_2 = 1000; \)

* The neighborhood function includes initially all neurons in the network (as we said the initial value is greater then half the diameter of the SOM around the winning neuron);

* This neighborhood function, \( h_{j,i}(n) \) shrink slowly with time. It is permitted for the neighborhood to decrease to a small value of only a couple of neurons, or even the shrink slowly with time. Since as I told before for the concrete classification problem under study
2.2 Basic SOM

we used bi-dimensional SOM, of 10 rows by 10 columns, the "radius" could be equal to 5, or even more (we select it equal to 6).
* In a correspondent way, the value of the $\tau_1$ from the formula 2.2.7 can be chosen, in a heuristic way as:

$$\tau_1 = \frac{1000}{\log(\sigma_0)}$$

(2.2.15)

2) The convergence phase

It is the second phase from the adaptive process of a SOM, which provides the fine-tuning, and a precise statistical quantification of the input phase. One of the drawbacks of this phase is that it requires a large number of iterations (epochs) in order to fulfil the fine-tuning. A rule of thumb says (see also [26]) that the number of epochs should be 500 times the number of neurons in the network. Since we use a 100-neurons network a recommended number of epochs should be in this way 5000. But I would like to add, as a comment, that the convergence was efficient even for a less number of iterations (for instance 2000).
* An other ”rule of thumb” says that the learning rate (n) should be kept at a low value (usually 0.01). It is not allowed to decrease to zero; that’s why it must be set a condition to be above a certain threshold.
* The neighborhood function $h_{j,i}(x)$ should contain only the nearest neighbor of the winning neuron, which may be even reduce to zero (only the winning neuron). In the figure 2.4, one can easily see how the adaptive phase works, precisely, how the topology of the SOM is changed according to the distribution of the input space.

Finally, from the initial condition of the two-dimensional lattice (seen in b), we get, function of the input data distribution (a), the lattice after the ordering phase (c) and after the convergence phase (d).

During the learning phase, the nodes that are topographically close in the array up to a certain geometric distance will activate each other to learn something from the same input.
Figure 2.4: Two-dimensional lattice driven by a two-dimensional distribution
k-Nearest Neighbor. Summary of SOM

The k-Nearest-Neighbor (KNN) Method

The distribution of the reference patterns of different classes usually overlap, whereby the simple Nearest Neighbor method does not define the class borders reliably. The traditional method, that we also implemented, is to consider K nearest neighbor reference vectors to the sample to be classified, and to identify the sample according to the majority of classes encountered among the K nearest neighbors. It can be shown that the class borders thereby defined are very closely the same as the statistically optimal borders derived according to the Bayesian theory of probability.

Conclusions about the SOM implementation After a long series of trial the parameters describing the unsupervised classifier that we implemented for the ischemia classification are as it follows: - the minimum number of epochs for the training phase : 3,000 having 1,000 epochs for the ordering phase (the rest for the convergence);
- the dimension of the SOM: 10 rows by 10 columns;
- initial nearby dimension is 6;
- nearby decrease rate: is 0.99 ;
- the type of the Self-Organizing Map: bi-dimensional lattice;
- the dimension of the input: 5 (number of principal components);
- the weights in function of which will ”be judged” (evaluated) the 5 principal components are as it follows:
  * weight 1 = weight 2 = weight 3 = 1;
  * weight 4 = 0.8;
  * weight 5 =0.5.
- initial learning rate : 0.5;
- the rate of decreasing the learning rate $\delta \eta$ is 0.95;
- minimum learning rate = 0.001;
- the type of distance which produced better results: MANHATTAN type distance;
- Class Assignment method: Majority voting (see k-Nearest Neighbor).

### 2.3 Learning Vector Quantization

**Reference Vectors. Voronoi Tessellation**

Vector quantization (VQ) is a classical signal-approximation method, that usually forms an approximation to the probability density function \( p(x) \) of the stochastic phenomenon \( p(x) \), using a finite number of as called codebook vectors \( m_i \in \mathbb{R}^n, i = 1, 2, \ldots, k \). Once we have chosen the "codebook" the approximation of \( x \) means finding the codebook \( m_c \) closest to \( x \). For this, we use a decision process as it follows:

\[
\| x - m_c \| = \min_i \{ \| x - m_i \| \} \quad (2.3.1)
\]

or, an equivalent form:

\[
c = \arg\min_i \{ \| x - m_i \| \} \quad (2.3.2)
\]

The quantization error, \( E \), which is defined below in 2.3.3, is considered as an optimal criterion for the selection of \( m_i \) in such a way that acquires a minimum value (for \( E \)).

\[
E = \int \| x - m_c \|^2 p(x)dx \quad (2.3.3)
\]

In all the above three equations, the norm \( (\| \|) \) is considered to be of Euclidean type. A concept that is useful for the illustration of the vector quantization problems, both in pattern recognition and neural networks in general it is called Voronoi Tessellation (tessellate is to arrange in a mosaic pattern), see also [33]. Below, in Figure 2.5, I represented how it looks the Voronoi tessellation partition for a "two-dimensional input space" \((\xi_1, \xi_2)\).
Figure 2.5: Partitions for a Voronoi type tessellation
The reference vectors are represented as points in the above two-dimensional space. Obviously, all the vectors \((\xi_1, \xi_2)\), have the same reference vector as their nearest neighbor if they are in the same partition. All the \(x\) vectors that have a particular vector as their closest neighbor define the Voronoi set. With other words, all the vectors in the corresponding partition of the Voronoi tessellation are said to constitute a Voronoi set. In the figure above, all the partitions are bordered by lines - in general the lines are in fact hyperplanes - such as each partition contains one reference vector, that is the ”nearest neighbor” to any other vector in that partition. These lines, also referenced as ”midplanes” of the neighboring reference vectors, constitute together the Voronoi tessellation.

*Learning Vector Quantization algorithm*

The *vector quantization* is a new technique that exploits the underlying structure of input vectors for the purpose of compressing the data. The input space is divided into a number of distinct regions and we define for each one of these a reconstruction vector. In the figure below, 2.6, we can see a Voronoi diagram with four cells. When we present a new vector from the input to the vector quantizer, the following algorithm is followed:

- first it is determined the region in which this vector lies;
- this new vector is represented using the correspondent reproduction vector of that region.

In this way, using the representation vectors to quantify a new input, for storage and for data transmission, important savings can be realized, at the expense of some distortion. *The code book* of the quantizer is constituted from the set of all reproduction vectors (from all the regions in the Voronoi diagram). The members of the code book are called *code words*. As I said, there is some distortion involved in the quantization of the data using code book. The vector quantizer with the minimum distortion it is called a Voronoi or nearest-neighbor quantizer, because the Voronoi cells about a set of points in an input space correspond to a partition of that space according to the nearest-neighbor rule based on the
Figure 2.6: Voronoi diagram having 4 cells
Euclidean metric. In the figure 2.6 there are 4 Voronoi cells with their associated Voronoi (or as called reconstruction) vectors. Each of these 4 Voronoi cells contains all the points that are closest to the reconstruction vector of that cell from the totality of such points. As I previously shown, the Self-Organizing Map performs in an unsupervised manner an approximation for computing the Voronoi vectors. The approximations are specified by the synaptic weights vectors of the neurons from the feature map. According with what is presented in the Figure 2.7, the adaptive pattern classification using a self-organizing feature map and learning vector quantization can be divided into two specific phases:

1) computation of the feature map;

2) final fine tuning of the feature map, which is called learning vector quantization.

The idea of learning vector quantization (LVQ) also originates from Professor Kohonen, see [32, 33], and there are three versions of this algorithm. We implemented LVQ1 which is an well-known one; the other two versions, LVQ2 and LVQ3 mainly differ by the speed of convergence. In general we can say that LVQ is a supervised learning technique that uses class information to move slightly the Voronoi vectors, so as to improve the quality of the classifier decision region. I will not insist too much on this technique because when SOM performs a good definition of classification decision regions, the LVQ does not improve
too much the classification performances. An other example about LVQ performance on
synthetic data is presented in [26], pp. 470, when one can see an improvement of less than
1%.

If we choose an input vector \( x \), randomly from the input space, there are two situations:
1) either the class label of the vector \( x \) and the class of the Voronoi vector agree, the Voronoi
vector \( (w) \) lightly moved in the direction of the input vector or:
2) they disagree and in this case the Voronoi vector is moved away from the input vector \( x \).
3) all the other Voronoi vectors are not modified at the presentation of the \( x \) input vector.

If we denote by \( \{x_i, i = 1, 2, \ldots, n\} \) the input vectors and by \( w_j, j = 1, \ldots, l \)
where \( l \) is the total number of neurons in the lattice, then we can express in a mathe-
matically, formal way what we expressed above (about the agreement and disagreement of
input and Voronoi vectors):

\[
\begin{align*}
1) & \text{ is equivalent with:} \\
    w_c(n + 1) &= w_c(n) + \alpha_n [x_i - w_c(n)] \\
2) & \text{ in case of disagreement is equivalent with:} \\
    w_c(n + 1) &= w_c(n) - \alpha_n [x_i - w_c(n)]
\end{align*}
\]

(2.3.4)

(2.3.5)

where \( 0 < \alpha_n < 1 \)

3) All the other Voronoi vectors are left unchanged.

As a conclusion, when due to some conditions (improper chosen training set for instance)
the LVQ algorithm performed well improving the classification performance of the SOM (in
average up to 5 \%). When we had a proper training set and a well-chosen topology for
the SOM the classification improvement was negligible, as I mentioned before. The conclusion is that the Learning-Vector-Quantization in these kind of situations, it performs an overtraining, having as an immediate effect the fact that the neural network "learns" the input data too well, having as a direct result the decrease in the generalization performance. That is the reason for continuing the search of new, flexible approaches in order to have better results for adaptive pattern classification problems (like the "The Mixture of Experts", presented in the Chapter 4).
3.1 Introduction to Feedforward networks

In a feedforward neural network - FNN, all the connections are directed such that the network forms a directed acyclic graph. It is considered to be the simplest type of NN. On the contrary, Feedback network can be defined as any network that is not a feedforward network. In this chapter we will discuss only about learning algorithms for feedforward NN.

The topology of a 3-layers (input - left, hidden - middle and output on the right) fully-connected FNN is emphasized in figure 3.1

The learning in NN is of two types:

- learning with a teacher, also known as supervised learning, and
- Learning without a teacher, also referred to as unsupervised learning
We are dealing in this chapter with supervised learning algorithms; among these, Gradient descent techniques are the most widely used in NN, [29].

The training of an FNN is usually accomplished by using a backpropagation (BP) algorithm that involves two phases [69, 94]:

- **Forward Phase.** During this phase the free parameters of the network are fixed, and the input signal is propagated through the network of 3.1 layer by layer. The forward phase finishes with the computation of an error signal

  \[ e_i = d_i - y_i \]  
  \[ (3.1.1) \]

  where \( d_i \) is the desired response and \( y_i \) is the actual output produced by the network in response to the input \( x_i \).

- **Backward Phase.** During this second phase, the error signal \( e_i \) is propagated through the network of 3.1 in the backward direction, hence the name of the algorithm. It is during this phase that adjustments are applied to the free parameters of the network so as to minimize the error \( e_i \) in a statistical sense.

### 3.2 Backpropagation Learning

The basic idea of the backpropagation learning algorithm is to repeat the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function \( E \):

\[
\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial \text{net}_i} \frac{\partial \text{net}_i}{\partial w_{ij}}
\]

\[ (3.2.1) \]
where $w_{ij}$ is the weight from neuron $j$ to neuron $i$, $s_i$ is the output and $net_i$ is the weighted sum of the inputs of neuron $i$. A simple gradient descent (as mentioned above) is used to achieve a minimal errorfunction:

$$w_{ij}(t+1) = w_{ij}(t) - \epsilon \frac{\partial E}{\partial w_{ij}}(t)$$

(3.2.2)

The choice of the learning rate $\epsilon$ has an important role by weighting the derivative of the errorfunction in the time needed until convergence. There are two situations:

- if it is set too small there is a need for too many steps in order to reach an acceptable solution;

- by contrary, a large learning rate will possibly lead to oscillations, preventing the error to fall below a certain value (which is most often user-defined).

One of the earliest proposed methods to overcome this situation is to introduce a momentum-term like below:

$$\Delta w_{ij}(t+1) = \epsilon \frac{\partial E}{\partial w_{ij}}(t) + \mu \Delta w_{ij}(t - 1)$$

(3.2.3)

It was believed that the momentum-term will render the learning procedure more stable and will accelerate the convergence in shallow regions of the errorfunction. However, the practical experience has shown that the optimal value of the parameter momentum $\mu$ is equally problem dependent as the learning rate $\epsilon$, and that we cannot accomplish any general improvement. That’s why several adaptive Learning Algorithms have been proposed, as are described here below.
3.3 Resilient Backpropagation-Rprop and variants

3.3.1 The standard Rprop algorithm

Keeping the notations from the above parameters for \( w_{ij} \) and \( E \), we have to mention that a necessary condition is the differentiability of \( E \) with respect to the weights. We consider as bias parameters the weights from an extra input; the superscripts indicate the learning epoch (the iteration).

In the Rprop learning algorithm the direction of each weight update is based on the sign of the partial derivative \( \frac{\partial E}{\partial w_{ij}} \).

A step-size, i.e. the update amount of a weight, is adapted for each weight individually. The main difference to other techniques is that the step-sizes are independent of the absolute value of the partial derivative.

There are two parts in which can be divided the Rprop algorithm: the first part, the adjustment of the step-sizes, is basically the same for all algorithms characterized in this section. For each weight \( w_{ij} \) an individual step-size \( \Delta_{ij} \) is adjusted using the following rule:

\[
\Delta_{ij}^{(t)} := \begin{cases} 
\eta^+ \cdot \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\
\eta^- \cdot \Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\
\Delta_{ij}^{(t-1)}, & \text{else,}
\end{cases}
\]

(3.3.1)

where, see Riedmiller-Braun [64] \( 0 < \eta^- < 1 < \eta^+ \).

If the partial derivative \( \frac{\partial E}{\partial w_{ij}} \) possesses the same sign for consecutive steps, the step-size is increased, whereas if it changes sign, the step-size is decreased.

The step-sizes are bounded by the parameters \( \Delta_{\min} \) and \( \Delta_{\max} \).

In the following, we describe the second part of the algorithm, the update of the weights, for the different Rprop versions.
3.3 Resilient Backpropagation-Rprop and variants

3.3.2 Rprop with and without Weight-Backtracking

a) Rprop with Weight-Backtracking

There are two cases in adjusting the weights $w_{ij}$.

Case A If the sign of the partial derivative has not changed, a regular weight update is executed:

\[
\text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} \geq 0 \text{ then } \Delta w_{ij}^{(t)} := -\text{sign} \left( \frac{\partial E}{\partial w_{ij}}^{(t)} \right) \cdot \Delta_{ij}^{(t)},
\]

(3.3.2)

where the sign operator returns +1 if its argument is positive, otherwise. If the partial derivative changes sign, the previous weight update is reverted as follows:

\[
\text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \text{ then } \Delta w_{ij}^{(t)} := -\Delta w_{ij}^{(t-1)} \text{ and } \frac{\partial E}{\partial w_{ij}}^{(t)} := 0.
\]

(3.3.3)

We set the stored derivative to zero above in the above equation in order to avoid an update of the learning rate in the next iteration, because the else branch in equation 3.3.1 becomes active.

In fact there are 2 alternatives: one as described and the second, by adding a flag to the algorithm. The new weights are given by:

\[
w_{ij}^{(t+1)} := w_{ij}^{(t)} + \Delta w_{ij}^{(t)}
\]

(3.3.4)

this algorithm is referred as $Rprop^+$, the superscript "+" it is to signalize the Weights-Backtracking.

