Cooperative Coevolutionary Mixture of Experts

A Neuro Ensemble Approach for Automatic Decomposition Of Classification Problems

Minh Ha Nguyen
B.Eng (First-class honors) in Computer Engineering, University of Canberra, Australia

submitted in partial fulfillment of the requirements
for the PhD degree at the
School of Information Technology and Electrical Engineering
University of New South Wales
Australian Defence Force Academy Campus

A thesis submitted for the degree of Doctor of Philosophy.

February 2006

Abstract

Artificial neural networks have been widely used for machine learning and optimization. A neuro ensemble is a collection of neural networks that works cooperatively on a problem. In the literature, it has been shown that by combining several neural networks, the generalization of the overall system could be enhanced over the separate generalization ability of the individuals. Evolutionary computation can be used to search for a suitable architecture and weights for neural networks. When evolutionary computation is used to evolve a neuro ensemble, it is usually known as evolutionary neuro ensemble

In most real-world problems, we either know little about these problems or the problems are too complex to have a clear vision on how to decompose them by hand. Thus, it is usually desirable to have a method to automatically decompose a complex problem into a set of overlapping or non–overlapping sub–problems and assign one or more specialists (i.e. experts, learning machines) to each of these sub–problems.

An important feature of neuro ensemble is automatic problem decomposition. Some neuro ensemble methods are able to generate networks, where each individual network is specialized on a unique sub–task such as mapping a subspace of the feature space. In real world problems, this is usually an important feature for a number of reasons including: (1) it provides an understanding of the decomposition nature of a problem; (2) if a problem changes, one can replace the network associated with the sub-space where the change occurs without affecting the overall ensemble; (3) if one network fails, the rest of the ensemble can still function in their sub–spaces; (4) if one learn the structure of one problem, it can potentially be transferred to other similar problems.

In this thesis, I focus on classification problems and present a systematic study of a novel evolutionary neuro ensemble approach which I call cooperative coevolutionary mixture of experts (CCME). Cooperative coevolution (CC) is a branch of evolutionary computation where individuals in different populations cooperate to solve a problem and their fitness function is calculated based on their reciprocal interaction. The mixture of expert model (ME) is a neuro ensemble approach which can generate networks that are specialized on different sub–spaces in the feature space. By combining CC and ME, I have a powerful framework whereby it is able to automatically form the experts and train each of them. I show that the CCME method produces competitive results in terms of generalization ability without increasing the computational cost when compared to traditional training approaches.

I also propose two different mechanisms for visualizing the resultant decomposition in high-dimensional feature spaces. The first mechanism is a simple one where data are grouped based on the specialization of each expert and a color–map of the data records is visualized. The second mechanism relies on principal component analysis to project the feature space onto lower dimensions, whereby decision boundaries generated by each expert are visualized through convex approximations.

I also investigate the regularization effect of learning by forgetting on the proposed CCME. I show that learning by forgetting helps CCME to generate neuro ensembles of low structural complexity while maintaining their generalization abilities.

Overall, the thesis presents an evolutionary neuro ensemble method whereby (1) the generated ensemble generalizes well; (2) it is able to automatically decompose the classification problem; and (3) it generates networks with small architectures.

Keywords

Artificial neural networks, automatic problem decomposition, binary classification, co-operative coevolution, data mining, ensemble learning, evolutionary computation, generalization, learning by forgetting, machine learning, mixture of experts, neuro ensemble, regularization, visualization, weight distribution, weight elimination.

Acknowledgements

I am deeply grateful to my supervisor A/Prof Hussein A. Abbass for his detailed and constructive comments, and for his untiring support and encouragement throughout my research. My utmost thanks.

I am also grateful to my co–supervisor Dr. Robert I. McKay for his comments and support. Our detailed discussions have raised many valuable afterthoughts and ideas for this thesis.

I wish to extend my gratitude to the academics and staff of the University College, Australian Defence Force Academy, University of New South Wales for their help along the way. I thank my colleagues in the Artificial Life and Adaptive Robotics Lab, University of New South Wales for their valuable discussions and encouragement.

My acknowledgement also goes to the numerous referees for their constructive comments on various parts of this thesis that have been published.

I sincerely thank my family and friends for their understanding, support and encouragement. I especially thank my sister, Ms Binh Minh Nguyen, for her help in programming.

The financial support of the University of New South Wales is gratefully acknowledged.

Minh Ha Nguyen

Canberra 2006

Declaration

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
APD	Automatic Problem Decomposition
BP	Back Propagation
CC	Cooperative Coevolution
CCME	Cooperative Coevolutionary Mixture of Experts
EANN	Evolutionary Artificial Neural Network
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
FFNN	Feed–forward Neural Network
GA	Genetic Algorithm
imgpl	Image plot

LF	Learning by Forgetting
ME	Mixture of Experts
MOP	Multi–objective Optimization Problem
NCL	Negative Correlation Learning
PC	Principal Components
PCA	Principal Component Analysis
pcapl	PCA-based plot
PDE	Pareto-frontier Differential Evolution
RBF	Radial Basis Function
WE	Weight Elimination

List of Publications

Publications arising from the research work conducted during the thesis duration are listed chronologically (latest first)

1. Nguyen M.H., Abbass H.A. and McKay R. (2006) Analysis of CCME: Coevolutionary Dynamics, Automatic Problem Decomposition and Effects of Regularization [submitted]

2. Nguyen M.H., Abbass H.A. and McKay R. (2006) A Novel Mixture of Experts Model Based on Cooperative Coevolution, Neurocomputing [to be printed]

3. Nguyen M.H., Abbass H.A. and McKay R. (2005) Stopping criteria for ensemble of evolutionary artificial neural networks, Applied Soft Computing, Volume 6, Issue 1, pp. 100-107. [This paper was listed in the top 25 articles within the journal, access on 27/1/2006, http://top25.sciencedirect.com/?journal_id=15684946]

4. Nguyen M.H., Abbass H.A. and McKay R. (2004) Diversity and neuro ensemble, In Evolutionary Computing in Data Mining, A. Ghosh and L. Jain (Eds), Physica-Verlag, Germany, pp. 125-153.

5. Teo J., Nguyen M.H. and Abbass H.A. (2003) Multi-Objectivity as a Tool for Constructing Hierarchical Complexity, The Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003), Chicago, Lecture Notes in Computer Science LNCS 2724, Springer-Verlag, pp. 483-494.

Chapter 1

Introduction

Artificial Neural Networks (ANNs) have been widely studied for their ability to correctly learn the true distribution of the data from a sample. This ability is called generalization. *Evolutionary Computation* (EC), a powerful tool for global optimization, has been quite successful in training ANNs. As described in the prominent review of Yao (Yao 1999), evolutionary methods can be applied at different levels of the ANN, notably the architecture and connection weights. When EC is used for training ANNs, the process is called *Evolutionary ANNs* (EANNs).

Most of the work in the ANN literature concentrates on finding a single network. However, a single network found using the training set alone may not be the best network on the test set (*i.e.* it may not generalize well). The network can either over–fit the data, or be undertrained on it. It has been found that combining several neural networks, can enhance the generalization of the whole system over the separate generalization ability of the individuals (Abbass 2003a; Abbass 2003b; Jimenez and Walsh 1998; Liu and Yao 1997; Liu and Yao 1999; Liu, Yao, and Higuchi 2000; Rosen 1996; Sharkey 1998; Tumer and Ghosh 1996; Ueda and Nakano 1996; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a). Ensembles of neural networks are called neuro ensembles.

One important application of an ensemble of ANNs is in automatic problem decomposition. Imagine a complex problem, where instead of using a single ANN to solve it, an ensemble is used in such a way that each of its individuals is specialized on a different aspect of the problem. For example, each individual might specialize on a different region of the input space. Imagine further, that a mechanism to combine these specialists' opinions is provided. Such an ensemble could potentially solve a complex problem with higher accuracy than a single network alone, because the subproblems might be simple enough for the individual ANNs to learn and specialize well.

This thesis is about neuro ensembles. I am interested in understanding the mechanisms underlying neuro ensembles. These include the role of diversity on the ensemble, the benefits of cooperative coevolution of the components of neuro ensembles, the effects of regularization on neuro ensembles and the role of neuro ensembles in automatic problem decomposition.

Although the thesis focuses on neuro ensembles, the analysis and visualization tools proposed in this thesis could also be used to understand other types of ensembles, whether ensembles of a single class of learners (as here), or ensembles of mixed types of learners.

1.1 Scope and thesis questions

This thesis is limited to the study of ensembles of artificial neural networks (neuro ensembles) on binary classification problems. More specifically, the thesis investigates the automatic problem decomposition property, which is useful for the classification task by enabling enhanced classification, of neuro ensembles. The three key questions which may be raised in the study of automatic problem decomposition are

1. How does the system divide a complex problem into simpler sub–tasks?
2. How does the system create the modules to work on these sub–tasks?
3. How does the system self-organize its modules to solve the sub–tasks efficiently and accurately?

An important related issue, motivated by the second question, is that it is highly desirable that the system maintain diversity between its modules. I note that a system with self–similar components has no obvious advantage over a system consisting of only one component, and therefore is not desirable.

In this thesis, I wish to specifically answer the following research question:

Can artificial evolution produce a self-organized neuro ensemble that automatically decomposes a complex problem?

In order to answer the main research question, it is necessary to study a number of aspects of neuro ensembles. These aspects can be grouped into the following sub questions to be answered in this thesis:

1. ***What is the role of diversity in neuro ensembles?*** An essential question for any research into ensembles is how to diversify the ensemble (Brown 2004; Kuncheva 2003b; Shipp and Kuncheva 2002). Without diversity, an ensemble offers no advantage over a single individual. Therefore understanding the role of diversity, and how to inject diversity into neuro ensembles, gives insights allowing better design of neuro ensemble. The experiments in Chapter 4 of the thesis are devoted to investigating the role of diversity in a number of neuro ensemble methods.

2. ***Can cooperative coevolution of both the gating mechanism and the experts generate good neuro ensembles?*** Many ensemble methods use a gate of some form to combine the results from different members. In the traditional neuro ensemble and evolutionary neuro ensemble literatures, the ensemble members and the gate are generally considered separately. Many systems use a fixed gating mechanism, which does not change depending on the data. However it is desirable that the gate have information about both the data, and the experts it is combining, before it decides on the best way to combine the individuals' outputs. This is the motivation of the mixture of experts (ME) method (Jacobs, Jordan, Nowlan, and Hinton 1991; Jacobs, Jordan, and Barto 1991). However ME uses gradient-based techniques,

which can easily be trapped in local minima. In the search space defined by the combination of complex data and the interaction of complex experts, local minima are highly likely. Given the strengths of evolutionary methods in handling complex fitness landscapes (Forrest 1993; Mitchell 1996), it is worth examining the combination of evolutionary approaches and ME models. In this thesis, I propose a method called the cooperative coevolutionary mixture of experts model (CCME). It combines the advantages of ME and cooperative coevolution, allowing the gate and the experts to coevolve and potentially helping the system to generalize better. Chapter 5 of the thesis is devoted to examining this approach, and determining whether it offers significant advantages.

3. ***Do ME and CCME automatically decompose a complex problem?*** The problem of automatic problem decomposition has been studied in many disciplines (Khare, Yao, Sendhoff, Jin, and Wersing 2005). Neuro ensembles provide effective learning methods, which many believe result from good problem decomposition (Sharkey 1996). It is highly desirable to test this. In Chapter 6 of this thesis, I provide visualization tools which help to understand the structure of the solutions provided by these ME-based methods. The tools allow us to examine the input spaces assigned to the individual experts by the gate; with the help of the visualization, I am able to see that the gate sub-divides the input space into regions, and assigns primary responsibility for different regions to different experts. Chapter 6 provides a detailed analysis of this issue.

4. ***Can regularization enhance neuro ensembles by generating low structural complexity while maintaining the same level of accuracy?*** In the literature of artificial neural networks, it has been shown that "learning by forgetting" can generate ANNs with lower structural complexity (Ishikawa 1996; Ishikawa and Yoshino 1993; Kozma, Kitamura, Malinowski, and Zurada 1995; Miller and Zurada 1997). This is important, since reduced structural complexity reduces the risk of over-

specialization (Bishop 1995). However I have not been able to find any previous studies on the use of learning by forgetting in neuro ensembles, and especially in ME. In this thesis, I investigate the effects of learning by forgetting on both the performance and the architecture of ME methods. I examine both the original ME method, and the CCME extension. This work is reported in Chapter 7.

5. ***Can learning by forgetting help neuro ensembles to automatically decompose problems?*** Learning by forgetting has been shown to promote a form of automatic problem decomposition by modularizing the architectures of the ANNs (Ishikawa 1996; Ishikawa and Yoshino 1993). In chapter 7, I examine whether learning by forgetting can assist with feature selection for ME and if so, whether this feature selection has any benefit for problem decomposition.

1.2 Structure of the thesis

This thesis is divided into three main parts:

- **Literature Review** (chapters 2 and 3): in chapter 2, I introduce the basic concepts of ANNs and evolutionary ANNs using EC. In chapter 3, I review in greater detail the literature on ensembles of ANNs (neuro ensembles). I also review the key concepts used in the later parts of the thesis. These include the Mixture of Experts model, the Cooperative Coevolution framework, regularization methods, and automatic problem decomposition in general.
- **Initial Investigation** (chapter 4): in chapter 4, I compare a number of state-of-the-art neuro ensemble methods which have been proposed in the literature. These include
  - the simple ensemble method, where a population of ANNs is evolved, and the best individuals are combined to form an ensemble.

  - an Island–based neuro ensemble, which uses the Island model from EC to generate diversity in the ensemble
  - the Negative Correlation Learning ensemble method proposed by Liu and Yao (Liu and Yao 1999; Liu, Yao, and Higuchi 2000)
  - an ensemble method based on multi–objective optimization

  This chapter provides an empirical study of different aspects of the above four neuro ensemble methods. It also examines the significance of diversity in ensemble learning.

- **CCME: A novel method for automatic problem decomposition** (chapters 5,6 and 7): in chapter 5, I introduce and investigate a novel method based on the *Mixture of Experts* (ME) model and the *Cooperative Coevolution* (CC) framework. In chapter 6, I introduce a set of novel visualization tools to specifically investigate automatic decomposition of classification problems. These tools are applied to analyze ME and CCME on binary classification problems. In chapter 7, ME and CCME are extended with regularization of the structure of the model. Two different regularization methods are investigated: structural learning by forgetting, proposed by Ishikawa (Ishikawa and Yoshino 1993), and the weight elimination method proposed by Weigend et al (Weigend, Rumelhart, and Huberman 1990).

1.3 Contributions of the thesis

At the core of this thesis, two major contributions can be identified. These are

- a novel machine learning method based on the concept of mixture of experts and the cooperative coevolution framework. I examine whether this method works well through comparisons with other methods, and statistical test(s) of significance.
- visualization tools to inspect whether or not a method is capable of automatically decomposing the problem.

The detailed contributions are summarized below (in order of their appearance):

- A comprehensive literature review of artificial neural networks, evolutionary artificial neural networks, neuro ensembles and mixture of experts (chapter 2 and 3). This provides an intensive background for the thesis and a short summarization of what has been done in the literature of these fields.

- An empirical study, emphasizing the role of diversity, in state-of-the-art neuro ensembles. I examine methods such as Negative Correlation Learning and multi–objective optimization. The studies include the effects of combinational mechanisms of the ensemble, the level of diversity, the usefulness of local search on evolution to find better solutions, the effect of noise injection and early stopping on generalization, and possible relationships between diversity and performance of the ensemble (chapter 4).

  The experimental results and analysis show that:

  - Different combination gates have little effect on the average performance of the four investigated ensemble methods
  - The diversity level maintained by negative correlation learning is poor, as suggested by McKay and Abbass's analysis (McKay and Abbass 2001)
  - Combining individuals improves the performance of the system
  - Local search helps evolution to find better solutions. This applies to both ensemble and individual-based methods
  - Noise injection shows interesting effects on performance enhancement, though the results are equivocal
  - Early stopping is useful in enhancing generalization. A variety of different stopping conditions, detecting minimum values of a variety of validation set properties such as the fitness of the ensemble or the average fitness of the population, can avoid over–fitting of the networks to the training data.

The empirical study also addresses the first research question, regarding the connection between diversity and performance of neuro ensembles. The results are perhaps surprising. They suggest that the assumed connection between diversity and performance of the ensemble is at best doubtful, and requires further verification. Our results are consistent with those reported by Kuncheva (Kuncheva 2003b).

- A novel Mixture of Experts model based on cooperative coevolution (chapter 5): The ME model is a neuro ensemble approach which can generate networks that are specialized on different sub-spaces in the feature space. Cooperative coevolution is a branch of evolutionary computation where individuals in different populations cooperate to solve a problem, and their fitness function is calculated based on their reciprocal interaction. By combining CC and ME, I have built a powerful framework in which it is possible to automatically form and train the experts. The CCME method produces competitive results in terms of generalization ability, without increasing the computational cost when compared to traditional training approaches. The experimental results and analysis in this chapter answer the second research question, in that cooperative coevolution of both the gate and the experts produces ensembles with better generalization ability, when compared with gradient-based non- evolutionary ME.

- An empirical study that provides a better understanding of ME and CCME. I examine a range of aspects such as the effects of error functions, learning rate, stopping criteria, number of hidden units per experts and number of experts, on the generalization performance of the models (chapter 5). The study suggests that

  - CCME and ME are robust to various error functions, learning rates, numbers of experts and numbers of hidden units;
  - early stopping is beneficial for ME and CCME;
  - the empirical study gives better insights into different aspects of neuro-ME

and CCME, which have not been investigated before, such as the effects of different error functions, learning rates, stopping criteria, number of hidden units and number of experts on the generalization performance of ME and CCME.

- Two novel visualization tools to analyze automatic problem decomposition in high-dimensional feature spaces (chapter 6). The first, simpler, mechanism groups data based on the specialization of each expert, generating a color–map of the data records. The second mechanism relies on *Principal Component Analysis* (PCA) to project the feature space onto lower dimensions, in which decision boundaries generated by each expert are visualized through convex hull approximations. These tools help to visualize how the models decompose the data into sub-regions, so that the input-output relationship in each sub-region is easier for the expert to learn. The tools also show how the methods discover potential modularity in the dataset, and match this modularity by assigning different experts to different modules in the problem. The PCA plots also indicate how increasing the number of experts and number of hidden units affect the way ME and CCME divide and conquer hard problems. The plots show that there is a range of optimal structural complexity in which increasing the structural complexity of the ensemble also increases the system's potentiality to find a better decomposition and solve the decomposed sub-regions with higher accuracy. Beyond this range, increasing complexity does not help the performance of the system. Although these tools are used in this thesis to analyze neuro ensembles, they are not limited to neural networks and can be used with any ensemble methods. They are useful for the analysis of automatic problem decomposition in general, independent of the representation. The tools and analysis address the third research question. about how ME and CCME automatically decompose a complex problem.

- The first empirical study of two different regularization mechanisms, *Learning by*

Forgetting (LF) and *Weight Elimination* (WE), on the performance and structure of ME and CCME models for binary classification problems (chapter 7). First, the study addresses the fourth research question, showing that regularization can generate ME and CCME with low structural complexity, while maintaining accuracy. Second, the study suggest that LF can discover the modularity of each expert in its own designated regions of the input space. Moreover, with the help of LF, the model can discover the importance of the input features and the relevance of the features for each expert. In other words, ME and CCME with LF can automatically assign different experts to different subsets of the input features. This empirical study addresses the last research question, in that LF helps mixtures of experts (and their extensions) to automatically decompose a complex problem. It also opens up a number of new questions for future research, such as whether LF can be used to perform feature selection on other types of neuro ensemble.

Chapter 2

An Overview of Artificial Neural Networks

Researchers have been attracted to the vast, sophisticated architecture and marvellous functionality of human and animal brains for centuries. In 1943, Warren McCulloch and Walter Pitts proposed the first artificial model for a neuron (the so-called McCulloch and Pitts model). Since then, numerous methods for building neuro-inspired computational models, ranging from simple to very sophisticated mathematical models, have been proposed and investigated. This field of study is generally known by the name of *Artificial Neural Networks* (ANNs).

To design an ANN, one needs to define a number of parameters. These include the number of neurons, the way they are connected, the learning rule, etc. In the traditional ANN field, networks were hand designed in a lengthy process, requiring an expert's knowledge in the domain of application (Balakrishnan and Honavar 1995). Later, with increasing interest in another biological process, the evolutionary process, it was proposed that *Evolutionary Computation* (EC) could provide a suitable tool to automatically construct and train an ANN (Yao 1999). The use of EC eases the dependence on human experts, both in terms of domain knowledge (EC can self-adapt), and in the lengthy process of hand design (EC can evolve the architecture of an ANN).

One of the main problems for ANN is to find a suitable network that generalizes well on unseen data/task (Bishop 1995). Often, the primary aim of training an ANN is to obtain one that can learn the underlying distribution of the data, and thus be able to predict unseen data accurately. Therefore, it is undesirable to have a network that over–fits the training data, as it will generalize poorly on unseen data. The avoidance of over–fitting - finding a suitable generalization - is an essential aspect of designing an ANN.

In recent years, Dietterich and Pennock (Dietterich 1997; Pennock, Maynard-Reid II, Giles, and Horvitz 2000) proposed an alternative idea to overcome the generalization problem, based on the committee of experts in human organizations. They argued that, by combining a set of experts with different generalization abilities, the whole system might possess a better generalization ability than the individual networks in isolation. Since then, various ideas have been proposed to design, construct and train such ensembles of experts. This idea was heavily applied in the field of ANN, and became a special branch, namely neuro ensembles (Jimenez and Walsh 1998; Liu and Yao 1997; Sharkey 1998; Sharkey 1996).

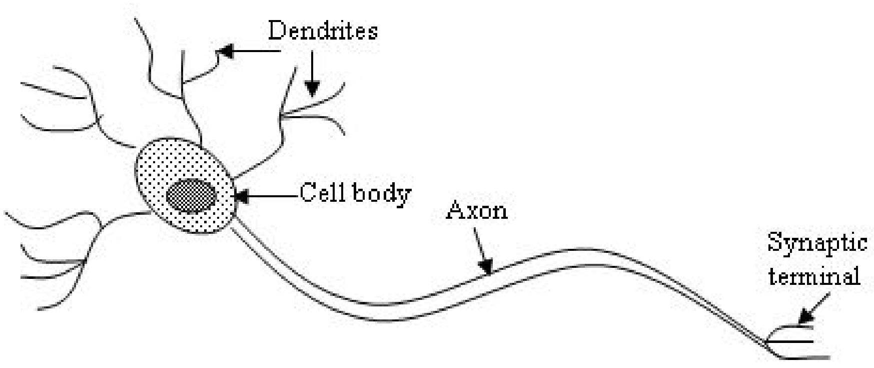
In the following sections, I present a brief overview of the ANN field from the basic structure of ANNs, to evolutionary–based methods, and finally to neuro ensembles and evolutionary neuro ensembles.

2.1 Artificial neural networks

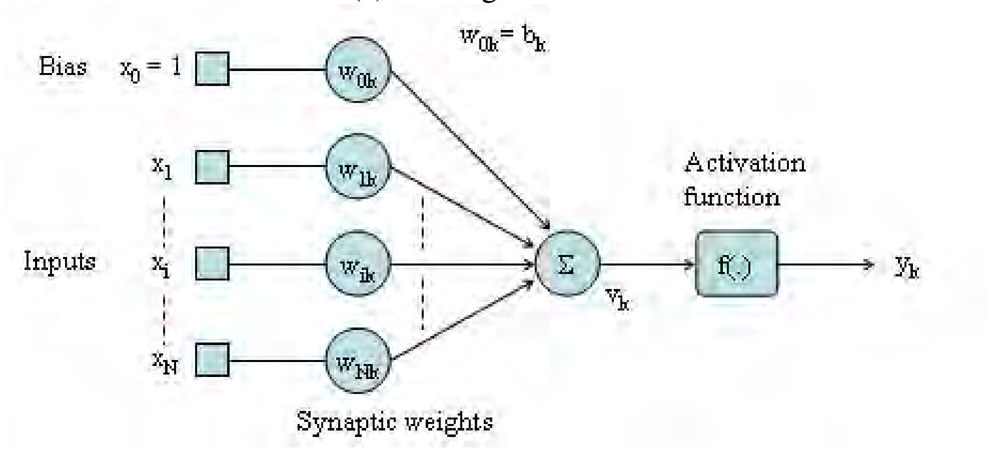
2.1.1 Artificial neurons

An ANN is a network of mathematical interconnections between processing units (Haykin 1999). A neural network can achieve a very large computational power by distributing its work over a massively parallel structure through specialized learning mechanisms.

Biological nervous systems come in various architectures. Some are simple, while others are complex. But all of these different types are composed of the same type



(a) a biological neuron



(b) an artificial neuron

Figure 2.1: Basic structure of (a) a biological neuron and (b) an artificial neuron

of building blocks, the neural cells or neurons.

A neuron receives signals and produces a response. In this sense, a neuron acts as a function which receives a set of inputs as arguments and produces an output. Figure 2.1 shows the basic structure of (a) a biological neuron and (b) an artificial neuron.

In the biological neuron, the inputs and responses are electrical pulses. The input pulses are passed to the neuron body through a set of channels (i.e. dendrites) and the output (if any) is passed forward through the axon. The contact points between different neurons are called the synapses. The synapses can open and close to allow or stop the current flow; they also direct the pulses in a well defined manner. The surface (membrane) of the neuron body contains a number of ionic channels in the form of pores. When the electrical impulse reaches the synapse, the impulse causes the synapsis to fuse to the membrane of the cell body, and a number of these ionic channels will be opened to let more ions flow to the interior of the neuron cell. This flow of ions changes the potential inside the cell and thus causes a response signal to be emitted. Depending on the type of ions flowing in, the cell potential can increase (excitatory) or decrease (inhibitory), and thus a different action is emitted. The number of ionic channels triggered at the arrival of a signal are different from one synapse to another. Thus, in the artificial model, the synapses are often associated with a weight value.

Analogously to this biological neuron, an artificial neuron often consists of a body which acts as a basic function, with a set of inputs represented by the number of incoming links (McCulloch-Pitts unit) or real values. These values are passed to the neuron through abstract links, which might be associated with weights as in the case of synaptic weights in the biological neuron. The neuron computes the function of the inputs, and emits an output value, which is transmitted to other components in the network. Thus, the artificial neurons can be thought of as (i) simplified versions of biological neurons or (ii) primitive mathematical functions.

2.1.2 Artificial neural networks

If an artificial neuron can be thought of as a primitive function, then an artificial neural network is a network of these basic functions. This distribution of information in a vast network allows an ANN to speed up the computation process and to represent complicated functions.

Different models of ANNs differ in the primitive functions used, the structure and interconnection patterns, and the timing of the transmission of information. Typical ANNs have the structure shown in Figure 2.2. The ANN can be considered as a composite function Φ of the sub functions $f_1, f_2, .., f_H$ evaluated at points $(x_1, x_2, ..., x_n)$. Such a function Φ is called the network function. Different sets of parameters $w_1, w_2, ..., w_W$ produce different network functions.

$$y_{output} = \Phi(f_1(\vec{x}, \vec{w}), f_2(\vec{x}, \vec{w}), ..., f_H(\vec{x}, \vec{w})) \tag{2.1}$$

where $\vec{x} = \{x_1, x_2, ..., x_n\}$ and $\vec{w} = \{w_1, w_2, ..., w_W\}$

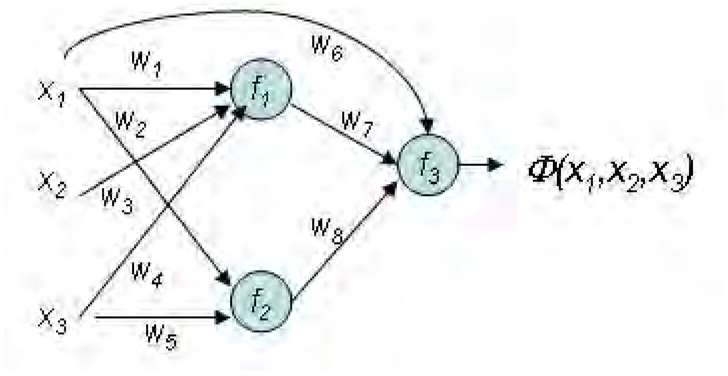


Figure 2.2: An ANN as a network of mathematical functions

Therefore, there are three essential elements to determine an ANN: (i) the structure of the neurons (i.e. their functions f_i's and thresholds θ_i's), (ii) the topology of the network (e.g. the number of nodes and how nodes are interconnected) and (iii) a learning algorithm to obtain a suitable set of weights $\{w_k\}$ of the network.

2.1.2.1 Computational units in ANNs

The simplest form of artificial neuron was proposed in 1943 by McCulloch and Pitts. In their model (Figure 2.3), each neuron receives, as inputs, binary signals through a set of un–weighted edges. These edges belong to one of two types: excitatory and inhibitory. The McCulloch and Pitts neuron computes its binary output as follows: if at least one of the incoming edges is inhibitory, and a 1 is received through this edge, the output of the unit is 0 (inhibited); otherwise, if the overall signal $x = \sum_i x_i$ through the excitatory edges is greater than or equal to a certain threshold θ, the unit fires a 1; otherwise a 0 if $x < \theta$. With this particular setup, the McCulloch and Pitts neuron model achieves its functionality by defining a suitable number of inhibitory and excitatory edges and assigning an appropriate threshold θ to the neuron.