The algorithm below describes in pseudo-code the above mentioned procedure.
for each \( w_{ij} \) do

if \( \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \) then

\[ \Delta_{ij}^{(t)} := \min(\Delta_{ij}^{(t-1)} \cdot \eta^+, \Delta_{\max}) \]

\[ \Delta w_{ij}^{(t)} := -\text{sign}\left( \frac{\partial E}{\partial w_{ij}}^{(t)} \right) \cdot \Delta_{ij}^{(t)} \]

\[ w_{ij}^{(t+1)} := w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \]

elseif \( \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \) then

\[ \Delta_{ij}^{(t)} := \max(\Delta_{ij}^{(t-1)} \cdot \eta^-, \Delta_{\min}) \]

\[ w_{ij}^{(t+1)} := w_{ij}^{(t)} - \Delta w_{ij}^{(t-1)} \]

else if \( \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} = 0 \) then

\[ \Delta w_{ij}^{(t)} := -\text{sign}\left( \frac{\partial E}{\partial w_{ij}}^{(t)} \right) \cdot \Delta_{ij}^{(t)} \]

\[ w_{ij}^{(t+1)} := w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \]

fi

od

Table 3.1: The Rprop+ algorithm with Weight-Backtracking

a) Rprop without Weight-Backtracking

As described in a recent book, [11] we describe here a version of the above \( Rprop^+ \). We omit the weight-backtracking and use always the right-hand of 3.3.2. Therefore, there is no need to store the previous weight updates; this version is known as \( Rprop^- \). Obviously, as shown in the table below, the pseudo-code description is simplified.

for each \( w_{ij} \) do

if \( \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \) then

\[ \Delta_{ij}^{(t)} := \min(\Delta_{ij}^{(t-1)} \cdot \eta^+, \Delta_{\max}) \]

else if \( \frac{\partial E}{\partial w_{ij}}^{(t-1)} \cdot \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \) then

\[ \Delta_{ij}^{(t)} := \max(\Delta_{ij}^{(t-1)} \cdot \eta^-, \Delta_{\min}) \]

fi

\[ w_{ij}^{(t+1)} := w_{ij}^{(t)} - \text{sign}\left( \frac{\partial E}{\partial w_{ij}}^{(t)} \right) \cdot \Delta_{ij}^{(t)} \]

od

Table 3.2: The Rprop- algorithm without Weight-Backtracking
3.4 Adaptive Stepsize Algorithm for On-line Training AOBP. Other On-Line Training Algorithms

These two algorithms, Rprop with and without Weight=Backtracking can also be improved, as is shown in a recent article, see Igel-Häusken [29] and so have resulted $iRprop^+$ and $iRprop^-$.  

3.4 Adaptive Stepsize Algorithm for On-line Training AOBP. Other On-Line Training Algorithms

I come back to the essential principles of supervised learning as iterated at the beginning of this chapter. The theory (and we have followed these) there are a number of heuristic that must be respected in order to get the full-benefits of a FNN architecture.

These are summarized here:

- Use neurons with antisymmetric activation functions (e.g., hyperbolic tangent function) in preference to nonsymmetric activation functions (e.g., logistic function). Two forms of activation functions very often found in the literature and implementations are sigmoid-function and threshold function.

- Shuffle the training examples after the presentation of each epoch; an epoch involves the presentation of the entire set of training examples to the network.

- Follow an easy-to-learn example with a difficult one.

- Preprocess the input data so as to remove the mean and decorrelate the data.

- Arrange for the neurons in the different layers to learn at essentially the same rate. This may be attained by assigning a learning rate parameter to neurons in the last layers that is smaller than those at the front end.

- Incorporate prior information into the network design whenever it is available.

BP learning in principle, may be implemented in one of the two basic ways:
• Sequential mode (also referred to as the on-line mode or stochastic mode or per-pattern mode): In this mode of BP learning, adjustments are made to the free parameters of the network on an example-by-example basis. The sequential mode is best suited for pattern recognition problems.

• Batch mode: In this second mode of BP learning, adjustments are made to the free parameters of the network on an epoch by-epoch basis, where each epoch consists of the entire set of training examples. The batch mode is best suited for nonlinear regression.

In the on-line training the MLP (multi-layer perceptrons) parameters are updated after the presentation of each training example, which can be sampled with or without repetition. Therefor, I can emphasize the advantages [40] of on-line training:

• they are the appropriate choice for learning a task which is either very large (as number of samples in the training set) or because the task it is slow-varying in time; implicitly means that they are the optimum solution for large MLPs.

• seems to be faster training method for small-size training sets and networks.

• The random order of presentation and updating per-pattern means that the search of weight space is stochastic and so, it helps avoiding local minima and provides a more natural approach for learning non-stationary tasks.

• The on-line methods are more robust than the batch methods in case of errors, omissions or redundant data in the training set, which can be corrected or rejected during the training phase.

• Additionally, training data can be generated easily and in great quantities in case of a operational system.
As mentioned before due to a better adaptability, the stochastic methods can be used to learn and track time varying functions and continuously adapt in a time-varying environment.

- It is simpler to implement.

- Finally, there is a less storage need for each weighted connection.

Now, of course there are a few drawbacks that I’m going to enumerate, together with proposed methods to alleviate this disadvantages.

- There is a certain sensitivity to learning parameters.

- The learning in batch-mode is generally faster than in on-line method.

We are dealing now with the second (and the major one) drawback and briefly describe two enhancements already known in the literature [40, 41].

As also the above described Rprop method, the AOBP relies on the adaptation of the step-size $\Delta_{ij}$. Maximizing or minimizing a complex function is a difficult task [2]. There are in general three classes of techniques:

- Local techniques which are based on local information (usual derivatives of higher orders), and can find local extrema.

- Global techniques, which do not rely (generally) on higher order derivatives, they usually involve a stochastic component and they are considerably slower in comparison with the local ones.

- There are also a number of hybrid techniques trying to combine the advantages of the above mentioned.
In this section we are dealing minimization as the kind of problem to be solved. Gradient descent makes changes to the weights which are proportional to the gradient; the constant of proportionality is called the stepsize (or learning rate). The AOBP algorithm in pseudo-code description is as it follows:

<table>
<thead>
<tr>
<th>ON-LINE TRAINING WITH ADAPTIVE STEPSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Initialize the weights (w^0, \Delta^0), and (K)</td>
</tr>
<tr>
<td>1: <strong>Repeat</strong> for each input pattern (p)</td>
</tr>
<tr>
<td>2: Calculate (E(w^p)) and then (\nabla E(w^p))</td>
</tr>
<tr>
<td>3: Update the weights: (w^{p+1} = w^p - \Delta^p E(w^p))</td>
</tr>
<tr>
<td>4: Calculate the steps size to be used with the next pattern (p + 1): (\Delta^{p+1} = \Delta^p + K &lt; \nabla E(w^{p-1}), \nabla E(w^{p-1}) &gt;)</td>
</tr>
<tr>
<td>5: <strong>Until</strong> the <em>termination condition</em> is met</td>
</tr>
<tr>
<td>6: <strong>Return</strong> the final weights (w^{p+1}).</td>
</tr>
</tbody>
</table>

Table 3.3: Algorithm 1: The AOBP proposed algorithm in pseudo-code

Where \(\Delta\) is the stepsize, \(K\) is the meta-steps size and, \(< \ldots >\) stands for the inner product. We have used all these learning paradigms described in the present chapter, i.e. AOBP, Rprop, iRprop, BPVS [41] and Scaled Conjugate Gradient for fast supervised learning [52] of feedforward NN, one per each cluster detected using of k-windows clustering method, see 4.3.
Chapter 4

Data Mining - New clustering methods

4.1 Data Mining

4.1.1 Introductory Concepts

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. However, clustering is a difficult problem combinatorially, and differences in assumptions and contexts in different communities has made the transfer of useful generic concepts and methodologies slow to occur.

Data analysis underlies many computing applications, either in a design phase or as part of their on-line operations. Data analysis procedures can be dichotomized as either exploratory or confirmatory, based on the availability of appropriate models for the data source, but a key element in both types of procedures (whether for hypothesis formation or decision-making) is the grouping, or classification of measurements based on either (i) goodness-of-fit to a postulated model, or (ii) natural groupings (clustering) revealed through analysis. Cluster analysis is the organization of a collection of patterns (usually represented
as a vector of measurements, or a point in a multidimensional space) into clusters based on similarity. It is important to understand the difference between clustering (unsupervised classification) and discriminant analysis (supervised classification). In supervised classification, we are provided with a collection of labeled (preclassified) patterns; the problem is to label a newly encountered, yet unlabeled, pattern. Typically, the given labeled (training) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern. In the case of clustering, the problem is to group a given collection of unlabeled patterns into meaningful clusters. In a sense, labels are associated with clusters also, but these category labels are data driven; that is, they are obtained solely from the data.

Clustering is useful in several exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification. However, in many such problems, there is little prior information (e.g., statistical models) available about the data, and the decision-maker must make as few assumptions about the data as possible.

### 4.1.2 A Taxonomy of Clustering Techniques

Different approaches to clustering data can be described with the help of the hierarchy shown in Figure 4.1 (other taxonometric representations of clustering methodology are possible; ours is based on the discussion in [30]). At the top level, there is a distinction between hierarchical and partitional approaches (hierarchical methods produce a nested series of partitions, while partitional methods produce only one). The taxonomy shown in (Figure 4.1 must be supplemented by a discussion of cross-cutting issues that may (in principle) affect all of the different approaches regardless of their placement in the taxonomy.

1. Agglomerative vs. divisive: This aspect relates to algorithmic structure and operation.

   An agglomerative approach begins with each pattern in a distinct (singleton) cluster, and successively merges clusters together until a stopping criterion is satisfied. A
divisive method begins with all patterns in a single cluster and performs splitting until a stopping criterion is met.

2. Hard vs. fuzzy: A hard clustering algorithm allocates each pattern to a single cluster during its operation and in its output. A fuzzy clustering method assigns degrees of membership in several clusters to each input pattern. A fuzzy clustering can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership.

3. Deterministic vs. stochastic: This issue is most relevant to partitional approaches designed to optimize a squared error function. This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labelings.

4. Incremental vs. non-incremental: This issue arises when the pattern set to be clustered is large, and constraints on execution time or memory space affect the architecture of the algorithm. The early history of clustering methodology does not contain many examples of clustering algorithms designed to work with large data sets, but the advent of data mining has fostered the development of clustering algorithms that minimize the number of scans through the pattern set, reduce the number of patterns examined during execution, or reduce the size of data structures used in the algorithms operations.

4.2 \( k \)-Means Clustering Algorithm

\( k \)-Means is one of the most popular data mining tasks, of partitional type. A partitional clustering algorithm obtains a single partition of the data instead of a clustering structure, such as the dendrogram produced by a hierarchical technique. Partitional methods
Figure 4.1: A taxonomy of clustering approaches
4.2 k-Means Clustering Algorithm

have advantages in applications involving large data sets for which the construction of a dendrogram is computationally prohibitive.

The most intuitive and frequently used criterion function in partitional clustering techniques is the squared error criterion, which tends to work well with isolated and compact clusters. The squared error for a clustering $\mathcal{L}$ of a pattern set $X$ (containing $K$ clusters) is

$$e^2(X, \mathcal{L}) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_{i}^{(j)} - c_{j} \|^2$$

where $x_{i}^{(j)}$ is the $i^{th}$ pattern belonging to the $j^{th}$ cluster and $c_{j}$ is the centroid of the $j^{th}$ cluster. The $k$-means is the simplest and most commonly used algorithm employing a squared error criterion [49].

The $k$-means consists of two main phases. During the first phase, a partition of patterns, in $k$ clusters is calculated, while during the second phase, the quality of the partition is determined. $k$-means is implemented by an iterative process that starts from a random initial partition. The latter is repeatedly recalculated until its quality function reaches an optimum. In particular, the whole process is built upon four basic steps:

1. selection of the initial $k$ means,
2. assignment of each pattern to a cluster with nearest mean,
3. recalculation of $k$ means for clusters, and
4. computation of the quality function.

The last three steps are performed iteratively until convergence. Most clustering algorithms which are variants of $k$-means have been proved convergent [72]. On the other hand, $k$-means-type algorithms often terminate at a local minimum. Formally, let $i_1, \ldots, i_n$ be the input patterns. Each of them is represented by a $d$-tuple

$$\{(a_{11}, a_{12}), \ldots, (a_{d1}, a_{d2})\}$$
where \( an_j, av_j, 1 \leq j \leq d \), denote, respectively, the name and the value of the \( j^{th} \) numerical attribute, whose domain is the set of reals \( \mathbb{R} \). Let the \( k \) first means be initialized to one of \( n \) input patterns \( i_{m1}, \ldots, i_{mk} \). These \( k \) means define the set \( C \) of clusters \( C = \{ C_j | 1 \leq j \leq k \} \). The essence of the algorithm is to minimize the following quality function:

\[
E = \sum_{j=1}^{k} \sum_{i \in C_j} q(i_l, i_{mj}).
\] (4.2.2)

In direct \( k \)-means \( q \) is defined by the squared Euclidean distance, thus \( q(x, y) = \| x - y \|^2 \), where \( \| . \| \) determines the Euclidean norm. Therefore, the direct \( k \)-means clustering algorithm is as follows:

**ALGORITHM DIRECT \( k \)-MEANS**

input \( k \)

initialize \( k \) means \( i_{m1}, \ldots, i_{mk} \)

repeat

for each input pattern \( i_l, 1 \leq l \leq n \)
do

assign \( i_l \) to \( C_j \) with nearest mean \( i_{mj} \), such as \( \| i_l - i_{mj} \|^2 \leq \| i_l - i_{mu} \|^2, 1 \leq j, u \leq k \)

for each cluster \( C_j \in C, 1 \leq j \leq k \)
do

recalculate the mean of patterns \( i_l \in C_j, i_{mj} = \frac{1}{|C_j|} \sum_{i \in C_j} i_l \)

where \( |C_j| \) defines the cardinality of \( C_j \)

compute the quality function \( q \)

until no object has changed clusters (or \( q \) does not change)

The direct \( k \)-means algorithm is computationally very expensive for large sets of patterns. It requires time proportional to the product of the number of patterns, the number of clusters and the number of iterations. More specifically, in the algorithm above, the first loop, for each iteration, has a time complexity \( O(ndk) \), the second \( O(nd) \) and the quality function is calculated in \( O(nd) \). Thus the whole algorithm has a time complexity \( O(ndkt) \), where \( t \) is the number of iterations. In practice, it holds that \( d, k, t \ll n \), where, once again:
• \( n \) is the number of input patterns;

• \( d \) is the dimensionality of input patterns;

• \( k \) is the number of clusters

• \( t \) it's the number of iterations

Note that the first loop has as a basic operation the calculation of the squared Euclidean distance of two numbers and it is this which we consider the basic unit of computational processing cost. The calculation of the quality function has the same basic operation, while the second loop has as a basic operation just the addition of two numbers. There are a number of modifications in the direct \( k \)-means algorithm improving either the computational complexity or the expressive adequacy. The latter is achieved by extending the direct \( k \)-means to work on categorical date (e.g., [27]) or on mixed data (e.g., [63]). Another related extension concerns the quality function, where different (dis)similarity measures have been proposed (e.g., [24, 25, 70]). Improvement of the computational complexity is achieved either by sophisticated initialization methods (e.g., [6, 13, 19]) or by reducing the number of (dis)similarity calculations (e.g., [3, 15, 23]). The \( k \)-windows algorithm is based on the latter approach.

### 4.3 \( k \)-windows Clustering Algorithm

In this section instead of constructing a global model for the pattern classification, we construct several local models, for neighborhoods of the state space. For this task, we use the novel \( k \)-windows clustering algorithm [93], to automatically detect neighborhoods in the state space. This algorithm, with a slight modification (unsupervised \( k \)-windows algorithm) has the ability to endogenously determine the number of clusters present in the
dataset during the clustering process. Once the clustering process is complete, a trained FNN acts as the local predictor for each cluster. In synopsis, the proposed methodology consists of the following four steps:

1. Identify the clusters present in the training set.

2. For each cluster, train a different FNN using for training patterns, patterns from this cluster solely.

3. Assign the patterns of the test set to the clusters according to their distance from the center of the cluster.

4. Use the trained FNNs to obtain the classification scores on the test set.

The unsupervised $k$-windows algorithm generalizes the original algorithm [93]. Intuitively, the $k$-windows algorithm tries to place a $d$-dimensional window (box) containing all patterns that belong to a single cluster; for all clusters present in the dataset. At first, $k$ points are selected (possibly in a random manner). The $k$ initial $d$-ranges (windows), of size $a$, have as centers these points. Subsequently, the patterns that lie within each $d$-range are identified. Next, the mean of the patterns that lie within each $d$-range is calculated. The new position of the $d$-range is such that its center coincides with the previously computed mean value. The last two steps are repeatedly executed as long as the increase in the number of patterns included in the $d$-range that results from this motion satisfies a stopping criterion. The stopping criterion is determined by a variability threshold $\theta_v$ that corresponds to the least change in the center of a $d$-range that is acceptable to recenter the $d$-range (Figure 4.2, left).

Once movement is terminated, the $d$-ranges are enlarged in order to capture as many patterns as possible from the cluster. Enlargement takes place at each dimension separately. The $d$-ranges are enlarged by $\theta_e/l$ percent at each dimension, where $\theta_e$ is user defined, and
4.3 $k$-windows Clustering Algorithm

$l$ stands for the number of previous successful enlargements. After the enlargement in one dimension is performed, the window is moved, as described above.

Once movement terminates, the proportional increase in the number of patterns included in the window is calculated. If this proportion does not exceed the user-defined coverage threshold, $\theta_c$, the enlargement and movement steps are rejected and the position and size of the $d$-range are reverted to their prior to enlargement values. Otherwise, the new size and position are accepted. If enlargement is accepted for dimension $d' \geq 2$, then for all dimensions $d''$, such that $d'' < d'$, the enlargement process is performed again assuming as initial position the current position of the window.

The pseudocode-like description of the proposed $k$-windows clustering algorithm is as follows:

This process terminates if enlargement in any dimension does not result in a proportional increase in the number of patterns included in the window beyond the threshold $\theta_c$ (Figure 4.2, right). In the figure the window is initially enlarged horizontally ($E1$). This enlargement is rejected since it does not produce an increase in the number of patterns included. Next the window is enlarged vertically, this enlargement is accepted, and the result of the subsequent movements and enlargements is the initial window to become $E2$.

The key idea to automatically determine the number of clusters, is to apply the $k$-windows
ALGORITHM $k$-WINDOWS

input $k,a,v$
initialize $k$ means $i_{m1},...,i_{mk}$ along with their
$k$ $d$-ranges $w_{m1},...,w_{mk}$ each of area $a$
repeat
  for each input pattern $i_l, 1 \leq l \leq n$
    do
      assign $i_l$ to $w_j$, so that $i_l$ lies within $w_j$
    for each $d$-range $w_j$
      do
        calculate it’s mean $i_{mj} = \frac{1}{|w_j|} \sum_{i_l \in w_j} i_l$
        and recalculate $d$-ranges
    until no pattern has changed $d$-ranges
enlarge $d$-ranges up to no significant change exists, in their initial mean
compute the ratio $r = \frac{1}{n} \sum_{j=1}^{k} |i_l \in w_j|$
if $r < v$
  do
    reexecute the algorithm

algorithm using a sufficiently large number of initial windows. The windowing technique of
the $k$-windows algorithm allows for a large number of initial windows to be examined, without
any significant overhead in time complexity. Once all the processes of movement and
enlargement for all windows terminate, all overlapping windows are considered for merging.
The merge operation is guided by a merge threshold $\theta_m$. Having identified two overlapping
windows, the number of patterns that lie in their intersection is calculated. Next the propor-
tion of this number to the total patterns included in each window is calculated. If the
mean of these two proportions exceeds $\theta_m$, then the windows are considered to belong to a
single cluster and are merged.
4.4 Myocardial ischemia problem revisited in the framework of $k$-windows Clustering Algorithm

The procedures for data preprocessing, feature extraction (i.e. Principal Component Analysis) have already been presented in Chapter 4. Following the extraction of principal components a noise reduction approach is used to improve these (PCA) coefficients. The utilization if advanced wavelet denoising technique has improved the classification results. The selected noise reduction approach was based on soft thresholding [20].

We have chosen five levels of wavelet decomposition and Daubechies-type wavelets. The classification methodology was described in the paragraph above.

4.4.1 Numerical Results

Numerical experiments were performed using a Clustering, and a Neural Network, C++ Interface built under the Red Hat Linux 7.3 operating system using the GNU compiler collection (gcc) version 3.2. The efficient supervised training of FNNs (i.e. Feedforward Neural Networks) is a subject of considerable ongoing research and numerous algorithms have been proposed to this end. In this work, we consider the following neural network training methods:

- Resilient Back Propagation (RPROP),
- Improved Resilient Back Propagation (iRPROP) [29],
- Scaled Conjugate Gradient (SCG),
- Adaptive On–Line Back Propagation (AOBP) [40],
- Back Propagation with Variable Stepsize (BPVS) [41],

After extensive experimentation the network architecture selected consisted of 8 nodes in the first hidden layer, 7 nodes in the second hidden layer, and two output nodes (5–8–7–2).
All FNNs were trained for 300 epochs on the patterns of the training set and subsequently their performance was evaluated on the test sets. This process was repeated 100 times for all the training algorithms considered. The classification capability of the trained FNNs with respect to the accurate pattern classification in the test sets are reported in Table 4.1.

<table>
<thead>
<tr>
<th>Test set</th>
<th>E103</th>
<th>E104</th>
<th>E106</th>
<th>E107</th>
<th>E108</th>
<th>E111</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RPROP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>75.60</td>
<td>86.63</td>
<td>59.78</td>
<td>79.07</td>
<td>73.95</td>
<td>92.19</td>
</tr>
<tr>
<td>std</td>
<td>0.39</td>
<td>0.96</td>
<td>2.20</td>
<td>2.48</td>
<td>1.13</td>
<td>2.28</td>
</tr>
<tr>
<td>max</td>
<td>76.21</td>
<td>88.87</td>
<td>66.04</td>
<td>85.76</td>
<td>75.85</td>
<td>94.55</td>
</tr>
<tr>
<td>min</td>
<td>74.76</td>
<td>84.34</td>
<td>57.58</td>
<td>73.67</td>
<td>71.57</td>
<td>87.51</td>
</tr>
<tr>
<td><strong>iRPROP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>75.64</td>
<td>87.07</td>
<td>59.87</td>
<td>78.48</td>
<td>73.91</td>
<td>91.84</td>
</tr>
<tr>
<td>std</td>
<td>0.54</td>
<td>0.73</td>
<td>2.88</td>
<td>1.62</td>
<td>1.10</td>
<td>4.00</td>
</tr>
<tr>
<td>max</td>
<td>76.53</td>
<td>88.29</td>
<td>69.17</td>
<td>81.71</td>
<td>75.75</td>
<td>94.92</td>
</tr>
<tr>
<td>min</td>
<td>74.36</td>
<td>84.37</td>
<td>57.48</td>
<td>73.24</td>
<td>71.97</td>
<td>80.41</td>
</tr>
<tr>
<td><strong>SCG</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>77.46</td>
<td>85.77</td>
<td>62.51</td>
<td>79.07</td>
<td>73.57</td>
<td>92.45</td>
</tr>
<tr>
<td>std</td>
<td>0.77</td>
<td>0.80</td>
<td>4.74</td>
<td>2.13</td>
<td>1.61</td>
<td>1.49</td>
</tr>
<tr>
<td>max</td>
<td>79.05</td>
<td>87.09</td>
<td>71.15</td>
<td>84.38</td>
<td>76.16</td>
<td>94.77</td>
</tr>
<tr>
<td>min</td>
<td>75.76</td>
<td>83.64</td>
<td>58.54</td>
<td>72.61</td>
<td>69.62</td>
<td>89.82</td>
</tr>
<tr>
<td><strong>BPVS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>72.50</td>
<td>84.29</td>
<td>67.94</td>
<td>83.28</td>
<td>70.00</td>
<td>93.91</td>
</tr>
<tr>
<td>std</td>
<td>0.02</td>
<td>0.45</td>
<td>0.17</td>
<td>0.07</td>
<td>0.007</td>
<td>0.03</td>
</tr>
<tr>
<td>max</td>
<td>72.57</td>
<td>85.27</td>
<td>68.08</td>
<td>83.34</td>
<td>70.01</td>
<td>93.99</td>
</tr>
<tr>
<td>min</td>
<td>72.49</td>
<td>84.06</td>
<td>67.72</td>
<td>83.17</td>
<td>69.99</td>
<td>93.86</td>
</tr>
<tr>
<td><strong>AOBP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mean</td>
<td>73.69</td>
<td>83.83</td>
<td>71.68</td>
<td>83.10</td>
<td>69.67</td>
<td>93.21</td>
</tr>
<tr>
<td>std</td>
<td>0.59</td>
<td>0.09</td>
<td>0.00</td>
<td>0.01</td>
<td>0.005</td>
<td>0.30</td>
</tr>
<tr>
<td>max</td>
<td>73.88</td>
<td>83.91</td>
<td>71.68</td>
<td>83.14</td>
<td>69.67</td>
<td>93.86</td>
</tr>
<tr>
<td>min</td>
<td>71.93</td>
<td>83.72</td>
<td>71.68</td>
<td>83.10</td>
<td>69.64</td>
<td>93.05</td>
</tr>
</tbody>
</table>

Table 4.1: Percentages of correct classification on the test sets over 100 iterations

In the datasets E103, E104, and E108, FNNs trained with RPROP and iRPROP outperformed all other methods. The drawback of these two methods is the relatively high standard deviation. SCG also suffers from the same drawback. The performance of RPROP and iRPROP on dataset E106 is discouraging. For the remaining datasets FNNs trained with BPVS and AOBP, produced the best results. A significant advantage of AOBP, and to
4.4 Myocardial ischemia problem revisited in the framework of $k$-windows Clustering Algorithm

Overall, for the datasets E104, E107 and E111, the classification ability of the proposed methodology is very good.

4.4.2 Conclusions

This chapter presents a methodology for automatic recognition of ischemic episodes, which draws from the disciplines of clustering and artificial neural networks. The methodology consists of four stages. To effectively partition the state space, the training patterns are subjected to clustering through the unsupervised $k$-windows algorithm. Subsequently, a different FNN is trained on each cluster. At the third stage, the patterns in the test set are assigned to the clusters identified in the training set. Finally, the trained FNNs are used to classify each test pattern. This methodology was applied to classify several test cases of the European ST-T database and the obtained results were promising.
Chapter 5

Radial Basis Function (RBF) Networks in Data Mining

5.1 Introduction

The emergence of neural network (NN) technology [26] offers valuable solutions to solve complicated data mining problems. Patterns arising both from commercial databases and from many engineering databases (as those that describe biosequences) involve data defined over a space that lacks the fundamental properties of distance metric spaces. This work constructs a proper distance metric for expressing the distance between values of features in symbolic domains. This metric owns some geometric properties that make it effective in the context of the regularization formulation of the Generalized Radial Basis Function (GRBF) networks. Regularization techniques impose the learning of a smooth functional from the network [23, 62]. Therefore, it is justifiable to expect from the network to be able of learning the underlying smooth dependence of the outcomes on the attributes, even in the presence of noise that induces the perturbation. The potential of this distance metric to regularize the solution of the GRBF networks is the theoretical justification of the improved performance related to the simple nearest neighbor schemes. The paper proceeds as follows: Section 2 presents the proposed Statistical Distance Metric (SDM). Section 3 discusses how the statistical distance is fitted in the context of GRBF networks. Section 4 introduces the
SDM within the framework of the GRBFs. Section 5 discusses the heuristic instance based parsing of the training set in order to improve the GRBF parameters (i.e. the selection of centers and their spreads). In the last section are presented the conclusions of the present work.

5.2 The Statistical Distance Metric (SDM)

The key problem for applications involving symbolic features is the definition of the distance metric. In domains where features are numeric, it is straightforward to compute the distance between two points in the pattern space in terms of a geometric distance. Indeed, the traditional RBF learning algorithms have been formulated and operate effectively in numeric domains with such distances. However, when the features are symbolic (as is usually in data mining applications using databases from bioinformatics or characteristic to a certain type of disease), the utilization of the traditional types of distances yields inadequate performance. There are two common approaches for handling symbolic information: one is the overlap method and the second the orthogonal representation [3, 76], both of them yielding poor performance in case of symbolic data. In order to be able to obtain an effective formulation of the distances between patterns with symbolic feature values we have adapted the distance measure proposed in [76]. This statistical distance measure takes into account the overall similarity of classification of all instances for each possible value of each feature. The method extracts with a statistical approach from the training set, a matrix that defines the distances between all possible values of a given feature. Therefore, a separate matrix for each feature is obtained. The distance measure for a specific feature is defined according to the following equation:

\[ d(V_A, V_B) = \sum_{i=1}^{N} \left| \frac{C_{A_i}}{C_A} - \frac{C_{B_i}}{C_B} \right|^k \]  

(5.2.1)
5.3 Generalized RBFs with the Statistical Distance Metric (SDM)

In the equation above, $V_A$ and $V_B$ denote two possible values for the feature, e.g. for the DNA promoter data they will be two nucleotides. The distance between the values is the sum over all the N classes. For example, for the DNA promoter example (discussed below) there are two classes, either the sequence is a promoter (i.e. a sequence that initiates a process called transcription) or not. The number of patterns for which the value $V_A$ ($V_B$) is classified to class $i$, is denoted by $C_{Ai}$ ($C_{Bi}$). Also, the total number of patterns of class A (B) is denoted by $C_A$ ($C_B$), and $k$ is a constant usually set to 1. These counts are computed over all patterns of the training set. It becomes easily evident that the more correlated are the classifications of patterns pertaining to two values for a feature the smallest is their statistical distance computed with equation (1). Therefore for feature values belonging to training set patterns with similar classifications a small statistical distance will be computed.

The distance between two patterns is obtained by a weighted sum of distances between the values of the individual features of these patterns:

$$D(X, Y) = \sum_{i=1}^{F} w_{fi}d(V_{X_i}, V_{Y_i})^r$$

(5.2.2)

where $F$ is the number of features, $w_{fi}$ accounts for the weight assigned to feature $f_i$ reflecting its significance and $r$ is a parameter that controls how distances between individual features scale for the computation of the total pattern distance (usually $r=1$ or 2). Also, $V_{X_i}$ and $V_{Y_i}$ denote the values for the $i$th feature of $X$ and $Y$.

5.3 Generalized RBFs with the Statistical Distance Metric (SDM)

The Generalized Radial Basis Functions networks explore the Tikhonov’s regularization theory for obtaining a good generalization performance, as described in [26, 62]. One prerequisite for the application of SDM distance type is to have enough training data for the accurate construction of the SDM space. However, the training sets of size large enough
for providing the essential information for generalization, provide also the necessary information for the computation of an effective distance matrix. In contrast to example based nearest neighbor learning schemes, the GRBF learns a smooth functional that weights the contribution of each example subject to the requirements imposed by the regularizing term for the smoothness of the solution. This fact is the theoretical explanation for the superior performance of GRBF networks related to the Instance Based Learning (IBL) schemes. A parameter of particular importance is the region of influence of the GRBF kernels that is determined by their spread parameter $\sigma$. This problem becomes more complicated within the domain of statistical distances and the heuristic suggestions of [26] to compute $\sigma$ as

$$\sigma = \frac{d_{\text{max}}}{\sqrt{2m}}$$  \hspace{1cm} (5.3.1)

where $d_{\text{max}}$ is the maximum distance and $m$ the number of RBF centers has not been proved effective in practice. In order to obtain an effective setting for the spread parameter, a sensible approach is to obtain at the first step an estimate of the average distance $d_{\text{av}}$, of patterns within the space defined with the SDM. Then the region of influence of the RBF kernels is designed by requiring that at a particular distance $\text{Spread}$ from the RBF center expressed in units of $d_{\text{av}}$, the attenuation of influence is decreased by $a$. Mathematically, this requirement is formulated as: $\exp(-DF \cdot d_{\text{av}} \cdot \text{Spread}) = a$ and therefore the required parameter $DF$ is derived as:

$$DF = \frac{-\log(a)}{\text{Spread} \cdot d_{\text{av}}}$$  \hspace{1cm} (5.3.2)

Values of these parameters that realize good results are for example $\text{Spread} = 5$ and $a = 0.01$ meaning that at a distance from an RBF center 5 times larger than the average distance between patterns, the influence of the RBF function attenuates with a factor of 0.01. The RBF centers own an influence at a distance $x$ from their center expressed by: $\exp(-DF \cdot x)$ However, since the above scheme trains globally the spreads of RBF centers the peculiarities and irregularities of the state space are ignored. An additional instance based learning step
that is described in the following section can estimate the relative importance of each RBF center and therefore can improve the performance of the designed RBF solution.