Although it has been claimed initially that a network of McCulloch and Pitts neurons can compute any logical function, the model has some undesirable features for simple use. First, because of the limited number of free parameters in the network, the

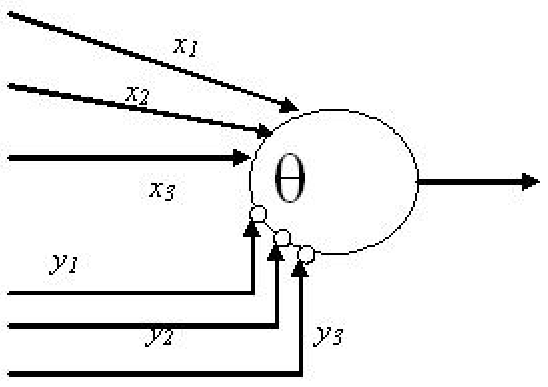


Figure 2.3: McCulloch and Pitts model of an artificial neuron

system is often inflexible when applied to different problems. Second, learning in the McCulloch and Pitts system is obtained via changing the connections between the units. Physically disconnecting and connecting these edges is more problematic than changing a numerical weight over the edges. Because of these two limitations, the weighted model is often more appealing than the simple McCulloch and Pitts model (Rojas 1995).

The second model for the artificial neuron is the perceptron (Figure 2.4) proposed by Frank Rosenblatt in 1958. A simple perceptron is a computing unit with threshold θ and a set of edges with weights $w_1, w_2, ..., w_n$. A perceptron receives n inputs $x_1, x_2, ..., x_n$ and outputs 1 if $\sum_{i=1}^{n} w_i x_i \geq \theta$ and 0 otherwise (Rojas 1995).

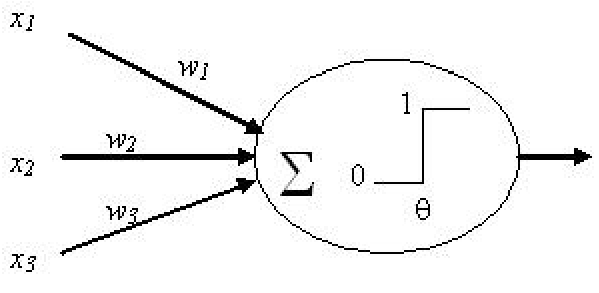


Figure 2.4: A perceptron model of an artificial neuron

A geometric interpretation of the perceptron is as a hyperspace dividing the input space into two half-spaces, where points belonging to one of the half-spaces are associated with an output of 1, while a 0 is associated with the other half (Figure 2.5).

With this structure, perceptrons can be considered as step functions of a linear

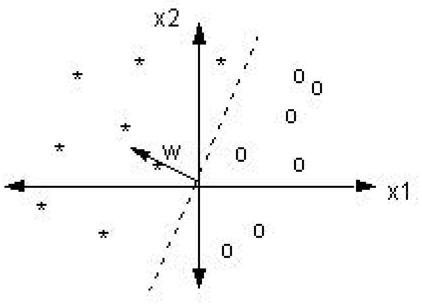


Figure 2.5: A geometric interpretation of the perceptrons

combination of the input units. However, a step function is non-differentiable. This is a serious limitation because it excludes many simple approximation methods in which gradient information (i.e. derivatives) play an essential role. Thus, for a more powerful computing unit, the step function is often replaced by a smooth threshold function, such as the sigmoidal function used in this thesis.

2.1.2.2 ANN topology

A single perceptron is powerful enough to compute a number of basic logical functions. However for more complicated situations, such as non-linearly separable problems, a combination of perceptrons is required (Haykin 1999; Rojas 1995). The arrangement of these basic computing units is a critical feature in defining the expressive power of an ANN. This arrangement is called the topology or architecture of the ANN.

As mentioned in the introduction of this chapter, the biological nervous system is laid out in a well-defined manner. The direction of current flow is very well defined in the makeup of the synapses.Thus artificial neural networks, imitating the nervous system, are often represented by a directed graph, with arrows as the direction of the signal flow (Figure 2.6a). As in directed graphs, an edge can be forward or feedback (Figure 2.6b). Depending on the existence of feedback edges, one can divide ANNs into two types: (i) a *Feed-forward Neural Network* (FFNN), which has no feedback links, and a *Recurrent Neural Network* (RNN), which has some type of feedback (or cyclic) edges.

Other important features of an ANN topology are the number of intermediate

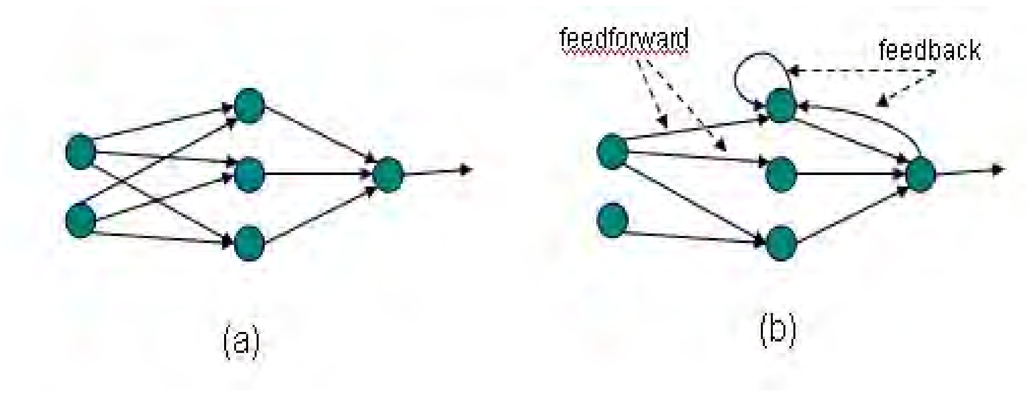


Figure 2.6: ANNs (a) as a directed graph (b) with feed–forward and feedback edges

layers of neurons, and the number of neurons in these layers. As the input layer serves as a "sensory" layer to receive external inputs and the output layer is to emit the final signal to the environment, the intermediate layers are the main computing mechanisms of an overall ANN. These layers are hidden from the external environment, leading to the names "hidden layers" and "hidden units". The number of these hidden units, and their complex arrangement, largely determine the expressive power of an ANN.

The third important aspect of the ANN topology is the mechanism by which the units are interconnected. Neurons can be partially or fully connected, depending on the requirements of the problem. A fully connected ANN is simpler to design, but generally contains redundancy, which may hinder the performance of the system. A partially connected ANN is more compact, more precise and faster than a fully connected one. However designing a proper pattern of partial connections requires extra effort from the designer and/or the system itself (if automatic methods are implemented).

In the scope of this thesis, I implement the most common ANN architecture, the so called feed-forward single-layer fully connected ANN. In other words, the implemented ANN consists of a single hidden layer without feedback edges; and every unit in one layer is connected to every other unit in the next layer (Figure 2.7).

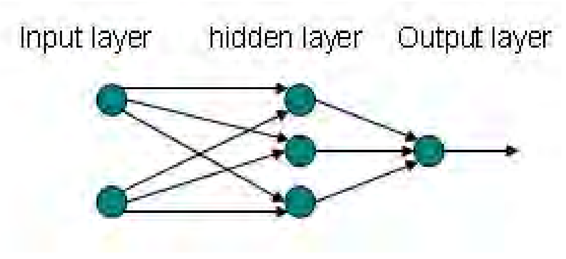


Figure 2.7: A feed-forward one-layered fully-connected ANN

2.1.2.3 Learning rules

An ANN is attractive because of its ability to learn and generalize. Learning is achieved by training the network with a set of training data. During the training process, the connection weights and threshold are adjusted.

The adjustment of weights is undertaken by a learning rule. There are four basic learning rules, namely error-correction, Hebbian, competitive and Boltzman learning (Haykin 1999). During the learning phase, the network produces a response y_k for each input pattern. However, this response may be different from the target output d_k. With some learning-rate parameter – which affects the rate of convergence and hence must be carefully selected – the synaptic weight adjustment $\Delta w_{kj}(n)$ of the four basic learning rules is summarized below (Haykin 1999).

Error-correction learning minimizes some cost function based on the difference between the actual and target outputs $e_k = y_k - d_k$. The adjustment to the synaptic weight is defined as $\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$. Hebbian learning is based on some simple rules, that the strength of a connection should be increased or decreased, depending respectively on whether the two neurons on its two sides excite or inhibit each other (i.e. have the same or different signs). The weight adjustment for the Hebbian rule is defined as $\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$. Thirdly, in competitive learning, the output neurons of a neural network compete against each others to be active. Then, the synaptic adjustment is $\Delta w_{kj}(n) = \eta[x_j(n) - w_{kj}]$ if neuron k wins the competition and 0 if it loses. Finally, Boltzman learning is based on information-theoretic and thermodynamic concepts: it is more likely that a network will jump from high energy to lower energy. The state of a random unit is flipped from 'on' to 'off' and vice versa, according to some probability based on the energy change and some temperature T. $\Delta w_{kj}(n) = \eta(\rho^{+}_{kj} - \rho^{-}kj)$ where ρ^{+}_{kj} and ρ^{-}_{kj} are respectively the conditional (on the condition that all visible neurons are clamped to a particular state determined by the environment) and unconditional correlations between the state of neurons j and k.

2.1.2.4 Classes of learning algorithms

There are three basic classes of learning algorithms: supervised, reinforcement and unsupervised learning. The main difference in these three classes is the degree of dependence of the system on the prior knowledge regarding the behavior of the function (decreasing in that order from fully dependent to independent). Supervised learning requires a priori knowledge in the training, i.e. the "teacher" has to set up a set of inputs with their corresponding outputs. ANNs for this purpose often use error-correction to adjust the synaptic weights. By contrast, reinforcement learning does not require explicit prior knowledge, this knowledge rather being received from the environment. The system constructs a trial network, and observes the reaction in the environment, using its observations to adjust the weights. Unsupervised learning happens when the system does not have any prior knowledge. The scope of this thesis is limited to supervised learning.

2.2 Feed–forward artificial neural networks

The previous sections introduced the basic details of ANNs. The following section provides an overview of a popular architecture in the ANN literature, the so called FFNNs. There are other types of architectures such as recurrent neural networks, associative networks, and *Radial Basis Functions* (RBF). But this thesis is limited to FFNN.

2.2.1 Back propagation

An ANN can be regarded as a complex composite function of the input signals. A key question is how to derive a suitable set of parameters (i.e. synaptic weights) to fit the function to a particular problem. The task of adjusting these weights is known generally as the learning problem. As I described in the previous section, a significant number of learning algorithms have been proposed in the literature of ANNs. Among them, the most popular, and most widely applied in industrial problems, is *Back Propagation* (BP).

The back propagation algorithm looks for the minimum of the error function in the weight space, using the method of iterative gradient descent. In other words, the set of weights that minimizes the error function is considered to be the solution of the system. Since this method requires the explicit computation of the gradient of the error function, it is essential for the error function to be continuous and differentiable. The back propagation algorithm can be summarized in the following components:

1. Initialization: BP starts with a set of zero values or random values for the synaptic weights and thresholds

2. Presentation of training patterns: the order and the mode to present training patterns to the ANN is an important aspect of BP. There are two basic modes: (i) sequential mode and (ii) batch mode.

3. Forward computation: given a pattern $(\vec{x}, \vec{d})$, forward $\vec{x}$ to the ANN and compute the output of each node. The computation can be divided into three sub steps:

   - Compute the weighted sum of the inputs to node j: $v_j^l = \sum_i^P w_{ij}^l y_i^{l-1}$ where w_{ij}^l is the weights of the incoming link to node j from node i and y_i^{l-1} is the output of node i. l is the current layer of node j and $l-1$ indicates the previous layer.

   - Compute the output signal of node j by applying the activation function $y_j^l = \varphi(v_j^l)$

- Compute the errors at the output layer depending on the error function in use. For example, for the sum-of-squares error function, the error of node j at the output layer is defined as $e_j = d_j - y_j^L$ where d_j is the target value and y_j^L is the computed output.

4. Backward computation: the error is propagated backward to the nodes in hidden layers. A local gradient is defined and computed for each node.

5. Weight update: weights are updated in the negative gradient direction: $w_{ij} = w_{ij} + \eta\frac{\partial E}{\partial w_{ij}}$ where η is the update step size, generally known as the learning rate.

6. Stopping criterion: the learning process is halted when a certain stopping criterion is met. The simplest stopping mechanism is fixing the number of iterations (called epochs). However, later on in this thesis, I will discuss why other types of stopping criteria may be preferable to this simple one.

2.2.1.1 Activation functions

1- Sigmoidal function

$$\varphi(x) = \frac{1}{1 + e^{-ax}} \tag{2.2}$$

The coefficient a determines the shape of the sigmoid curve (Figure 2.8)

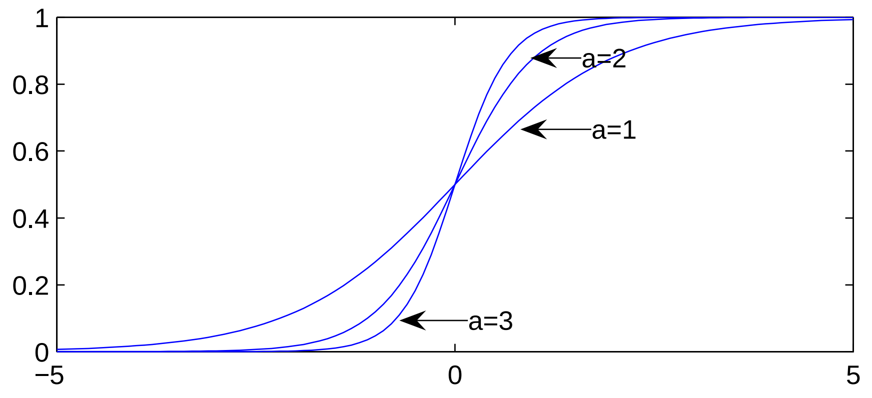


Figure 2.8: Sigmoids with different coefficients

The derivative of the sigmoid with respect to x is

$$\frac{d\varphi(x)}{dx} = \frac{ae^{-ax}}{(1+e^{-ax})^2} = a\varphi(x)(1-\varphi(x)) \tag{2.3}$$

2- Hyperbolic tangent function:

Another popular activation function in the literature is the so called hyperbolic tangent (Figure 2.9).

$$\varphi(x) = a\frac{1-e^{-bx}}{1+e^{-bx}} \tag{2.4}$$

The derivative of the hyperbolic tangent with respect to x is

$$\frac{d\varphi(x)}{dx} = \frac{b}{a}(a-\varphi(x))(a+\varphi(x)) \tag{2.5}$$

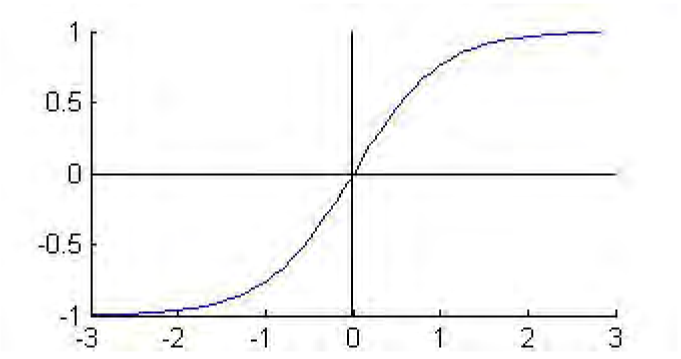


Figure 2.9: Hyperbolic tangent

The advantage of this activation function is its symmetry about the origin as seen in Figure 2.9. In fact, any squashing function, which is continuous and differentiable, can be used as an activation function for ANN, but in this thesis, I use the sigmoid function.

2.2.1.2 Sequential vs. batch mode

There are two principal training modes for presenting the training examples and updating the weights: sequential and batch training modes.

1. Sequential training mode (also known as online or stochastic training mode): in this training mode, the ANN weights are updated after each exposure of each training pattern (Algorithm 1), with the patterns typically being presented in random order.

2. Batch training mode (also known as off-line): in this mode, the weight corrections $\triangle_1 w_{ij}$, $\triangle_2 w_{ij}$, $\triangle_N w_{ij}$ are computed for weight w_{ij} with patterns $1..N$. The overall correction is computed as the sum of these per-pattern weight corrections $\triangle w_{ij} = \sum_k^{k=N} \triangle_k w_{ij}$. Weights are updated at the end of an epoch, when all patterns have been presented to the system.

Each of these training modes has its own benefits and drawbacks. The batch mode exactly follows the negative gradient direction, but can be rapidly trapped in a local minimum. On the other hand, the sequential mode, with its stochastic nature, does not exactly follow the gradient, and is thus less predictable than the batch mode. However sequential BP can jump out of local minima. It is arguable that "adding some noise to the gradient direction can help to avoid falling into shallow local minima of the error function" (Rojas 1995). In this thesis, I use the sequential update mode.

Algorithm 1 The back propagation algorithm in sequential mode

```
1: randomly select a set of initial weights
2: repeat
3:   for each pattern $\vec{x}, \vec{d}$ in the training set do
4:     forward the input $\vec{x}$ and compute the output of nodes in the ANN.
5:     compute the errors at the output level.
6:     pass the errors backward, and compute the partial derivatives of the errors with
       respect to each weight $\frac{\partial E}{\partial w}$.
7:     update the weights in the negative gradient direction. $\triangle w = -\eta \frac{\partial E}{\partial w}$
8:   end for
9: until the halting criteria are satisfied
```

2.2.1.3 Learning rate

The learning rate is a measure of the size of the step in the weight space taken in the direction of the minimum error. A small step size results in smoother trajectory but slower convergence. A large step size speeds up the learning process but may cause instability (i.e. oscillation) in the system (Haykin 1999). Usually, a carefully tailored trial-and-error approach has to be adopted, to find the most suitable learning rate for a particular problem. Other methods automatically adding a correction factor, such as adding a momentum component, to the error function can also be used.

2.2.2 The generalization problem for BP

The goal of training an ANN is not to find a model that exactly fits the training data. Such a model often performs poorly on unseen data, because instead of learning the true underlying distribution (i.e. function) of the data, it memorizes the training data. The ability to perform well on unseen data is known generally as the generalization ability of an ANN. To understand this generalization phenomenon, two concepts, namely bias and variance, have been introduced. The generalization error is decomposed into three components (Brown 2004): a bias term, which deals with the ability of the model to fit a particular training set, a variance term corresponding to the flexibility of the model to different training sets and a third term corresponding to the existing random noise in the training data. This last error term is not related to the model itself and is thus ignored in the bias/variance trade-off problem. The bias and variance terms are usually in conflict with each other, in the sense that a better-fitting model (high bias) must result in less flexibility (low variance), and vice versa. Theoretically, the best compromise or trade-off between bias and variance terms would result in an acceptable generalized model (Bishop 1995; Haykin 1999). In practice, it is often impossible to know in advance a perfect bias/variance tradeoff, because of the complexity of the ANN. These terms are deep-rooted in different aspects of the ANN, such as the model selection method, the architecture complexity, the learning algorithm, and the magnitudes of the weights. Numerous methods have been proposed in the literature for finding a good region for this trade-off but so far it is still an elusive concept in practice.

2.2.2.1 Cross validation

In most cases, only the training data are known. Without some kind of testing of the model, it is very hard to avoid the system over–fitting the training data. One way to deal with this problem is to divide the training data into two subsets: a training set and a validation set. The validation set acts as a test of the model on unseen data. The training set is used in the normal fashion, to train the system. The validation set is used to estimate the generalization performance of the ANN in a number of ways. For example, the validation set can be used to compute the performance of the ANN, if knowing the generalization is the only concern. Alternatively, it can be used to stop the training process early, as I describe below.

Although dividing the training data into a training set and a validation set is helpful, the ANN is now additionally subject to bias in the division process itself. To solve this problem, statistical concepts are often applied. Instead of a one-time division, the system can be repeatedly partitioned into a number of partitions at random points. In this thesis, I adopt the traditional k-fold cross validation, in which the dataset is partitioned into k disjoint subsets using stratified sampling. In each fold, two subsets are chosen as the test and validation sets, and the remaining subsets are combined to form the training set. For example, in the first fold, subsets 1 and 2 are used as test and validation and a combination of $3 \rightarrow k$ is used as training; in the second fold, 2 and 3 are used as test and validation and a combination of 1 and $4 \rightarrow k$ is used as training and so on. This technique results in k folds of data setups. The system is then run with each fold, and an average error over the test sets of the k folds is reported as the system's performance.

2.2.2.2 Early stopping of training

One way to handle the generalization problem is to stop the training process early, before it over-learns the training data. This idea takes advantage of the cross validation mechanism, using the validation set to predict the generalization ability of the system. The smoothed curves (because actual curves are often much more rugged, with many raises and dips) of the training and validation errors are shown in Figure 2.10. As I can see in these curves, there exists a point t where the system generalizes best. After that point, the training curve still reduces, while the validation curve starts to increase. In other words, after this turning point, the system starts to over–fit the training data and generalize poorly. This turning point can be approximated by the minimum on the validation curve (i.e. minimum error on the validation set), so that the training process should be stopped when this point is reached.

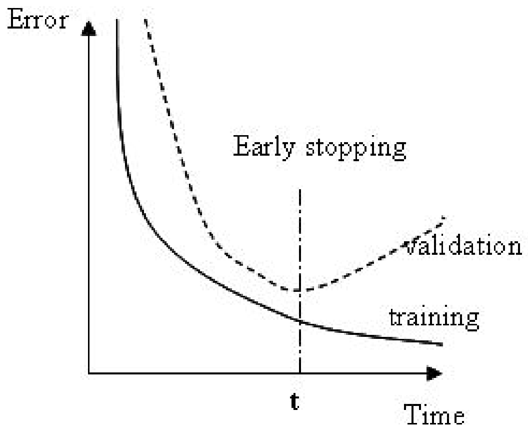


Figure 2.10: Early stopping based on minimum error on validation set

2.2.3 Growing and pruning FFNN

To solve real world problems, often a highly structured ANN is preferable to a large size one. The concerns here are, how to decide on the right size of the ANN while maintaining the performance, and more importantly how to organize the ANN into a suitable structure. There are generally two ways to achieve this objective: (i) growing an ANN from a minimum structure (i.e. the constructive mechanism) and (ii) pruning a large network to remove unnecessary redundancy (the so called destructive mechanism).

2.2.3.1 Growing

Constructive techniques, such as the Tower (Gallant 1990), the BPtower (Hayward 1999), the Tiling (Mezard and Nadal 1989) algorithms, and the famous Cascade Correlation algorithm (Fahlman and Lebiere 1990), share the same principles: hidden nodes are added and connected to existing nodes when the system still makes errors (classification problems) or when the error curve reaches a certain plateau (regression problems). It is argued that, by incrementally increasing the network complexity, via adding hidden nodes, the system is able to adapt itself to the best compact structure that minimizes the error. These methods are reported to perform well on artificial datasets. But in practical situations, the benefit of such constructive mechanisms has not yet been confirmed (Campbell 1997). First, adding hidden nodes may not reduce the error, due to the complex interactions among the nodes and weights of the ANN. Second, difficulties in detecting the optimum size mean that the constructed network is often overgrown, so that it over–fits the training data and thus generalizes poorly. To prevent this over-complexity, some form of early stopping may be implemented, to halt the growing process at the correct moment. Another way to solve this problem is to implement, in parallel, a pruning mechanism which reduces the network complexity and regularizes the network structure.

2.2.3.2 Pruning

While the constructive methods attempt to increase the network complexity by adding hidden nodes to a minimum ANN architecture, the destructive techniques look at the problem from the opposite view: regularizing the complexity of ANN by decaying the weights and/or pruning near-zero connections from a complex ANN. Pruning redundant weights and nodes has a number of benefits. Firstly, the resulting structure is compact, sot that it is less computationally expensive to train, and less likely to over–fit the data (Haykin 1999). Secondly, the relevance of nodes and connection patterns are highlighted more clearly, resulting in a better and/or modularized structure (Campbell 1997). Thirdly, some forms of regularization (e.g. weight decay) smooth out the predicted function curve, and thus are less likely to fit the noise in the training data. Large weight values produce a “mapping with regions of large curvature”, while relatively small weights produce approximately linear network mapping, since with small weights, the sigmoidal function in the neurons is activated in the approximately linear central region (around zero) (Bishop 1995).

Since I will investigate a number of regularization methods in this thesis (Chapter 7), the following subsection is devoted to providing an overview of some well known pruning methods in the literature of FFNN. The destructive techniques could be approximately categorized into (1) complexity regularization, (2) physically removing synapses or nodes and (3) a combination of (1) and (2).

2.2.3.2.1 Complexity regularization In complexity regularization methods, a penalty term is added to the usual performance measure, to smooth out the network mapping.

$$R(\mathbf{w}) = E_p(\mathbf{w}) + \lambda E_r(\mathbf{w}) \tag{2.6}$$

where $E_p(\mathbf{w})$ is the normal performance measure (e.g. error function) and $E_r(\mathbf{w})$ is the penalty function for the complexity of the network and λ is the regularized coefficient or penalty weight. Training is performed on this composite error function. A network with a good fit to the training data will result in a smaller value for E_p while one with a smoother mapping will result in a smaller value for E_r. The resulting network mapping is a compromise between fitting the data and smoothing the curve (Bishop 1995).

- *Weight decay*: the simplest form of network regularization is the weight decay method.

$$E_r(\mathbf{w}) = \frac{1}{2}\sum_i w_i^2 \tag{2.7}$$

  where w_i involves all weights and biases in the current artificial neural network. The result is an ANN with smaller values of weights, and often better generalization ability. To prune this ANN, weights with magnitudes less than a certain threshold are considered redundant, and their corresponding links are removed from the ANN (Bishop 1995).

- *Weight elimination*: However, a problem with the simple weight-decay mechanism is that the system prefers networks with many small weights to those with a few large weights. To fix this problem, Weigend et al (Weigend, Rumelhart, and Huberman 1990) proposed another regularizing term, the so called weight elimination.

  $$E_r(\mathbf{w}) = \sum_i \frac{w_i^2}{w_0^2 + w_i^2} \tag{2.8}$$

  where w_0 is a scaling factor, which is often set to 1. This modified decay term favors ANN with a few large weights over those with many small weights, so that weights are more susceptible to elimination (whence comes the name) (Bishop 1995).

- *Laplace prior regularization*: when considering the weight decay technique in the light of the Bayesian framework, it can be viewed as incorporating a Gaussian prior. This raises an interesting question, whether other prior distributions could be applied in complexity regularization. One of the better distributions for the regularization term is the Laplace prior (Goutte and Hansen 1997; Williams 1993; Williams 1995).

  $$E_r(\mathbf{w}) = \sum_i |w_i| \tag{2.9}$$

  Williams (Williams 1993; Williams 1995) argued that a Laplace weight prior is more suitable than a Gaussian for feed–forward neural networks with no direct input-output connections. They base this on the consistency principle, which demands that the prior for the weights should be a function of the weight magnitude $|w|$ alone and “the maximum entropy distribution for a non-negative quantity constrained to have a given mean is the exponential distribution” (Williams 1995).

Thus a Laplace density $\frac{1}{2}\alpha e^{-\alpha|w_j|}$ is a suitable choice.

2.2.3.2.2 Weight and node pruning The key question in network complexity pruning is how to define the relative importance of the weights, in order to prune the least important. Some measure of importance, or saliency, of weights has to be defined. The complexity regularization method above uses the simplest form of salience – the magnitude of weights, with the argument that smaller weights are less important than larger ones (Bishop 1995). However, this simplistic form of argument may not capture the actual characteristics of the network weight space, so that pruning may not be efficient.

Weight pruning takes advantage of the same error function as used in the ANN training. The methods in this category, such as Optimal Brain Damage (Le Cunn, Denker, and Solla 1990) and Optimal Brain Surgeon (Hassibi and Stork 1993), are based on the second-order derivatives of the error surface, and use this information to trade off the ANN performance and complexity (Campbell 1997).

Weights are not the only thing which can be pruned. Complete nodes may be removed as well. The simplest way is to measure the salience of nodes, based on the difference between the cost functions with and without the node. However, this technique is quite slow, as the system must be repeatedly trained with and without each of the available nodes (Bishop 1995).

2.2.3.2.3 Combining regularization with node pruning - the case of *Structural Learning by Forgetting* (SLF): One of the well-known methods to combine magnitude-based pruning with physical node pruning is structural learning by forgetting (Ishikawa 1996; Miller and Zurada 1997; Kozma, Kitamura, Malinowski, and Zurada 1995). SLF is rooted in the Laplace prior regularization, adding a penalty term of $\sum_i |w_i|$ to the training error function for back propagation. The derivative of this penalty term with respect to the individual weight $|w_i|$ is $sgn(w_i)$ where sgn is the sign function which is 1 if $|w_i| > 0$ and -1 otherwise.

The updated weights for BP thus becomes $\triangle w_i = -\eta \frac{\partial E}{\partial w_i} - \eta \varepsilon sgn(w_i)$ where

$-\eta\frac{\partial E}{\partial w_i}$ is the usual negative gradient of the error function, η is the learning rate and $\varepsilon' = \eta\varepsilon$ is the regularization coefficient. After the system is trained for a number of epochs with this modified cost function, the system enters the second stage, in which hidden nodes are forced to become either 0 (inactive) or 1(active). Ishikawa (Ishikawa 1996) reported that SLF was able to discover a skeletal form of ANN with relatively good performance.

In Chapter 7, I will present an experimental study of two regularization techniques: weight elimination and a modified version of SLF.

2.2.4 Problems with gradient descent approach

Although gradient descent is the most popular ANN method, especially in industrial problems, because of its simple implementation and its efficiency, it still has some serious drawbacks. First, any method based on gradient descent suffers from the risk of being trapped in local minima. By descending to a valley of the error function, the gradient-based method can guarantee to reduce the total error to a local minimum. However, it is impossible for these simple gradient descent methods to escape from a local minimum once it reaches that point. Unless a perturbing mechanism is introduced, to corrupt the parameters and thus bring the system out of the trapped local minimum, (as is done in the generalized gradient techniques (Bishop 1995; Rojas 1995)), the system will remain trapped forever. Second, with gradient descent, not every function can be learnt (Sutton 1986). This can be seen clearly in problems where the error surface is multimodal or when there are discontinuities in the error function.