5.4 Instance-based learning for the determination of the parameters of the RBF networks

It is highly desirable to exploit the reliable examples as centers of the RBF network. Also, the more reliable an example is, larger should its region of influence be when the example is used as an RBF center. The extent of the region of influence is expressed with the spreading parameter $\sigma$ of the RBF center. A heuristically driven learning strategy is adopted for the determination of the examples that should be used as RBF centers and of their widths. The proposed GRBF training approach consists of two steps. At the first step, the Instance Based Learning (IBL) step, successive learning steps evaluate the potential of each example for serving as an RBF center, i.e. how representative the example is. This step is of a heuristic type and it tries to discover the reliability and the importance of the training examples with an instance based learning scheme that resembles the functionality of PEBLS [76]. This solution can be implemented with nearest neighbor schemes and if it is viewed as an input-output mapping it tends to create many class boundaries and discontinuous "islands" of misclassified regions placed near erroneously classified examples. The structure of the decision boundaries is smoothed and most of the regions with artifacts are extracted to reject the influence of noisy examples at the designed classification system. These examples do not yield satisfactory performance at the initial IBL step, so they are not selected as RBF centers. The second learning step constructs the Green's matrix with the estimated spreads of the Gaussian kernels estimated from the heuristically driven first step. During the first step, an empirical approximation to the solution is constructed. There are three basic approaches that can be exploited at the first heuristic learning pass.
1) The one pass approach is an exemplar weighting method that is used in conjunction with the nearest neighbor parameter. The learning is accomplished with only one pass through the training examples. At this training step, for each training instance its $k$ nearest neighbors are found from among the remaining training set. If $j$ neighbors have a matching class then the weight is assigned to the current instance according to the simple formula:

$$\text{weight} = 1 + k - j$$

Therefore, the more the class of the exemplar is reinforced by its neighbors, the less the weight (i.e. the more reliable the exemplar is). Algorithmically, the one pass instance based learning algorithm takes the form:

for each pattern $P$ of the training set do begin
1. detect the $k$ nearest neighbors to $P$ from the training set according to the SDM;
2. Let $j =$ number of nearest neighbors with the same class label as the class of $P$;
3. Set the weight parameter that quantifies the reliability of the exemplar as weight = $1 + k - j$
end;

The other two approaches that we tested for the weighting were 2) the used correct and, respectively, 3) the increment method.

5.5 Applications

We have applied the GRBF based solutions to a variety of data mining problems both from the engineering domain and from the commercial databases domain. Below we describe shortly one application from bioinformatics, one example from data mining of commercial databases and finally some examples using medical databases from the UCI (University of California, Irvine) Machine Learning Repository, from the address: (http://www.ics.uci.edu/ mlearn/MLRepository.html).
The first application concerns the prediction of promoter sequence [3]. This task involves predicting whether or not a given subsequence of a DNA sequence is a promoter, i.e. a sequence of genes that initiates a process called transcription. The data set contains 106 examples, 53 of which were positive examples (promoters) and the rest negative ones. A training pattern consists of a sequence of 57 nucleotides (features) from the alphabet a, c, g and t with the respective classification (promoter or not promoter). Since the available number of patterns were small the classification performance was tested with the leave-one-out methodology, i.e. repeatedly trials have been performed by training on 105 examples and testing on the remaining one. The computed performance was 2/106 (i.e. an average of 2 errors over 106 trials) versus 4/106 for a competitive experiment that used the KBANN neural network model [81].

We can observe from the Table 1 that the utilization of IBL within the framework of GRBFs improves the generalization performances obtained with the classic PEBLS algorithm. However, we cannot easily conclude that a particular IBL learning approach (from those described in the previous section) is better.

Table 5.1: Performances of the proposed GRBF + IBL data mining algorithm

<table>
<thead>
<tr>
<th>Database</th>
<th>PEBLS</th>
<th>GRBF</th>
<th>GRBF+IBL One-pass</th>
<th>GRBF+IBL Used-correct</th>
<th>GRBF+IBL Increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothyroid</td>
<td>97.90</td>
<td>98.04</td>
<td>98.33</td>
<td>98.34</td>
<td>98.29</td>
</tr>
<tr>
<td>Breast cancer</td>
<td>94.23</td>
<td>95.8</td>
<td>96.01</td>
<td>96.12</td>
<td>96.08</td>
</tr>
<tr>
<td>Iris</td>
<td>94.62</td>
<td>95.2</td>
<td><strong>96.22</strong></td>
<td>96.2</td>
<td>95.09</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>76.59</td>
<td>78.23</td>
<td>79.45</td>
<td><strong>84.31</strong></td>
<td>81.29</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>63.45</td>
<td>62.98</td>
<td>65.9</td>
<td>72.5</td>
<td><strong>74.56</strong></td>
</tr>
<tr>
<td>Heart Disease</td>
<td>81.90</td>
<td>82.34</td>
<td>82.28</td>
<td>83.20</td>
<td>85</td>
</tr>
<tr>
<td>Audiology</td>
<td>77.90</td>
<td>78.91</td>
<td><strong>81.06</strong></td>
<td>81.03</td>
<td>79.44</td>
</tr>
</tbody>
</table>
5.6 Conclusions

Neural network algorithms for learning are very effective in domains in which all features have numeric values. At these domains, the examples are treated as points and distance metrics obeys to standard definitions. However the usual domain of data mining applications is the symbolic domain. The utilization of the traditional distance metrics for data mining with neural networks usually results in modest results. The paper has adapted a SDM for application in the context of the GRBF neural networks. This distance metric extends the area of effectiveness of GRBF neural networks to the symbolic world. The results indicate that the generalization potential of neural networks can be utilised for patterns with symbolic features when the learning and evaluation algorithms are designed with the statistically extracted distance metric.

Future work to upgrade further the proposed GRBF and IBL hybrid data mining algorithms can proceed along many different directions, such as: finding optimal multisplits (for numerical attributes), or using simulated annealing algorithm (for the discretization of the continuous attributes). Concrete results of this technique applied in Bioinformatics were published by my colleagues in [48]. The material of this chapter was published in ”Lecture Notes in Computer Science” 2035 Springer 2001, ISBN 3-540-41910-1, and presented at ”The 5th Pacific-Asia Conference- PAKDD 2001” , Hong Kong, China, April 16-18, 2001, [91].
Chapter 6

The Network SOM (NetSOM) and Ischemia Detection

6.1 Hierarchical Mixture of Experts (HME)

The ”Mixture-of -Experts” type of neural networks belong to a special class of neural networks, called ”Committee Machines”. These machines are considered to be universal approximators. Generally, the Mixture of Experts (ME) approach works by dividing the input space into subspaces and using an as called ”gating network” for distributing the information to be classified to the various experts in the architecture. With other words, according to the principle ”divide and conquer” a complex computational task is solved by dividing it into a number of computationally simple tasks and then combines the solutions of those tasks. The ”local experts” (LE) can be any kind of neural networks (MLP-multilayer perceptrons, Radial Basis Function, Support Vector Machines) which are trained individually. The Multi-Layer Perceptrons (MLP) trained with one type of Back-propagation algorithm relies on a form of global optimization in its design. The Radial Basis Function is well known for local optimizations on it’s design. The Support Vector Machines exploits the Vapnik-Chervonenkis (VC) dimension theory for it’s design.

Depending on the problem at hand we can choose neurons from one of the above presented classes, or combinations of neurons from different classes. The combination of local
Figure 6.1: Block diagram of a committee machine based on the "Mixture of Experts"
experts and global expert is said to constitute a committee machine. The so-called “Gating network” performs the task of a mediator among the local experts. The local experts are chosen first so they work best in regions (subspaces) of the input space according to the probabilistic generative model described. For the probabilistic generative model we implemented a small dimension self-organizing map which is known (previously presented in Chapter 2) for the good performance in self-organization (by unsupervised learning performs a “coarse” classification- in fact help us to define the subspaces).

In the Figure 6.2 it is presented a hierarchical mixture of experts (HME), a structure that has two levels of neural networks. It constitutes a natural extension of the notion of ME presented above. The illustration below is for four experts, and it has a "like a tree" architecture, where the gating networks sit at the various nonterminals of the tree and experts sit at the leaves of the tree. What is presented above is a general framework for the HME. Each LE type network has a definition file, in which are specified parameters like for instance in the case of feedforward architecture trained with backpropagation: - the initial learning rate ;
- $\alpha$, the parameter from the exponential belonging to the logistic function;
- the momentum;
- the number of epochs (for the stop condition acts either this maximum number of epochs or a cost function, like the mean-square-error);
- the number of layers;
- the input layer dimension;
- the output layer dimension;
- the number of nodes in the hidden layer;
- the threshold on which (on MSE) at which it stops the current learning process; All these parameters are specified in a separate ”definition file” for the local experts. The preliminary results are encouraging and we aim also at integrating in the architecture the
Figure 6.2: Block diagram of a committee machine based on the "Mixture of Experts"
"feature extraction" layers in order to have an unitary approach - a network that takes data from a ECG recorder (or from a database, like in our case) and perform all the preprocessing phase, feature extraction and classification procedure.

6.2 Description of NetSOM

The Network Self-Organizing Map (NetSOM) is an extension to the Self-Organizing Map (SOM) that is designed in order to cope with complex application domains. Recently, neural network models with strong mathematical basis for obtaining outstanding, near optimal performance have been developed. A notable example is the Support Vector Machine [26], [85] capable of obtaining good generalization performance, on the basis of the training patterns alone (without incorporating a priori knowledge for the problem). These approaches however, require a form of extensive optimization (e.g. solving a quadratic-programming problem [85]) with a computational complexity that does not scale well with the size of the problem. The poor scaling behavior seems to be a common feature of all the neural models that attempt near optimal performance. This fact has its roots to the well known "course of dimensionality" and to the corresponding combinatorial explosion of the problem state space [6], [85]. Therefore, a device for dividing complex application domains onto ordered subspaces becomes of particular importance. The NetSOM is consisted of two components:

* The internal SOM;
* The local experts.

The internal SOM is a SOM usually of a small size that is optimized for the task of separating subspaces. It is trained over the whole training set. Its task is to perform an effective state space partitioning and to map a subspace to each of its neurons. Classification
Figure 6.3: Main steps for NetSOM training
decisions are not performed by the internal SOM except at the cases of unambiguous subspaces where an entropy-based criterion is used to quantify the ambiguity (i.e. uncertainty about class assignment). The local experts that are supervised neural models optimized for a particular subspace created for a NetSOM neuron handle the ambiguous subspaces. Figure 6.3 outlines the main steps involved in NetSOM training. The training set consisted of the Wavelet Denoised PCA coefficients that describe the ST-T Segment is used for the unsupervised training of the internal SOM. Then the ambiguous neurons i.e. those neurons for which the uncertainty of class assignment is significant, are identified with an entropy based criterion that will be described below. Then for these ambiguous neurons local training and testing sets are created for the training of the local experts.

The classification task proceeds with the steps outlined by Figure 6.4. Initially, the pattern is fed to the internal SOM. If the winning neuron is one that is not ambiguous, the internal SOM outputs as the pattern class the class of the winning neuron. In the other case, the local expert connected with the ambiguous neuron is used to perform the classification decision. In order to gain insight to the potential of NetSOM for mapping ordered local subspaces of the problem over the lattice it is beneficial to investigate some relevant
properties of the basic SOM model that constitutes the internal SOM. The operation of the SOM can be considered as a kind of a nonparametric regression [33], with important implicit regularization potential. This regularization potential is important from the point of view of the NetSOM model since it permits the development of a disciplined methodology for hierarchical state-space partitioning using a properly designed SOM at the top level at the hierarchy. Indeed, the SOM fits a number of ordered discrete reference vectors (corresponding to the weights of its neurons) to the distribution of vectorial input samples. Continuous functions are therefore approximated with the reference vectors of the SOM, since the local interactions involved in SOM training impose the regularity of the topologically ordered lattice. This approximation initially describes the gross features of the input space and as the number of SOM neurons is increased, hierarchically more detail of the peculiarities of the input space is represented. This view of the SOM brings some correspondence with the function approximation/regularization capability of traditional neural networks [10, 26, 61].

The template ordering potential of the SOM yields to subspaces of the global space that become ordered over the neuron lattice. This subspace ordering is supported experimentally by observing that the architecture of the local experts for adjacent neurons of the internal SOM tends to be similar. Stated differently, when an effective local expert for an internal SOM’s neuron is constructed, then it is highly likely that the effective architectures for the adjacent experts do not vary significantly (at least when the internal SOM’s size is large enough to perform a sufficient amount of state space splitting). The Radial Basis Function (RBF) networks have produced better results compared with the Multilayer Perceptrons trained with the classical backpropagation algorithm [26], as local experts of the NetSOM. The RBF networks explore the Tikhonov’s regularization theory for obtaining generalization performance. They try to obtain a tradeoff between a term that measures the fitness of the solution to the training set and one that evaluates the smoothness of the solution.

Denoting as $\mathbf{x}_i, d_i, \mathbf{F}(\mathbf{x}_i)$ the input vectors, the desired responses and the corresponding
6.2 Description of NetSOM

realizations of the network respectively this tradeoff can be formulated with a cost function as [26, 61, 62]:

\[ C(F) = C_s(F) + \lambda C_r(F) \]  \hspace{1cm} (6.2.1)

where

\[ C_s(F) = \frac{1}{2} \sum_{i=1}^{N} [d_i - F(x_i)]^2 \]
\[ C_r F = \frac{1}{2} \| DF \|^2 \]

The \( C_s F \) is the standard error term that accounts for the fitting to the training set in the least squares error sense, \( \lambda \) is a positive real number called the regularization parameter; and \( C_r F \) is the regularized term that favors the smoothness of the solution. The later term is the most important from the point of view of generalization performance. The operator \( D \) is a stabilizer because it stabilizes the solution by providing smoothness. In turn, a smooth solution is significantly more robust to erroneous examples of the training set. However, the design of a proper generalization performance for RBF networks still remains a difficult and complex issue that involves heuristic criteria. Specifically, \( m \) fixed centers are selected at random from the training patterns. Their spread \( \sigma \) is common and is computed according to the empirical formula \( \sigma = d_{max}/\sqrt{2m} \), where \( d_{max} \) is the maximum distance between the chosen centers. Finally, the only parameters that need to be learned are the linear weights of the output layer, which are computed with the pseudoinverse method [26]. The number of centers that yielded good generalization performance is \( m=500 \) and the regularization parameter \( \lambda \) is in the range 0.1 to 0.3. The objective of obtaining better performance over regions of the input space that lie at the regions of class boundaries leads to the significant modification of the basic SOM algorithm at the NetSOM model. Speaking precisely, the NetSOM is a new framework for combining neural network models (local experts) in order to improve the performance from that obtained with the utilization of each single neural
network alone.

6.3 ECG and Myocardial Ischemia

Myocardial ischemia is caused by a lack of oxygen and nutrients to the contractile cells. Frequently, it may lead to myocardial infarction with its severe consequence of heart failure and arrhythmia that may even lead to patient death. The sudden cardiac death (SCD) in general has been attributed to the sudden and unexpected ventricular arrhythmias in the group of patients with CAD (coronary artery disease) who has no prior history of arrhythmias [53]. Therefore, identification and treatment of high-risk patients before they experience a major arrhythmic event has a great impact in the problem of myocardial infarction and SCD; myocardial ischemia is a major marker of these dangerous events [66]. Recently, considerable interest has been directed toward the ventricular repolarization (VR) because ST-T complex changes may be a marker of the faulty myocardium electrical activity and, further on, this might lead to ventricular fibrillation (VF) and even SCD [35, 67, 79]. The ST-T Complex of the ECG represents the time period from the end of the ventricular depolarization to the end of corresponding repolarization in the electrical cardiac cycle. Changes in the values of measured amplitudes, times, and duration on the ST-T complex are used to detect and quantify ischemia noninvasively from the standard ECG [22, 35, 39, 74, 75]. The ECG is a widespread noninvasive examination that provides information about the electrical activity of the heart tissue. Different degrees of the severity of ischemic progression can be described in terms of ECG features [9, 21, 35]. For our study only the annotated ischemic episodes were analyzed, without taking into account the short duration ones, e.g. those that may be related to Prinzmetal’s angina [21].

T-wave or repolarization alternans (TWA) is the beat-to-beat fluctuation in T-wave morphology and area, being an intrinsic property of the ischemic myocardium [89]. There is
mounting evidence [34, 89] that the onset of the ST segment preceded the changes in the TWA amplitude and the appearance of arrhythmias. So, because the time course of the onset and offset of TWA during occlusion-release sequence of a cardiac vessel coincide with the spontaneous appearance of malignant arrhythmias it is an important marker of SCD risk [89].