There are a number of ways to ameliorate the problem of local minima. One is to carefully select a set of good initial weights, in a promising region for locating the global optimum. This is often inefficiently done through trial and error. An alternative is to occasionally add a small perturbation to the ANN weights, to move it out of any trapped local minima. A key issue is, how and when to add such interference to the system, so that the interference does not destroy the knowledge learnt so far. A third alternative is to change the architecture of the ANN. Altering the architecture (e.g. number of hidden layers, number of hidden units) changes the shape of the overall error function, and thus by chance may arrive at a better error function.

However applying concepts from biological evolution to solve multi-modal problems might offer a better solution to these drawbacks in gradient descent learning. In the next section, I will discuss the application of evolutionary computation (EC) to enhance the learning of ANNs. EC techniques are not based on gradient learning, so they suffer less difficulty when problems are multi–peaked and/or the error surface is discontinuous. However, the most important advantage of EC is that it can automatically generate and adapt the architecture and weights of the artificial neural network without intensive domain knowledge.

2.3 Evolutionary artificial neural networks

Evolutionary Artificial Neural Networks (EANN) are a special class of training algorithms for neural networks. They are based not only on learning, but also on evolution. A biological brain does not develop into a complex system solely based on adjusting its connection weights, but also on, throughout generations, the evolution of its architecture and learning rules. In other words, evolution (over the generations) changes the genetic code of the neural network, and learning (over the lifetime of an individual) tunes the network to a particular environment (Gruau 1994b; Gruau 1994a). This biological analogy motivates researchers to apply evolutionary techniques to evolve the connection weights, architectures and learning rules of ANNs. The results support the hypothesis that, by both evolving and learning, neural networks could advance in their complexity, power and adaptability (Nolfi and Floreano 1999; Yao 1999).

2.3.1 A brief glance on evolutionary computation

Evolutionary computation refers to "a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution" (Yao 1999). Lying at the heart of an evolutionary algorithm (EA) is a pool of individuals (i.e. population) and a set of evolutionary operators (e.g. selection, crossover and mutation). The individuals are encoded in a specific type of representation, for example a bit string in the case of *Genetic Algorithms* (GA). The Darwinian principle of natural evolution is that individuals are evolved through the process of mating, which combines and sometimes mutates the genetic information stored in the species' DNA strands. The individuals are often competing to survive in the environment. The stronger individual, a concept depending on the evaluated function, has a better chance to survive and spread its genetic information. After a number of generations (possibly millions of years in natural evolution), this system is able to converge to better adaptive and more complex species. Imitating this natural process, an evolutionary algorithm starts with an initial random population, and iteratively applies certain biologically inspired evolutionary operators to modify the genetic information in a fitness improving direction. As a result, at the end of n generations or when a certain stopping criterion is met, the system may have converged to a better region of the search space and thus may produce fitter solutions. Algorithm 4.2.4.1 summarizes a typical evolutionary algorithm.

Algorithm 2 An evolutionary algorithm

```
1: generate an initial population
2: evaluate the individuals in the population
3: repeat
4:     select parents based on their fitness.
5:     apply operators such as crossover and mutation to the parents to create offsprings.
6:     evaluate offsprings.
7:     form the population for the next generation from the offsprings and/or the current
       population.
8: until halting criterion is met
```

EC has a powerful ability to deal with large, complex search problems. The change from gradient search may allow EC to escape from local minima. These advantages motivate the applications of EC to ANN design problems. Pioneering work, e.g. (Montana and Davis 1989; Fogel, Fogel, and Porto 1990), in EANN attempted to code the connection weights to genomes suitable for genetic algorithms. Later work attempted to evolve the next level in complexity of an ANN, namely the architecture of the ANN, or a combination of both architecture and weights. In addition, there have been a number of attempts to evolve the learning algorithm itself. The following section will briefly review some trends in evolutionary artificial neural networks.

2.3.2 Evolution of the connection weights

Connection weights are often adjusted during training by minimizing the error function between target and actual outputs, as discussed in previous sections. One major problem for gradient search is the risk of being trapped in a local minimum of the error function (Yao 1999). Evolutionary methods, which work well in global optimization problems, could solve this problem (Barlett and Downs 1990; Beer and Gallagher 1992; Caudell and Dolan 1989; Dill and Deer 1991; Fogel, Fogel, and Porto 1990; Fogel 1993; Fogel, Wasson, and Boughton 1995; De Garis 1991; Hansen and Meservy 1996; Heistermann and Eckardt 1989; Kinnebrock 1994; Koeppen, Teunis, and Nickolay 1997; Koza and Rice 1991; Likartsis, Vlachavas, and Tsoukalas 1997; Osmera 1995; Porto, Fogel, and Fogel 1995; Prados 1992; Saravanan and Fogel 1995; Secton, Dorsey, and Johnson 1998; Srinivas and Patnaik 1991; Thierens 1996; Topchy and Lbedko 1997).

In order to apply evolution, the connection weights are encoded into chromosomes. Yao (Yao 1999) mentioned that a neural network with its connection weights was often represented as a binary or real-valued string. In a binary representation, each weight is represented as a string of bits. The weights of the connections threading the same hidden and output nodes are located together on the chromosome, to permit the retention of building blocks. Obviously, this method is designed to work with Genetic Algorithms. In a real-valued representation, each weight is represented as a real number; and each individual of the population is a vector of real-valued weights. In this case, *Evolutionary Programming* (EP) or *Evolutionary Strategies* (ES) could be used. In both representations, to evaluate the fitness of individuals, the individual chromosome has to be converted back to a full network. This network is then trained, and the error function is computed, to define the fitness of the individual.

Although evolutionary computation could work well for global search on a complex, multimodal or non-differentiable surface, it is often slow compared to gradient search based techniques such as BP (Yao 1999). Thus, a number of researchers have combined these two methods and obtained impressive results. In these hybrid methods, an EA is used to locate promising search regions, and local search is used to locate optimum solutions in these regions.

Evolution of connection weights is simple, but it can work only on a predefined fixed architecture. Thus, the optimal solutions obtained are limited to a particular architecture, which may not be the optimal architecture for the required task. Moreover, it is often not simple to manually design the network architecture, due to the extensive domain knowledge required. This amount of knowledge is often not available for the designer. Evolutionary techniques can overcome such problems by letting evolution determine the network connections.

2.3.3 Evolution of the ANN architecture

During the evolution of the architecture, evolutionary computation is applied to the connectivity and transfer function of the nodes (Alba, Aldana, and Troya 1993; Angeline, Sauders, and Pollack 1994; Bornholdt and Graudens 1992; Dodd 1991; Gruau 1994b; Gruau 1994a; Koza and Rice 1991; Likothanassis, Georgopoulos, and Fotakis 1997; Liu and Yao 1996; Maniezzo 1994; Marshall and Harrison 1991; Miller, Todd, and Hedge 1989; Mondada and Floreano 1995; Pujol and Poli 1998; Ragg and S.Gutjahr 1997; Sarkar and Yegnanarayana 1997; Schaffer, Caruana, and Eshelman 1990; Stanley and Miikkulainen 2002; Tang, Chan, Man, and Kwong 1995; Tsakonas and Dounias 2002; White and Ligomenides 1993; Yao and Liu 1997; Yao and Liu 1998a; Yao and Liu

1998b). There are direct and indirect encoding schemes, which are described below.

In the direct scheme, either only the connections of the architecture are directly specified, in a two-dimensional matrix in which each entry represents whether a connection between any two nodes is active (1) or inactive (0). Alternatively, both the architecture and connection weights may be encoded (Yao 1999). The mapping between phenotype (the actual neural network) and genotype (the encoded chromosome) is one-to-one, which means that one genotype can produce one and only one corresponding network.

Abbass (Abbass 2002a) encodes the connections weights and the number of hidden units into an individual, and uses a multi–objective optimization algorithm called *Pareto-frontier Differential Evolution* (PDE) (Abbass 2002b; Abbass and Sarker 2002) to decide on the network's weights and the number of hidden units. Yao and Liu (Yao and Liu 1997) designed EPnet, which uses evolutionary programming with only mutation and rank-based selection to evolve both the architecture and connection weights. Ferdinando et al (Di Ferdinando, Calabretta, and Parisi 2001) used GA to evolve two schemes: (i) both architecture and connection weights of a network, (ii) only the architecture.

While direct schemes encode every connection into the genome, the indirect methods encode only some details of the architectures. There are a number of different methods, Yao (Yao 1999) contains a good review of some. Among them, the developmental (or grammatical) rule methods are most popular. A developmental rule is a recursive equation, that, when repetitively applied, will generate the connection matrix.

Kitano (Kitano 1990) developed a context free, deterministic grammar encoding method which used Lindenmayer-systems (L-systems) to evolve a connection matrix. Nolfi and Parisi proposed a neural growing method (Nolfi and Parisi 1992), using a fixed length genetic string to encode both the architecture and connection weights. In their method, the neurons and their axonal growth are positioned in a 2D Cartesian plane. A connection is formed when the axonal growth of one neuron reaches another neuron. In the final stage, the isolated and interconnected neurons and axons are removed from the space. Gruau (Gruau 1994b) developed another developmental grammar encoding method called Cellular Encoding. Cellular encoding "encodes families of similar structured Boolean Neural Networks" as a set of grammar trees, which keep the instructions (graph transformations) to manipulate the cells and their parameters. Eggenberger (Eggenberger 1997; Eggenberger 2000), Bongard and Pfeifer (Bongard and Pfeifer 2003) and Dellaert and Beer (Dellaert and Beer 1994) inspired by biology, proposed various models based on the artificial genetic regulatory system.

2.3.4 Evolution of learning rules

The third level deals with the evolution of learning rules. Unlike the first two levels, which encode and evolve the static properties (i.e. weights and architecture of the ANN), methods in this level attempt to evolve the dynamic properties of the learning process. Key questions are (i) how to understand and characterize the dynamics of learning and (ii) how to measure learning in order to encode it into certain type of genotype (e.g. bit string or array of numerical values). Current researchers following this direction study simplified versions of the learning mechanism.

Chalmers (Chalmers 1990) encoded the complex forms of weight-space dynamics into simple linear genomes, as functions of local information about ANN connections. More specifically, the method encoded a scaling value and ten coefficients of a linear function of four parameters of a synapse link, namely the activations of the input and output units, the training signal on the output unit and the weight of the connection, and their six pairwise products. The ANN architecture in this system was fixed. The system was able to evolve various successful learning mechanisms, including the well-known delta rule (Chalmers 1990). Neirotti and Caticha (Neirotti and Caticha 2003) applied genetic programming to generate populations of programs that implement algorithms used by ANN classifiers to learn a rule in supervised learning mode. A few other methods (Baxter 1992; Bengio and Bengio 1990; Crosher 1993; Kim, Jung, Kim, and Part 1996; Kuscu 1995; Radi and Poli 2003) have been proposed. However because of the extreme complexity of the learning mechanism, this field of study has not advanced very far, the representations and solutions being still limited to simplistic evolution of parameters.

Chapter 3

A Review Of Neuro Ensembles

The problem of generalization (e.g. bias/variance dilemma) is a challenge in the machine learning field (see Chapter 2). A number of experiments have shown that the overall generalization ability of a committee of experts can be better than each expert alone (Jimenez and Walsh 1998; Liu and Yao 1997; Liu and Yao 1999; Liu, Yao, and Higuchi 2000; Rosen 1996; Sharkey 1998; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a).

3.1 A theoretical perspective: why an ensemble is better than a single network

An ensemble is a collection of predictors that can be combined to give an overall prediction.

In Chapter 2, we have discussed the issues of the so called bias/variance dilemma, which can be restated as follows: the best model on a training dataset should minimize the difference between the model's output and the expected output. However, this minimization process has not considered the possibility of noise in the data. It will also attempt to fit any noise that may exist in the training set and thus, when presented with unseen data, it may perform poorly. Such a model is said to have small bias and high

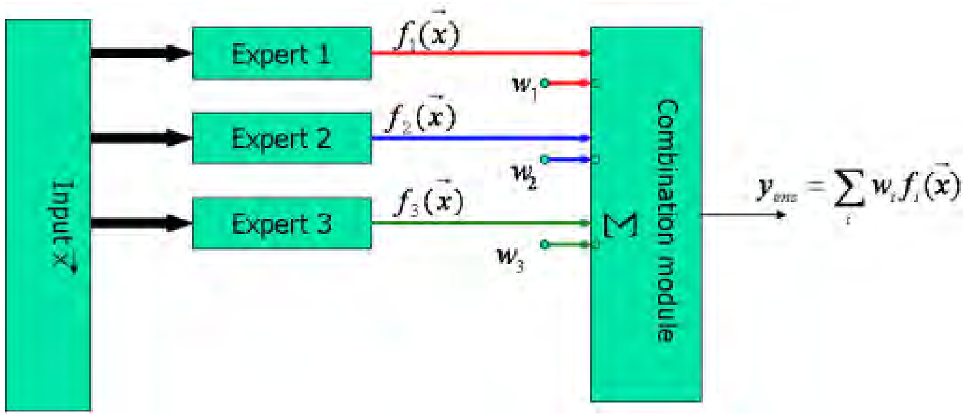


Figure 3.1: Ensemble of predictors

variance. On the contrary, a model with small variance and high bias is one that is less dependable on the training set (e.g. a linear model). The motivation is to find a trade-off between these two components: the bias and the variance.

The motivation behind ensemble research is to find such a tradeoff, by combining models which possess different bias/variance abilities. Intuitively, the average of these different biases/variances approaches the optimal tradeoff. Some researchers have attempted to derive a theoretical framework to prove that this intuition is valid (Krogh and Vedelsby 1995; Tumer and Ghosh 1996; Ueda and Nakano 1996).

3.1.1 Ambiguity decomposition

Krogh and Vedelsby (Krogh and Vedelsby 1995) have shown in the regression context that the quadratic error of an ensemble of predictors is guaranteed to be less than the average quadratic error of the individual predictors.

$$(f_{ens} - d)^2 = \sum_i w_i(f_i - d)^2 - \sum_i w_i(f_i - f_{ens})^2 \tag{3.1}$$

where $f_{ens} = \sum_i w_i f_i$ is the combination of the individual estimators' outputs f_i.

The first term is the weighted average of the errors of the predictors (i.e. accuracy) while the second term measures the amount of variability (i.e diversity) of the predictors in the ensemble, and thus is called the Ambiguity term. Since the ambiguity term is always positive, it guarantees that the ensemble error is always less than the weighted average of the individual predictors. However, although increasing the ambiguity term reduces the overall ensemble error from the weighted average of the members' errors, it also tends to increase the individuals' errors and consequently increases the first term. In other words, diversity alone is not enough, a correct balance between accuracy and diversity is essential to guarantee a better error for the ensemble (Brown 2004).

3.1.2 Bias-variance-covariance decomposition

Although the ambiguity decomposition is useful in increasing the accuracy of the ensemble, it has not attacked the problem of generalization. Ueda and Nakano (Ueda and Nakano 1996) have derived another useful decomposition, based on the original Bias-Variance decomposition for a single estimator.

If the output of the ensemble is a simple average of the individual outputs,

$$f_{ens} = \frac{1}{M}\sum_i f_i \tag{3.2}$$

then, the mean square error of the ensemble is

$$E[(f_{ens} - d)^2] = \overline{bias}^2 + \frac{1}{M}\overline{var} + (1 - \frac{1}{M})\overline{covar} \tag{3.3}$$

where $\overline{bias}$, $\overline{var}$ and $\overline{covar}$ are the average conditional bias, conditional variance, and conditional covariance averaged over all individual estimators respectively (a more detailed derivation may be found in (Ueda and Nakano 1996)).

$$\overline{bias} = \tfrac{1}{M}\textstyle\sum_{i=1}^{M} bias\{f_m\}$$

$$\overline{var} = \tfrac{1}{M}\textstyle\sum_{i=1}^{M} var\{f_m\}$$

$$\overline{covar} = \tfrac{1}{M(M-1)}\textstyle\sum_i \sum_{j,j\neq i}^{M} Cov\{f_i, f_j\}$$

This decomposition indicates that the generalization error of the ensemble also depends on the correlation between the individual estimators. Thus, an ensemble in which individual experts are uncorrelated (i.e. diverse) generalizes better. However, this decomposition is limited to the simple average combination method for regression predictors.

3.1.3 Error correlation in classification

Tumer and Ghosh (Tumer and Ghosh 1996) attempted to provide a theoretical framework for decomposing the error function in a classification context. They assumed that individual classifiers estimate the a posteriori class probability. These estimates are then averaged, to provide the final estimation of the ensemble. Since the estimated boundary may not be the same as the optimum boundary, the total error can be decomposed into two terms: the normal Bayes error E_{bayes} and an added error E_{add}^{ens}:

$$E_{total} = E_{bayes} + E_{add}^{ens} \tag{3.4}$$

Because the Bayes error cannot be altered, the ensemble can only be enhanced by changing the second term. The added error of the ensemble E_{add}^{ens} was derived as follows:

$$E_{add}^{ens} = \frac{1 + \delta(M-1)}{M} E_{add}^{ind} \tag{3.5}$$

where E_{add}^{ind} is the added error rate of the individual classifiers, assumed to be equal for all classifiers. δ is the correlation between the errors of each classifier. Thus, if the errors of the classifiers are fully correlated (i.e. $\delta = 1$), no improvement happens with the ensemble, i.e $E_{add}^{ens} = E_{add}^{ind}$. However, if the errors are uncorrelated (e.g $\delta = 0$), then the added error of the ensemble is reduced by M, i.e. $E_{add}^{ens} = \frac{E_{add}^{ind}}{M}$.

3.2 Forming and selecting the individual networks

By combining the opinions of several “different” individuals or “experts”, the system’s generalization ability can be enhanced (Krogh and Vedelsby 1995; Tumer and Ghosh 1996; Ueda and Nakano 1996; Yao and Liu 1996). Consequently, the ensemble literature mainly focuses on two important issues, namely accuracy and diversity. Accuracy refers to the ability of each individual predictor to learn and predict the correct underlying distribution of the data; and diversity (or “difference”) refers to different bias/variance trade-offs.

3.2.1 Accuracy issues in classification

Bouckaert (Bouckaert 2002) defined the accuracy of an ensemble under 0-1 loss as “the one minus the 0-1 loss of the classifier, i.e. A = 1 -L” where the loss function is a function “that compares the prediction of a classifier with the true value of y and maps it onto a real value” (y is the data distribution). In (Bouckaert 2002), Bouckaert derived a framework for the upper and lower bounds of ensemble accuracy based on the mean accuracy of the individual classifiers. His proof stated that ensemble accuracy can never exceed twice the mean accuracy of the individuals, and can never decrease below twice the mean accuracy minus 1.

Dietterich (Dietterich 1997) defined an accurate classifier as one which has an error rate of better than random guessing on new x values. He also examined the three fundamental reasons why an ensemble works better than its individual components: (i) statistically: by combining a set of classifiers, the chance of selecting a wrong classifier is reduced, (ii) computationally: a classifier with local search often gets stuck in local minima and thus, by combining classifiers with different local search points, the system could better approximate the true unknown function, and (iii) representationally: a weighted sum of hypotheses drawn from the hypothesis space H may expand the space of representable functions.

3.2.2 Diversity issues

Diversity requires that the individual members of an ensemble be different. In many cases, accuracy and diversity are in conflict. It is sometimes not recognised that selecting a set of most accurate predictors may result in an ensemble of self-similar ones. This is undesirable, because a self-similar ensemble is no better than a single predictor. Hence, in designing neuro ensembles, one has to take into account these two tightly coupled performance indicators.

Researchers have been attempting to verify the relationship between diversity and the ensemble's generalization (Kuncheva 2003b). To build this understanding, one has to answer a number of questions: (i) how to define and measure diversity, (ii) which diversity promotion mechanisms to use, and (iii) how to use diversity to improve the ensemble? (Kuncheva 2003b).

3.2.2.1 How to define diversity?

A number of researchers (Krogh and Vedelsby 1995; Ruta and Gabrys 2001; Tumer and Ghosh 1996; Ueda and Nakano 1996) have attempted to provide a framework to decompose the error term of the ensemble in order to isolate the contributions/effects of accuracy and diversity on the overall performance. Often, a correlation term between individuals in the ensemble is identified as the source of diversity. Following these different decompositions, (Rosen 1996; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a; Liu, Yao, and Higuchi 2000) derived a negative correlation concept to promote diversity among the neural network members of the ensemble. Their results showed a possible reduction in the ensemble error.

However, as shown in (McKay and Abbass 2001) and in our own experiments in chapter 4, the diversity of the ensemble with negative correlation can be very poor. As ensemble diversity is often defined as the total differences (or distances) between all pairs of classifiers in the ensemble; therefore, an ensemble with high diversity is one in which each classifier is as different (or further away) from all others as possible. McKay and Abbass revealed that negative correlation learning only acts to push the members of the ensemble away from their mean, but not necessarily away from each other. Therefore low diversity with negative correlation learning is not a particular surprise. A simple demonstration of such situation can be seen in Figure 3.2. The experts of ensemble in (a) are grouped in two dense clusters, therefore although the distance from each expert to the ensemble mean is high, the total distances between all pairs of experts (i.e. diversity) are low. On the other hand, the experts of ensemble in (b) are pushed away from each other, thus, the total distance is higher.

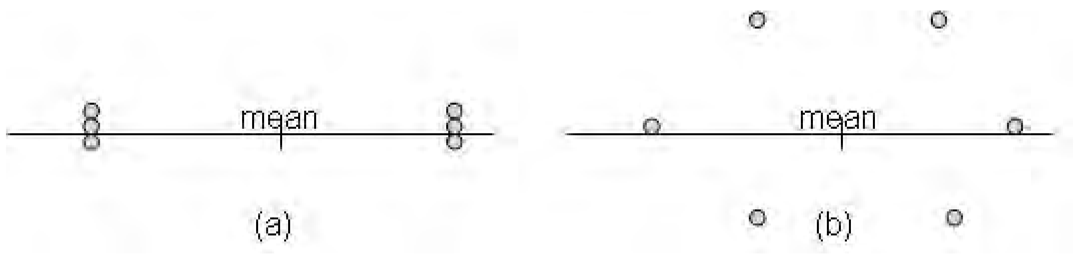


Figure 3.2: An example of ensembles with (a) low and (b) high diversity

Moreover, as noted by Kuchenva (Kuncheva 2003b), the results of correlating different diversity measures to the ensemble errors are discouraging. There is not yet any obvious link between injecting diversity into the ensemble, and improving its generalization ability. In other words, it is still an open question whether the diversity of the ensemble does assist in generalization, and if so, in what way it is useful.

3.2.2.2 Diversity measures

In the literature of ANNs and EC, a number of diversity measures have been proposed. They range from statistically to mathematically to biologically inspired measures. Krogh and Vedelsby (Krogh and Vedelsby 1995) defined diversity as the degree of disagreement (ambiguity) between two classifiers. Some researchers (Cunningham and Carney 2000; Cho, Ahn, and Lee 2001; Cho and Ahn 2001; Kuncheva 2003b) borrowed different measurements from the field of information theory to define diversity, notably entropy (e.g. the Kullback-Leibler distance). Others applied statistical concepts (e.g. Kappa statistics, Q-statistics, etc..) to compute the dissimilarity between classifiers (Kuncheva 2003b; Banfield, Hall, Bowyer, and Kegelmeyer 2003).

Diversity can be measured on different levels in an ensemble. Kuncheva (Kuncheva 2003b) distinguished between the data point level and the classifier level. In the former, the diversity is computed based on the entropy of the distribution of class labels among the classifiers with respect to a certain data point; thus, the diversity in a dataset with N data points is the average of N within-population diversity measurements, corresponding to the populations at the N data points. In the latter level, some pairwise measure of diversity between two classifiers (e.g. measure of disagreement (Kuncheva 2003b; Skalak 1997; Zenobi and Cunningham 2001)) is defined, and the diversity of the ensemble is taken to be the average over all possible pairs.

3.2.2.3 Diversity promotion mechanisms

The next question is how to design such a mechanism to promote diversity in the population. Diversity has been studied in a wide range of fields. Biologists have struggled for decades to understand the mechanisms causing the diversity in the biosphere (bio–diversity) (Standish 2002). EC researchers have been trying to solve the problem of early (premature) convergence and diversity promotion for a considerable time (Laumanns, Thiele, Deb, and Zitzler 2002; Horn, Goldberg, and Deb 1994; Spears, Jong, Bäck, Fogel, and de Garis 1993; Toffolo and Benini 2003; Zitzler 1999). Furthermore, in the literature of ensembles of classifiers, a number of theoretically motivated mechanisms have been proposed, aiming to promote useful diversity among the members of an ensemble of classifiers (Brown 2004; Brown, Wyatt, Harris, and Yao 2004; Chandra and Yao 2004; Liu, Yao, and Higuchi 2000; Rosen 1996; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a; Zenobi and Cunningham 2001). Diversity promotion mechanisms will be discussed in section 3.4.

3.3 Gates in ensemble learning

Currently there are four popular combination methods, namely, majority voting, winner-take-all, simple averaging and weighted averaging. These are described below.

Majority voting In this combination method, the output of the ensemble is the binary output value which receives the vote of the majority of networks in the ensemble.

Winner-take-all In the winner-take-all method, the ensemble output is the output of the network whose output is maximally different from the classification threshold (e.g. 0.5).

$$y_{ens}(\overrightarrow{x}_i) = \arg_{max,j}(|\hat{y}_j(\overrightarrow{x}_i) - threshold|) \quad j \in [1, N_c] \tag{3.6}$$

Simple averaging In the simple averaging method, the output of the ensemble is the mean of the individual outputs of the networks.

$$y_{ens}(\overrightarrow{x}_i) = \sum_{j=1}^{N_c} \hat{y}_j(\overrightarrow{x}_i)/N_c \tag{3.7}$$

Weighted averaging In the weighted average method, a weight α_j is assigned to each ANN member in the ensemble. The output of the ensemble is a linear combination of the output of the individual networks with the assigned weights as the coefficients.

$$y_{ens}(\overrightarrow{x}_i) = \sum_{j=1}^{N_c} \alpha_j \hat{y}_j(\overrightarrow{x}_i)/N_c \tag{3.8}$$

Peronne and Copper (Perrone and Cooper 1993) presented a theoretical framework for simple averaging, and generalized (through weights) the averaging mechanisms. Theoretically, the generalized averaging mechanism should perform better than simple averaging (Perrone and Cooper 1993) in the mean square error sense. Following this demonstration, Jimenez (Jimenez and Walsh 1998) attempted to derive a procedure to dynamically weight the certainties of the individual classifiers. Zhou et al (Wu, Zhou, and

Chen 2001; Zhou, Wu, and Tang 2002; Zhou, Wu, Jiang, and Chen 2001) also weighted the contribution of the members to the generalization improvement, and thus, selected the highly contributing members to form a weighted average ensemble. Fumera and Roli (Fumela and Roli 2002) theoretically and experimentally compared simple averaging and weighted averaging fusion gates. Their results showed that "weighted averaging significantly improves the performance of simple averaging only for ensembles of classifiers with highly imbalanced performance and correlation". Thus, the advantage of weighted averaging over simple averaging is quite small. This argument was supported by the experimental results in (Ueda and Nakano 1996). Pennock et al. (Pennock, Maynard-Reid II, Giles, and Horvitz 2000) examined different combination gates for the classification problem. They identified several properties for the combination gates such as universality, independence of irrelevant information, scale invariance, neutrality, symmetry and positive responsiveness. Their analysis showed that no combination gate possesses all of these properties.

Beside these popular fusion methods, Sharkey (Sharkey 1998) mentioned a few other methods. Sharkey divided the combination gates into four categories: (1) averaging and weighted averaging, (2) non-linear combining methods (e.g. voting and rank-based), (3) supra Bayesian, which is based on the probability distribution of the experts' opinions , and (4) stacked generalization.

3.4 Review of diversity promotion mechanisms

3.4.1 Methods based on data manipulation

3.4.1.1 Data sampling

One popular method to generate diverse members for an ensemble is to sample the training data into different subsets. The purpose is to create subsets of training data such that each subset has at least some data that do not exist in other subsets. Because each subset is used to train a member of the ensemble, it is likely that the ensemble members are different from each other, i.e. the errors are likely to have reduced correlation, compared with ensemble members trained on the same data.

A number of techniques can be used to sample data from a training dataset. The simplest method is to divide the training data into disjoint datasets (Sharkey 1998; Sharkey 1996). However, this method works better with large datasets than with small datasets, where the resultant disjoint subsets may be small enough for a neural network to over-fit the data. Krogh and Vedelsby (Krogh and Vedelsby 1995) used cross-validation as their sampling method. A cross-validation strategy, also called leave-some-out, works by removing a number of data points out of the training data to form the validation set. A network is trained on the training set and tested on the validation set. A k-fold cross-validation is a mechanism in which the training set is divided into k disjoint subsets. The training set is composed of $(k-1)$ subsets; and the k-th subset forms the validation set. One way to use cross-validation is to train individual networks on k different training sets, and use the different validation sets to estimate the generalization error. Another approach is to test the validity of a method by training the networks with k different collections of data (a collection here refers to the corresponding subsets of training, validation and testing data); and then use the average error of the k collections as the true estimated error of the method.

Efron (Efron 1982) introduced a re–sampling method called bootstrapping; and Breiman (Breiman 2000) proposed another re–sampling method called bagging. Both involve sampling the dataset, with replacement, into N subsets. However in bagging, where the probability given to a pattern is $\frac{1}{N}$ across N data points in the original training set, an adaptive scheme will change this probability in favor of the data that are currently misclassified.

3.4.1.2 Boosting and arcing

These techniques are also based on sampling the training data, but the difference lies in the adaptive re–sampling scheme. The original boosting method (Schapire 1990) works as follows: create the first network and train it on a subset of the training data. This network is then used to select a training set for the second network by selecting a distribution of correctly classified and misclassified patterns out of the remaining patterns. Then the second trained network will jointly work with the first one to filter the patterns for the next network. In summary, this process will add a network based on the disagreement of the previous trained networks on "new" data.

The original boosting method has a problem in that it requires a large dataset. To solve this problem, Freund and Schapire (Freund and Schapire 1995; Freund and Schapire 1996) proposed an algorithm that adaptively resampled and combined - hence the name arcing - the data in such a way that the weights in the re–sampling were increased for the most misclassified cases; the combining was done by weighted voting. The best known variant of arcing is the so called Adaboost (Schapire 1999), which can be used with small training datasets by assigning a weight to each data pattern. These weights are updated to favor the patterns most misclassified by the networks of the current round. A number of modified methods based on Adaboost have been proposed. Oza (Oza 2003) proposed AveBoost, which uses a training example weight vector based on the performance of all preceding networks, rather than the previous one alone. Kuncheva (Kuncheva 2003a) attempted to derive a mathematical framework for the error bound of different versions of Adaboost, namely aggressive Adaboost and Conservative 1 and Conservative 2 Adaboosts. These modified versions of Adaboost differ from the original in the way they punish previous successful or unsuccessful outcomes.

3.4.1.3 Other data varying methods

Beside these popular methods to vary the training data (and hence vary the resultant members of the ensemble), there are a number of other methods. These include distortion of the input data by adding noise (Raviv and Intrator 1999), or selecting different feature subsets of the feature space (Zenobi and Cunningham 2001; Oliveira, Sabourin, Bortolozzi, and Suen 2003; Opitz and Shavlik 1996). Another way is to select the input data from different data sources, (*e.g.* different sensory data (Sharkey 1998; Sharkey 1996).

Tumer and Gosh (Tumer and Ghosh 1996) tested four different methods based on correlation reduction: (i) cross-validation to partition training data; (ii) pruning input features to generate new training sets; (iii) re–sampling, and (iv) weighted averaging of individual networks, where the weights are a function of the inputs (so that in each input region, one network gets more weight than the others). Their results showed that except for the first method, which showed some promising results, the methods did not provide significant improvement over the base results. Moreover, it was difficult to fine tune the methods since "a small change in the design step lead to large changes in combiner performance" (Tumer and Ghosh 1996).

3.4.2 Methods based on explicit diversity mechanisms

3.4.2.1 Diversity based on networks' correlation

Knogh and Vedelsby (Krogh and Vedelsby 1995) has proposed another bias/variance decomposition designed for ensembles of predictors. They defined a term "Ambiguity", which measures the disagreement of a member with the overall ensemble output (weighted average) (Krogh and Vedelsby 1995). The ensemble ambiguity was computed as the weighted average of the individual networks' ambiguities. Thus the error of the ensemble, E, could be decomposed into the weighted average of the generalization error of the individual networks, $\overline{E}$, minus the weighted average of the ambiguities, $\overline{A}$.

$$E = \overline{E} - \overline{A} \tag{3.9}$$

This ambiguity term captured all correlations between networks (Krogh and

Vedelsby 1995). Their decomposition scheme suggested that increasing the negative correlation between networks could reduce the ensemble error. Based on this suggestion, a number of methods have been proposed for incorporating negative correlation during the training of individual networks in the ensemble.

Rosen (Rosen 1996) incorporated a decorrelation measure that penalizes correlations between networks. Similarly, Liu and Yao (Liu, Yao, and Higuchi 2000; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a) proposed the negative correlation learning method, which adds the correlation between networks as a penalty term to the error function. This penalty term was computed based on the disagreement of every network output in the ensemble with the ensemble's output (using the average). In other words, negative correlation methods aims to diversify the member networks by decorrelating them.

3.4.2.2 Diversity based on evolutionary computation

Another way to generate diversity in the ensemble is to borrow diversity mechanisms from other disciplines, such as evolutionary computation. Lee et al (Cho, Ahn, and Lee 2001; Cho and Ahn 2001) applied speciation to diversify their evolutionary neural networks. Abbass (Abbass 2003b) proposed to use the Pareto-based multi-objective method to evolve a group of networks that differ from each other in respect of the objectives. He suggested a number of objectives which could be used to promote diversity between networks. The first approach was to contrast the architecture's complexity with the generalization error of the ensemble. Another approach depended on sampling the data, where the dataset is divided into subsets, whose training errors serve as the objectives. The third suggestion was to inject noise to distort the output, and use the distorted output as the second objective in competition with the original output. The second approach was found to produce the best results.

3.4.2.3 Other diversity mechanisms

As mentioned above, noise injection into the output could serve as a source of diversity. The effects of noise injection in the output level range from changing the bias/variance distribution to diversifying the models. Christensen (Christensen 2003) introduced an output distortion mechanism, in which the outputs are distorted by displacing them by amounts selected from a fixed set of predetermined values. The distortion was conducted such “that the average distortion applied to each data points output values is exactly zero”. Their results showed an interesting characteristic, contrasting with the common belief that individual networks in an ensemble should have low bias and high variance. He showed that large bias networks might also be useful, if the correlation component between them were large enough to overcome the bias component.

Most of the above ensemble methods used the same architecture for the individual networks. Another way to handle the diversity issue was to vary the networks’ architectures. Abbass (Abbass 2003b) used the number of hidden units as a possible objective in his multi-objective-based technique. Yao et al also introduced architecture variations in their EPNet model (Liu, Yao, and Higuchi 2000; Yao and Liu 1996; Yao and Liu 1997). Renner (Renner 1999) used cascade correlation as the constructive method; however the selected neural networks did not require diversity on the architecture level. Thus, although the study used the cascade correlation method, the source of diversity actually derived from the data varying techniques.

3.4.3 Using diversity as a criterion to select the members for an ensemble

So far, we have recorded that diversity is widely used in constructing the individual networks for the ensemble. Another important component that may determine the success of an ensemble technique is a pruning process (through which members should be removed from the ensemble). Selection is an important issue because:

- using identical networks in the ensemble is pointless. Thus, a first heuristic goal is to remove identical networks.
- networks with very low accuracy may deteriorate the whole ensemble. According to the ambiguity formula (Krogh and Vedelsby 1995), $E = \overline{E} - \overline{A}$, including high error networks will greatly increase the first term, which is undesirable unless there is a more-than-compensating beneficial increase in the second term of at least $\overline{E}$ (Christensen 2003).
- networks that are positively correlated should not be included together. This argument is derived directly from the ambiguity formula. Positive correlation between networks entails a reduction in $\overline{A}$, and thus a possible increase in the ensemble's error, E.

There are a number of different ways to make use of diversity in network selection. Perrone and Cooper (Perrone and Cooper 1993) introduced the use of the correlation matrix C_{ij}, where the entries represent the correlations between networks. They discussed how this correlation matrix could be ill-conditioned, due to duplicate and nearly duplicate networks. Thus, it is essential to remove these duplicates. The correlation matrix, representing diversity, could be used to remove the duplicates. They also came up with an inequality condition for adding sets to the ensemble.

Similarly, Zhou et al (Zhou, Wu, and Tang 2002) derived a condition for removing networks from an ensemble. They proposed a method called GASEN (Zhou, Wu, and Tang 2002; Wu, Zhou, and Chen 2001; Zhou, Wu, Jiang, and Chen 2001), which assigned weights to individuals according to the contribution of the networks to the ensemble. These weights, used to prune the networks, were evolved using a genetic algorithm programmed to find an optimum contribution distribution of the individual sets in the ensemble.

Aksela (Aksela 2003)) compared six different methods for selecting the members, based on six diversity measures: (i) Correlation between errors (ii) Q-statistics (iii)

Mutual information (iv) Ratio between different and same errors (v) Weighted count of errors and correct results (vi) Exponential error count. The results favor the exponential error count method, which weights identical errors made by classifiers in an exponential fashion, and normalizes it with the number of data points where all members are correct.

Navone et al (Navone, Verdes, Granitto, and Ceccatto 2000) proposed an incremental approach. At every stage, the system searches for a new member which is partially anti-correlated with the current members in the ensemble. However, in their paper (Navone, Verdes, Granitto, and Ceccatto 2000), diversity does not directly result from their method but is rather a theoretical result stemming from the assumption that an added member must reduce the ensemble error. It was not clear if this theoretically expected diversity did in fact emerge in the ensemble.

Banfield et al (Banfield, Hall, Bowyer, and Kegelmeyer 2003) defined another diversity measure - the Percentage Correct Diversity Measure - which counts the number of data instances which are correctly classified by between 10% and 90% of the classifiers. In addition, they proposed two different diversity based mechanisms to thin out the classifiers in the ensemble's pool. However, they used C4.5 decision trees as the classifiers in their research, rather than neural networks.

Oliveira et al (Oliveira, Sabourin, Bortolozzi, and Suen 2003) introduced multi-objective methods to two aspects of ensemble classification: feature selection and member selection. The argument is that Pareto optimality displays characteristics suitable for diversity and ensemble (Abbass 2002a; Kuncheva 2003b; Oliveira, Sabourin, Bortolozzi, and Suen 2003). The second level incorporates selecting the subset of networks that promote maximum recognition rate, and maximization of ambiguity (diversity) using a Pareto-based approach. Although the method seems to perform well, there is no concrete validation of the claim that the members are diverse.

Both (Oliveira, Sabourin, Bortolozzi, and Suen 2003) and (Abbass 2003b) used *Multi–objective Optimization Problem* (MOP) as the main mechanism to promote diversity among the members of the ensemble. However, because of the crisp comparison being used in MOP, the networks could be very similar, yet MOP could still pick out a subset with “highest” diversity, though “highest” may result from a very small difference from the lesser values, e.g. $3.5 > 3.4999$.

Lazarevic and Obradovic (Lazarevic and Obradovic 2001) derived their pruning method based on an unsupervised clustering algorithm, where the distance between clusters represents the diversity of classifiers. The algorithm incrementally increases the number of clusters until the diversity between clusters deteriorates, then the clusters are removed, leaving only their centroids.

3.5 Automatic problem decomposition

One important application of ensembles is *automatic problem decomposition* (APD). Often, a real world problem is too complicated for a single individual to solve. The Divide-and-Conquer strategy has proven effective in many of these complicated situations. The questions that may arise in this research area are (i) how to divide the problem into simpler tasks, (ii) how to assign individuals to solve these subtasks and (iii) how to put the whole system back together. As mentioned previously, most real–world problems are too complex to be hand–designed, thus, it is desirable to have a method to automatically decompose a complex problem into a set of overlapping or disjoint sub problems, and to assign one or more specialists to each of these subproblems. The remaining question is how to combine the outputs of these experts if the decomposition scheme is not known in advance.

In the previous section, we have discussed a number of combining methods, including methods to generate weights for the experts. In these methods, the weight-generating module is often disconnected from the training of the experts themselves. However, since these weights more or less represent the networks - one might consider these weights acting as the confidence levels of the experts - we argue that it is more practical to train the experts together with the weight-generating module.

A second way to create a modular ensemble of ANNs is to apply a new paradigm from evolutionary computation research, namely cooperative coevolution. Its main motivation is to generate a system of subcomponents that must cooperate for it to perform well. A number of authors have investigated this direction (Garcia-Pedrajas, Hervas-Martinez, and Ortiz-Boyer 2005; Khare, Yao, Sendhoff, Jin, and Wersing 2005). In both of these approaches, fixed size artificial neural networks are trained on some aspect of the problem. Another possible way to accommodate automatic task decomposition is to grow, prune and differentiate an ANN to a modularized structure (Ishikawa and Yoshino 1993). In a sense, this third method is a bridge between a single huge modularized ANN and an ensemble of ANN.

3.5.1 Mixture of experts

Jacobs (Jacobs, Jordan, Nowlan, and Hinton 1991; Jacobs, Jordan, and Barto 1991) has proposed a method of ensemble training and combination called the mixture of experts model. It is based on the Divide-and-Conquer principle. In their method, instead of assigning a set of combinational weights to the experts, an extra gating component is used to compute these weights dynamically from the inputs (Figure 3.3). This gating component is trained together with other experts through a specially tailored error function. The training localizes the experts into different subsets of the data while improving the system's performance. In the ME model, experts may be of any type, e.g. ANNs or C4.5 trees, but the gating is often an artificial neural network. Jordans and Jacobs (Jordan and Jacobs 1992; Jordan and Jacobs 1994) also extended the model to the so-called hierarchical mixture of experts, in which each component of the ME model is replaced with a Mixture of Experts. Since Jacobs' proposal of the ME model in 1991, there has been a huge volume of literature on the topic.

Some researchers (Alexandre, Campilho, and Kamel 2004; Jordan and Xu 1993; Kang and Oh 1997) derived a statistical understanding of the workings of the ME model. Waterhouse (Waterhouse, MacKay, and Robinson 1996; Waterhouse 1997) and

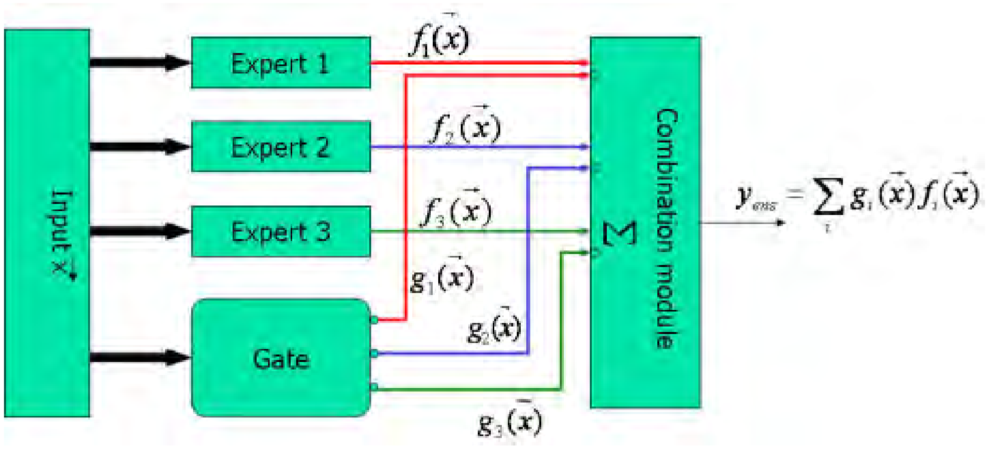


Figure 3.3: ME architecture

Moerland (Moerland 1997a) have applied the Bayesian framework to design and explain the ME model. According to their interpretation, the ME outputs can be considered as estimates for the a-posteriori probabilities of class membership (Moerland 1997a); thus, the Bayesian framework can be used to design the training error function (Bishop 1995) and estimate the parameters for the ME model (Waterhouse, MacKay, and Robinson 1996).

Besides the traditional ME model, a large number of variants have been put forward. Waterhouse and Cook (Waterhouse and Cook 1997) and Avnimelech and Cook (Avnimelech and Intrator 1999) proposed to combine ME with the boosting algorithm. They argued that boosting encourages classifiers to be experts on those patterns that the previous experts disagree on. As a result, it should be able to split the dataset into regions for the experts in the ME model, and thus ensure localization for experts. On the other hand, the dynamic gating function of the ME ensures a good combination of classifiers (Avnimelech and Intrator 1999). Tang et al (Tang, Heywood, and Shepherd 2002) tried to explicitly localize the experts by applying a self organizing map to partition the input space for the experts. Wan and Bone (Wan and Bone 1996) used a mixture of radial basis function networks to partition the input space into statistically correlated regions, and to learn the local covariation model of the data in each region.

3.5.2 Co-evolved ensembles

In recent years, the use of multi-population-based EAs has attracted attentions as the next level of parallelization. One architecture uses coevolution, based on the co-evolution of multiple species in real ecosystems. There are two types of coevolutionary systems, namely competitive (Rosin and Belew 1995; Rosin and Belew 1997) and cooperative coevolution (Potter 1997; Potter and De Jong 2000). In this thesis, we focus on applying cooperative coevolutionary systems to the mixture of experts model.

A coevolutionary algorithm is an evolutionary algorithm, usually involving multiple populations, in which the fitness of each individual depends on its interaction with individuals from other populations. In other words, the fitness of an individual results from the individual reciprocal interaction with other individuals. However, one may ask about the nature of this interaction. Waterhouse (Waterhouse 1997) defined the following classification of measures defined over a system:

1. **Objective measure**: A measurement of an individual is **objective**, if the measure considers that individual independently from any other individuals, apart from any scaling or normalization effect.

2. **Subjective measure**: A measurement of an individual is **subjective** if the measure is not objective

3. **Internal measure**: A measurement of an individual is **internal** if the measure influences the course of evolution in some way.

4. **External measure**: A measurement of an individual is **external** if the measure cannot influence the course of evolution in any way.

5. **Definition of a coevolutionary algorithm**: An EA that employs a subjective internal measure for fitness assessment.

However, from this definition, it is hard to distinguish the coevolutionary concept from the more traditional EA concept for a single-population. Is it essential to have multiple populations for a coevolutionary algorithm? In the literature of coevolution, this is still a muddy area. However it is widely accepted that a coevolutionary algorithm should involve more than one population, and the fitness evaluation of individuals in one population should be influenced by the fitness evaluation of other individuals from other populations.

However, what kind of interaction should an individual in one population have, for fitness evaluation, with indivduals from the other populations? Should it resemble the predator-prey relationship, where each species has to derive better and better strategies in order to survive (competitive coevolution)? Or should the problem be considered as a "big picture", in which each sub–population evolve to cope with one part of this whole picture (cooperative coevolution)? The answer depends on the nature of the problem. In the next section, we will limit our discussion to the cooperative coevolutionary algorithm, since it is what is needed in this thesis.

3.5.3 Cooperative co-evolutionary algorithms

Potter and De Jong (Potter 1997; Potter and De Jong 2000) developed a general framework to construct and evaluate cooperative coevolution. The architecture involves two or more isolated sub–populations interacting through a fitness evaluation mechanism. The species are encouraged to cooperate with one another by rewarding them based on how well they cooperate. These species in general are isolated from each other, i.e. they stay in different niches or sub–populations, and evolve without interaction with each other other than through the fitness measure.

Algorithms 3 and 4 describe the principal procedures for operating a CC. A primary driver of the system is the collaborative fitness evaluation module (Figure 3.4). It distinguishes CC from traditional multi-population EAs. An individual has to form a successful collaboration with representatives from other populations, in order to obtain higher fitness. This credit assignment scheme enforces collaboration between populations. However, a number of problems remains unsolved under this scheme.

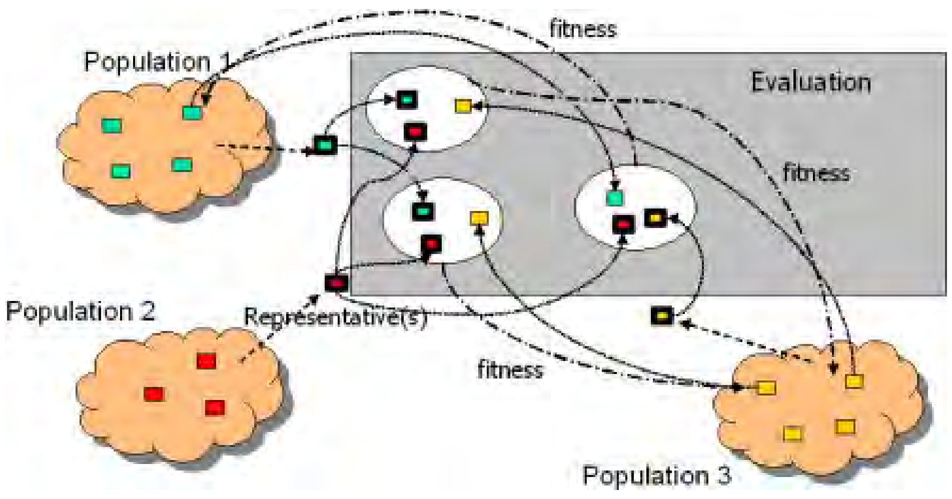


Figure 3.4: Cooperative coevolutionary algorithm

First, when sub–populations interact, it is possible that they may became self-similar. Especially, when the number of individuals is large, it is possible that sub–populations overlap. How to push the sub–populations apart from each other, so that they can find separable niches to live, is a key open research question. Potter et al (Potter 1997; Potter and De Jong 2000) suggested decreasing/increasing the number of sub–populations when the system stagnates. However, determining the stagnation of a system is a non-trivial question. A simplistic approach to detecting stagnation is to note when the best fitness does not improve. However, in some situations, the best fitness may stabilize while evolution is trying to escape from large valleys. Thus finding a better way to measure evolutionary activity is an important open research topic.

Second, suppose we have good populations occupying different niches of the search space. An important question is how, when bringing them together through collaboration, to define a suitable fitness function to model their interaction. The appropriate form of collaboration is largely dependent on the problem itself, and the way the task is decomposed into subtasks for each sub–population to specialize on.

Algorithm 3 Canonical cooperative coevolutionary algorithm
(Potter 1997)

```
1:  t = 0
2:  for each species S do
3:      initialize $P_t(S)$ to random individuals from a specified range.
4:  end for
5:  for each species S do
6:      evaluate the fitness of individuals in $P_t(S)$
7:  end for
8:  while the termination condition is false do
9:      for each species S do
10:         select individuals for reproduction from $P_t(S)$ based on fitness
11:         apply genetic operators to reproduction pool to produce offspring
12:         evaluate fitness of offspring
13:         replace members of $P_t(S)$ with offspring to produce $P_{t+1}(S)$
14:     end for
15:     t = t + 1
16: end while
```

Algorithm 4 Fitness evaluation of individuals from species S
(Potter 1997)

```
1: choose representatives from other species
2: for each individual $z$ from S requiring evaluation do
3:     form collaboration between $z$ and representatives from other species
4:     evaluate fitness of collaboration by applying it to target problem
5:     assign fitness of collaboration to $z$
6: end for
```

3.5.4 Cooperative coevolutionary neuro ensembles

Some recent work directly applies the concept of cooperative coevolution to the design of ensembles.

- **Cooperative coevolution with multiobjective optimization (CCMOP)** (Garcia-Pedrajas, Hervas-Martinez, and Ortiz-Boyer 2005): CCMOP has two levels of evolution. There is a network level, consisting of sub–populations which contributes ANNs to the ensemble. There is a separate ensemble level, which evolves the gate and corresponding combinational weights for the ensemble. Multiobjective Optimization algorithms are used at the network level to evolve a set of well-performed, regularized, cooperative and diverse ANNs; and at the ensemble level to create accurate and uncorrelated ensembles. The use of a MOP mechanism has its own advantages and disadvantages. The benefit of the method is that different criteria (i.e. objectives) are considered at the same time, which fits naturally with the previously noted bias/variance decomposition or accuracy/ambiguity decomposition. One can consider each of these conflicted terms as an objective for the MOP algorithm.

- **Cooperative coevolution with radial basis function ANNs (CCRBF)**(Khare, Yao, Sendhoff, Jin, and Wersing 2005): CCRBF also has two levels of evolution: a module level and an ensemble level. Both the module and the ensemble level make use of RBF networks. Each sub–population in the module level is designed to solve a particular sub-task of the whole problem, and the ensemble level provides the combination of these module networks. The main disadvantage of this method is its dependence on credit assignment. That is, the fitness of each module is decided by the contribution of the module to the whole system. To solve the problem of a fixed number of modules, Khare et al (Khare, Yao, Sendhoff, Jin, and Wersing 2005) used Potter's idea of adding and removing sub–populations whenever the system's fitness stagnates for a long period. Despite its problems, CCRBF has merit in that both the structures and the parameters of the modules and the ensemble can be evolved within the framework.

3.5.5 Regularization in the light of task decomposition and neuro ensembles

I have discussed a number of regularization methods in chapter 2. This section will look at regularization in task decomposition.

One can look at the biological nervous system as a single complex system, or as a set of separate networks corresponding to different functions, with some form of central network to direct and control these sub-networks. For example, there might be a network for walking and another for eating, together with a central network to tell the body when to walk and when to eat. The distinction between these two views are somewhat blurred. The same fuzzy line may be drawn between the fields of single ANNs and ensembles of ANNs, especially in the light of the Mixture of Experts (Jacobs, Jordan, Nowlan, and Hinton 1991), Dyn–Co (Hansen 2000) and Negative Correlation Learning (Liu, Yao, and Higuchi 2000). Brown (Brown and Wyatt 2003) has drawn an interesting connection between these methods. An ME can be considered as an intermediate link between single ANN and a modular system of ANNs. Dyn–Co with its special penalty term can alternate between ME and a closed kind of ensemble. NCL fills in the gap between Dyn–Co and a pure ensemble of ANNs (Brown and Wyatt 2003). In other words, the distinction between modular and ensemble systems is no longer clear. In fact, one can consider a neuro ensemble as a very large decentralized network, in which a specialized gating module (e.g. a switch) provides a link between the individual experts (Figure 3.5).

In the previous chapter, I mentioned that regularization can remove redundant connections in an artificial neural network. With careful design, it is possible to construct a regularization scheme organizing the connection patterns of the ANN into a modular-

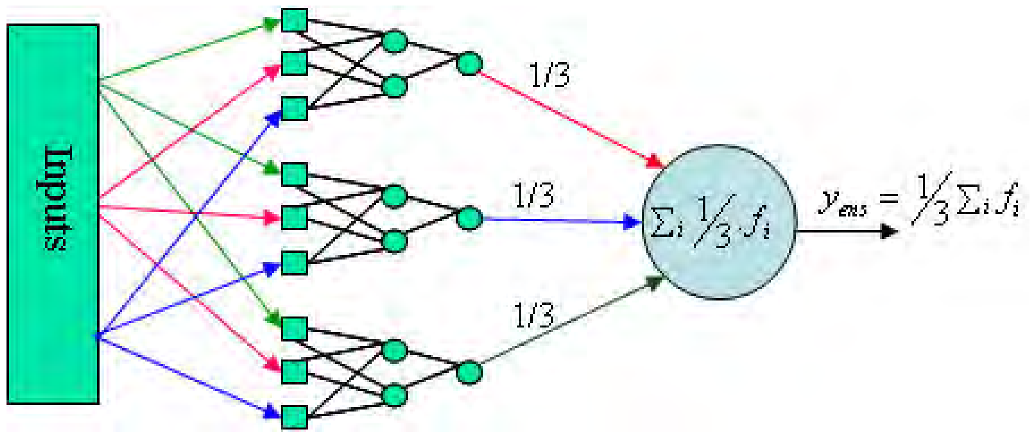

Inputs
1/3
1/3
1/3
$\Sigma_i \frac{1}{3} \cdot f_i$
$y_{ens} = \frac{1}{3}\Sigma_i f_i$

Figure 3.5: An ensemble with averaging combinational mechanism as a super ANN

ized structure.

3.5.5.1 Regularization as a method to modularize the system

Methods such as learning by forgetting have successfully created a form of modularized structure for task decomposition (Ishikawa and Yoshino 1993), by skeletonizing the ANN until it reflects the regularity in the training data. This method works well if the data possesses clear regularization characteristics. Although learning by forgetting has shown its merits in regularizing ANN, it has not been applied to the mixture of experts.

3.5.5.2 Regularization at the level of modules

At this level, regularization might help to better fit the individual ANN to the subproblem. It might reduce the redundancy and complexity so that the individual ANN is localized and generalized to the assigned subtask. This ensures that the subcomponents are different and localized. Waterhouse (Waterhouse 1997) applied weight decay to regularize the expert network in the Mixture of Experts model. Ramamurti and Ghosh (Ramamurti and Gosh 1996) used a penalty term to force the gate network in the Mixture of Experts model to produce softer splitting in the input space, so that each expert can observe more training data before deciding on its expertise. Since more training data entails less likelihood of overfitting, this allows experts to generalize better in their sub regions of input space. Abbass (Abbass 2003a) suggested using the complexity of the neuro ensemble (i.e. the number of synapses and/or hidden nodes) as one objective in the multiobjective optimization-based evolutionary neuro ensemble. Garcia-Predajas et al (Garcia-Pedrajas, Hervas-Martinez, and Ortiz-Boyer 2005) and Jin et al (Jin, Okabe, and Sendhoff 2004) applied regularization as an objective in the evolution of the individual ANN.

3.5.5.3 Regularization at the ensemble level

At this higher level, regularization can be used to prune out an overall individual neural network if it does not contribute to the ensemble. Another use of regularization at this level is to determine a suitable ensemble size. For example, Kondo et al (Kondo, Hatanaka, and Uosaki 2005) used the number of RBF networks as an objective in their MOP-based ensembles of RBF networks.

3.6 Conclusion

In this chapter I have reviewed the literature of neuro ensembles. It has been found that combining several neural networks can enhance the generalization of the whole system beyond the separate generalization ability of the individuals. The two main issues for neuro ensembles, accuracy and diversity, have also been discussed in detail. I have especially reflected on how these issues drive neuro ensemble research toward finding, selecting and combining a set of suitable individuals to form an ensemble. The discussion raises a number of important questions in the neuro ensemble literature, such as how to define, measure and promote diversity within the ensemble.

Finally, this chapter gives an overview of automatic problem decomposition. The review looks at how neuro ensembles and other mechanisms, such as cooperative coevolution or learning by forgetting, promote automatic decomposition. The review also describes a number of the background ideas for this thesis, including the mixture of experts model, the cooperative coevolution framework and the learning by forgetting regularization. In the literature of ANN, of neuro ensembles, and of evolutionary computation, each of these techniques has been separately analyzed in the context of automatic problem decomposition, however, a combination of them has not previously been introduced. In this thesis, I will show how these mechanisms can enhance each other, and how their integration produces an effective neuro ensemble approach to automatic decomposition of classification problems.

Chapter 4

An Empirical Study of Neuro Ensembles

This chapter is based on (Nguyen, Abbass, and McKay 2004).