The first stage of ischemia is characterized by T-wave amplitude increase without simultaneous ST segment change. As the ischemia extends transmurally through the myocardium, the intracellular action potential shortens and the injured cells become hyperpolarized. This hyperpolarization in turn produces an "injury current" which is reflected at the ECG as a horizontal ST segment deviation [88]. At the final stage the ischemia is so extensive that the terminal portion of the active depolarization waveform, represented on the ECG by the QRS complex, is altered [9]. This stage is usually associated with myocardial necrosis. The key in treating ischemia is its early detection at the first stages. Myocardial ischemia is indicated at the ECG by the alteration of the ventricular repolarisation waveform. As the ECG is recorded easily and non-invasively, it becomes very important to provide means for reliable ischemia detection from ECG analysis. The capability for accurate and early detection of an acute ischemic event is critical for the establishing the proper treatment.

A common way to discover myocardial ischemia from the long-term ECG, is the detection of an alteration of the ventricular repolarization waveform. However, ischemic changes in the ECG frequently affect the entire repolarization wave shape and therefore are inadequately described by isolated features, such as ST slope, ST-J amplitude, and positive and negative amplitudes of the T-wave, even if these are obtained as an average of several signal samples [73].

Figure 6.5 illustrates the named parameters above. These approaches depend upon the
accurate detection of the J-point on the ECG, which is the inflection point following the S wave. However, in many cases the ST segment is sloped or is influenced by noise, and therefore becomes impossible to reliably recognize this point [50]. This inability constitutes a major drawback to the methods that rely upon the computation and utilization of local features. The approach proposed in the current work avoids the utilization of local, isolated features by exploring the proper neural network techniques with the goal of maximizing the performance of the ST-T segment automatic recognition.

Initially, the Principal Component Analysis (PCA) method [26] is applied and the first five PCA coefficients are extracted. The PCA proved to be [35] the most effective method for separating the signal from noise and no other filtering could produce further improvements. It concentrates the signal information in the minimum number of parameters and it defines the domain where the signal and the noise are well separated. A representative basis set was computed from a database containing a lot of normal and abnormal ST-T segments. The PCA coefficients have been used to train the Network Self-Organizing Map (NetSOM) neural network that is an extension to the Self-Organizing Map of Kohonen [32, 33]. Each
6.3 ECG and Myocardial Ischemia

input pattern consisted of the first five PCA coefficients.

The training phase of the NetSOM consists of an initial SOM like phase that arranges the state space of the problem. The NetSOM exploits the topological ordering properties of the SOM in order to arrange over the lattice simpler subspaces of the original complex state space of the problem that are ordered regularly over the lattice. The accomplished ordering of effective network architectures facilitates significantly the design of hierarchical mixture of experts schemes for confronting complex domains, as the ST-T Segment recognition problem. Indeed, for neurons adjacent at the lattice, the network architectures that demonstrated adequate performance tend to have significant similarity. The scaling properties of the NetSOM resemble those of the SOM since the training complexity of the local experts grows linearly with their number, that in turn scales linearly with the size of the problem (assuming each local expert handles a domain of about the same order of magnitude).

The NetSOM has allowed the building of a modular neural system based on ST-T Segment analysis that outperformed solutions based on a single type of recognition engine. In addition the system can be modularly extended both quantitatively (increasing the training set size) and qualitatively (replacing local experts with better learning paradigms).

Next section describes the stages of preprocessing applied at the ECG signals of the European ST-T database recordings. The purpose of these steps is to create an effective description of the ST-T Segment for input to the neural classification devices. Chapter 2 outlined the Self-Organizing Map (SOM) model to prepare what the precedent section (4.2) introduced- the proposed Network Self-Organizing Map (NetSOM) extension for the hierarchical partitioning of complex state spaces. This model exploits effectively an adapted SOM in order to perform a clever partitioning of the state space that retains some useful ordering properties. Currently, Multilayer Perceptrons trained with the back-propagation algorithm and Radial Basis Function networks have been tested as local experts in the
context of NetSOM. Section 5 presents and discusses the classification effectiveness of the plain SOM and compares it with the obtained performance with the NetSOM with Multilayer Perceptrons and Radial Basis Function networks as local experts. Finally, Section 4.5 (Ischemic episodes detection using NetSOM) presents the conclusions along with some directions onto which further research can proceed for improvements.

6.4 The preprocessing of ECG signals

6.4.1 The European ST-T Database

The ECG signals of the European ST-T Database are a set of long-term Holter recordings provided by eight countries [78]. This database consists of 90 continuous two-channel records, each two hours in duration, taken from ambulatory ECG recordings from 79 patients. The leads that were used included modified leads V1, V2, V3, V4 and V5 and modified limb leads I and III (MLI and ML III). The recordings from modified leads V1-V5 have been used for analysis.

In accordance with the electromagnetic current dipole theory, the reduction of the distance between the recording electrodes and the cardiac signal source will enhance the detection of the subtle alterations in the repolarization patterns [88]. Therefore, in order to benefit from the better signal quality of the precordial (“chest”) leads V1, ..., V5, only signals from them have been analyzed. The ECG signals have been obtained by digitizing the outputs of analog recorders with a sampling frequency of 250 Hz. Each record of the European ST-T Database contains at least one ST or T episode. Cardiology specialists have annotated these episodes in the original database. The localization of the ST and T episodes has been accomplished according to the following set of rules [22]:

(a) ST segment deviations are measured relatively to a reference waveform, which is
usually selected from the first 30 seconds of that specific record.

(b) ST episodes must contain an interval of at least 30 sec in duration, during which the absolute value of the ST deviation is no less than 0.1 mV.

(c) The beginning (ending) of an ST episode is located by searching backward (forward), until a beat is found with ST deviation less than 0.05 mV.

(d) The beginning (ending) of a T-wave episode is located by searching backward (forward) from the time at which the absolute T deviation first (last) exceeds 0.2 mV. When an interval of at least 30 seconds is detected during which the absolute T deviation does not exceed 0.2 mV, the end (the beginning) of that interval defines the beginning (the end) of the episode.

6.4.2 ECG signal preprocessing

The main aim of the ECG signal preprocessing is to prepare a compact description of the ST-T complex, composed from the ST Segment and the T-wave, for input to the classification device (the Self Organizing Map in this case) with the minimum loss of information. From the samples composing each beat, a window of 400 msec is selected (100 samples at the 250 Hz sampling frequency). This signal component will form the input to the Principal Component Analysis in order to describe most of its content within a few (i.e. five) coefficients. In order to have a reference for the extraction of the relevant segment, the position of the R-peak (within the QRS complex) should be detected. In the literature there are a large number of available algorithms that perform this task. An approach based on amplitude and the first derivative of the signal [5] is selected. Baseline wander rejection is implemented using an approach based on cubic splines [6]. The continuous component was eliminated from the extracted ST-T segment. The start of the ST-T Segment was selected at approximately 60 msec after the detected R peak. However, in the database, there are both patients with bradycardia and tachycardia (abnormal heart rates). Therefore, a
more flexible approach that accounts for heart rate variations is required. The selection of the distance between the S point and the previously detected R-peak is correlated with the heart rhythm of the patient. The distance between the R-peak and the J point is in the range of 45 - 80 msec. Due to the fact that the correction of the ST-T length using the Bazett’s formula [5], yields to a similar PCA basis function set [22, 35], an analysis approach is selected with a fixed time window of 400 msec. This assumption is valid for the set of first 5 PC we used for representation. During the ST-T segment extraction, we manually rejected a small number of ST-T segments, considered as particularly noisy.

6.4.3 Principal Component Analysis

Feature extraction refers to a process whereby the input data space is transformed into a feature space that although it has the same dimensionality as the original data space, it can represent the data set more accurately within the constraints imposed by having a reduced number of features at the representation. Therefore, the original data space undergoes dimensionality reduction as the feature space is constructed. The term dimensionality reduction refers to the fact that each \( m \)-dimensional data vector \( \mathbf{x} \) of the original data space can be represented with \( d \) numbers, where \( d < m \), and yet most of the intrinsic information content of the data set is retained.

In mathematical terms, the Principal Component Analysis (PCA) method (also known as Karhunen-Loeve Transformation) transforms a set of correlated random variables of dimensionality \( m \), to a set of uncorrelated (in terms of their second order statistics) variables according to the direction of maximum variance reduction in the training set [73]. The \( d \) uncorrelated variables correspond to the subspace decomposition based on the first \( d \) principal components of the input data. This decomposition is in terms of second order statistics optimum, in the sense that it permits an optimal reconstruction of the original
data in the mean-square error sense (subject to the dimensionality constraint).

The linear PCA can be implemented with powerful, robust techniques as the Singular Value Decomposition (SVD) that guarantee numerical accuracy and stability [26]. To the contrary, the robustness of nonlinear PCA technique is questionable, due to the involved local minima that generally do not allow the detection of the optimal solution [39]. Therefore, when the variables of interest are mostly linearly correlated the linear PCA becomes a highly effective solution.

For this particular application, PCA has performed well for the extraction of representative vectors for the ST-T Segments with only five coefficients. We conclude therefore that at the particular dimensionality reduction problem there is not sufficient evidence that the successive samples of the ECG signal are correlated in complex nonlinear ways that will render PCA an ineffective method of choice [73, 74].

Figure 6.6 illustrates the shape of the five basis functions. Figure 3 in an early article of our group, [8] demonstrates that the ST-T Segment can be reconstructed effectively with the first five PCA projections that represent about 98.1% of the total signal energy. The PCA projection coefficients are then fed to the SOM nonlinear device in order to perform the complex (and highly nonlinear) classification decision about the category pertaining to each analysis case (i.e. normal, abnormal, artifact). The first Principal Component (PC) the second one and the third (but to a less extent) represent the dominant low frequency component of the ST-T segment; the fourth and fifth contain more high frequency energy (they have more noise).
Figure 6.6: Illustration of the principal component analysis, showing shape of all PCA basis functions
Table 6.1: Results obtained with internal SOM of NetSOM as main classification tool: average ischemic beat sensitivity: 73.57 %; average ischemic beat predictivity: 68.31 %

<table>
<thead>
<tr>
<th>Record</th>
<th>Attained Performance</th>
<th>Record</th>
<th>Attained Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sensitivity %</td>
<td>predictivity %</td>
<td></td>
</tr>
<tr>
<td>e0103</td>
<td>66.5</td>
<td>60.3</td>
<td>e0129</td>
</tr>
<tr>
<td>e0104</td>
<td>69.7</td>
<td>63.5</td>
<td>e0139</td>
</tr>
<tr>
<td>e0105</td>
<td>67.7</td>
<td>60.4</td>
<td>e0147</td>
</tr>
<tr>
<td>e0106</td>
<td>65</td>
<td>60</td>
<td>e0148</td>
</tr>
<tr>
<td>e0107</td>
<td>79.1</td>
<td>71.2</td>
<td>e0151</td>
</tr>
<tr>
<td>e0108</td>
<td>64</td>
<td>60.3</td>
<td>e0154</td>
</tr>
<tr>
<td>e0111</td>
<td>81.8</td>
<td>74.7</td>
<td>e0159</td>
</tr>
<tr>
<td>e0112</td>
<td>67.6</td>
<td>60.7</td>
<td>e0162</td>
</tr>
<tr>
<td>e0113</td>
<td>68.9</td>
<td>62.1</td>
<td>e0166</td>
</tr>
<tr>
<td>e0114</td>
<td>71.2</td>
<td>65.8</td>
<td>e0202</td>
</tr>
<tr>
<td>e0118</td>
<td>63.2</td>
<td>59.9</td>
<td>e0203</td>
</tr>
<tr>
<td>e0119</td>
<td>74.5</td>
<td>64.5</td>
<td>e0204</td>
</tr>
<tr>
<td>e0121</td>
<td>72</td>
<td>67.6</td>
<td>e0205</td>
</tr>
<tr>
<td>e0122</td>
<td>75.1</td>
<td>70.1</td>
<td>e0206</td>
</tr>
<tr>
<td>e0127</td>
<td>81.9</td>
<td>74</td>
<td>e0207</td>
</tr>
</tbody>
</table>

6.5 Ischemic episodes detection using NetSOM

In the time series representation of the principal components (PC’s), see 6.4, the ischemic episodes appear as peaks.

6.5.1 Results

After denoising the time series of the Principal Components, 1.4, we applied these new data at the inputs of NetSOM. The task of the neural network is to discriminate these peaks from artifacts and also to learn how to discriminate the "weak ischemic episodes", i.e. the episodes with duration of less than 30 seconds that are not annotated in the European ST-T Database. During the evaluation experiments we have observed that the regularization potential of the SOM allows the tolerance to a small number of possible errors at the
annotations of the ST-T patterns (annotated by the cardiology experts) that constitute the training set. Specifically, given the existence of a significant number of correctly annotated training set patterns (i.e. ST-T segment beats) at the neighborhood of an incorrectly labeled one (due to observation error), many of these patterns will vote for their correct class at about the same neuron where the incorrect pattern has voted. Here, it should be emphasized that the assignment of a class label to each neuron of the internal SOM is performed according to a majority-voting scheme [33]. This scheme acts as a local averaging operator defined over the class labels of all the patterns that activate that neuron as the winner (and therefore are located at the neighborhood of that neuron). The typical majority-voting scheme considers one vote for each winning episode. An alternative more ”analog” weighted majority voting scheme weights the votes each by a factor that decays with the distance of the voting pattern from the winner (i.e. the largest the distance the weakest the vote). The averaging operation of the majority and weighted majority voting schemes effectively attenuates the artifacts of the training set patterns. An alternative, internal SOM calibration method selects as the class of the neuron the class of the nearest pattern to that neuron (i.e. according to the nearest-neighbour rule [26]). This method does not perform well at the task of canceling noise and it leads to less overall classification performance. The performances of both the majority voting and the weighting majority voting schemes are similar for both the task of rejecting artifacts and for the final classification. In the context of NetSOM the utilization of either majority or weighted majority voting for the internal SOM offers an additional advantage. These schemes allow the degree of class discrepancy for a particular neuron to be readily estimated. Indeed, by counting the votes at each SOM neuron for every class, an entropy criterion for the uncertainty of the class label of neuron m can be directly evaluated, as [36]:

\[ H(m) = - \sum_{k=1}^{N_c} p_k \log(p_k) \]
where $N_c$ denotes the number of classes and $p_k = V_k/V_{total}$ is the ratio of votes $V_k$ for class $k$ to the total number of votes $V_{total}$ to neuron $m$. Clearly, the entropy is zero for unambiguous neurons and increases as the uncertainty about the class label of the neuron increases. Therefore, with these voting schemes the regions of the SOM that are placed at ambiguous regions of the state space can be readily identified. For these regions local training sets and testing sets are created and a local neural network optimized for that region is created and trained.

The training set consists of about 50,000 ST-T segment data for the computation of the principal components. This set is constructed by using samples taken from 12 records (different from those used at the testing sets). The extracted training set patterns are selected from the relatively ”flat” regions of the PCA projection coefficient signals. Also, the two main classes (i.e. normal and ischemic) are represented by an approximately equal number of samples.

The evaluation of the SOM and the NetSOM models has been performed on another 30 records out of the 90 records of the European ST-T database. From these records testing sets have been constructed. The whole test set contained principal component projection coefficients from approximately 120,000 ECG beats. Tables 1, 2 and 3 present the classification performance for ischemic beat classification. The classification performance is evaluated by means of the beat sensitivity and beat predictivity. The ischemic beat sensitivity is defined as the ratio of the number of detected ischemic beats, matching the database annotations to the number of annotated ischemic beats. Also the ischemic beat predictivity is defined as the ratio of the number of correctly detected beats to the total number of ischemic beats detected. The internal SOM already performs well given that it has an increased size in order to perform the classification directly. The related performances are described with Table 1. The results obtained by using an internal SOM organized as a 10x10 lattice of neurons. The performed experiments have illustrated that this size yields the best results
for direct classification. Also, with the utilization of the Manhattan distance measure [33] better results are obtained in comparison to the alternative Euclidean measure. Although the internal SOM is trained with the usual SOM unsupervised training algorithm [33], it has the potential to obtain classification accuracy close to those reported in [39, 74, 75] with supervised neural models. The supervised module of the NetSOM is a modular part that theoretically can take the form of any neural network. We have developed the NetSOM with Multilayer Perceptrons trained with Backpropagation and Radial Basis Function networks as local experts. The Multilayer Perceptrons construct a global approximation to the target function by performing a type of stochastic approximation [26]. The Radial Basis Function networks exploit a different type of approximation that involves local fitting to the dynamics of the target function. The locality of RBF approximation fits well with the locality of subspaces that the NetSOM creates. Also, the RBF networks address the issue of regularization in a more disciplining mathematically way through the Tikhonov regularization theory [10, 26, 61, 62]. Therefore, we should expect better results from the RBF networks as local experts compared with their multilayer perceptron counterparts. Indeed, this is verified experimentally by the results displayed by Tables 2 and 3. The internal SOM size at the NetSOM experiments was 4x4. It created 10 subspaces and therefore 10 local experts have been constructed to handle each one of them. Clearly, the remaining 6 neurons were not justified as ambiguous according to the described entropy criterion.