The literature of neuro–ensemble has been growing over the last few years. Hence, I decided to investigate the effectiveness of some ensemble methods in order to gain insight and understanding of their properties and the way they function, and to construct the basic experimental framework for the later chapters 5,6,7. The investigation focuses on six methods, namely (i) a simple evolutionary computation with no ensemble (*i.e.* the best network found all over an evolutionary run is selected), (ii) a memetic, through back propagation, evolutionary computation with no ensemble, (iii) ensembles built using simple evolutionary computation, (iv) ensembles built using an island model, (v) ensembles built using evolutionary computation with negative correlation learning, and (vi) ensembles built using evolutionary Pareto multi–objective optimization. Besides exploring the previous methods, I also use two generalization improvement methods: noise injection and early stopping criteria. The first generalization method adds a Gaussian noise in the evolution with the aim of introducing stochastic noise to the fitness landscape hoping that the evolutionary method will escape a local optima. The second generalization improvement method uses some criteria, such as performance on a validation set, to select intermediate networks in the evolutionary run. These networks may not be the best found network in terms of training error but may have the potential to generalize better.

4.1 Methods

In this preliminary study, I investigate a number of state-of-the-art neuro ensembles in the literature. These include (i) the simple neuro ensemble whose members are taken straightforwardly from an evolved population, (ii) the island neuro ensemble, whose members are evolved separately in a set of islands, (iii) the negative correlation neuro ensemble, whose networks are evolved and trained using the negative correlation learning proposed by Yao and Liu (Liu and Yao 1999; Liu, Yao, and Higuchi 2000), and (iv) the multi–objective optimized neuro ensemble, whose networks are evolved using Pareto-based multi–objective optimization (Abbass and Deb 2003). Although there are a large number of evolutionary algorithms in the literature, in this thesis, I use the self-adaptive $(\mu + \lambda)$ evolutionary strategies (Schwefehm 1981), which have been successfully applied to evolve ANNs (Binner and Kendall 2002; Eggenberger 2000; Eggenberger 2001; Gabryel, Cpalka, and Rutkowski 2005; Greenwood 1997; Magoulas, Plagianakos, and Vrahatis 2001) to evolve a population of ANNs to be used in the ensemble. The evolved ANNs are fused together using three popular gating mechanisms in the neuro ensemble literature: majority voting, simple averaging and winner-take-all.

4.1.1 Self-adaptive evolutionary strategies $(\mu + \lambda)$

Evolutionary Strategies (ESs) (Back 1996; Rechenberg 1973; Rechenberg 1994) were invented for numerical optimization. Let $\vec{x}$ be an n dimensional solution vector $(x_1, x_2, \ldots, x_n)$ for problem $P1$ and $\vec{\sigma}$ be the corresponding step–length $(\sigma_1, \sigma_2, \ldots, \sigma_n)$. Let μ be the number of parents, where each parent z_k is the pair $(\vec{x}_k, \vec{\sigma}_k)$. In the first generation, μ parents are generated at random. In each subsequent generation, λ children are generated from the μ parents through recombination and mutation as follows: let $y_j = (\vec{x}_j, \vec{\sigma}_j)$ be the j^{th} child to be generated from the two parents $z_k = (\vec{x}_k, \vec{\sigma}_k)$ and $z_l = (\vec{x}_l, \vec{\sigma}_l)$. The child is generated either by discrete recombination or arithmetic recombination as follows: for each variable x_{ji} in $\vec{x}_j$, do $x_{ji} = x_{ki}$ or x_{li} for discrete recombination, or $x_{ji} = (x_{ki} + x_{li})/2$ for arithmetic recombination. The same recombination takes place for the step–size vectors σ. The child is then mutated as follows:

$$\vec{x'}_j = \vec{x}_k + \vec{R}_k \tag{4.1}$$

$$\vec{\sigma'}_j = \vec{\sigma}_k \tag{4.2}$$

where $\vec{R}_k$ is a random vector according to a Gaussian distribution with zero mean and standard deviation $\vec{\sigma}_k$; that is the probability, $Prob(R_{ki})$, of the random number $R_{ki} \in \vec{R}_k$ is

$$Prob(R_{ki}) = \frac{1}{\sqrt{2\pi}\sigma_{ki}} \exp^{-\frac{R_{ki}}{2\sigma_{ki}}^2} \tag{4.3}$$

There are two primary variants of ESs, based on the replacement mechanism. In the first variant, $ES(\mu + \lambda)$, λ children are generated from μ parents. The parents in the next generation are the μ best solutions among the total $\mu + \lambda$ solutions. In the second variant, $ES(\mu, \lambda)$, λ children are again generated from the μ parents, but now the parents in the next generation are the μ best solutions among the λ child solutions from the previous generation.

The step–size σ can vary during the evolutionary process. In this case, the algorithm is called a self–adaptive evolutionary strategy. The well–known one–fifth success rule is commonly used, where the step–size increases if the ratio of successful mutations (mutations which produced children better than their parents) to all mutations is greater than 1/5. Schwefehm (Schwefehm 1981), proposes log-normal self–adaptation, where a rotation angle is used to adapt the search towards coordinates which are likely to be correlated. Given τ and τ', the step size σ_j is perturbed as follows

$$\sigma'_{ji} = \sigma_{ji} \exp^{(\tau' N(0,1) + \tau N_j(0,1))} \tag{4.4}$$

where, the values of τ and τ' are suggested to be

$$\tau = \frac{1}{\sqrt{\left(2\sqrt{(n)}\right)}} \tag{4.5}$$

$$\tau' = \frac{1}{\sqrt{(2n)}} \tag{4.6}$$

The complete self–adaptive evolutionary strategy algorithm is depicted in Algorithm 5.

Algorithm 5 The self–adaptive evolutionary strategy $(\mu + \lambda)$.

```
 1: randomly generate $\mu$ parents, where each parent $z_k = (\vec{x}_k, \vec{\sigma}_k)$.
 2: set $\tau = \left(\sqrt{\left(2\sqrt{(n)}\right)}\right)^{-1}$ and $\tau' = \left(\sqrt{(2n)}\right)^{-1}$.
 3: repeat
 4:   repeat
 5:     select two parents $z_k = (\vec{x}_k, \vec{\sigma}_k)$ and $z_l = (\vec{x}_l, \vec{\sigma}_l)$ at random to generate child
        $\vec{y}_j = (\vec{x}_j, \vec{\sigma}_j)$.
 6:     discrete recombination: for each variable $x_{ji}$ and step size $\sigma_{ji}$ in $\vec{y}_j$, do ($x_{ji} =$
        $x_{ki}$ and $\sigma_{ji} = \sigma_{ki}$ ) or ($x_{ji} = x_{li}$ and $\sigma_{ji} = \sigma_{li}$).
 7:     mutation: For each $x_{ji}$ and step size $\sigma_{ji}$ in $\vec{y}_j$
            $x'_{ji} = x_{ji} + \sigma_{ji} N_j(0,1)$                    (4.7)
            $\sigma'_{ji} = \sigma_{ji} \exp(\tau' N(0,1) + \tau N_j(0,1))$   (4.8)
 8:     select the best $\mu$ individuals among all the $\mu + \lambda$ parents and children.
 9:   until $\lambda$ children are generated
10: until the halting criteria are satisfied
```

4.1.2 Simple neuro ensembles

An ensemble (Sharkey 1998) is a set of redundant networks, each by itself would “provide a solution to the same task”, i.e. predict the target output of an input vector. Thus, the output of an ensemble can be computed by feeding the input pattern to the individual neural nets, getting a prediction from each, and combining their outputs in a predefined fashion, namely a combination gate. The algorithm is presented in Algorithm 6.

Algorithm 6 General neuro ensemble algorithm

```
1: Input: Population $P$ of neural networks. A set of $M$ examples
2: for each example in the dataset do
3:     for each network in the ensemble do
4:         present the example to the network to get a predicted output
5:     end for
6:     using the combining gate to combine the outputs of all the networks in the en-
       semble to obtain a common value. This is the output of the ensemble for each
       example
7:     compare the ensemble's output with the actual value to compute the error rate
       (fitness).
8: end for
```

4.1.3 Island model and ensemble of diversified neural networks using island model

The Parallel Island model (Adamidis 1994; Back 1996; Lin, Yao, and Macleod 1996) is one type of diversity-promotion mechanisms often used for distributing the computation in an evolutionary algorithm. The main difference between the island model and a normal evolutionary computation method is the division of the population into sub–populations in a number of isolated islands. The sub–population occupying each island is updated independent from other sub–populations. Occasionally, members in neighboring islands are migrated from one island to another. In one variant of the algorithm, the best individuals from one island replace the worst individuals of neighboring islands. The connections between islands can be random or fixed in a sequential and cyclic manner. Algorithm 7 shows the neuro ensemble with island models.

Algorithm 7 The self-adaptive evolutionary strategy ($\mu' + \lambda'$) with island models.

```
1: for each island $p$ do
2:    randomly generate $\mu'$ parents
3:    set migration count to 0
4: end for
5: repeat
6:    repeat
7:       for each island $p$ do
8:          generate $\lambda'$ children from $\mu'$ parents.
9:          select the best $\mu'$ individuals among all the $\mu' + \lambda'$ parents and children.
10:         when the stopping criteria are satisfied, stop.
11:         increment migration count.
12:      end for
13:   until the migration interval is reached
14:   for each $p$ = 1 to $N_{island}$ do
15:      find the best individual $z_{p,best}$ in the top $\mu'$ individuals of island $p$
16:      find the worst individual $z_{p+1 modI,worst}$ in the top $\mu'$ individuals of island $p + 1\, mod\, N_{island}$
17:      exchange the individuals $z_{p,best}$ and $z_{p+1\, mod\, N_{island},worst}$ between islands $p$ and $p + 1\, mod\, N_{island}$
18:      reset migration count to 0.
19:   end for
20: until the halting criteria are satisfied
```

4.1.4 Negative correlation learning and ensemble of evolutionary negatively correlated neural networks

Negative Correlation Learning, proposed by Yao and Liu (Liu and Yao 1999; Liu, Yao, and Higuchi 2000), aims to maximize the difference among the members of the ensemble by adding a penalty term in each member network's mean squared error (MSE) function. These penalty terms are computed based on the correlation of the individuals. The networks are trained using normal back propagation (Haykin 1999).

Let M be the total number of members in the ensemble (often termed ensemble size). For each pattern $\vec{X}^p$ in the training set, the output of the ensemble F^p is computed as the average of the output $\hat{Y}^p(m)$, $m = 1 \ldots M$ of the M individual members.

$$F^p = \frac{1}{M}\sum_{m=1}^{M}\hat{Y}^p(m) \tag{4.9}$$

The error function for network m is defined by

$$Error^p(m) = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{2}(\hat{Y}^p(m) - Y^p)^2 + \frac{1}{P}\sum_{p=1}^{P}\lambda\Phi^p(m) \tag{4.10}$$

$\Phi^p(m)$ is called the penalty function of network m and pattern p. This represents the correlation between the networks.

$$\Phi^p(m) = (\hat{Y}^p(m) - F^p)\sum_{l \neq m}(\hat{Y}^p(l) - F^p) \tag{4.11}$$

The partial derivative of the error $Error^p(m)$ with respect to the output of network m is

$$\frac{\partial Error^p(m)}{\partial \hat{Y}^p(m)} = (\hat{Y}^p(m) - Y^p) - \lambda(\hat{Y}^p(m) - F^p) \tag{4.12}$$

In other words, the only difference between Negative Correlation Learning and BP is an additional penalty term of $\lambda(\hat{Y}^p(m) - F^p)$ to the error function.

4.1.5 Ensemble of diversified neural networks using multi–objective optimization

Pareto-based methods are potential diversity mechanisms (Abbass and Deb 2003; Abbass 2003b; Kuncheva 2003b). In this set of experiments, I investigate the feasibility of using Pareto ranking as a source of diversity for the ensemble. In the multi–objective optimization literature, diversity has been also a main concern. As a result, a number of methods (Coello Coello 2002; Van Veldhuizen 1999; Zitzler 1999) were invented to solve this problem. A state-of-the-art method, Strength Pareto Evolutionary Algorithm, was proposed by Zitzler (Zitzler 1999), where the Pareto ranking algorithm was introduced as a diversity mechanism. The algorithm takes into account the crowding (based on a special dominance/nondominance ratio) of the population in different regions and promotes diversity by avoiding crowded regions. This ranking method results in a wide spread of solutions on the Pareto frontier. To be fair in the comparison, Zitzler's Pareto ranking algorithm is implemented together with the Evolutionary Strategy algorithm introduced earlier.

Algorithm 8 shows Zitzler's Pareto ranking method, in which an elitist set P' is created to hold the accounted non-dominated solutions in the whole population P. For each member i of the elitist set, a strength function (i.e. crowding fitness) is defined as the number of individuals in the whole population P being dominated by i, normalized by N, the total number of individuals in the whole population. Hence, the fitness of a non-dominated individual is always less than 1. Next, for each dominated member j in the population P, a fitness function is computed as the sum of the strength of the elitist members that dominate j. To ensure that the non-dominated solutions have better chance (i.e. lower crowding fitness) to survive, a value of 1 is added to the fitness of the dominated solutions.

The Pareto ensemble method uses Pareto ranking as the diversity promoting mechanism. It is similar to the evolutionary ensemble algorithm, the only difference lies

Algorithm 8 Pareto ranking algorithm.

```
1: Input: Population $P$. Population size N
2: create an empty pool $P'$
3: copy all non-dominated individuals into $P'$
4: for each non-dominated solution $z$ in $P'$ do
5:     define a strength function
           $$s(z) = \frac{N_1}{N+1} \qquad (4.13)$$
       where $N_1$ is the number of individuals in P that are dominated by $z$. Assign a
       fitness $f(z) = s(z)$ to item $z$
6: end for
7: for each dominated solution in $P - P'$ do
8:     assign a fitness $f(z) = 1 + \sum_{t \in P', t\, dominates\, z} s(t)$
9: end for
10: rank the population in ascending order of the fitness
```

in the use of Pareto ranking as the mechanism for selection. The training set is split into two subsets and the multi–objective problem is then to minimize the training error on each subset.

4.1.6 Other over–fitting avoidance methods

4.1.6.1 Noise injection

A random Gaussian noise with zero mean and a standard deviation of σ is added to the training error (fitness) of the ensemble in each generation. This disturbance gives networks with bad training fitness more chances to survive because often the ones with less ability to memorize the training data are the ones which can generalize well. This noise addition also has the effect of changing the bias/variance distribution of each individual network (Breiman 2000).

4.1.6.2 Stopping criteria

In this chapter, I investigate three different criteria for choosing the optimal generation for the ensemble. The first criterion is to use the population of the last generation to form the ensemble; this is widely used in the literature (Liu, Yao, and Higuchi 2000; Yao and Liu 1996; Yao and Liu 1998a). The second criterion is to form the ensemble in each generation and evaluate it on the validation set. The ensemble corresponding to the minimum validation error is selected (Algorithm 10). The third criterion is to form the ensemble from the members of the population achieving the minimum average fitness on the validation set (Algorithm 11).

Algorithm 9 Without early stopping

```
 1: for $gen = 0$ to the maximum number of generations do
 2:     run any techniques.
 3: end for
 4: form the ensemble $E$ by combining the individuals of the population
```

Algorithm 10 Minimum error of ensemble on validation set

```
 1: initialize ensemble $E$ to empty, initialize the minimum validation fitness $f_{min}$ to 1.
 2: for $gen = 0$ to the maximum number of generations do
 3:     conduct the selected method
 4:     form the ensemble $E_{current}$ of all the individual in the population
 5:     compute the ensemble fitness $f$ on the validation set
 6:     if $f < f_{min}$ then
 7:         set $E = E_{current}$
 8:     end if
 9: end for
10: compute and report the testing error of $E$
```

Algorithm 11 Minimum average error of population on validation set

```
 1: initialize the population $P'$ to empty, initialize the minimum average validation fit-
    ness $f_{min}$ to 1.
 2: for $gen = 0$ to the maximum number of generations do
 3:     conduct the selected method
 4:     compute the validation fitness of each individuals in the population and compute
        the average $f_{avg}$
 5:     if $f_{avg} < f_{min}$ then
 6:         set $P' = P_\mu$
 7:     end if
 8: end for
 9: form the ensemble $E$ out of the population $P'$
10: compute and report the testing error of $E$
```

4.1.7 Gating methods for combining members in the ensemble

As discussed in Chapter 3, there are many ways to combine the individual ANNs to form the ensemble. In the experiments of this thesis, I exploit the three most popular combination (gating) methods in the ensemble literature: majority voting, average and winner-take-all.

4.1.8 Diversity measure

In this initial investigation, I apply a simple diversity measure based on the average hamming distance of the predicted outputs of all possible pairs of individuals in the networks. In other words, let N be the number of instances in the training set, M the number of classifiers in the ensemble, and $\overrightarrow{y}_m = y^i_m, i = 1..N, m = 1..M$, where each entry y^i_m corresponds to the binary predicted class of example i in the training set for classifier m. Then, the diversity measure for the two binary vectors $\overrightarrow{y}_j$ and $\overrightarrow{y}_k$ is

$$d_{jk} = \sum_{i=1}^{N} (a(y^i_j, y^i_k)) \tag{4.14}$$

where

$$a(y^i_j, y^i_k) = \begin{cases} 0 & if\, y^i_j = y^i_k \\ 1 & otherwise \end{cases}$$

Finally, the diversity measure of the ensemble is

$$D_{ens} = \sum_{p=1}^{M} \sum_{q=1, q \neq p}^{M} (d_{pq}) \tag{4.15}$$

4.2 An empirical study of neuro ensembles

4.2.1 Hypotheses

The experiments are designed around seven hypotheses; these are:

Hypothesis 1: Combination of local search through learning and evolution improves the generalization ability

The first set of experiments is designed to analyze the effect of combining learning with evolution. In the literature of evolutionary artificial neural networks, it is still debatable if combining learning and evolution will be better than each alone.

Hypothesis 2: An ensemble of neural networks perform better than individual networks

The second set of experiments is used to verify the claim by many researchers (Jimenez and Walsh 1998; Liu and Yao 1997; Liu and Yao 1999; Liu, Yao, and Higuchi 2000; Rosen 1996; Sharkey 1998; Tumer and Ghosh 1996; Ueda and Nakano 1996; Yao and Liu 1996; Yao and Liu 1997; Yao and Liu 1998a; Zhou, Wu, and Tang 2002) that an ensemble of classifiers performs better than individual ones.

Hypothesis 3: Different combination methods yield similar results

In this set of experiments, the members of the ensemble are combined using different combination gates. I verify the effect of these different combination gates.

Hypothesis 4: Noise injection improves the generalization ability of the ensemble

Noise could be injected in different levels of the system. In this experiment, I choose to inject a random Gaussian noise to training fitness of the networks in the ensemble with the hope that it could reduce the pressure of evolution to over–fit the networks.

This disturbance gives networks with bad training fitness more chances to survive because often the ones with less ability to memorize the training data are the ones which can generalize well. This noise addition also has the effect of changing the bias/variance distribution of each individual network (Breiman 2000).

Hypothesis 5: Early stopping is useful in avoiding over–fitting

The next set of experiments is motivated by the observation frequently seen in the neural network literature that early stopping could reduce the chance of over–fiting and thus improving the network's generalization ability. However, in the literature of ensemble, it is not yet clear if different stopping criteria affect the generalization ability of the ensemble. Therefore, this experiment is designed to explore possible use of early stopping and possible stopping criteria.

Hypothesis 6: Useful diversity is an important feature of ensemble techniques

Diversity, as seen in the literature review, is a very interesting and important issue in the ensemble. However, most of the papers in the field do not report the actual diversity of the members in their ensembles. In this initial investigation, three different diversity mechanisms, namely negative correlation learning, island model and the Pareto-based bootstrapping model, are examined in two aspects: accuracy and diversity.

The aim here is not to claim if any method outperforms the other. Since there are many different elements that could affect greatly the performance of a method, and since the aim of this investigation is to gain an understanding on different mechanisms, I choose a common evolutionary method with a common parameter setting for all three mechanisms. There is no guarantee that the chosen evolutionary method and parameters are optimized for each mechanism.

Hypothesis 7: Architecture complexity is important in the problem of over–fitting

	Number of Instances	Number of Attributes	Continuous Attributes	Discrete Attributes
(i) small datasets				
Breast Cancer	699	9	9	
Australian credit card	690	14	6	8
Diabetes	768	8	8	
Liver Disorder	345	6	6	
Tic Tac Toe Endgame	958	9		9

Table 4.1: Five datasets from UCI machine learning repository database

It is widely noted that architecture complexity plays an important role in the generalization ability of neural networks. Small simple networks have less chance to memorize the training data and, thus, have more bias and less variance. On the other hand, large and complex networks could capture more of the data but run into the problem of over–fitting the data. Hence, the problem of finding a suitable architecture for each problem set is a difficult problem. In this experiment, the architecture is varied by changing the number of hidden units in the hidden layer of the feed–forward neural networks.

4.2.2 Datasets

In these preliminary experiments, the hypotheses are tested on five standard datasets (Table 4.1) taken from the UCI Machine Learning Repository (Appdendix B) : the breast cancer Wisconsin, the Australian credit card assessment, the diabetes, the liver disorder and the Tic Tac Toe Endgame.

In the experiments, I use ten-fold cross validation for each dataset. A dataset is divided into ten subsets using the stratified sampling method. For each fold, the distribution of data in the test/validation/training sets is 1/1/8. For each fold, a different prefixed random seed is used to generate the required random numbers (e.g. network weights, noise, crossover and mutation rates) for the method. The method is trained using the training set, stopped by one of the different criteria, and the ensemble obtained by combining the population at the stopping point is tested on the test set. The final result is the average of the ten-fold results.

4.2.3 Parameters

Each individual feed–forward neural network consists of a single hidden layer with 6 hidden units (except in the architecture complexity experiments where 1 to 6 hidden units are used). The evolution is run for a maximum of 1000 generations, unless an early stopping criterion is satisfied, with a population of (μ = 20 and λ = 80) for all methods except the island model, where 10 islands of (μ' = 2 and λ' = 8) are used. The migration interval in the island model method is set to 10 generations.

For the negative correlation learning method, I use a learning rate of 0.1 for BP with 10 epochs of learning. For the negative correlation penalty coefficient, I also test different values ranging from 0.1 to 0.5. The initial experiment showed that 0.2 is a suitable value for most of the datasets, and thus, it is used in the NCL based experiments.

For the ensemble using MOP, the two objectives are the training errors of the neural network on two disjointed subsets of the training data.

Finally, for the ensemble method using noise distortion, a Gaussian noise of zero mean and 0.01 standard deviation is added to the computed training fitness of each individual neural network.

4.2.4 Results and analysis

The experiments, as discussed above, are carried out on six group of methods: (i) the simple evolutionary computation, (ii) the memetic (through back propagation) evolutionary computation, (iii) the ensemble of simple evolutionary computation, (iv) the ensemble of evolutionary neural networks using the island model, (v) the ensemble of evolutionary negatively correlated neural networks, and (vi) the ensemble of evolutionary Pareto based neural networks. Each group consists of a with-noise and a without-noise experiments. The means and standard deviations of the testing errors in these twelve experiments are recorded based on the three stopping criteria. Obviously, there are a number of different dimensions to be considered such as the use of local search, the effect of noise injection, the various architecture complexity as measured by the number of hidden units, and different diversity mechanisms. The large amount of experimental results are grouped under the seven hypotheses. To verify a hypothesis, an error value is computed across all other dimensions.

4.2.4.1 Local search helps evolution to find better solutions

Table 4.2 shows the results of the first set of experiments, in which a simple EC approach is used to evolve, with and without BP learning, the best neural network to classify the five datasets. These results show that integrating BP to EC (Algorithm 12) improves the performance of three out of the five datasets. The performance enhancement of the Liver Disorder (16%) and Australian Credit Card (11.2%) datasets and the performance worsening of Breast Cancer Wisconsin dataset (13.3%) are statistically significant. The results in Table 4.2 implies that BP learning, on the average, helps EC to find better solutions for the classification problem.

Algorithm 12 An evolutionary algorithm with back propagation

```
1: generate an initial population
2: evaluate the individuals in the population
3: repeat
4:     select parents based on their fitness.
5:     apply operators such as crossover and mutation to the parents to create offsprings.
6:     apply BP to the offsprings
7:     evaluate offsprings.
8:     form the population for the next generation from the offsprings and/or the current
       population.
9: until halting criterion is met
```

4.2.4.2 Ensemble performs better than individuals

The second set of experiments compares the simple ensemble with the best individual (without ensemble) approaches. The results are summarized in Table 4.3. The

Dataset	EC	EC + BP learning
Breast Cancer Wisconsin	**0.030(0.017)**	0.034(0.021)
Australian Credit Card	0.150(0.043)	**0.134(0.041)**
Diabetes	0.242 (0.055)	**0.231 (0.044)**
Liver Disorder	0.374 (0.054)	**0.318 (0.085)**
Tic Tac Toe End Games	0.289 (0.041)	0.291 (0.049)

Table 4.2: Memetic effect - means and standard deviations of the error rates of evolutionary ANN with and without BP learning. Bold face in each row indicates statistically significant result at confidence level 95%.

results show that the simple ensemble method performs better than the best individual in three out of the five datasets, especially the improvement in the Breast Cancer and the Liver Disorder datasets are quite significant (10.1% and 10.7%). In the only dataset (Tic Tac Toe) where the best individual outperforms the ensemble, the improvement is actually quite small (3.6%). As a result, ensemble of neural networks on average performs better than the best neural network in the population.

Dataset	Best Individual	Simple Ensemble
Breast Cancer Wisconsin	0.030(0.017)	**0.027 (0.014)**
Australian Credit Card	0.150(0.043)	**0.140 (0.052)**
Diabetes	0.242(0.055)	0.242(0.056)
Liver Disorder	0.374(0.054)	**0.336(0.075)**
Tic Tac Toe End Games	**0.289(0.041)**	0.299(0.052)

Table 4.3: Mean (and standard deviation) error rates of the best individual and the whole ensemble. Bold face in each row indicates statistically significant result at confidence level 95%.

4.2.4.3 Combination gates perform similarly

Table 4.4 shows that the performance of different gates is similar since each gate outperforms the other two on at most 2 out of 5 datasets. This suggests that on the average, the three different gates perform the same.

4.2.4.4 Noise injection improves generalization

Table 4.5 shows the results of the comparison of with and without noise disturbance in each dataset. Except in the Breast Cancer Wisconsin dataset, where three out of five methods perform better without noise addition, the remaining four datasets exhibit

Dataset	Majority voting	Averaging	Winner-take-all
Breast Cancer Wisconsin	0.027(0.014)	0.027(0.017)	0.027(0.017)
Australian Credit Card	0.140(0.050)	**0.133(0.048)**	0.139(0.050)
Diabetes	0.238(0.043)	0.235(0.041)	**0.232(0.037)**
Liver Disorder	0.286(0.038)	0.286(0.038)	0.286(0.038)
Tic Tac Toe End Games	**0.212(0.042)**	**0.212(0.042)**	0.223(0.044)

Table 4.4: Mean (and standard deviation) error rates of ensembles with three different combining gates: (i) majority voting, (ii) averaging, (iii) winner-take-all. Bold face in each row indicates statistically significant result at confidence level 95%.

a clear preference for noise injection. Also, from Table 4.5, noise intrusion is favorable for Simple Ensemble and Ensembles with Multi-Objective Optimization methods where four out of five datasets display preference on noise.

In summary, the results from Table 4.5 suggest that the performance could be improved by introducing a Gaussian noise into the fitness of the neural network during fitness computation. Especially, noise addition works well with Simple Ensemble and Ensembles with MOP methods. A possible reason is that noise addition reduces the pressure of selecting overly-fit network using the training fitness.

4.2.4.5 Early stopping is useful in avoiding over–fitting

Table 4.6 displays the results of the set of experiments to verify hypothesis 5: early stopping performs better than stopping at the maximum number of generations. As seen from the outcomes, using two early stopping criteria: minimum of the ensemble on validation and minimum average of population fitness on validation outperforms (4 of 5 datasets per method show preference) the last generation criterion for "Simple Ensemble", "Ensembles with Island model", and "Ensembles with Negative Correlation Learning". Only "Ensembles with MOP" shows a slight favor of the last generation criterion. The results support the hypothesis that early stopping is beneficial.

4.2.4.6 Useful diversity is an important feature of ensemble techniques

Table 4.7 displays the results of classification of five datasets using four different ensemble methods: "Simple Ensemble", "Ensembles with Island model", "Ensem-

Breast Cancer		
Method	Without Noise	With Noise
EC	**0.030(0.017)**	0.032(0.016)
SE	**0.027(0.014)**	0.028(0.019)
IsE	0.027(0.017)	**0.025(0.017)**
NCLE	**0.027(0.017)**	0.030(0.017)
MOPE	0.030(0.018)	**0.027(0.017)**

Australian Credit Card Assessment		
Method	Without Noise	With Noise
EC	0.150(0.043)	**0.142(0.048)**
SE	0.140(0.052)	**0.137(0.044)**
IsE	**0.133(0.048)**	0.149(0.043)
NCLE	0.144(0.040)	**0.137(0.044)**
MOPE	**0.140(0.050)**	0.144(0.057)

Diabetes		
Method	Without Noise	With Noise
EC	0.242(0.055)	**0.235(0.045)**
SE	0.242(0.056)	**0.240(0.060)**
IsE	**0.232(0.037)**	0.246(0.055)
NCLE	0.236(0.050)	**0.226(0.057)**
MOPE	0.236(0.051)	**0.226(0.042)**

Liver Disorder		
Method	Without Noise	With Noise
EC	0.374(0.054)	**0.321(0.054)**
SE	0.336(0.075)	**0.315(0.068)**
IsE	0.350(0.085)	**0.303(0.080)**
NCLE	**0.286(0.038)**	0.295(0.054)
MOPE	0.326(0.074)	**0.298(0.073)**

Tic Tac Toe End Games		
Method	Without Noise	With Noise
EC	**0.289(0.041)**	0.322(0.047)
Se	0.299(0.052)	**0.281(0.062)**
IsE	0.311(0.034)	**0.277(0.042)**
NCLE	0.212(0.042)	**0.195(0.039)**
MOPE	0.278(0.042)	**0.272(0.026)**

Table 4.5: Effect of noise injection for each dataset (SE=Simple Ensemble, IsE=Ensembles with island, NCLE=Ensemble w/NCL, MOPE=Ensemble w/MOP). Presented results are means and standard deviations of error rates. Bold face in each row indicates statistically significant result at confidence level 95%.

bles with NCL", and "Ensembles with MOP". The values show that the Island model performs the best in three datasets and Negative Correlation Learning performs the best in the remaining two datasets.

The next question is whether these diversity mechanisms do generate diversity. Diversity measure as described in section 4.1.8 is used. Table 4.8 shows the diversity measures of five datasets across different stopping criteria and diversity methods. Moreover, figures 4.1, 4.2, 4.3, 4.4, and 4.5 plot the Diversity over the generations of the four different diversity mechanisms for the five Datasets.

Looking across different ensemble methods with different diversity promoting mechanisms, ensembles generated with the Island model show very good diversity level for most cases, while "Simple Ensembles" and "Ensembles with multi–objective Optimization" are reasonably good in promoting diversity. However, surprisingly, "En-

Dataset	Last generation	Minimum on validation	minimum average
Simple Ensemble			
Breast Cancer Wisconsin	0.035(0.028)	**0.027(0.014)**	0.030(0.021)
Australian Credit Card	0.149(0.056)	**0.140(0.059)**	**0.140(0.052)**
Diabetes	0.243(0.047)	**0.242(0.056)**	0.248(0.039)
Liver Disorder	0.338(0.066)	0.347(0.071)	**0.336(0.075)**
Tic Tac Toe End Games	**0.299(0.052)**	0.308(0.049)	0.301(0.053)
Ensembles with Island model			
Breast Cancer Wisconsin	0.219(0.246)	0.030(0.012)	**0.027(0.017)**
Australian Credit Card	0.150(0.057)	**0.133(0.048)**	0.142(0.051)
Diabetes	0.244(0.041)	**0.232(0.037)**	0.235(0.038)
Liver Disorder	0.422(0.071)	**0.350(0.085)**	0.368(0.064)
Tic Tac Toe End Games	**0.311(0.034)**	0.317(0.047)	0.323(0.034)
Ensembles with Negative Correlation Learning			
Breast Cancer Wisconsin	0.044(0.028)	0.028(0.019)	**0.027(0.017)**
Australian Credit Card	0.169(0.048)	0.147(0.044)	**0.144(0.040)**
Diabetes	0.264(0.058)	**0.236(0.050)**	0.249(0.051)
Liver Disorder	**0.286(0.038)**	0.292(0.071)	0.295(0.071)
Tic Tac Toe End Games	0.217(0.038)	0.224(0.040)	**0.212(0.042)**
Ensembles with multi–objective Optimization			
Breast Cancer Wisconsin	0.032(0.020)	0.032(0.017)	**0.030(0.018)**
Australian Credit Card	**0.140(0.050)**	0.146(0.052)	0.142(0.044)
Diabetes	0.248(0.057)	**0.236(0.045)**	0.246(0.052)
Liver Disorder	**0.326(0.074)**	0.347(0.078)	0.347(0.080)
Tic Tac Toe End Games	**0.278(0.042)**	0.293(0.052)	0.284(0.050)

Table 4.6: Comparison of different stopping criteria (i) last generation (ii) minimum on validation (iii) minimum average of population on validation. Presented results are means and standard deviations of error rates. Bold face indicates statistically significant result at confidence level 95%.

Breast Cancer Wisconsin			
Simple Ensemble	Ensemble + Island	Ensemble + NCL	Ensemble + MOP
0.027(0.014)	**0.027(0.017)**	0.027(0.017)	0.030(0.018)
Australian Credit Card			
Simple Ensemble	Ensemble + Island	Ensemble + NCL	Ensemble + MOP
0.140(0.052)	**0.133(0.048)**	0.144(0.040)	0.140(0.050)
Diabetes			
Simple Ensemble	Ensemble + Island	Ensemble + NCL	Ensemble + MOP
0.242(0.056)	**0.232(0.037)**	0.236(0.050)	0.236(0.045)
Liver Disorder			
Simple Ensemble	Ensemble + Island	Ensemble + NCL	Ensemble + MOP
0.336(0.075)	0.350(0.085)	**0.286(0.038)**	0.326(0.074)
Tic Tac Toe End Games			
Simple Ensemble	Ensemble + Island	Ensemble + NCL	Ensemble + MOP
0.299(0.052)	0.311(0.034)	**0.212(0.042)**	0.278(0.042)

Table 4.7: Mean (and standard deviation) error rates of ensembles with and without diversity promotion. Bold face in each row indicates statistically significant result at confidence level 95%.

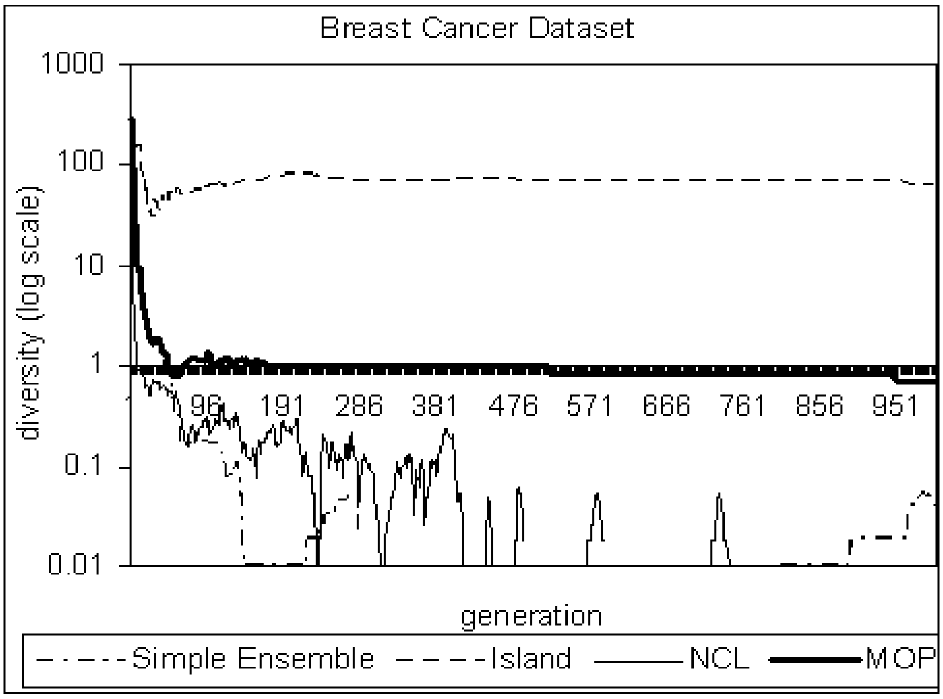


Figure 4.1: Diversity measure of the Breast Cancer Wisconsin Dataset

sembles with Negative Correlation Learning" shows a very poor diversity measure. This fact is supported by the analysis of McKay and Abbass (McKay and Abbass 2001) which showed that Negative Correlation Learning does not promote diversity among networks.

Considering the four lines in each figure(4.1, 4.2, 4.3, 4.4, and 4.5), which correspond to four different methods, "Ensembles with Island model" and "Ensembles with MOP" have the best diversity curves (top two lines in each graph), and the "Ensembles with Negative Correlation" curves are very low for all dataset. These results agrees with the remark that the former two methods produce higher diversity among members than the other methods. Also, it confirms that Negative Correlation Learning produces very small diversity among the members.

The diversity analysis and performance analysis across different methods raises an interesting fact: the Island model yields best performance and the highest diversity level, while the Negative Correlation Learning model also performs well but has very small diversity. Also, the simple ensemble and the ensemble with MOP have quite good diversity levels but with poor performance. In conclusion, I have not found any useful

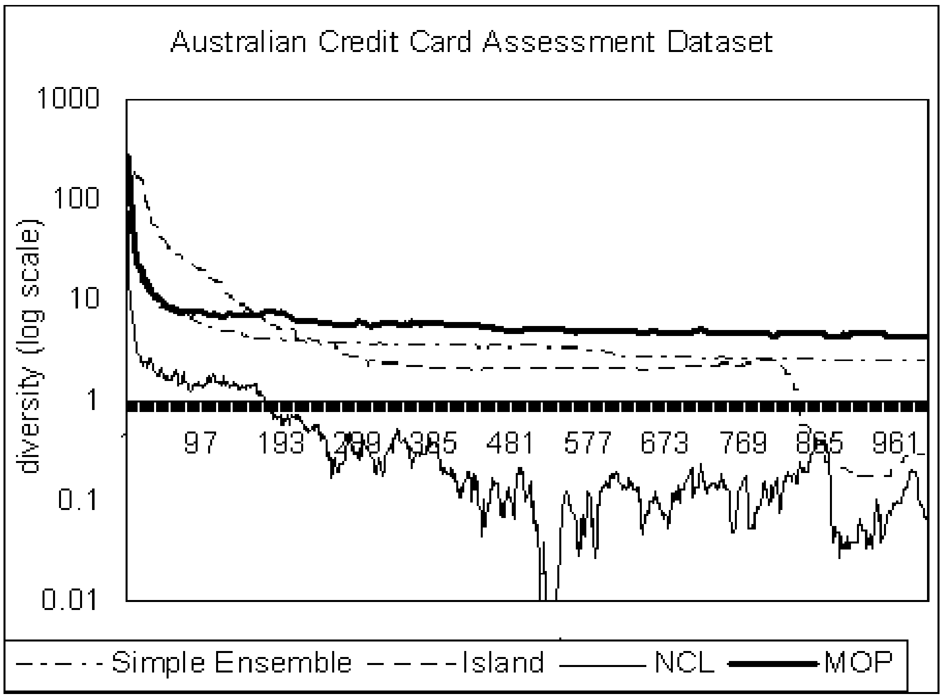


Figure 4.2: Diversity measure of the Australian Credit Card Assessment Dataset

connection between diversity and performance when building an ensemble.

One possible reason is that unhealthy diversity could deteriorate the ensemble performance. Let us revisit Krogh's equation:

$$E = \overline{E} - \overline{A} \tag{4.16}$$

Promoting diversity among members of the ensemble will change the Ambiguity term $\overline{A}$ which contains all the correlations among the individual classifiers. However, large diversity in the ensemble may greatly reduce the average accuracy of the ensemble $\overline{E}$. Thus, it is possible that unhealthy diversity, which reduces $\overline{E}$ much faster than $\overline{A}$, can degrade the performance of the ensemble. In the future, more experiments may be required to fully analyze the relationship of diversity and accuracy.

Diversity across different stopping criteria Looking across the columns in Table 4.8, the minimum on validation (column 2) stopping criterion shows reasonably high level of diversity across three stopping criteria. The second best is the minimum average of

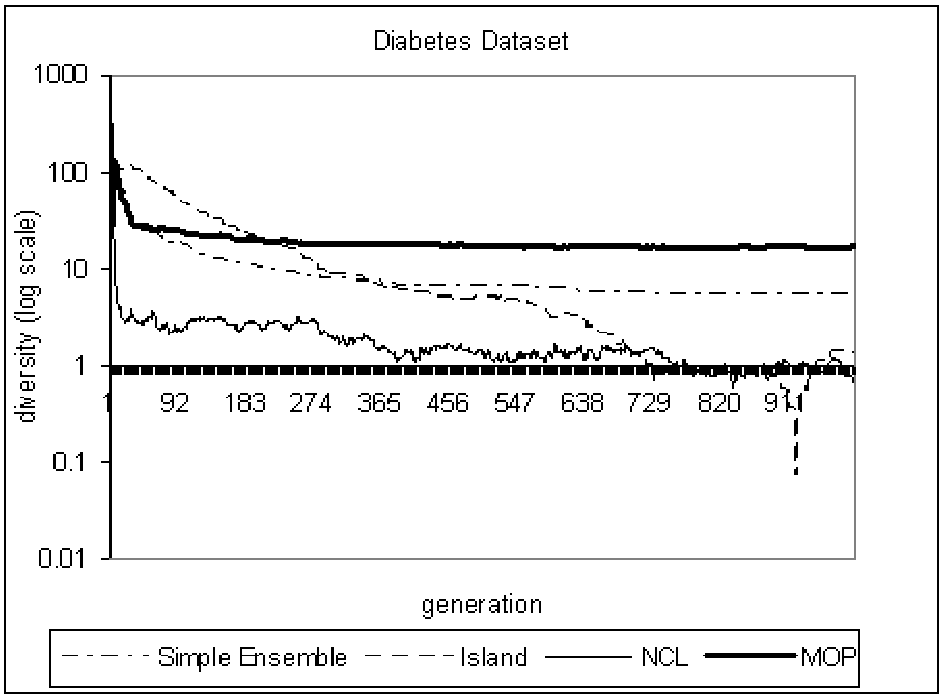


Figure 4.3: Diversity measure of the Diabetes Dataset

population on validation (column 3) and the lowest is the last generation criterion. This result, together with the previous conclusion that early stopping performs better than last generation, suggests a connection between high diversity and the good performance of early stopping.

Also, Figures 4.1, 4.2, 4.3, 4.4, and 4.5 show the diversity decaying over time (Note: since the diversity measure decreases quickly to zero, the graph is plotted in log scale). From the figures, since diversity tends to reduce over time, an early stopping often implies higher diversity among the individuals.

4.2.4.7 Architecture complexity is important

This final set of experiments is designed to testify any connection of the neural network's architecture complexity and the performance of the ensemble. Table 4.9 presents the results of different number of hidden units on the performance of the ensemble for three datasets: the Australian credit card, the Diabetes and the Tic Tac Toe End Games. Looking across the three datasets, it is obvious that there are no particular fixed

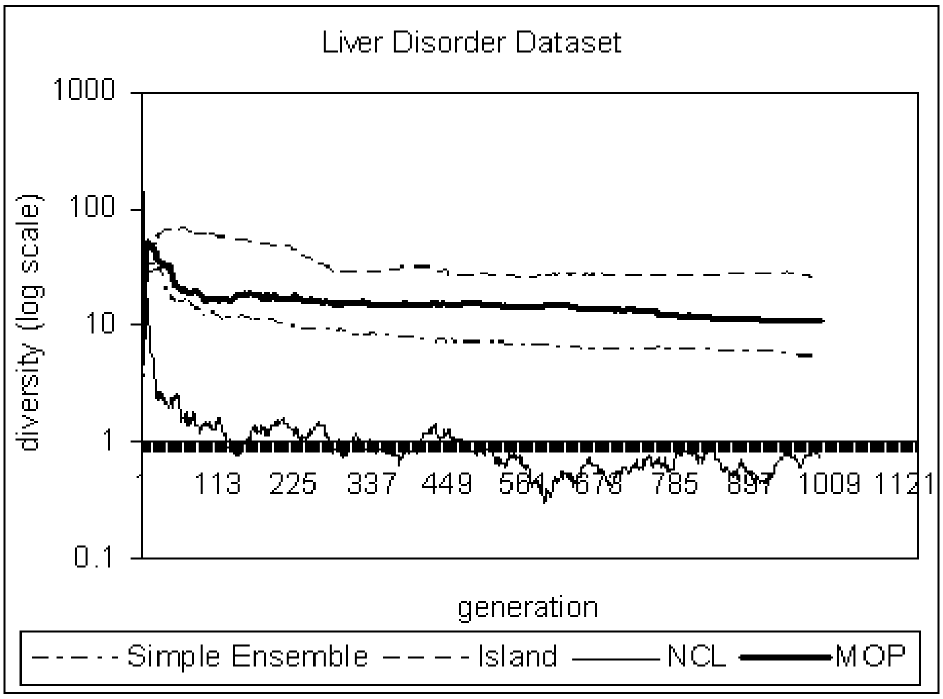


Figure 4.4: Diversity measure of the Liver Dataset

number of hidden units that can perform best for all datasets with all methods.

Looking down the columns in Table 4.9, the island model appears to perform comparably well across various number of hidden units while NCL favors less complex networks and MOP prefers more complex ones.

The results in Table 4.9 back the claim that architecture complexity of the neural network is an important factor to be considered when designing an ensemble method. Moreover, and what is more interesting, the level of network complexity depends on the diversity promoting mechanism.

4.3 Conclusion

In this chapter, I have presented an overview of some state-of-the-art methods in the field of neuro ensemble. A number of experiments were conducted to investigate various aspects of methods used to build neuro ensembles. The findings verify some points raised by other researchers in the field, and also raise a number of interesting and

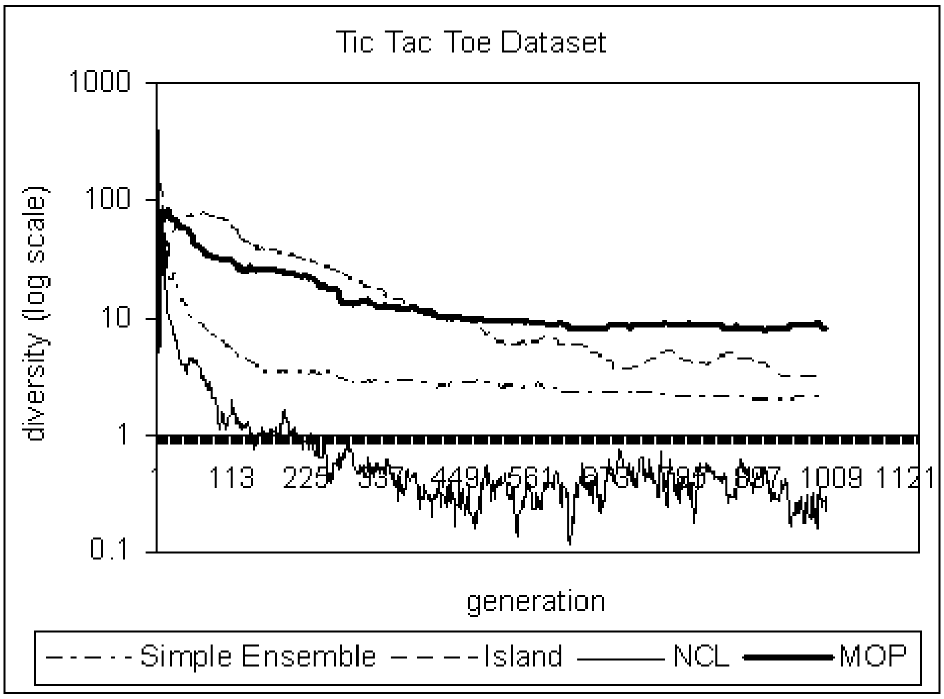


Figure 4.5: Diversity measure of the Tic Tac Toe End Games Dataset

potential directions for later chapters.

The key points from the experimental analysis could be summarized as : (1) different combination gates have little effect on the performance of the ensemble, (2) the diversity level maintained by negative correlation learning is poor, which was suggested by McKay and Abbass's analysis (McKay and Abbass 2001), (3) combining individuals (ensemble) improves the performance of the system.

Some interesting open directions are raised from the experimental results. Those are: (1) local search does help evolution to find better solutions, this is applicable for both ensemble and individual-based methods, (2) noise injection shows interesting effects on performance enhancement, though the improvement is not yet clear from the initial experiments, (3) early stopping is useful in enhancing generalization, where using different minimum values in the validation fitness, such as the fitness of the ensemble or the average fitness of the population, can avoid the networks to over–fit the training data, (4) architecture complexity of the neural networks is an important factor to be considered when designing the ensemble of EANNs, (5) the connection between diversity and per-

Objective	last generation	min ensemble on validation	min average of population
Breast Cancer Wiscosin			
Simple Ensemble	0.039(0.118)	**28.042(20.423)**	3.203(2.730)
Emsemble Island	**59.912(57.168)**	**91.205(41.182)**	**22.853(20.531)**
Emsemble NCL	0.000(0.000)	**116.646(139.802)**	1.542(1.342)
Emsemble MOP	0.721(1.727)	**56.438(63.541)**	4.009(4.815)
Australian Credit Card Assessment			
Simple Ensemble	2.381(2.537)	**78.141(42.795)**	**20.452(22.799)**
Emsemble Island	0.285(0.857)	**85.813(58.874)**	**16.838(15.241)**
Emsemble NCL	0.060(0.151)	**31.515(83.813)**	3.600(4.638)
Emsemble MOP	4.478(3.769)	**48.713(30.609)**	12.235(9.887)
Diabetes			
Simple Ensemble	5.167(3.165)	**51.202(27.868)**	**18.474(19.099)**
Emsemble Island	1.282(3.600)	**64.383(41.064)**	**28.635(22.122)**
Emsemble NCL	0.743(0.528)	6.460(9.215)	3.552(2.457)
Emsemble MOP	16.985(10.014)	**34.303(15.857)**	**25.267(10.204)**
Liver Disorder			
Simple Ensemble	5.360(4.938)	**23.387(17.791)**	**16.083(19.161)**
Emsemble Island	**24.479(12.618)**	**62.334(12.630)**	**44.370(19.247)**
Emsemble NCL	0.734(0.938)	2.415(1.185)	1.681(1.208)
Emsemble MOP	10.717(7.366)	**44.920(36.224)**	**23.461(13.196)**
Tic Tac Toe End Games			
Simple Ensemble	2.048(2.970)	8.608(13.847)	3.836(4.208)
Emsemble Island	2.977(4.559)	**59.845(32.055)**	**37.045(46.293)**
Emsemble NCL	0.215(0.209)	3.856(2.879)	1.815(2.509)
Emsemble MOP	8.173(10.266)	**39.862(25.063)**	**18.239(19.456)**

Table 4.8: Diversity measure of five Datasets across different diversity mechanisms and stopping criteria. The bold values are the high diversity measure (members more disagree) in contrast to plain numbers which correspond to little difference among ensemble members.

formance of the ensemble is still a myth to be verified, this is supported by the results of Kuncheva's experiments (Kuncheva 2003b).

Australian Credit Card Assessment			
Hiddens	Island model	NCL	MOP
1	**0.134(0.054)**	**0.133(0.044)**	0.152(0.035)
2	0.147(0.047)	**0.133(0.050)**	0.146(0.040)
3	0.139(0.052)	0.139(0.049)	0.143(0.045)
4	0.150(0.045)	0.144(0.048)	0.142(0.049)
5	0.149(0.051)	**0.133(0.039)**	0.144(0.057)
6	**0.133(0.048)**	0.144(0.040)	**0.139(0.048)**
Diabetes			
Hiddens	Island model	NCL	MOP
1	0.246(0.046)	**0.225(0.056)**	0.246(0.055)
2	**0.236(0.050)**	**0.229(0.062)**	**0.235(0.074)**
3	0.244(0.061)	**0.226(0.069)**	0.247(0.044)
4	**0.234(0.058)**	0.230(0.063)	0.242(0.059)
5	**0.232(0.042)**	0.231(0.068)	**0.238(0.056)**
6	**0.232(0.037)**	0.236(0.050)	**0.236(0.051)**
Tic Tac Toe			
Hiddens	Island model	NCL	MOP
1	0.318(0.015)	0.305(0.057)	0.311(0.050)
2	0.322(0.044)	0.309(0.045)	**0.261(0.045)**
3	0.312(0.046)	0.286(0.033)	0.313(0.055)
4	**0.300(0.048)**	**0.249(0.034)**	0.297(0.030)
5	0.313(0.037)	**0.241(0.071)**	0.302(0.047)
6	**0.311(0.034)**	**0.212(0.042)**	**0.279(0.042)**

Table 4.9: Mean (and standard deviation) error rates of ensembles with various architecture complexity in terms of the number of hidden units ("Hiddens" column). Bold face in each row indicates statistically significant result at confidence level 95%.

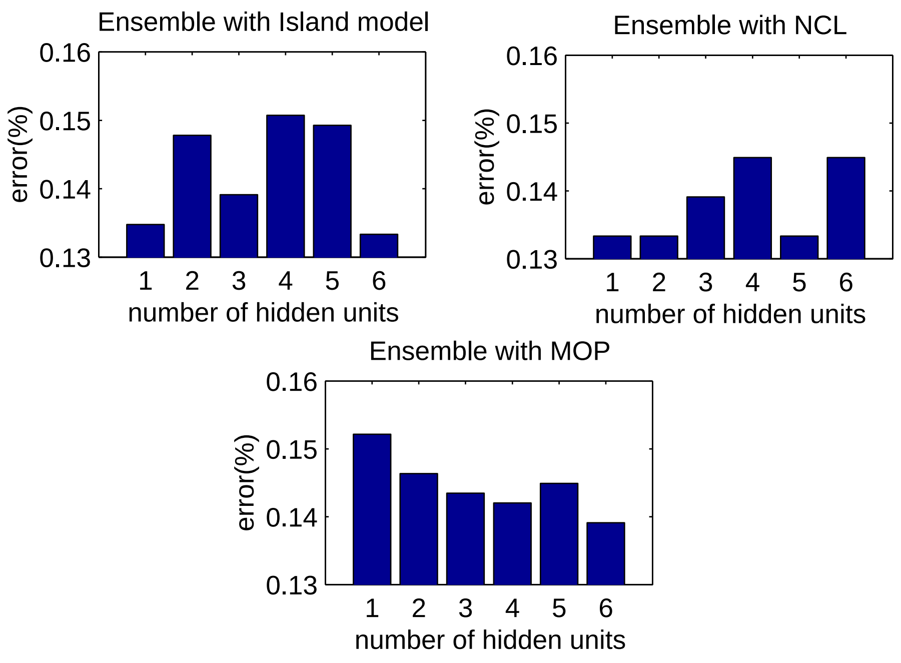


Figure 4.6: Error rate vs number of hidden units for the Australian Credit Card Assessment dataset

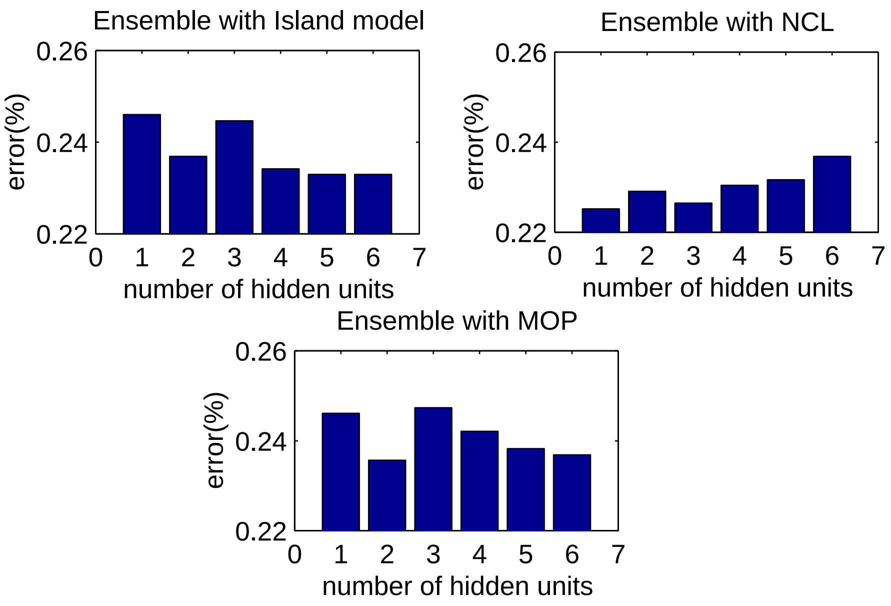


Figure 4.7: Error rate vs number of hidden units for the Diabetes dataset

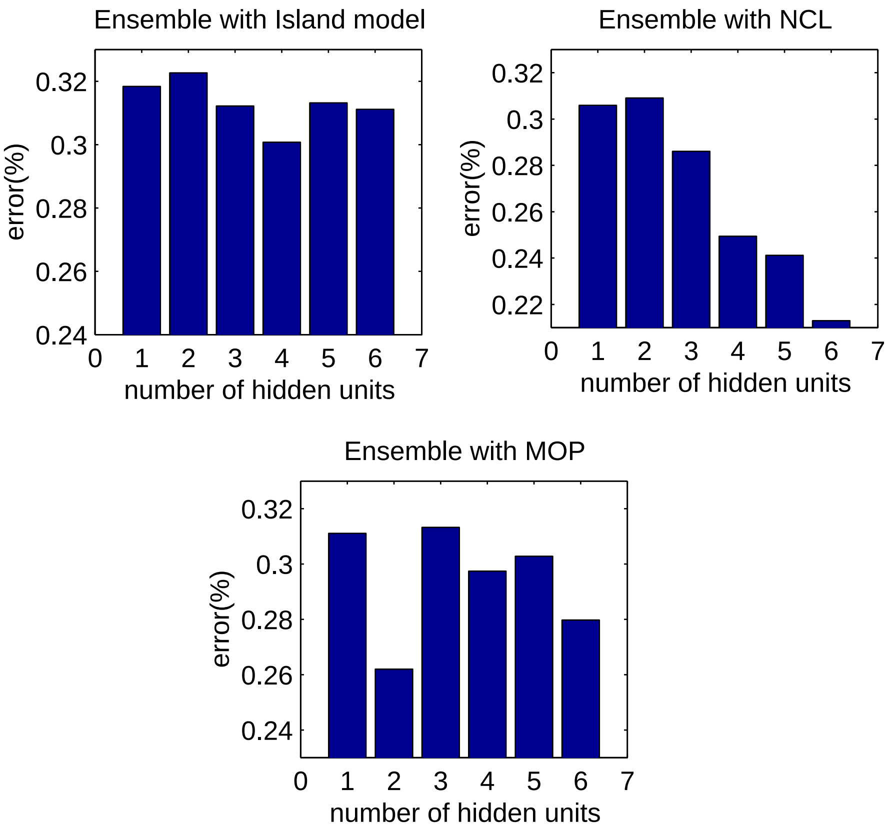


Figure 4.8: Error rate vs number of hidden units for the Tic Tac Toe End Games dataset

Chapter 5

Cooperative Coevolutionary Mixture of Experts

In chapter 2, I have explained why evolutionary computation algorithms are efficient heuristic methods to solve computationally difficult problems. ECs are often more effective than gradient based methods (such as BP) in difficult, rugged, multimodal and/or discontinuous search spaces. In difficult problems, they tend to find better solutions in the search space. In chapter 3, the concept of cooperative coevolution (CC) is discussed as a framework to evolve diverse and modularized neuro ensembles. The biologically inspired collaboration of modules allows the ensemble to emerge as a well-structured system.