6.5.2 Conclusions

This work has proposed a new supervised extension to the Self-Organizing Map (SOM) model [32, 33, 65]. The extended model is called Network Self-Organizing Map (NetSOM) since the ordering potential of the SOM is exploited for the task of generating effective partitions of the global state space onto a number of subspaces. A specialized neural network for each subspace (local expert) is created and trained to cope with the peculiarities of its
Therefore, in the context of NetSOM, the internal SOM accomplishes mainly the task of state space partitioning and the local experts the task of pattern classification. The NetSOM integrates an initial unsupervised learning phase that organizes the problem domain and the final supervised training of the local experts. The unsupervised learning phase is uniform because it involves always the training of the internal SOM. To the contrary, any well performing supervised neural model (also and non-neural as e.g. fuzzy) can be used as local expert. The performance attained with the unsupervised internal SOM fitted properly to the peculiarities of the problem is already very good and can be compared with other approaches [39, 73, 74]. The NetSOM obtains further performance by exploiting the ordering properties of the SOM for the clever partitioning of the problem state space at regions that become domains of local experts (rather than by directly obtaining a classification with the SOM approach). The NetSOM is a modular architecture that can be improved along many directions. Future research proceeds on developing and testing more effective local expert networks as the Support Vector Machines [26, 85]. These results can be seen in an other article of our group, [58].

Also, the utilization of different frameworks for self-organization as the Adaptive Subspace Self-Organizing Map (ASOM) [33] and information theoretic frameworks for self-organization [26, 36] can improve the phase of state space partitioning. All these research efforts on the NetSOM are with the general philosophy that the best network architecture depends on the structure of the problem that is confronted. Therefore, for complex problems with irregular state spaces a device capable of integrating effectively multiple architectures as the presented NetSOM can perform better than individual architectures.
Table 6.2: Results obtained using NetSOM with multilayer perceptrons as local experts: average ischemic beat sensitivity = 75.9%; average ischemic beat predictivity = 72.5%

<table>
<thead>
<tr>
<th>Record</th>
<th>Attained Performance</th>
<th>Record</th>
<th>Attained Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sensitivity %</td>
<td>predictivity %</td>
<td></td>
</tr>
<tr>
<td>e0103</td>
<td>77.7</td>
<td>72.5</td>
<td>e0129</td>
</tr>
<tr>
<td>e0104</td>
<td>79.2</td>
<td>63.5</td>
<td>e0139</td>
</tr>
<tr>
<td>e0105</td>
<td>75.4</td>
<td>60.4</td>
<td>e0147</td>
</tr>
<tr>
<td>e0106</td>
<td>65</td>
<td>61.2</td>
<td>e0148</td>
</tr>
<tr>
<td>e0107</td>
<td>82.3</td>
<td>71.2</td>
<td>e0151</td>
</tr>
<tr>
<td>e0108</td>
<td>69.5</td>
<td>67</td>
<td>e0154</td>
</tr>
<tr>
<td>e0111</td>
<td>90.6</td>
<td>83.5</td>
<td>e0159</td>
</tr>
<tr>
<td>e0112</td>
<td>71.4</td>
<td>69.2</td>
<td>e0162</td>
</tr>
<tr>
<td>e0113</td>
<td>80</td>
<td>78.1</td>
<td>e0166</td>
</tr>
<tr>
<td>e0114</td>
<td>81.1</td>
<td>78.2</td>
<td>e0202</td>
</tr>
<tr>
<td>e0118</td>
<td>67.5</td>
<td>64.2</td>
<td>e0203</td>
</tr>
<tr>
<td>e0119</td>
<td>64.9</td>
<td>62</td>
<td>e0204</td>
</tr>
<tr>
<td>e0121</td>
<td>72.7</td>
<td>69.8</td>
<td>e0205</td>
</tr>
<tr>
<td>e0122</td>
<td>76</td>
<td>73.5</td>
<td>e0206</td>
</tr>
<tr>
<td>e0127</td>
<td>88</td>
<td>84.6</td>
<td>e0207</td>
</tr>
</tbody>
</table>
6.5 Ischemic episodes detection using NetSOM

Table 6.3: Results obtained using NetSOM with RBF Networks as local experts: average ischemic beat sensitivity = 77.7%; average ischemic beat predictivity=74.1%

<table>
<thead>
<tr>
<th>Record</th>
<th>Attained Performance</th>
<th>Record</th>
<th>Attained Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sensitivity %</td>
<td>predictivity %</td>
<td></td>
</tr>
<tr>
<td>e0103</td>
<td>77.7</td>
<td>72.5</td>
<td>e0129</td>
</tr>
<tr>
<td>e0104</td>
<td>79.2</td>
<td>63.5</td>
<td>e0139</td>
</tr>
<tr>
<td>e0105</td>
<td>75.4</td>
<td>60.4</td>
<td>e0147</td>
</tr>
<tr>
<td>e0106</td>
<td>65</td>
<td>61.2</td>
<td>e0148</td>
</tr>
<tr>
<td>e0107</td>
<td>82.3</td>
<td>71.2</td>
<td>e0151</td>
</tr>
<tr>
<td>e0108</td>
<td>69.5</td>
<td>67</td>
<td>e0154</td>
</tr>
<tr>
<td>e0111</td>
<td>90.6</td>
<td>83.5</td>
<td>e0159</td>
</tr>
<tr>
<td>e0112</td>
<td>71.4</td>
<td>69.2</td>
<td>e0162</td>
</tr>
<tr>
<td>e0113</td>
<td>80</td>
<td>78.1</td>
<td>e0166</td>
</tr>
<tr>
<td>e0114</td>
<td>81.1</td>
<td>78.2</td>
<td>e0202</td>
</tr>
<tr>
<td>e0118</td>
<td>67.5</td>
<td>64.2</td>
<td>e0203</td>
</tr>
<tr>
<td>e0119</td>
<td>64.9</td>
<td>62</td>
<td>e0204</td>
</tr>
<tr>
<td>e0121</td>
<td>72.7</td>
<td>69.8</td>
<td>e0205</td>
</tr>
<tr>
<td>e0122</td>
<td>76</td>
<td>73.5</td>
<td>e0206</td>
</tr>
<tr>
<td>e0127</td>
<td>88</td>
<td>84.6</td>
<td>e0207</td>
</tr>
</tbody>
</table>
The Network SOM (NetSOM) and Ischemia Detection
Chapter 7

Support Vector Machines - introduction and application within sNet-SOM

7.1 Support Vector Machines - SVM

7.1.1 Fundamentals of Statistical Learning Theory

The problem of classification is to estimate a function \( f : \mathbb{R}^N \rightarrow \{\pm 1\} \) using input-output training data \(((x_1, y_1), \ldots, (x_l, y_l) \in \mathbb{R}^N \times \{\pm 1\})\) such that \( f \) will correctly classify unseen examples \((x, y)\), i.e. \( f(x = y) \). These examples \((x, y)\) are generated from the same underlying probability distribution \( P(x, y) \), as the training data. The quantity to be minimized in order to obtain generalization performance is the risk (or prediction risk)

\[
R[f] = \int \frac{1}{2} |f(x) - y| dP(x, y)
\]  

(7.1.1)

Since \( P(x, y) \) is unknown we can only minimize the empirical risk

\[
R_{emp}[f] = \frac{1}{l} \sum_{i=1}^{l} |f(x_i) - y_i|.
\]  

(7.1.2)

Vapnik-Chervonenkis (VC) theory [85] shows that is imperative to restrict the class of functions from which \( f \) is chosen to one that has a capacity suitable for the amount of the
available training data. The developed theory provides bounds on the generalization error for a chosen significance level. They relate the number of examples, the training set error and the complexity of the hypothesis space to the generalization error [13]. The inductive principle of structural risk minimization [4, 86] introduces a systematic method to minimize these bounds by considering both the empirical risk and the capacity of the function class that is accounted by the Vapnik-Chervonenkis (VC) dimension $h$. For binary classification, $h$ is the maximal number of points that can be separated by the functions that can be implemented with the learning machine into two classes in all possible $2^h$ ways. An important VC bound is the one outlined by the following theorem (Theorem 7.1.1), which provides a bound on the rate of uniform convergence of the training error to the classification error, for a set of classification functions with VC dimension $h$, read also [86].

**Theorem 7.1.1.** If $h < l$ is the VC dimension of the class of functions that the learning machine can implement, then for all functions of that class, with probability of at least $1 - \frac{1}{p}$, the bound

$$R[f] \leq R_{emp}[f] + \varphi\left(\frac{h}{l}, -\frac{\log(p)}{l}\right)$$

(7.1.3)

holds, where the confidence term $\varphi$ is defined as

$$\varphi\left(\frac{h}{l}, -\frac{\log(p)}{l}\right) = \sqrt{\frac{h\left(\log\left(\frac{2l}{h} + 1\right) - \log(\frac{2}{p})\right)}{l}}$$

(7.1.4)

This theorem states clearly the dependence of the generalization error $R[f]$ on the VC dimension parameter $h$. The Support Vector Machine is perhaps the only model where this important parameter can be explicitly controlled. This is done by enforcing maximum separation between the patterns of different classes with the construction of the optimal
linear separating hyperplane, usually in a very high-dimensional feature space where the input data are mapped by means of a kernel function.

The key concepts of the Support Vector Machine approach to the implementation of the principle of structural risk minimization are briefly presented. Details can be found in [85]. This material serves also as the theoretical basis for the comprehension of the results of the SVM supervised expert implementation.

The Support Vector (SV) algorithm implements Structural Risk Minimization based on a structure of separating hyperplanes imposed on a dot product space $X$. For a set of pattern vectors $x_1, \ldots, x_l \in X$, these hyperplanes can be written as $\{ x \in X : w \cdot x + b = 0 \}$, where $w$ is an adjustable weight vector and $b$ is a bias. In order to enforce the uniqueness of the separating hyperplane we require

$$
\min_{i=1,\ldots,l} |w \cdot x_i + b| = 1
$$

(7.1.5)

i.e. the data point closest to the separating hyperplane has a distance of $\frac{1}{\|w\|}$ and the smallest distance between a positive and a negative example is $\frac{2}{\|w\|}$. The later distance (i.e. $\Delta = \frac{2}{\|w\|}$) is referred to as the margin $\Delta$ of separation. Hyperplanes constrained with (7.1.5) are termed canonical hyperplanes and the hyperplane that maximizes $\Delta$ with minimization of $w$ is the optimal separating hyperplane. The following important theorem provides a rigorous way of controlling the SVM generalization performance with the computation of the appropriate weight vector $w$ for the maximization of the margin $\Delta$ [86, 85].

**Theorem 7.1.2.** Let the $l$ training set vectors $x_1, x_2, \ldots, x_l \in X$ ($X$ is the dot product space) belong to a sphere $S_R(a)$, of diameter $D$, and center at $a$, i.e. $S_R(a) = \{ x \in X : \|x - a\| < \frac{D}{2} \}$, $a \in X$. Also, let $f_{w,b} = \text{sgn}((w \cdot x) + b)$ be canonical hyperplane decision functions, defined on these points. Then the set of $\Delta$-margin optimal separating hyperplanes
has the VC-dimension \( h \) bounded by the inequality

\[
h \leq \min(\lfloor D^2/\Delta^2 \rfloor, n) + 1
\]  

(7.1.6)

where \( \lfloor x \rfloor \) denotes the integer part of \( x \).

Theorem 7.1.2 states that control over the VC dimension (i.e. complexity) of the optimal hyperplane can be exercised independently of the dimensionality \( n \) of the input space, by properly choosing the margin of separation \( \Delta = \frac{2}{\|w\|} \). Applying the framework of the structural risk minimization for linear machines, a set of separating hyperplanes of varying VC dimension is constructed such that the decrease of the VC dimension occurs at the expense of the smallest possible increase in training error. The SVM imposes a structure on the set of separating hyperplanes by constraining the Euclidean norm of the weight vector \( w \), in order to minimize the VC dimension of the learning machine, according to Theorem 7.1.2.

### 7.1.2 Support Vector Machines

Suppose we are given a set of examples \((x_1, y_1), \ldots, (x_l, y_l)\), \( x_i \in X, y_i \in \{\pm 1\} \) and we assume that the two classes of the classification problem are \textit{linearly separable}. In this case, we can find an optimal weight vector \( w_0 \) such that \( \|w_0\|^2 \) is minimum (in order to maximize the margin \( \Delta = \frac{2}{\|w_0\|} \) of Theorem 7.1.2) and \( y_i \cdot (w_0 \cdot x_i + b) \geq 1 \), \( i = 1, \ldots, l \).

The support vectors are those training examples that satisfy the equality, i.e. \( y_i \cdot (w_0 \cdot x_i + b) = 1 \). They define two hyperplanes. The one hyperplane goes through the support vectors of one class and the other through the support vectors of the other class. The distance between the two hyperplanes is maximized when the norm of the weight vector
\[ \|w_0\| \text{ is minimum. This minimization can proceed by maximizing the following function with respect to the variables } \alpha_i (\text{Lagrange multipliers}) \text{ [85]}: \]

\[ W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot (x_i \cdot x_j) \cdot y_i \cdot y_j \quad (7.1.7) \]

subject to the constraint: \(0 \leq \alpha_i\). If \(\alpha_i > 0\) then \(x_i\) corresponds to a support vector.

The classification of an unknown vector \(x\) is obtained by computing

\[ F(x) = \text{sgn}(w_0 \cdot x + b), \text{ where } w_0 = \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot x_i \quad (7.1.8) \]

and the sum accounts only \(N_s \leq l\) nonzero support vectors (i.e. training set vectors \(x_i\) whose \(\alpha_i\) are nonzero). Clearly, after the training, the classification can be accomplished efficiently by taking the dot product of the optimum weight vector \(w_0\) with the input vector \(x\).

\[ \text{minimize } \frac{1}{2} w \cdot w + C \cdot \sum_{i=1}^{l} \xi_i \quad (7.1.9) \]

The case that the data is not linearly separable is handled by introducing slack variables \((\xi_1, \xi_2, \ldots, \xi_l)\) with \(\xi_i \geq 0\) [15] such that, \(y_i(w \cdot x_i + b) \geq 1 - \xi_i, i = 1, \ldots, l\). The introduction of the variables \(\xi_i\), allows misclassified points, which have their corresponding \(\xi_i > 1\). Thus, \(\sum_{i=1}^{l} \xi_i\) is an upper bound on the number of training errors. The corresponding generalization of the concept of optimal separating hyperplane is obtained by the solution of the optimization problem given by equation 7.1.9 above, subject to:

\[ y_i \cdot (w \cdot \xi_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, i = 1, \ldots, l \quad (7.1.10) \]

The control of the learning capacity is achieved by the minimization of the first term of 7.1.9 while the purpose of the second term is to punish for misclassification errors. The
Support Vector Machines - introduction and application within sNet-SOM

Parameter $C$ is a kind of regularization parameter, that controls the tradeoff between learning capacity and training set errors. Clearly, a large $C$ corresponds to assigning a higher penalty to errors.

Finally, the case of nonlinear Support Vector Machines should be considered. The input data in this case are mapped into a high dimensional feature space through some nonlinear mapping $\Phi$ chosen a priori [85]. The optimal separating hyperplane is then constructed in this space. The corresponding optimization problem is obtained from (7.1.7) by substituting $x$ by its mapping $z = \Phi(x)$ in the feature space, i.e. is the maximization of $W(\alpha)$:

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - 1/2 \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot (\Phi(x_i) \cdot \Phi(x_j)) \cdot y_i \cdot y_j$$ (7.1.11)

Also, the constraint $0 \leq \alpha_i$, becomes $0 \leq \alpha_i \leq C$ (assuming the nonseparable case). When it is possible to derive a proper kernel functional $K$ such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, the mapping $\Phi$ is not explicitly used. Conversely, given a symmetric positive kernel $K(x, y)$, Mercer’s theorem [85] states that there exists a mapping $\Phi$ such that $K(x, y) = \Phi(x) \cdot \Phi(y)$. By designing a kernel $K$ that satisfies Mercer’s condition, the training algorithm is reformulated to the maximization of

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - 1/2 \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \cdot \alpha_j \cdot K(x_i, x_j) \cdot y_i \cdot y_j$$ (7.1.12)

with the constraint $0 \leq \alpha_i \leq C$, and the decision function becomes

$$F(x) = sgn\left(\sum_{i=1}^{l} \alpha_i \cdot y_i \cdot K(x_i, x) + b\right)$$ (7.1.13)

With different expressions for inner products $K(x_i, x)$ we can construct different learning machines with arbitrary types of decision surfaces (nonlinear in input space). The best known kernel types are the polynomial and the radial basis. Polynomial kernels specify polynomials of any fixed order $d$ for the inner product in the corresponding feature space,
i.e.

\[ K(x_i, x) = ((x_i \cdot x) + 1)^d \]  \hspace{1cm} (7.1.14)

Radial Basis Function (RBF) kernels construct decision functions of the form:

\[ F(x) = \text{sgn}\left( \sum_{i=1}^{d} \alpha_i \cdot y_i \cdot \exp\left(-|x - x_i|^2 / \sigma^2 \right) + b \right), \]

with kernel of the type:

\[ K(x_i, x) = \exp\left(-|x - x_i|^2 / \sigma^2 \right). \]

In the RBF case, the SVM training algorithm determines both the centers (support vectors) \( x_i \), the corresponding weights \( \alpha_i \) and the threshold \( b \).