In this chapter, I will first investigate a popular modularization method in the ensemble literature, the so called mixture of experts (ME) model. The main advantage of the ME model over other ensemble methods is its ability to automatically decompose problems by using a special architecture and training scheme, which forces the experts to specialize on different regions of the input space while maintaining their cooperation (Jacobs, Jordan, Nowlan, and Hinton 1991).

More importantly, I will introduce a novel method that combines the ME model with the CC mechanism. CC allows for incorporating evolutionary computation into back propagation-based ME. This can enhance the learning method when compared with simple gradient descent search. Moreover, the CC framework has an inherently decomposition nature, so we expect it to assist the ME rather than hinder it. Last but not least, CC as proposed by Potter (Potter 1997; Potter and De Jong 2000) has the potential to adapt the number of species to the problem, by increasing or decreasing the number of sub–populations when the system stagnates. This will be helpful in designing ensembles for problems where the desirable ensemble size is not known a priori, so that the size can be an emergent property rather than predefined by human experts.

The ME model imposes an external diversity force, to drive the components into different local regions, and therefore ensures diversity between the sub–populations. This diversity is useful because it is an emergent property, while ensuring performance, rather than being imposed by the users. Moreover, the ME model is localized in the sense that each expert is responsible for a sub region of the input space. This suits very well the localization requirement of the CC framework, in which each species preferably occupies a local niche in the environment.

5.1 Mixture of experts

Jacobs et al (Jacobs, Jordan, Nowlan, and Hinton 1991) introduced the ME model based on the principle of divide and conquer. ME is a method to derive and combine a series of localized models through a dynamic system. Considering a learning problem, it is desirable that each expert accumulate and specialize its expertise in a subset of the input space. An extra gating network, also looking at the input vectors, can decide which expert is in charge of which input instance.

If the problem has a distinct natural decomposition, it would be possible to derive such a decomposed system by hand. However, in most real-world problems, we either know little about the problem, or the problem is too complex, for us to have a clear vision on how to decompose it by hand. Thus, it is desirable to have a method to automatically decompose a complex problem into a set of overlapping/non-overlapping sub problems, and to assign one or more specialists (i.e. experts, learning machines) to each of these subproblems.

The remaining issue is how to combine the outputs of these experts if the scheme for decomposition is not known in advance. In the previous chapter, I have discussed a number of gating mechanisms, including methods to generate dynamic weights for the experts. However, even in these methods, this weight-generation module is often independent of the training of the experts themselves. However, the weights in effect represent the networks - one might consider the weights as the confidence levels of the experts. Thus it is more practical to train the experts together with the weight-generation module, rather than separately.

5.2 Architecture

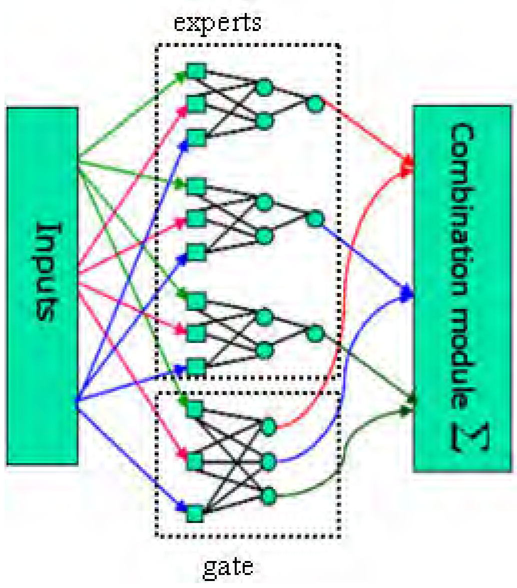


Figure 5.1: ME architecture

In the original form (Jacobs, Jordan, Nowlan, and Hinton 1991), an ME consists of a number of experts joined by a gate (Figure 5.1). All have access to the input space X. Each component can be any type of classifier, including simple feed–

forward multi–layer neural networks as in this thesis. Given a dataset $(\vec{x^i}, \vec{d^i}), i = 1..N$, where $\vec{x^i}$ and $\vec{d^i}$ are the attributes and targets of case i, the output $y(\vec{x^i})$ of an ME is the weighted average of the individual experts' outputs $y_m(\vec{x^i}), m = 1..M$ with the weights $g_m(\vec{x^i}), m = 1..M$ produced by the gate network.

$$y(\vec{x^i}) = \sum_{m=1}^{M} g_m(\vec{x^i}) y_m(\vec{x^i}) \tag{5.1}$$

5.3 A probabilistic interpretation of the ME model

The gate output could be considered as the probability that an expert m is selected (5.4.1). To ensure that $g_m(\vec{x^i})$ satisfies the axioms of probability theory (i.e. $g_m(\vec{x^i}) \geqslant 0$ and $\sum_{m=1}^{M} g_m(\vec{x^i}) = 1$), one could apply a soft-max function on the raw outputs $z_m(\vec{x^i})$ of the gate. For example, applying softmax on a linear gate network produces the following outputs

$$g_m(\vec{x^i}) = \frac{\exp(z_m(\vec{x^i}))}{\sum_{j=1}^{M} \exp(z_j(\vec{x^i}))} \tag{5.2}$$

Another way to view a ME output is as the conditional density of target $\vec{d^i}$ given input $\vec{x^i}$. If the output $y_m(\vec{x^i})$ of each expert m, $m = 1..M$, could be viewed as the conditional density of target $\vec{d^i}$ given input $\vec{x^i}$ for that expert $y_m(\vec{x^i}) = \phi(\vec{d^i}|\vec{x^i})$, then the internal function of each expert can be expressed as

$$p(\vec{d^i}|\vec{x^i}) = \sum_{m=1}^{M} g_m(\vec{x^i}) \phi_m(\vec{d^i}|\vec{x^i}) \tag{5.3}$$

Given the above equation, one can apply different probability distributions in place of $\phi_m(\vec{d^i}|\vec{x^i})$. In the following section, I will describe a number of distributions, and the corresponding error functions, used in the literature of ME.

5.4 Training the ME model

There are many ways to look at the ME model – as a simple ensemble model or as a probabilistic model. Depending on the point of view, different motivations arise for different error functions. While a residual cancelling error function is sufficient for a simple ensemble, more localized and competitive error functions are preferred for the ME model to take full advantage of its special architecture. Furthermore, if a ME is considered in the light of the Bayesian framework, where the gate is estimating the apriori probability that an expert is chosen for each pattern, Gaussian and Bernoulli distributions could be used to design a suitable error function.

5.4.1 Error functions

Bishop has devoted a whole chapter in his book (Bishop 1995) to discussing various error functions for regression and classification problems. Since the scope of this thesis is binary classification, this section summarizes a number of possible error functions for ME in binary classification problems. Because a binary class has only two possible values, 0 and 1, to simplify the notation, $\vec{d^i}$ is replaced with d^i, and g^i_m and y^i_m are used interchangeably with $g_m(\vec{x^i})$ and $y_m(\vec{x^i})$.

5.4.1.1 Residual cancelling error function

Originally, Jacobs et al (Jacobs, Jordan, and Barto 1991) assumed the final output of the whole system was a simple linear combination of the experts' outputs. The gating network generated the weights, which indicated the contribution of the localized experts on the overall input space.

$$E^i = \|d^i - \sum_m g^i_m y^i_m\|^2 \tag{5.4}$$

where d^i is the target of case i and y^i_m is the output of expert m on case i. g^i_m is the generated combination weight corresponding to expert m and case i. This error function compares the target to a blend of experts' outputs. Thus, "to minimize the error, each local expert must make its output cancel the residual error that is left by the combined effects of all the other experts" (Jacobs, Jordan, Nowlan, and Hinton 1991). Although this strong coupling forces the experts to cooperate amicably, it does not encourage expert diversity. Nevertheless, this is the error function used in most methods in the ensemble literature. To overcome this similarity problem, one could inject diversity into the system as discussed in the previous chapters.

5.4.1.2 Competitive and localized error function

Instead of injecting diversity in the system, a simpler approach is to repair the error function so as to encourage competition among the networks for each input case. Jacobs et al introduced such a competitive and localized error function in (Jacobs, Jordan, Nowlan, and Hinton 1991). Instead of letting each expert cancel out the residual error, the experts are forced to consider the whole output vector. "As a result, the goal of a local expert on a given training case is not directly affected by the weights within other local experts" (Jacobs, Jordan, Nowlan, and Hinton 1991). The system tends to assign each expert to each training case, and hence, localize the experts.

$$E^i = \sum_i g^i_m \|d^i - y^i_m\|^2 \tag{5.5}$$

5.4.1.3 Probabilistic error function

Another way to view the system is through the Bayesian framework. An ME model could be interpreted as an input conditional mixture model, with the data assumed to be generated from a series of processes. Each data point $(\vec{x}^i, d^i)$ is assumed to be generated by a process m. There exists a probability distribution $P(Z)$ such that each z^i_m is the decision to use process m for case i. With this probabilistic interpretation, each expert in the ME system models a process, while the gating network models the decision probability distribution $P(Z)$ (Waterhouse 1997). The likelihood of target d^i given inputs

$\vec{x}^i$ and parameters $\{\vec{w}_1, \vec{w}_2, ..., \vec{w}_M, \vec{v}\}$ is modelled by the ME as follow:

$$P(d^i|\vec{x}^i, \vec{w}_1, \vec{w}_2, ..., \vec{w}_M, \vec{v}) = \sum_{m=1}^{M} P(m|\vec{x}^i, \vec{v}) P(d^i|\vec{x}^i, \vec{w}_m, m) \tag{5.6}$$

where $\{\vec{w}_1, \vec{w}_2, ..., \vec{w}_M, \vec{v}\}$ is the parameter space, $\vec{w}_m$ are the weights of an expert network m, and $\vec{v}$ the weights of the gating network. $P(m|\vec{x}^i, \vec{v})$ is the conditional probability of the gating network to select expert m. $P(d^i|\vec{x}^i, \vec{w}_m, m)$ is the conditional probability of expert m to produce output d^i. This conditional probability is the underlying mechanism of the expert network. By varying this function, one can achieve the desired behavior.

Jacobs et al (Jacobs, Jordan, Nowlan, and Hinton 1991) used the Gaussian distribution.

$$P(d^i|\vec{x}^i, \vec{w}_m, \sigma_m) = \frac{1}{\sqrt{2\pi\sigma_m^2}} \exp(-\frac{1}{\sigma_m^2}(d^i - y_m^i)^2) \tag{5.7}$$

where σ_m^2 is the variance for expert m.

However, for a classification, it is more appropriate to use a cross entropy between the target d^i and the outputs of the experts y_m^i (Bishop 1995):

$$\phi_m(d^i|\vec{x}^i, \vec{w}_m) \equiv P(d^i|\vec{x}^i, \vec{w}_m) = (y_m^i)^{d^i} + (1 - y_m^i)^{(1-d^i)} \tag{5.8}$$

The overall error function E for the ME model is defined as the negative log likelihood:

$$E = -\sum_i log \sum_{m=1}^{M} P(m|\vec{x}^i, \vec{v}) P(d^i|\vec{x}^i, \vec{w}_m, m) \tag{5.9}$$

A more useful error function for each pattern i is defined as

$$E^i = -log \sum_{m=1}^{M} P(m|\vec{x}^i, \vec{v}) P(d^i|\vec{x}^i, \vec{w}_m, m) \tag{5.10}$$

5.4.2 Training ME with back propagation

I will use BP as a local search operator in this thesis. The modified BP for the ME method is presented in Algorithm 13 as follow:

5.5 Cooperative coevolutionary mixture of experts

CC is different from a traditional multi-population EA in its fitness evaluation scheme. In chapter 3, I have identified two important issues with respect to the CC method, namely the diversity between the sub–populations and the scheme to combine the components to form the whole system. ME can address both issues. If the fitness evaluation component - at the heart of the CC - is implemented with a ME, then ensuring sub–population diversity (arising from the localization nature of ME) is interwoven nicely with evolving a collaboration scheme (through the gate mechanism of the ME). Thus the integration of CC and ME (namely Cooperative Coevolutionary Mixture of Expert or CCME) overcomes both the inter-diversity and the collaboration problems of the traditional CC.

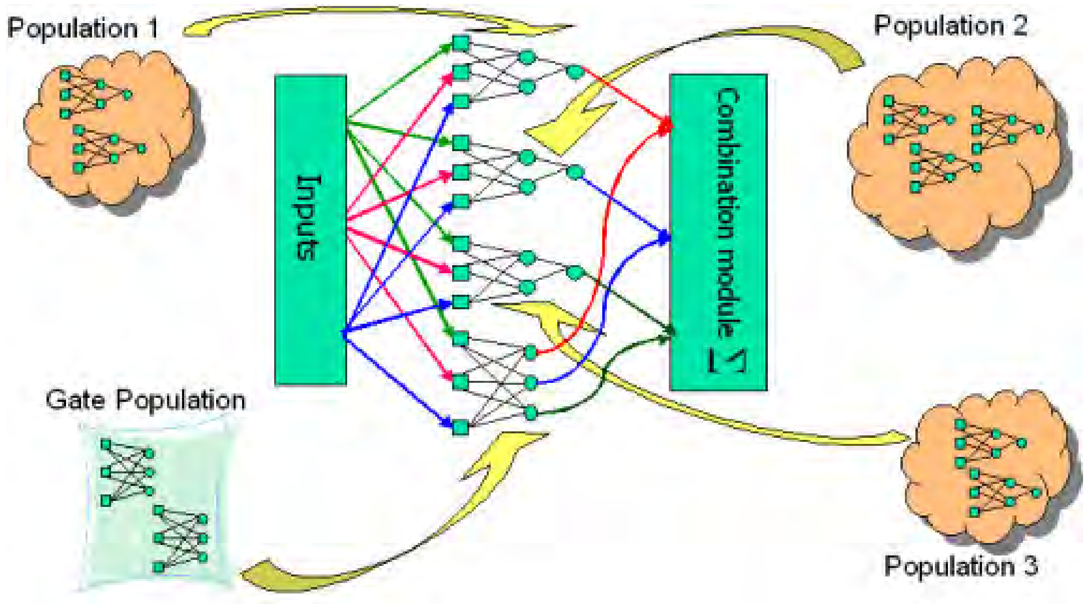


Figure 5.2: CCME architecture

Figures 5.2 illustrates the architecture of the CCME model. The CCME model

Algorithm 13 ME learning with back propagation

```
1:  randomly generate the weights for the ME
2:  for $epoch = 0$ to the maximum number of epoches do
3:      for each pattern $i$ do
4:          forward $\{\vec{x}^i, d^i\}$ through the gate network. Output $z^i_m = \vec{v}_m\vec{x}^i, m = 1..M$
5:          apply softmax: $g^i_m = \exp(z^i_m)/\sum_{j=1}^{M}\exp(z^i_j)$.
6:          forward $\{\vec{x}^i, d^i\}$ through each expert $m$, compute the conditional density
            $\phi_m(d^i|\vec{x}^i)$ (or $\phi^i_m$).
7:          compute the posterior probability $h^i_m = g^i_m\phi^i_m/\sum_{j=1}^{M} g^i_j\phi^i_j$
8:          compute the error terms $\delta^i_{g,m}$ for the gate outputs using the following partial
            derivative of the error function $E^i$ (see Appendix A for a derivation of the for-
            mula)
                $$\delta^i_{g,m} \equiv \frac{\partial E^i}{\partial z^i_m} = g^i_m - h^i_m \qquad (5.11)$$
9:          pass $\delta^i_{g,m}$ backward, and update weights $\vec{v}$ of the gate network
10:         compute the error term $\delta^i_m$ of the output of expert $m$ using the following sets of
            equations
                $$\delta^i_m \equiv \frac{\partial E^i}{\partial a^i_m} = \frac{\partial E^i}{\partial y^i_m}\frac{\partial y^i_m}{\partial a^i_m} \qquad (5.12)$$
            where $y^i_m = g(a^i_m)$ (e.g. $y^i_m = \frac{1}{(1+\exp(-a^i_m))}$ if sigmoid activation is used) and
            $a^i_m = \vec{w}_m\vec{x}^i$. Since
                $$\frac{\partial E^i}{\partial y^i_m} = -\frac{g^i_m}{\sum_{j=1}^{M} g^i_j\phi^i_j}\frac{\partial \phi^i_m}{\partial y^i_m} \qquad (5.13)$$
            Therefore,
                $$\delta^i_m = -\frac{g^i_m}{\sum_{j=1}^{M} g^i_j\phi^i_j}\frac{\partial \phi^i_m}{\partial a^i_m} \qquad (5.14)$$
            For a multi–nomial conditional density $\phi^i_m = (y^i_m)^{d^i} + (1 - y^i_m)^{(1-d^i)}$ (binary
            classification) and a sigmoid activation, the error term $\delta^i_m$ is simplified to:
                $$\delta^i_m = h^i_m(y^i_m - d^i) \qquad (5.15)$$
11:         pass $\delta^i_m$ backward and update weights $\vec{w}_m$ of expert $m$.
12:     end for
13: end for
```

consists of a number of sub–populations, each of which can be considered as a supply pool for one of the experts in the ME model, so that the experts do not depend entirely on a single initialization. Since in this thesis, the experts are ANNs, these sub–populations are pools of ANNs. To evaluate the fitness of an individual k in a sub–population j, it is required, in the CC scheme, that the other sub–populations contribute a representative (or perhaps representatives) to the evaluation module. This representative can be the best individual in the sub–population (greedy algorithm) or a randomly selected one (or in the case of multiple representatives, both might be used). The evaluation module will assemble individual k together with these representatives to form a complete system (e.g. a full ensemble) and evaluate it. The performance of the complete system is assigned back to the individual i as its fitness. In other words, the fitness of an individual measures how well it works with the other populations, hence the term cooperative.

In our model, the evaluation module is implemented as an ME. In other words, to evaluate the fitness of an individual k, k is assembled with the representatives from the other sub–populations to form an ME. This ME is then trained using BP for a number of epochs to evaluate the fitness. In the literature of EC, one often finds two different modes combining training and evolution. In the Lamarkian model, the lifetime training is encoded back to the individual while in Darwinian evolution it is not (though the Baldwin effect may lead to the gradual incorporation of this learned behavior back into the genotype even in Darwinian evolution) (Cantu-Paz and Kamath 2005). Each of these types has its own advantages and weakness.

I have found that Lamarkian is more efficient in my model. The reason is that unlike traditional CC, where training is just a local search, training is important here for turning the ensemble into a true ME by localizing its components, and hence finding suitable local niches for the species. Without encoding the changes through local search, as in Darwinian evolution, back to the individual, the system has to depend on the weaker and slower evolutionary pressure instead of this strong force to drive the experts to different areas of the search space. Consequently, Lamarkian is more efficient and therefore implemented throughout this thesis.

Another design concern is whether elitism should be used, and if so, how. I have tested the algorithm without elitism. The system takes much longer time to converge and often good solutions are lost. Especially with the Lamarkian ME, in which the training weights are coded back to the individual, champions are important. Let us take a simplified example of two sub–populations (Figure 5.3).

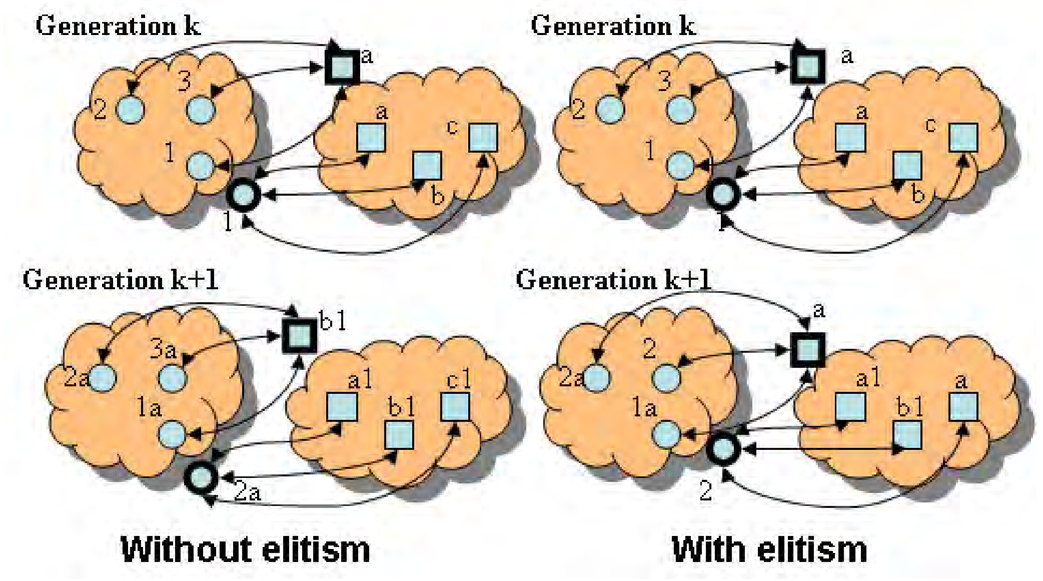


Figure 5.3: CCME with two sub–populations, without and with elitism

At generation k, I train a, b, c to work well with a clone of 1 and 1, 2, 3 to work with a clone of a. Because the trained weights are coded back to the individual as per the Lamarkian approach, individuals in the following generation are $a1$, $b1$, and so on instead of the original a, b, etc. Assuming that the best combination out of these six is between 2 and a, let us call it $2 \longleftrightarrow a$. At generation $k+1$, without elitism, the representatives are $2a$ and $b1$, which have the best fitness in their populations. However, $2a$ was trained previously to work well with the old a and $b1$ was trained to work well with old 1. Now both a and 1 have been lost. There is a possibility that (i) the system can never find such a good combination, or (ii) the information in $2a$ and $b1$ may completely disrupt the previously evolved direction. In this event, the system will fluctuate between finding a good direction and losing it. With elitism in place, as in Figure 5.3, where the best collaboration in a generation is saved back to the population for the next generation, the performance of a system becomes monotonically non-decreasing (i.e. stays as it is or increases). As in Figure 5.3, since the best collaboration in generation k is $2 \longleftrightarrow a$, 2 and a become the representatives. In the next generation, these two are replaced only if a better collaboration is found. The complete CCME algorithm is summarized in Algorithm 14.

Algorithm 14 ME learning with cooperative coevolution

```
1: initialize the populations $P_j$. Initialize empty pools $P'_j$. Set $f_{Eclone} = 0$.
2: for $gen = 0$ to the maximum number of generations do
3:     clone the best individual of each population $P_j$ to $P'_j$ (elites)
4:     apply evolutionary operators to each population $P_j$
5:     for each individual $z$ in each population $P_j$ do
6:         form an ensemble E consisting of $z$ and $P'_k, k \neq j$
7:         apply local search to E using BP and ME
8:         compute fitness $f_E$ of E in terms of classification accuracy rate, assign this fit-
           ness to individual k.
9:         if $f_E > f_{Eclone}$ then
10:            copy E to $E_{clone}$ and set $f_{Eclone} = f_E$
11:        end if
12:    end for
13:    copy components of $E_{clone}$ (the best ensemble in the current generation) to the
       corresponding populations $P_j$. Empty $P'_j$.
14: end for
15: apply the selecting criterion to select and output the desired ensemble.
```

5.6 ME on classification problems

I will apply the original ME model to a number of classification problems. The data sets are drawn from the UCI Machine Learning Repository and the StatLog database. The following experiments will establish a baseline for comparison against the CCME method proposed in this thesis. In term of complexity, I distinguish between the ensemble complexity (i.e. the number of experts in the ensemble) and the network complexity (i.e. the number of hidden units in each expert's neural network). These concepts will be used in chapter 5,6 and 7. In the following sections, I will present the results for the following investigations:

1- ME with different error functions

This set of experiments investigates different error functions proposed in the literature of the Mixture of Experts: (1) Gaussian error function, (2) cross-entropy error function, (3) competitive and localized error function and (4) residual cancelling error function.

2- ME with different learning rates

This set of experiments is designed to investigate the effect of different learning rates, an important aspect of any learning algorithm, on the performance of the ME model.

3- ME with and without early stopping

In chapter 4, I have investigated the benefits of early stopping on improving generalization. This set of experiments is conducted to confirm that early stopping improves the generalization ability of the ME model. The experiments are conducted on two criteria: (i) the minimum error on validation as an early stopping criteria (I call this "val") and (ii) the last generation (I call this "end" or "without early stopping").

4- ME with different ensemble sizes

The next set of experiments is designed to test the effect of ensemble complexity - in terms of the number of experts - on the performance of the mixture of experts.

5- ME with different network complexity

The last set of experiments investigates the effect of network complexity - in terms of the number of hidden units in each individual ANN of the ensemble - on the performance of ME.

5.6.1 Experimental setups

To investigate the above aspects of the ME model, I use the following general setup (unless stated otherwise) - which were investigated in the preliminary study (Chapter 4) and published in (Nguyen, Abbass, and McKay 2004):

The model consists of three experts (except in the experiments on different ensemble size), each of which is a feed–forward ANN with one hidden layer consisting of three hidden nodes. The nodes in the expert ANN are all sigmoidal. The gate component is a feed–forward ANN, which has three linear output nodes corresponding to three experts and no hidden nodes. The outputs of the gate network are passed through a softmax function to obtain probability-like values.

Since the best error function for a binary classification mixture of experts model is naturally the cross entropy error function (Bishop 1995), I will apply the cross entropy error function for the training process.

An initial set of experiments was used to determine suitable parameters. Overall, a value of 320,000 for the number of epochs was found to be suitable. The learning rate is taken to be 0.1. The ensemble at the last epoch (without early stopping) or the ensemble with the minimum validation error (early stopping) is tested against the testing data. The averaged test error rate (%) is reported as the performance of the ensemble.

Datasets

As in chapter 4, I will test the ME model on five standard datasets downloaded from the UCI Repository. Appendix B provides more detailed descriptions of these datasets.

Ten-fold cross validation is performed on each dataset: the available data are divided into ten disjoint subsets using stratified sampling. The testing/validation/training sets are taken at the ratio 1/1/8. For each fold, a new random seed is supplied, while I maintain the same random sees across different setups to have a fair comparison. The results are averaged over these ten folds.

The conventions adopted for presenting results in this thesis

For many of these experiments, a table of results is presented. Each row represents a dataset, while the column represents the investigated factor. The paired student t–test is used to compute the significance of the results in each row. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best. In some cases, to have a better comparison, I plot the performance (i.e. averaged error rates) of the method together with the standard deviations (error bars) in a box plot.

In more complex problems with multiple levels and multiple factors, ANOVA (analysis of variance) is a preferable test of significance to the traditional student t–test (Festing and Altman 2005; Gelman 2005; Hopkins 2000). Throughout this chapter, the ANOVA test and plots will be used when necessary. In an ANOVA plot, using the "multcompare" function in MATLAB on the statistical structure of an ANOVA test, each group average is represented by a symbol and an interval around the symbol. An overlapping of the intervals of two groups' averages implies the groups are not significantly different; while disjoint intervals imply they are significantly different (MathWorks 1997).

5.6.2 ME with different error functions

In this experiment, I will investigate different error functions for the ME model. Jacobs et al (Jacobs, Jordan, Nowlan, and Hinton 1991) proposed three different error functions: the residual cancelling (Res), the competitive (Com) and the Gaussian error function (Gau). According to their original investigation, the Gaussian error function seemed to perform better than the other two methods. In addition to these three error functions, a number of researchers have pointed out that a cross-entropy error function (Cro) is appropriate for classification problems (Bishop 1995; Moerland 1997b; Tang, Heywood, and Shepherd 2002; Waterhouse 1997). Thus, in this section, I will test the ME model against these four error functions.

Table 5.1 presents the error rates of these four error functions for the five datasets. The differences of these error functions for four out of five datasets are not statistically significant. Figure 5.4, which plots the performance and two-way ANOVA test of all data across all datasets, confirms this. Thus, because of the appropriateness of the cross entropy error function for binary classification problems (Bishop 1995; Moerland 1997b; Tang, Heywood, and Shepherd 2002; Waterhouse 1997), I will use the cross entropy error function throughout the chapter.

Dataset	Gaussian	Cross Entropy	Compete	Residual	Confidence
Breast Cancer	0.0372(0.0138)	0.0372(0.0193)	0.0386(0.0135)	0.0386(0.0135)	
Australian credit card	0.1449(0.0355)	*0.1580(0.0345)*	0.1435(0.0316)	**0.1275(0.0391)**	95%
Diabetes	0.2369(0.0633)	0.2537(0.0572)	0.2343(0.0668)	0.2369(0.0543)	
Liver Disorder	0.2785(0.0658)	0.2850(0.0793)	0.2756(0.0600)	0.2846(0.0796)	
Tic Tac Toe	0.1743(0.0547)	0.1847(0.0499)	0.1994(0.0435)	0.2078(0.0465)	

Table 5.1: Mean (and standard deviation) error rates of ME with different error functions. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

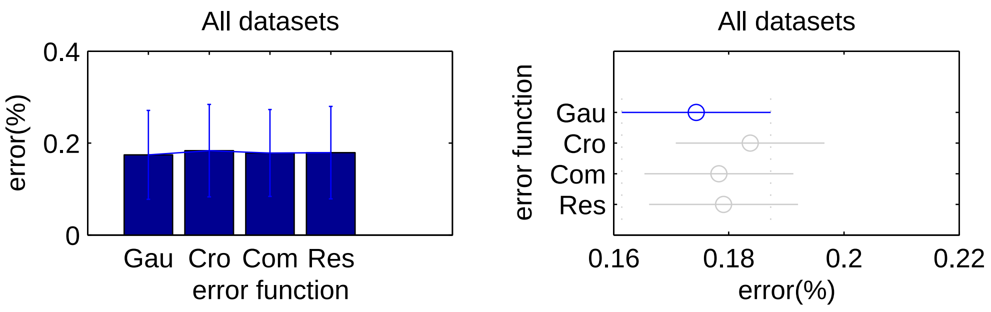


Figure 5.4: Error rates and ANOVA test for four different error functions for ME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different

5.6.3 ME with various learning rates

In this set of experiments, three different pairs of learning rates are used, respectively for the expert ANNs and for the gate ANN: (a)0.1 and 0.1, (b)0.1 and 0.01, and (c)0.01 and 0.1. The aim is to see if different learning rates for the experts and the gates have any effect on the overall performance of the ME model.