### 7.1.3 Model Order Selection

The SVM is perhaps the only model which permits disciplined model order selection for the optimization of the generalization performance. This makes it highly suited for the implementation of the supervised expert in the context of sNet-SOM.

In order to formulate a means for obtaining the best possible generalization by controlling characteristics of the learning machine we utilize the ideas of [71] at the context of determining the kernel degree, which yields the best generalization from the training data transferred to the supervised expert. We work with polynomial type of kernel, i.e. as described with equation (7.1.14).

With the assumption that the bound of Theorem 7.1.2 gives a reliable indication of the actual VC dimension, the VC-dimension can be estimated as

\[ h \approx c_1 \cdot h_{est} = R^2 \cdot \| w \|^2, \quad R = D/2 \]  \hspace{1cm} (7.1.15)

with some \( c_1 < 1 \).
At the approximation (7.1.15) \( ||w|| \) is determined by the Support Vector Algorithm in feature space, using the kernel. Thus, in order to compute \( h_{est} \), we need to compute \( R \), the radius of the smallest sphere enclosing the training data in feature space. This task is formulated with the following quadratic programming problem [85] p. 428–430, [71]:

\[
\text{minimize } R^2 \text{ subject to } \|z_i - z^*\| \leq R^2
\]  \hspace{1cm} (7.1.16)

where \( z_i = \Phi(x_i) \) and \( z^* \) is the center of the sphere enclosing the images of all the training data in feature space, that is to be determined. This optimization problem can be solved as in [71] and yields the radius of the minimal sphere enclosing the training data in feature space. Finally, the optimal polynomial support vector machine model for the supervising expert is selecting as the one with the minimal estimated VC dimension \( h_{est} \).

For example for \( N = 2045 \) patterns transferred to a supervised expert some results concerning the polynomial SVM model selection are presented at the Table 7.1. We observe that the model with the smallest estimated VC dimension \( h_{est} \) obtains the best generalization performance. We explain also that the dimension \( N_F \) of the polynomial feature space (2nd column of Table 7.1) is obtained as \( N_F = (N + d - 1)!/(d! \cdot (N - 1)!) \) with \( N \) the dimensionality of input vectors (\( N = 35 \) in our case), and \( d \) the degree of the polynomial kernel.

The training of Support Vector Machines involves the solution of a quadratic programming optimization problem which generally has worst-case computational complexity of order \( N_s^3 \), where \( N_s \) is the number of support vectors. This complexity arises from the need to perform an inversion of an \( N_s \) by \( N_s \) Hessian matrix and limits the applicability of SVMs to small scale problems. Currently some decomposition algorithms have been proposed that extent the support vector optimization to large data sets by exploiting decomposition schemes [55]. However, although these approaches alleviate the problem they still require significant computational resources and demand special properties from the training set.
7.2 The supervised Network Self-Organizing Map: sNet-SOM

Table 7.1: Generalization Performance of SVM classifiers with polynomial kernels of various degrees. We can observe that the estimated VC dimension of the polynomial SVM model, is much smaller than the dimensionality of the feature space. The testing set performance attains its maximum when the VC dimension of the learning machine is at its minimum.

<table>
<thead>
<tr>
<th>Degree (d) of polynomial classifier</th>
<th>Dimension of the feature space</th>
<th>Estimated VC dim of the SVM model</th>
<th>Testing Set Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>630</td>
<td>17</td>
<td>72.4%</td>
</tr>
<tr>
<td>3</td>
<td>7770</td>
<td>12</td>
<td>81.3%</td>
</tr>
<tr>
<td>4</td>
<td>73815</td>
<td>27</td>
<td>65.9%</td>
</tr>
<tr>
<td>5</td>
<td>575757</td>
<td>38</td>
<td>64.7%</td>
</tr>
<tr>
<td>6</td>
<td>3838380</td>
<td>65</td>
<td>62.4%</td>
</tr>
<tr>
<td>7</td>
<td>22481940</td>
<td>161</td>
<td>58.6%</td>
</tr>
</tbody>
</table>

in order to obtain computational efficiency (i.e. the approach of [55] relies on many of the support vectors having corresponding Lagrange multipliers \(\alpha_i\) at the upper bound, \(\alpha_i = C\)). Therefore, it is far better to limit the size of the support vector optimization problem, with a device like the proposed sNet-SOM, without sacrificing the accuracy of the final solution.

7.2 The supervised Network Self-Organizing Map: sNet-SOM

The Supervised Network Self-Organizing Map (sNet-SOM) is the proposed extension to the Self-Organizing Map (SOM) of Kohonen [32, 33, 65] designed in order to cope with complex application domains by a flexible combination of the SOM approach with supervised learning schemes. Recently, neural network models with strong mathematical basis for obtaining outstanding, near optimal performance have been developed. A notable example capable of obtaining good generalization performance, on the basis of the training patterns alone (without incorporating a priori knowledge for the problem) is the Support Vector Machine [26, 85]. These approaches however, require a form of extensive optimization (e.g. solving a quadratic-programming problem [47, 55, 85] with a computational complexity
that does not scale well with the size of the problem. The poor scaling behavior seems to be a common feature of all the neural models that attempt near optimal performance. The root of this fact is the well-known "curse of dimensionality" and the corresponding combinatorial explosion of the problem state space [6]. Therefore, a device for dividing a complex application domain to a part that corresponds to regions that are placed near class boundaries (where it is difficult to perform classification decisions) and a part that is unambiguous (i.e. within class regions) becomes of particular importance. Usually the unambiguous part accounts for most of the state space. Since it can be managed with the computationally efficient adaptation of the SOM algorithm in the context of sNet-SOM, the computational requirements for the total problem are reduced substantially without a reduction of the classification accuracy. Moreover, since the performance of many supervised learning algorithms deteriorates for large problems (e.g. due to the local minima trapping problem in gradient descent algorithms [10, 26]) the sNet-SOM at these cases improves the pure supervising learning solution in addition to reducing substantially the computation time. The sNet-SOM consists of two components:

- The Classification Partition SOM (CP-SOM).
- The supervised expert network.

The size of the CP-SOM is dynamically expanded with an adaptive process optimized for the task of the detection of the "difficult" ambiguous regions of the state space. It is trained over the whole training set. The dynamic growth is based on the criterion of neuron ambiguity (i.e. uncertainty about class assignment), which is quantified with an entropy measure that is defined over the CP-SOM nodes. This differs from the local quantization error approach of [1] that grows the map at the nodes that accumulate the largest quantization error. We developed the entropy based growing technique because it accounts class ambiguity much better than the accumulated local error. The local error depends on the
SOM quantization performance and thus can be large even with no class ambiguity, while the entropy directly and objectively quantifies the ambiguity. Classification decisions are performed with the CP-SOM only at the unambiguous part of the state space that corresponds to the neurons of small entropy. The supervised expert handles the ambiguous subspace. Below we discuss the CP-SOM and the supervised expert network in detail.

7.2.1 The Classification Partition SOM (CP-SOM)

The Classification Partition SOM (CP-SOM) is initialized with a few nodes (usually four) and grows nodes to represent the input data. Weight values of the nodes are self-organized according to a new method inspired by the SOM algorithm. This method has significantly lighter computational demands, because the fine-tuning of the weights is avoided and the CP-SOM is usually of small size. The self-organization process maps properties of the original high-dimensional data space onto the lattice consisted of CP-SOM nodes. The property of most interest is the ambiguity of class assignment over the state space and it is mapped to the ambiguity of the winning neuron quantified by an entropy criterion. The CP-SOM learning algorithm is as follows:

1. **Initialization phase.**
   Initialization of the weight vectors of the starting nodes (usually four) with random numbers within the domain of feature values (i.e. of the PCA coefficients, at the application considered).

2. **Adaptation phase**

   (a) Present input \( x_k \) to the network and determine the weight vector that is closest to the input vector mapped to the current feature map (winner), using usually
either Manhattan or Euclidean distance.

(b) Adapt weight vectors $w_j$ only for nodes $j$ in the neighborhood of the winner $i$
and for the winner itself according to the following formula:

$$w_j(k+1) = \begin{cases} w_j(k), & j \notin N_k \\ w_j(k) + \eta(k)\Lambda_k(d(j,i)) & j \in N_k \\ (x_k - w_j(k)), & j \notin N_k \end{cases}$$

(7.2.1)

where the learning rate $\eta(k), k \in N$, is a monotonically decreasing sequence
of positive parameters, $N_k$ is the neighborhood at the $k$th learning step and
$\Lambda_k(d(j,i))$ is the neighborhood function implementing different adaptation rates
even within the same neighborhood.

The neighborhood function $\Lambda_k(d(j,i))$ depends on the distance $d(j,i)$ between node $j$
and the winning node $i$. It decreases monotonically with increasing distance from the
winning neuron (i.e. nodes closer to the winner are adapted more) like in the standard SOM algorithm. The initial neighborhood, $N_0$, includes all the map. Unlike the
standard SOM, these parameters (i.e. $N_k, \Lambda_k(d(j,i)))$ do not need to shrink with time
and can be kept constant i.e. $N_k = N_0, \Lambda_k(d(j,i)) = \Lambda_0(d(j,i))$. This is explained
by the following: Initially, the neighborhood is large enough to include the whole
map. The CP-SOM starts with a much smaller size than a usual SOM; thus a large
neighborhood is not required to train the whole map at the first learning steps (e.g.
with 4 nodes initially at the map, a neighborhood of 1 only is required). A training
epoch is defined as the training of the CP-SOM with a fixed number of neurons at its
lattice. As training proceeds, during subsequent training epochs, the area defined by
the neighborhood becomes localized near the winning neuron, not by shrinking the
vicinity radius (as in the standard SOM) but by enlarging the SOM with the dynamic
growing.
3. Expansion Phase

The objective of controlling the number of training patterns that correspond to the ambiguous regions (and therefore that will be handled with the supervised expert) is the motivation for a modification of the basic SOM algorithm that leads to its dynamic expansion. The number of these training patterns should adhere with the computational limitations of the supervising learning algorithm (upper bound). Moreover, there should be enough patterns for providing the essential information for establishing good generalization over the ambiguous regions (lower bound). The expansion phase follows the adaptation phase. This phase is controlled by two parameters playing the role of upper and lower bounds. The first one, the parameter $\text{SupervisedExpertMaxPatterns}$ specifies a limitation to train effectively the supervised expert when the size of the training set exceeds its value. Also, for obtaining effective generalization, the other variable, the $\text{SupervisedExpertMinPatterns}$ controls the lower bound on the size of the training set. The expansion phase consists of the following steps:

3.1 Calibration of the map with a majority voting scheme. At this step not only the class of each node, but also a parameter $HN_i$ characterizing the entropy of the nodes is computed for every node $i$. This parameter is computed according to Equation 7.2.2 that is discussed below.

3.2 Detection of the neurons whose class assignments are ambiguous, referred to as the ambiguous neurons. A neuron is considered ambiguous if its node entropy $HN_i$ exceeds a threshold value (denoted by the parameter $\text{NodeEntropyThresholdForConsideringAmbiguous}$).

3.3 Evaluation of the map over the whole training set in order to compute the number
of training patterns that correspond to the ambiguous neurons. This number is denoted by the variable $\text{NumTrainingSetAtAmbiguous}$.

3.4 if $\text{NumTrainingSetAtAmbiguous} > \text{SupervisedExpertMaxPatterns}$ then
/* the number of patterns remaining for supervising training should not exceed the limitations of the supervising solution for effective training */

3.4.1.a Perform map expansion by inserting smoothly at the neighborhood of each ambiguous neuron a number of neurons that depends on its fuzziness (i.e. the more uncertain is the class of a neuron the more neurons are inserted over its neighborhood). The neuron insertion process is not described in detail since it follows the guidelines of [1].

3.4.1.b Repeat the adaptation phase after the dynamic extension of the map in order to adjust the new neurons at the appropriate positions at the lattice and re-execute the previous steps of the expansion phase reentering to test the conditions of 3.4

elseif $\text{NumTrainingSetAtAmbiguous} < \text{SupervisedExpertMinPatterns}$ then
/* few training patterns are passed to the supervising expert */

3.4.2 Reduce the parameter $\text{NodeEntropyThresholdForConsideringAmbiguous}$ that controls whether or not a node will be considered as ambiguous. Thus, restarting from Step 3.2, more nodes will be considered as ambiguous and the size of the supervised expert training set will be increased.

else
/* number of patterns transferred to the supervising expert is within the desired limits */

3.4.3 Generate training and testing sets for the supervised expert. Further supervising training will be performed with these sets by the supervising learning algorithm in order to better resolve the ambiguous parts of the
This algorithm exploits well the topological ordering that the basic SOM provides and increases the resolution of the representation over regions of the state space that lie near class boundaries. At this point, it should be emphasized that simply increasing the SOM size with the adaptive extension algorithm until each neuron represents unambiguously a class (i.e., $HN_i$ is low for all the nodes and zero for the case of overfitting) yields to a SOM configuration that although fits to the training set, fails to generalize well.

The training set consisted of the 35 Wavelet Denoised PCA coefficients that describe the ST-T Complex and its neighbours (2 beats on each side). This set is used for the unsupervised training and the dynamic expansion of the CP-SOM. Then the ambiguous neurons i.e. those neurons for which the uncertainty of class assignment is significant, are identified with the entropy criterion. Afterwards, training and testing sets are created for the supervised expert and the dynamic expansion phase of the CP-SOM is executed until the training set size remaining for supervising training is within the desired limits. These sets consist only of the patterns that are represented by the ambiguous neurons.

The classification task proceeds by feeding the pattern to the CP-SOM. If the winning neuron is one that is not ambiguous, the CP-SOM classifies by using the class of the winning neuron. In the other case, the supervised expert is used to perform the classification decision.

It should be noted here that the CP-SOM can be used to confront directly all the classification problem. The only parameter that requires a significant change is the size of the map that needs to be enlarged in order to provide better class resolution over the state space of the problem. The attained performance is directly comparable to those achieved for this task with other network types proposed at the literature [39, 73, 74, 75]. Nevertheless
using solely the CP-SOM, there remain regions of the state space where complex decision boundaries should be enforced in order to separate effectively between the different classes. The sNet-SOM obtains further generalization performance by separating the patterns at these regions with supervising learning schemes.

The assignment of a class label to each neuron of the CP-SOM is performed according to a majority-voting scheme [33]. This scheme acts as a local averaging operator defined over the class labels of all the patterns that activate that neuron as the winner (and accordingly are located at the neighborhood of that neuron). The typical majority-voting scheme considers one vote for each winning occurrence. An alternative more "analog" weighted majority voting scheme weights the votes each by a factor that decays with the distance of the voting pattern from the winner (i.e. the largest the distance the weakest the vote). The averaging operation of the majority and weighted majority voting schemes effectively attenuates the artifacts of the training set patterns. An alternative CP-SOM calibration method selects as the class of the neuron the class of the nearest pattern to that neuron (i.e. according to the nearest neighbor rule [26]). This method does not perform well at the task of canceling noise and it leads to less overall classification performance. The performances of both the majority voting and the weighting majority voting schemes are similar for both the task of rejecting artifacts and for the classification task.

In the context of sNet-SOM the utilization of either majority or weighted majority voting for the CP-SOM is essential. These schemes allow the degree of class discrepancy for a particular neuron to be readily estimated. Indeed, by counting the votes at each SOM neuron for every class, an entropy criterion for the uncertainty of the class label of neuron \( m \) can be directly evaluated, as [36]:

\[
HN(m) = - \sum_{k=1}^{N_c} p_k \cdot \log p_k
\]  

(7.2.2)
where $N_c$ denotes the number of classes and $p_k = V_k/V_{total}$, is the ratio of votes $V_k$ for class $k$ to the total number of votes $V_{total}$ to neuron $m$. Clearly, the entropy is zero for unambiguous neurons and increases as the uncertainty about the class label of the neuron increases. The upper bound of $H(m)$ is $\log N_c$, and corresponds to the situation where all classes are equiprobable (i.e. the voting mechanism does not favor a particular class). Therefore, with these voting schemes the regions of the SOM that are placed at ambiguous regions of the state space can be readily identified. For these regions the supervised expert is designed and optimized for obtaining adequate generalization performance.

The computation time for each iteration of the adaptation phase of the SOM algorithm (which is the most time consuming) scales with a factor that is almost linear to the size of the training set. As a general rule, the number of iterations that are performed at the adaptation phase of the CP-SOM is much smaller than at the corresponding phase of the standard SOM algorithm. Specifically, we have determined experimentally that a good initial value of the learning rate is $\eta(k) = 0.2$ and having it to decrease in 10 steps to 0.02 yields good results (thus the factor for decreasing the learning rate is $\exp \left( \frac{\log(0.1)}{10} \right) = 0.7943$. The CP-SOM only separates the state spaces without extensive fine tuning. However, the required number of neurons depends on the number of classes that are to be separated and the difficulty of the separation (i.e. is independent of the training set size). Since the size of CP-SOM is small, and the number of training epochs (i.e. map expansions) is also very small (3 to 5 map expansions are usually sufficient) the unsupervised phase of the sNet-SOM algorithm scales linearly with the size of the pattern set (although with a large scaling factor). Moreover, for large training sets we can train the CP-SOM with a randomly selected subset and still extract the important information for the class ambiguity over the state space. Thus, in this case, the learning complexity is even less than linear and can be considered to be bounded by a large constant that depends upon the size of the randomly selected subset. In contrast, supervised learning approaches, as i.e. the Radial Basis Function networks
have computational demands in terms of memory and processing resources that scale with a factor that is about cubic to the size of the training set. Practically, it was infeasible at our computing environment to train Radial Basis Function networks with a size more than 2000 X 500 where 2000 is the training set size and 500 the number of hidden units (i.e. RBF centers). Therefore, for the specific problem with about 9000 patterns at the training set it is very difficult to train effectively an RBF network to accomplish directly the classification task.

7.2.2 The Supervised Expert Network

The supervised expert network has the task of discriminating over the state space regions where the class decision boundaries are complex. The supervised expert network should be of a local approximation type and it should incorporate formalism in its design for obtaining adequate generalization performance. Appropriate neural network models that fulfill these requirements are the Radial Basis Function (RBF) and the Support Vector Machines (SVM). The SOM prototype vectors create piecewise linear class boundaries [33] that are usually not effective for resolving the class ambiguity over all the regions of the state space. Moreover, even if the learning procedure enlarges the CP-SOM adaptively until each of its neurons represent unambiguously a single class, this solution addresses only the minimization of the training error and ignores the generalization performance. In the absence of a formal setting for designing generalization, the decision boundaries that the CP-SOM constructs for the ambiguous regions are not expected to cope well for discriminating new patterns. To the contrary, a Support Vector Machine implementation of the supervised expert offers the potential to construct near perfect decision boundaries. For example, the results discussed in [26] illustrate close to optimal separation for a classification problem involving overlapping Gaussian distributions. Below we discuss the implementation of the supervised expert with a Radial Basis Function and with Support Vector Machine, emphasizing on the later choice
that provides better generalization performance and disciplined design.

**Radial Basis Function supervising expert**

The Radial Basis Function networks explore the Tikhonov’s regularization theory for obtaining generalization performance. They try to obtain a tradeoff between a term that measures the fitness of the solution to the training set and one that evaluates the smoothness of the solution. Denoting $\mathbf{x}_i, d_i, F(\mathbf{x}_i)$, the input vectors, the desired responses and the corresponding realizations of the network respectively this tradeoff can be formulated with a cost function as [33, 62]:

$$C(F) = C_s(F) + \lambda \cdot C_r(F) \quad (7.2.3)$$

where,

$$C_s(F) = \frac{1}{2} \sum_{i=1}^{I} [d_i - F(\mathbf{x}_i)]^2 \quad (7.2.4)$$

$$C_r(F) = \frac{1}{2} \|DF\|^2 \quad (7.2.5)$$

The $C_s(F)$ is the standard error term that accounts for the fitting to the training set in the least squares error sense, $\lambda$ is a positive real number called the *regularization parameter* and $C_r(F)$ is the regularized term that favors the smoothness of the solution. The later term is the most important from the point of view of generalization performance. The operator $D$ is a *stabilizer* because it stabilizes the solution by providing smoothness. In turn, a smooth solution is significantly more robust to erroneous examples of the training set. Nonetheless, the design of a proper generalization performance for RBF networks still remains a difficult and complex issue that involves heuristic criteria for the selection of centers and of their parameters [83]. The sNet-SOM with an RBF network as supervised expert has been configured to grow adaptively until about 2000 patterns (i.e. $\text{SupervisedExpertMaxPatterns} = 2000$) map onto the ambiguous neurons. This size of the training set is appropriate for
Support Vector Machines - introduction and application within sNet-SOM

an RBF solution by means of a numerically effective approach. Specifically, \( m = 500 \) fixed centers are selected at random from the training patterns. Their spread \( \sigma \) is common and is computed according to the empirical formula \( \sigma = d_{\text{max}} / \sqrt{2 \cdot m} \), where \( d_{\text{max}} \) is the maximum distance between the chosen centers, that for the normalized 35-dimensional input vector is

\[
\sigma = \sqrt{35} / \sqrt{2 \cdot 500} = 0.1871.
\]

The only parameters that need to be learned are the linear weights of the output layer, which are computed with the pseudoinverse method [26]. The number of centers that yielded good generalization performance is \( m = 500 \) with the regularization parameter \( \lambda \) at the range 0.1 to 0.3.

The RBF supervised expert is simple to design and to implement. However, it is not an easy task to estimate the important parameter \( \text{SupervisedExpertMinPatterns} \). The RBF networks do not include sufficient tools to access the generalization performance and (time consuming) empirical cross-validation methods are usually needed [26]. A simple, yet effective heuristic is to transfer to the RBF supervising expert approximately the maximum number of patterns that it can handle effectively. We have used \( \text{SupervisedExpertMinPatterns} = 800 \) both for the RBF and SVM supervised expert.

**Support Vector Machine supervised expert**

Since the main objective of the sNet-SOM is to use the supervised expert in order to generalize effectively at the state space regions where the plain SOM can’t generalize well we have fitted a supervised model with improved generalization abilities, the Support Vector Machine (SVM) network [15, 86, 85].

The SVM obtains high generalization performance without the need to add a priori knowledge even when the dimension of the input space is high. Moreover, it is perhaps the model that allows the more accurate formal assessment of the generalization performance. This fits well within the framework of sNet-SOM. Below we attempt a rigorous assessment of
the generalization performance of the supervised expert. We provided above a methodology
for the selection of an SVM model, see section 7.1.2, well fitted to the number of transferred
patterns (i.e. model order selection). All notation and key concepts have been defined
above.

7.3 Applications of sNet-SOM

The first application of sNet-SOM we have done, was to revisit the problem of myocardial
ischemia episodes detection. The same database was used, i.e. the first European ST-T
Database, which was presented in a previous section, 6.4.

The main aim of the ECG signal preprocessing is to prepare a compact description
of the ST-T complex, composed from the ST Segment and the T-wave, for input to the
classification device (the sNet-SOM in this case) with the minimum loss of information.
Below we concern ourselves first to the extraction of the relevant signal component from
the ST-T database. Later, we briefly present the dimensionality reduction process which is
accomplished with the Principal Component Analysis (PCA) method. The ST-T segment
of each beat is obtained by first determining the S point and the end of the T wave denoted
by $T_f$. Once those fiducial points are found, the ST-T segment is taken as starting in S
and ending in $T_f$. In particular, the Q and S points of each beat are found by applying
the zero-crossing technique to the difference signal. In order to identify the end of the T
wave ($T_f$), the QT (ms) interval is estimated. The relation between QT and RR intervals
is determined according to [31], by three linear regressions expressing the slopes 0.116 for
heart rates $< 60$ beats/min (i.e. $RR > 1000$ ms ), 0.156 for heart rates from 60 to 100
beats/min (i.e. $600 < RR < 1000$ ms) and 0.384 for heart rates $> 100$ beats/min (i.e.
$RR < 600$ ms):
Table 7.2: Some results characterizing the performance of the QRS detector

<table>
<thead>
<tr>
<th>Record number</th>
<th>Total No. of detected Beats</th>
<th>No. of normal annotated Beats</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0103</td>
<td>7221</td>
<td>7290</td>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>E0104</td>
<td>7979</td>
<td>7700</td>
<td>20</td>
<td>279</td>
</tr>
<tr>
<td>E0106</td>
<td>7132</td>
<td>7152</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>E0108</td>
<td>6818</td>
<td>6593</td>
<td>15</td>
<td>225</td>
</tr>
<tr>
<td>E0111</td>
<td>7515</td>
<td>7534</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>E0118</td>
<td>7200</td>
<td>7080</td>
<td>120</td>
<td>79</td>
</tr>
<tr>
<td>E0122</td>
<td>11338</td>
<td>11358</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>E0127</td>
<td>9386</td>
<td>9391</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>E0129</td>
<td>5564</td>
<td>5566</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>E0139</td>
<td>10256</td>
<td>10631</td>
<td>375</td>
<td>98</td>
</tr>
<tr>
<td>E0147</td>
<td>6344</td>
<td>6371</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>E0151</td>
<td>7499</td>
<td>7545</td>
<td>46</td>
<td>3</td>
</tr>
<tr>
<td>E0154</td>
<td>6698</td>
<td>6782</td>
<td>84</td>
<td>10</td>
</tr>
<tr>
<td>E0159</td>
<td>9135</td>
<td>9147</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>E0166</td>
<td>6384</td>
<td>6396</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>E0202</td>
<td>9854</td>
<td>9804</td>
<td>12</td>
<td>50</td>
</tr>
</tbody>
</table>

\[ QT = \begin{cases} 
0.384 \cdot RR + 99, & RR < 600 \text{ ms} \\
0.156 \cdot RR + 236, & 600 \text{ ms} < RR < 1000 \text{ ms} \\
0.116 \cdot RR + 277, & RR > 1000 \text{ ms} 
\end{cases} \]

We accepted only ST-T Complexes placed between two QRS complexes labelled as normal for reducing the effects of ectopic and other abnormal beats. As a result, the beats that correspond to artefacts have been excluded from the PCA calculation. The isoelectric level is approximated as the average signal value during the 20 ms interval, beginning 80 ms prior to the R peak. A criterion is implemented to leave out the current beat from the training set, if its isoelectric level is different by more than 0.2 mV from that of the previous or of the following beat. The remaining 82856 ST-T patterns constituted the training set.
used for learning the PCA basis functions. The PCA transform though is calculated from a set of vectors of the same dimension. As the extracted data have variable length due to the heart rate variability, each ST-T segment was resampled to 100 samples using polyphasic filtering for interpolation and decimation in time [84]. The average of the beats of the first 30 sec from each signal (all normal beats) was taken as a standard reference. It was subtracted from all resampled segments of the respective signal, to emphasize the relative changes along the time.

Again the Principal Component Analysis (PCA) transformation was selected as the tool for reducing the dimensionality of the extracted ST-T samples. The same method of singular value decomposition (svd), 6.4, was used to extract the optimum of PC (principal component) coefficients in order to represent the ST-T segment of the current beat. Yet, in order to extract the morphological information of the ST-T episodes, but in the same time to distinguish them better from artifacts, was designed a simple way to accomplish this objective, by feeding a window of ST-T Complex evolution to the neural network (the sNet-SOM above described), instead of a single beat information.

After trial and error we decided that a window consisted of 5 beats is appropriate. However, the central beat should have more importance. A simple way to accomplish this is to replicate it a number of times (e.g. three times) at the consideration of the input. Thus, by feeding three times the central beat, the input becomes of the form

\[( b_{n-2} \ b_{n-1} \ b_n \ b_{n+1} \ b_{n+2} )\]  \hspace{1cm} (7.3.1)

where \( b_i \) represents the 5 Principal Component projection coefficients of the \( i \)th beat. Hence, instead of using a 5-dimensional input (i.e. five Principal Component coefficients for the current ST-T segment) we use a 35-dimensional vector. Following the extraction of principal components a noise reduction approach is used to improve these coefficients. The selected noise reduction approach relies on the possibility that we have to modify
the properties of the PCA coefficients signal by processing its Wavelet Transform (WT) modulus maxima and to reconstruct the corresponding function [46, 45, 43]. The utilization of Wavelet Denoising at the domain of Principal Component coefficients has resulted in an improvement at the classification performance. The denoised PCA projection coefficients are then fed to the sNet-SOM nonlinear device in order to perform the complex (and highly nonlinear) classification decision about the category pertaining to each analysis case (i.e. normal, abnormal). The method with the modular architecture of sNet-SOM described in this chapter improved with a few percents the results that we previously obtained with NetSOM (described in the previous chapter), and these results are presented in our article [58] and after, my colleagues used it in the problem of Gene Array Expression Analysis, [48].
Chapter 8

Conclusions and Future Work Directions

Currently I am addressing several issues related to the work we presented in this thesis. The main objective of this work was to develop novel methods based on combination of neural networks with other advanced methods like multiresolution analysis, PCA and ICA (Independent Component Analysis) for difficult pattern classification tasks and the application of it to biomedical signal and image analysis and representation.

The NetSOM and sNet-SOM are modular architectures that can be improved along many directions. The utilization of different frameworks for self-organization as the Adaptive Subspace Self-Organizing Map (ASOM) [33] and information theoretic frameworks for self-organization [26, 36] can improve the phase of the state space partitioning. Another main direction for the improvement of the sNet-SOM performance is the incorporation of more advanced distance metrics to its algorithms, as the Bayesian one proposed in [28]. In order to enhance the exploratory potential of the sNet-SOM, the Sammon distance preserving nonlinear mapping can be adapted. The Sammon mapping allows an effective visualization of the intrinsic structure of the sNet-SOM codebook vectors even at the unsupervised case. The incorporation of the presented sNet-SOM dynamic growing algorithms as a front end processing within Bayesian network structure learning algorithms [10] is also an open area
for future work. Furthermore, we currently formulate the SOM algorithm within the framework of the Modified Value Distance Metric (MVDM) [16, 95]. This will permit us to use its partitioning potential for coping with complex data mining applications, including symbolic.

All these research efforts on the sNet-SOM are with the general philosophy that the best network architecture depends on the structure of the problem that is confronted. Therefore, for complex problems with irregular state spaces a device capable of integrating effectively multiple architectures as the presented sNet-SOM can perform better than individual architectures. The achievement of sNet-SOM as described in 7.2 was done with the cost of increasing the complexity of the NN, accordingly increasing of course of the computational cost and the difficulty of understanding the paradigms involved in it’s implementation. Also, in my opinion the problem of overfitting is more difficult to deal with in the case of such complex pattern analysis tools like sNet-SOM. In order to overcome these, I have started to work on nearly optimal models (like Support Vector Machines) using sparse approximations, but based on a generalization of SVM originated from the work of Professor Vladimir Vapnik, [85, 87]. It is an approach strongly connected to inverse problem for the signal / image representation and it’s aimed to be applied for large scale data mining applications.

The list of people that have used the theory and the software described in the articles accompanying the present thesis is, of course, under continue development, but, at the moment of editing this Thesis, i.e. July - September 2004, we could enumerate at least 16. The shareware code in this thesis is available at request by email at LiviuVladutu@ieee.org, vrahatis@math.upatras.gr or dtas@math.upatras.gr (the last 2 for the k-Windows Clustering software afferent to Chapters 3 and 4 in this Thesis).
Bibliography


[31] J. Karjalainen, M. Viitasalo, M. Manttari, and V. Manninen, *Relation between QT intervals and heart rates from 40 to 120 beats/min in rest electrocardiograms of men*
and a simple method to adjust QT interval values, J Am Coll Cardiol 23 (1994), 1547–1553.