Tables 5.2 and 5.3 present the results of these three sets of learning rates with and without early stopping. These results are averaged over the ten folds. The two-way ANOVA (analysis of variance) test is used to test the significance of the learning rate factor (Figure 5.5).

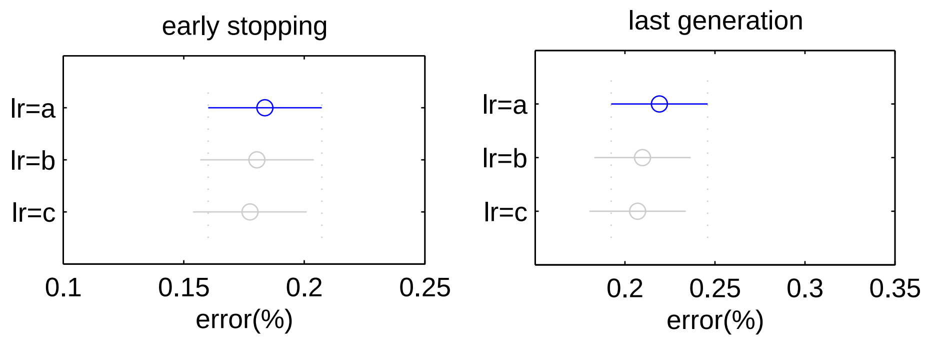


Figure 5.5: ANOVA test on error rates for learning rate factor for ME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different

The ANOVA test shows that the small variations in the learning rates that I tested are not a significant factor in the ME model with and without early stopping. The results in tables 5.2 and 5.3 confirm this observation. Looking at each dataset separately, with and without early stopping, only the breast cancer dataset, and the Tic Tac Toe dataset without early stopping, show a significant preference for particular learning rates. It is hard to generalize the results of learning rates on some datasets to other. However, I undertook a number of similar experiments for other setups, and the results were consistent with this section.

Dataset	(a)$r_{exp} = 0.1$ $r_{gate} = 0.1$	(b)$r_{exp} = 0.01$ $r_{gate} = 0.1$	(c)$r_{exp} = 0.1$ $r_{gate} = 0.01$
Breast Cancer	0.0372(0.0193)	0.0400(0.0162)	0.0386(0.0165)
Australian credit card	0.1580(0.0345)	0.1580(0.0440)	0.1464(0.0345)
Diabetes	0.2537(0.0572)	0.2304(0.0648)	0.2395(0.0598)
Liver Disorder	0.2850(0.0793)	0.2815(0.0595)	0.3167(0.0672)
Tic Tac Toe	0.1847(0.0499)	0.1919(0.0548)	0.1461(0.0340)

Table 5.2: Mean (and standard deviation) error rates of ME with different sets of learning rates on validation stopping criterion.

Dataset	(a)$r_{exp} = 0.1$	(b)$r_{exp} = 0.01$	(c)$r_{exp} = 0.1$	Confidence
	$r_{gate} = 0.1$	$r_{gate} = 0.1$	$r_{gate} = 0.01$	
Breast Cancer	*0.0644(0.0215)*	0.0500(0.0263)	**0.0343(0.0168)**	95%
Australian credit card	0.1768(0.0304)	0.1884(0.0421)	0.2014(0.0450)	
Diabetes	0.2889(0.0680)	*0.2904(0.0639)*	**0.2644(0.0745)**	90%
Liver Disorder	0.3421(0.0610)	0.3371(0.0963)	0.3596(0.0720)	
Tic Tac Toe	*0.2234(0.0386)*	0.1826(0.0414)	**0.1754(0.0425)**	95%

Table 5.3: Mean (and standard deviation) error rates of ME with different sets of learning rates without early stopping. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

5.6.4 ME with and without early stopping

In this experiment, the ME model is tested against two different stopping criteria: early stopping based on validation set; and no early stopping. The results in Table 5.4 are the averages of the ten-fold results over three sets of learning rates in the previous section (i.e. 30 results). Figure 5.6 displays the ANOVA plot of two different stopping criteria for the five datasets.

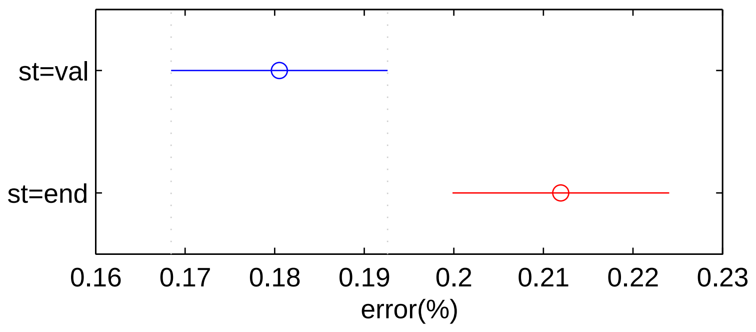


Figure 5.6: ANOVA on error rates to test the effect of stopping criterion on the performance of ME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

The ANOVA results indicate that early stopping is significantly better (with a confidence level of 95%) than omitting early stopping (i.e. last generation). This is confirmed by the results in table 5.4, where the results based on the validation early stopping criterion outperform the last generation criterion for all five test datasets with significance level of at least 95%.

In the previous two sets of experiments, I found that small variations of the

Dataset	(a) early stopping	(b) without early stopping	Confidence
Breast Cancer	**0.0386(0.0168)**	0.0496(0.0245)	95%
Australian credit card	**0.1541(0.0370)**	0.1889(0.0396)	99%
Diabetes	**0.2412(0.0594)**	0.2812(0.0676)	95%
Liver Disorder	**0.2944(0.0686)**	0.3463(0.0757)	99%
Tic Tac Toe	**0.1743(0.0498)**	0.1938(0.0449)	95%

Table 5.4: Mean (and standard deviation) error rates of ME with (a) early stopping based on minimum error on validation set and (b) without early stopping. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

learning rates are not a significant factor, while early stopping is significantly beneficial. Thus in the rest of this chapter, a learning rate of 0.1 is used for both the experts and the gate ANNs. Also, validation set performance is used as the criterion for early stopping. Only the results for early stopping are presented.

5.6.5 ME with various ensemble size

In this set of experiments, I test the ME model on different ensemble size to see if this factor has any effect on the performance of the model. As foreshadowed, the learning rates and stopping criterion are fixed. I compare ensembles with five different sizes of 3, 6, 9, 12 and 15 experts. The results are presented in figure 5.7. The figures in the left column display the error rates of the ensemble with different ensemble sizes (i.e. number of experts). The right column presents the corresponding two-way ANOVA test on different ensemble sizes. It is obvious that ME is quite robust to the ensemble size, i.e. the ensemble size does not significantly affect the performance of the ensemble. Only the Tic Tac Toe endgame dataset shows a significant deterioration when the ensemble size is increased to 15. The rest of figure 5.7 shows no significant effect. The ANOVA plot of the ensemble size factor across all datasets in Figure 5.8 confirms the insignificance of the ensemble size factor.

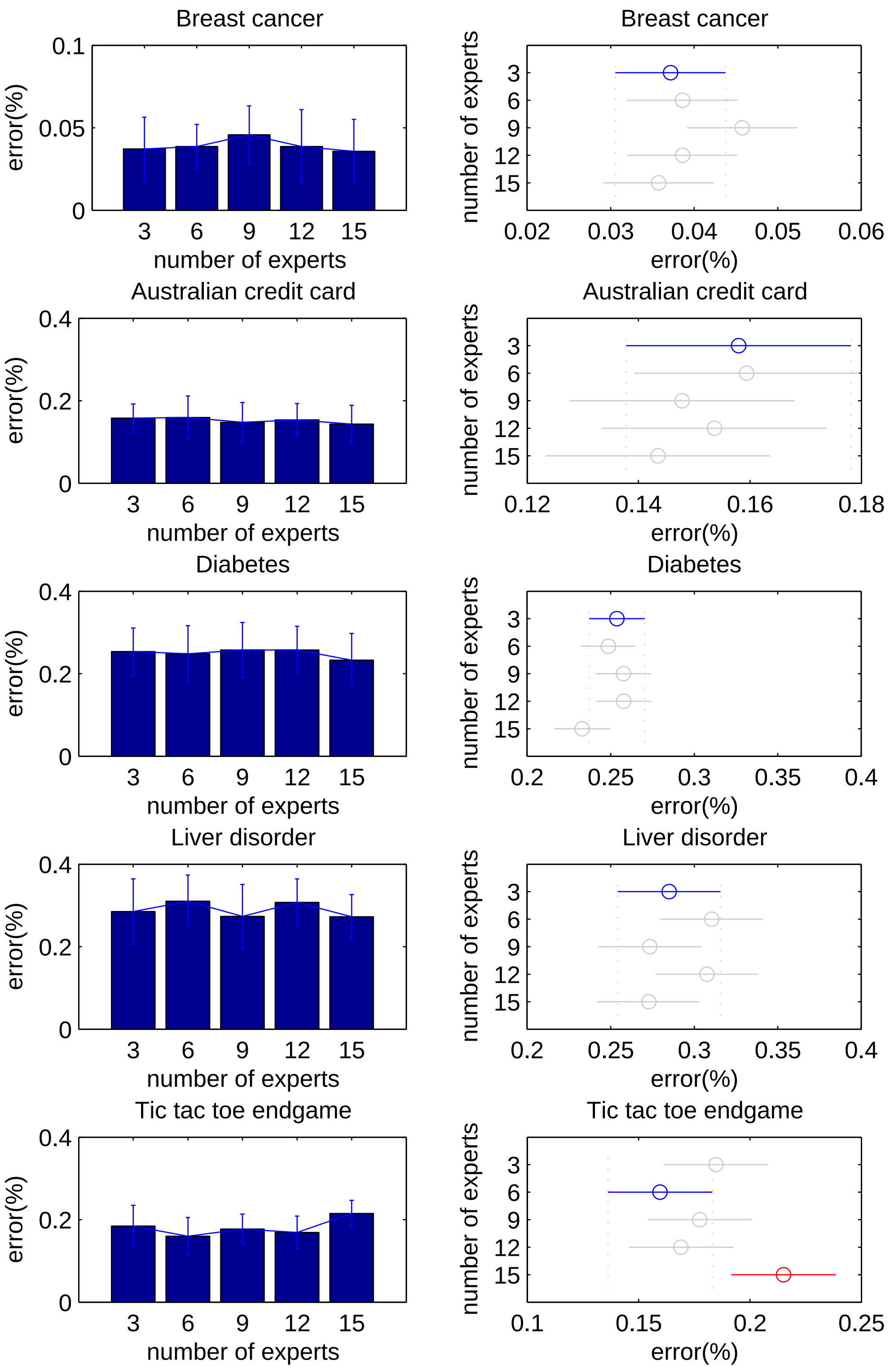


Figure 5.7: Error rate plots (left) and ANOVA tests (right) for five datasets on different ensemble size. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

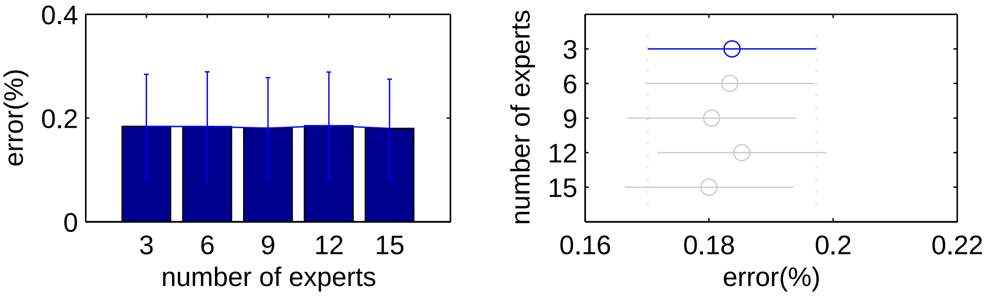


Figure 5.8: Mean (and standard deviation) error rates and ANOVA test for the ensemble size factor for ME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

5.6.6 ME with various network complexity

In the last set of experiments, I test the ME model with different numbers of hidden units in each individual ANN of the expert. I test the ME model against four different values of the number of hidden nodes: 3, 5, 7 and 9. The performance (left) and the corresponding ANOVA plot (right) are presented in figure 5.9. From figure 5.9, only the Tic Tac Toe endgame dataset shows a significant improvement between the smallest ANN (number of hidden nodes = 3) and the other architectures. The plots indicate the robustness of the ME model to an unnecessary increase in the number of hidden units. It is obvious that three hidden units are enough for all datasets except the Tic Tac Toe , which requires five hidden nodes. The gradual increase in the average performance (left plot in Figure 5.10) suggests that larger ANNs, in terms of the number of hidden units in each individual expert ANN, have slightly better average errors. However, the significance plot of the network complexity factor across all datasets in Figure 5.10 does not provide enough significant evidence to support the advantage of the larger ANNs.

5.7 CCME on classification problems

Analogously to section 5.6, CCME is tested against five different factors: (i) error functions, (ii) learning rates, (iii) stopping criteria, (iv) ensemble complexity in term of number of networks, and (v) network complexity in term of number of hidden units.

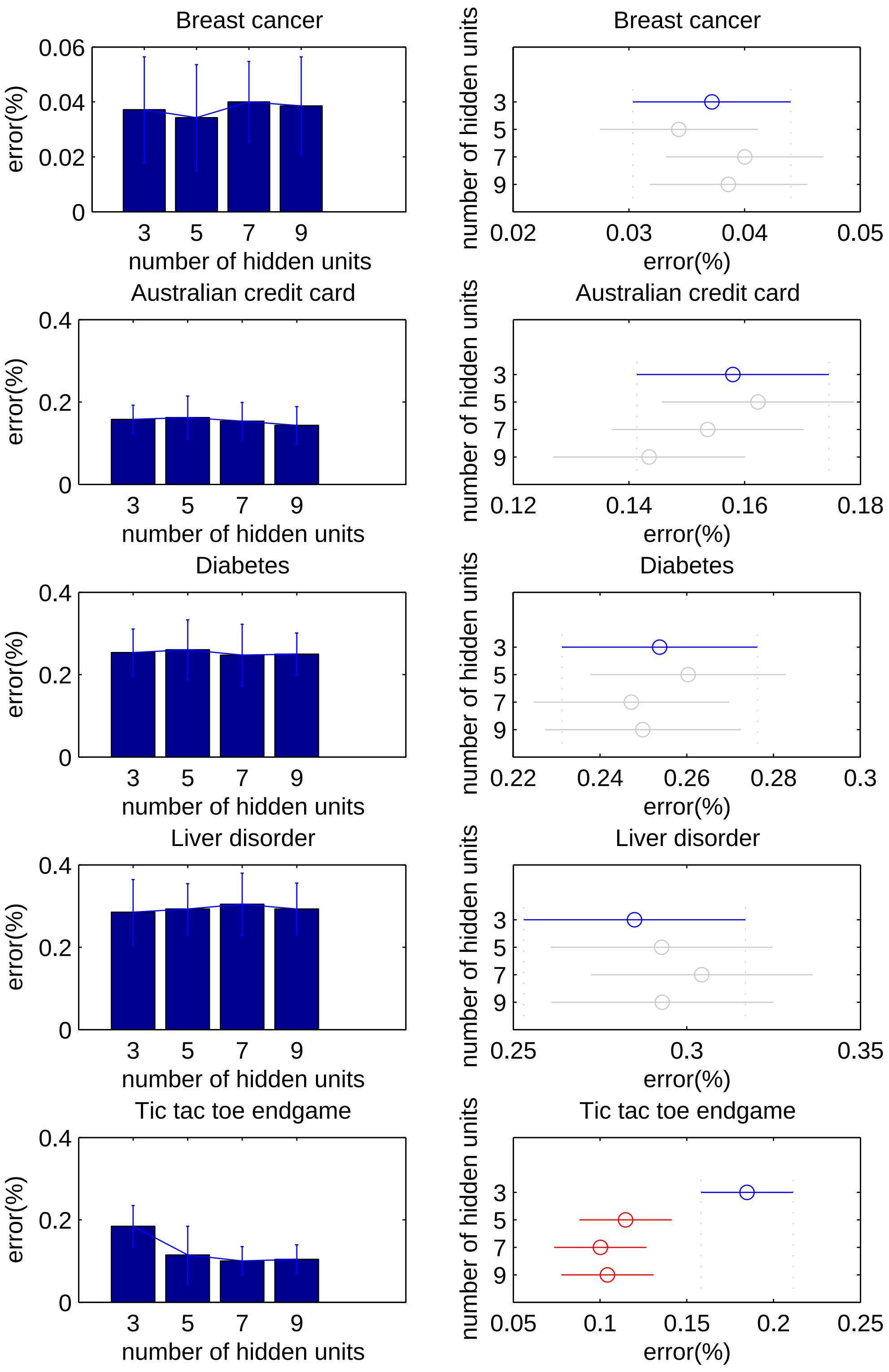


Figure 5.9: Error rate plots (left) and ANOVA tests (right) for five datasets on different network complexity. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

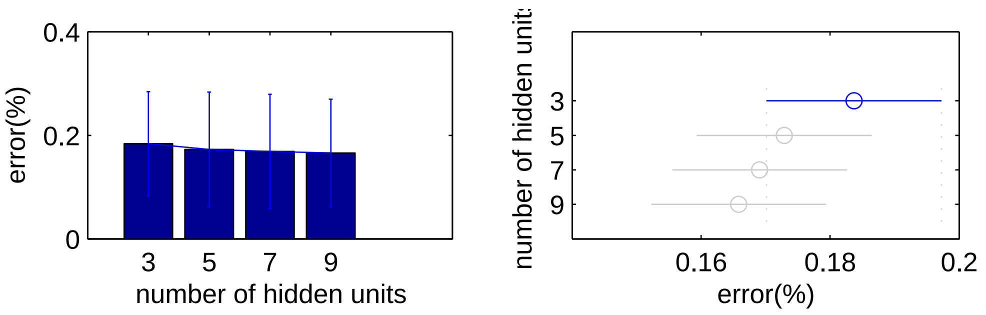


Figure 5.10: Mean (and standard deviation) error rates and ANOVA test for the network complexity factor for ME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

5.7.1 Experimental setups

In the experiments, a self–adaptive (20+20)-ES is applied as the Evolutionary Algorithm. The number of populations/species is the ensemble size plus the gate population. For example, to generate a system with an ensemble size of 3, I create three populations for these three experts and the fourth population for the gating network. I apply a coevolution scheme in which each individual is evaluated against the best in the other populations. Although there are other reasonable selection schemes, it is outside the scope of this thesis to investigate the effect of these schemes.

The same setup from section 5.6 is applied to the ME model at the heart of the CCME algorithm. In other words, each expert is a simple sigmoidal feed-forward neural network with one hidden layer consisting of three hidden nodes (unless noted otherwise). The gating network has no hidden layer and consists of as many linear outputs - with softmax - as the number of experts.

Back propagation is employed to train the ME model (i.e. local search). The learning rates are 0.1 for all components of the model unless specified otherwise. The probabilistic cross entropy function is used as the training error function of the ME local search.

The system is run for 200 generations; local search is performed for 10 epoches per generation per individual. In each generation, the best found ensemble is stored in a place holder and passed back to the population at the end of the evaluation process (i.e.

elitism = 1 per sub–population).

5.7.2 CCME with different error functions

Table 5.5 presents the error rates of these four error functions for the five datasets. Figure 5.11, which plots the performance and two-way ANOVA test of all data across all datasets, implies that the difference between these four error functions is not statistically significant. However, the cross entropy error function is the best among the four error functions (Figure 5.11). Therefore, I will use the cross entropy error function throughout the chapter. This is consistent with the the literature (Bishop 1995; Moerland 1997b; Tang, Heywood, and Shepherd 2002; Waterhouse 1997).

Dataset	Gaussian	Cross Entropy	Compete	Residual	Confidence
Breast Cancer	0.0300(0.0196)	0.0300(0.0142)	0.0372(0.0167)	0.0286(0.0165)	
Australian credit card	0.1420(0.0541)	0.1261(0.0381)	0.1464(0.0490)	0.1406(0.0355)	
Diabetes	0.2485(0.0449)	0.2317(0.0460)	**0.2290(0.0474)**	*0.2538(0.0587)*	95%
Liver Disorder	0.3051(0.0865)	0.2994(0.0786)	*0.3395(0.0667)*	**0.2790(0.0757)**	95%
Tic Tac Toe	*0.1555(0.0306)*	**0.1294(0.0381)**	*0.1775(0.0635)*	*0.1754(0.0541)*	95%

Table 5.5: Mean (and standard deviation) error rates of CCME with different error functions. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

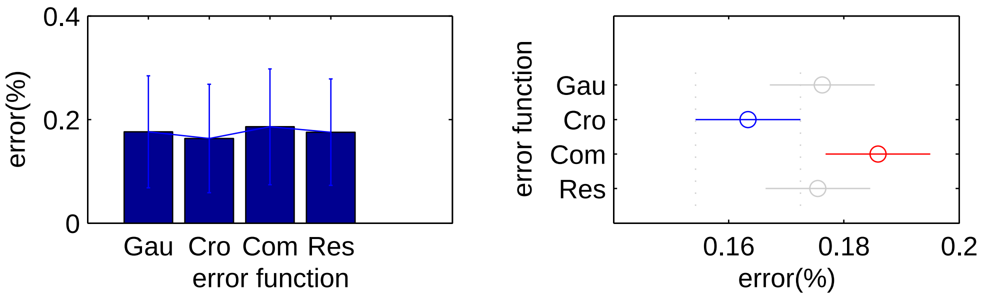


Figure 5.11: Error rates and ANOVA test for four different error functions for CCME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

5.7.3 CCME with different learning rates

Tables 5.6 and 5.7 present the performance of CCME on three different sets of learning rates for two different stopping criteria: minimum on validation and without early stopping respectively. The first set corresponds to the case of learning rate 0.1 for both the gating network and the experts. The second and third sets test the cases when the experts and the gating network have different learning rates (0.01,0.1) and (0.1,0.01) respectively.

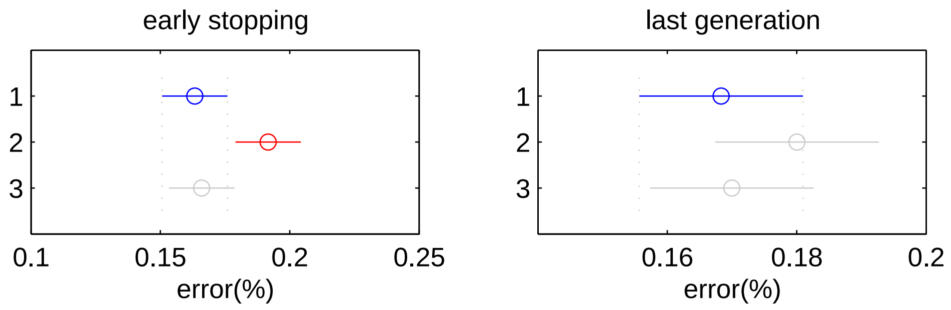


Figure 5.12: ANOVA test on error rates for learning rate factor for CCME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

The results in Table 5.6 show that the first set of learning rates is the best among the three in four out of five datasets while the results for the last generation (Table 5.7) is inconclusive. On the one hand, the ANOVA test on the learning rate factor (figure 5.12, considering all datasets, shows that for early stopping, a learning rate of 0.1 for both the experts and the gate is statistically significantly preferable to the other two situations. On the other hand, the difference in the last generation is not statistically significant for any set of learning rates although set (a) has the best average performance out of the three. Throughout this chapter, I will use a learning rate of 0.1 for both the expert ANNs and the gate ANN (unless stated otherwise).

Dataset	(a)$r_{exp} = 0.1$ $r_{gate} = 0.1$	(b)$r_{exp} = 0.01$ $r_{gate} = 0.1$	(c)$r_{exp} = 0.1$ $r_{gate} = 0.01$	Confidence
Breast Cancer	0.03(0.0142)	0.0329(0.0151)	0.0329(0.0152)	
Australian credit card	**0.1261(0.0381)**	0.1362(0.0356)	*0.1464(0.0301)*	95%
Diabetes	0.2317(0.0460)	0.2303(0.0477)	0.2265(0.0377)	
Liver Disorder	**0.2994(0.0786)**	*0.3346(0.1062)*	0.3021(0.0813)	90%
Tic Tac Toe	0.1294(0.0381)	*0.2243(0.07)*	**0.1221(0.0563)**	99%

Table 5.6: Different learning rate for CCME with stopping criteria based on minimum on validation set (a) $r_{exp} = 0.1$, $r_{gate} = 0.1$) and (b) $r_{exp} = 0.01$, $r_{gate} = 0.1$ and (c) $r_{exp} = 0.1$, $r_{gate} = 0.01$. Presented results are mean and standard deviation of error rates. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

Dataset	(a)$r_{exp} = 0.1$ $r_{gate} = 0.1$	(b)$r_{exp} = 0.01$ $r_{gate} = 0.1$	(c)$r_{exp} = 0.1$ $r_{gate} = 0.01$	Confidence
Breast Cancer	0.0329(0.0135)	**0.03(0.0142)**	*0.0415(0.0195)*	95%
Australian credit card	*0.1536(0.035)*	**0.1348(0.0355)**	0.1493(0.0306)	
Diabetes	0.2475(0.0607)	0.2434(0.0517)	0.2421(0.0623)	
Liver Disorder	0.2845(0.0861)	0.2937(0.0913)	0.299(0.0743)	
Tic Tac Toe	0.1231(0.0435)	*0.1982(0.0655)*	**0.1179(0.0428)**	99%

Table 5.7: Different learning rate for CCME without early stopping (a) $r_{exp} = 0.1$, $r_{gate} = 0.1$) and (b) $r_{exp} = 0.01$, $r_{gate} = 0.1$ and (c) $r_{exp} = 0.1$, $r_{gate} = 0.01$. Presented results are mean and standard deviation of error rates. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

Dataset	(a) early stopping	(b) without early stopping	Confidence
Breast Cancer	0.0319(0.0162)	0.0348(0.0144)	
Australian credit card	**0.1362(0.0346)**	0.1459(0.0346)	95%
Diabetes	**0.2295(0.0425)**	0.2443(0.0564)	95%
Liver Disorder	0.3120(0.0879)	**0.2924(0.0815)**	95%
Tic Tac Toe	0.1586(0.0721)	**0.1464(0.0623)**	99%

Table 5.8: Different stopping criteria for CCME (a) minimum error on validation set and (b) no early stopping. Presented results are mean and standard deviation of error rates. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

5.7.4 CCME with different stopping criteria

The average error rates (over ten fold results of three learning rates) are presented in tables Table 5.8. The performance (Table 5.8) and ANOVA tests (Figure 5.13) give mixed results. Early stopping is preferable on two of the five datasets, while ”last generation” is preferable on another two (significance level 95%). Overall, early stopping shortens the training time while maintaining a comparable competence, so I will use early stopping throughout the remainder of this chapter.

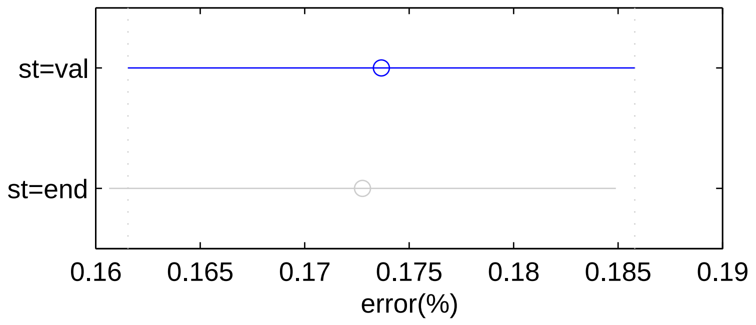


Figure 5.13: CCME: ANOVA test for the stopping criteria factor. Overlapping intervals of groups’ averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

5.7.5 CCME with various ensemble size

The performance of CCME with different ensemble sizes are presented in figure 5.14. It is obvious that CCME is quite robust to the ensemble size, i.e. the ensemble size does not significantly affect the performance of the ensemble. Only the Tic Tac Toe

Endgame dataset shows a significant improvement when the ensemble size is increased from 3 experts to a larger size. The rest of the figure 5.14 shows statistically insignificant effects. The results across all datasets are presented in Figure 5.15. The ANOVA plot shows that none of the ensemble sizes has a statistically significant effect on the performance of the ensemble, although the ensemble with 9 experts seems to be the best among the five configurations.

5.7.6 CCME with various network complexity

The performance of CCME with different networks complexity is presented in figure 5.16. The plots indicate the robustness of the CCME model to an unnecessary increase in the number of hidden units. It is obvious that three hidden units were enough for all datasets except the Tic Tac Toe which requires five hidden nodes. This conclusion is backed up by the plot across all datasets in Figure 5.17. Although the overall plot (Figure 5.17) suggests that too simple and too complex (in term of number of hidden units) ANNs are not ideal, the lack of a statistically significant difference makes it inconclusive.

5.7.7 Understanding evolutionary dynamics of CCME

In this section, the evolutionary dynamics of CCME are analyzed to gain insight into evolutionary behavior in the model.

Training and validation accuracy over generation plots

Figures 5.19 display the training and validation accuracy over generation curves of Diabetes, Liver Disorder and Tic Tac Toe Endgame datasets. In this section, the training and validation accuracies of the best ME in each generation are considered as the accuracies of the system in that generation. The plots contrast the training accuracy with the corresponding validation accuracy. As expected with an elitism-based coevolutionary system (Popovici and Jong 2005), the training accuracy (also the fitness in this

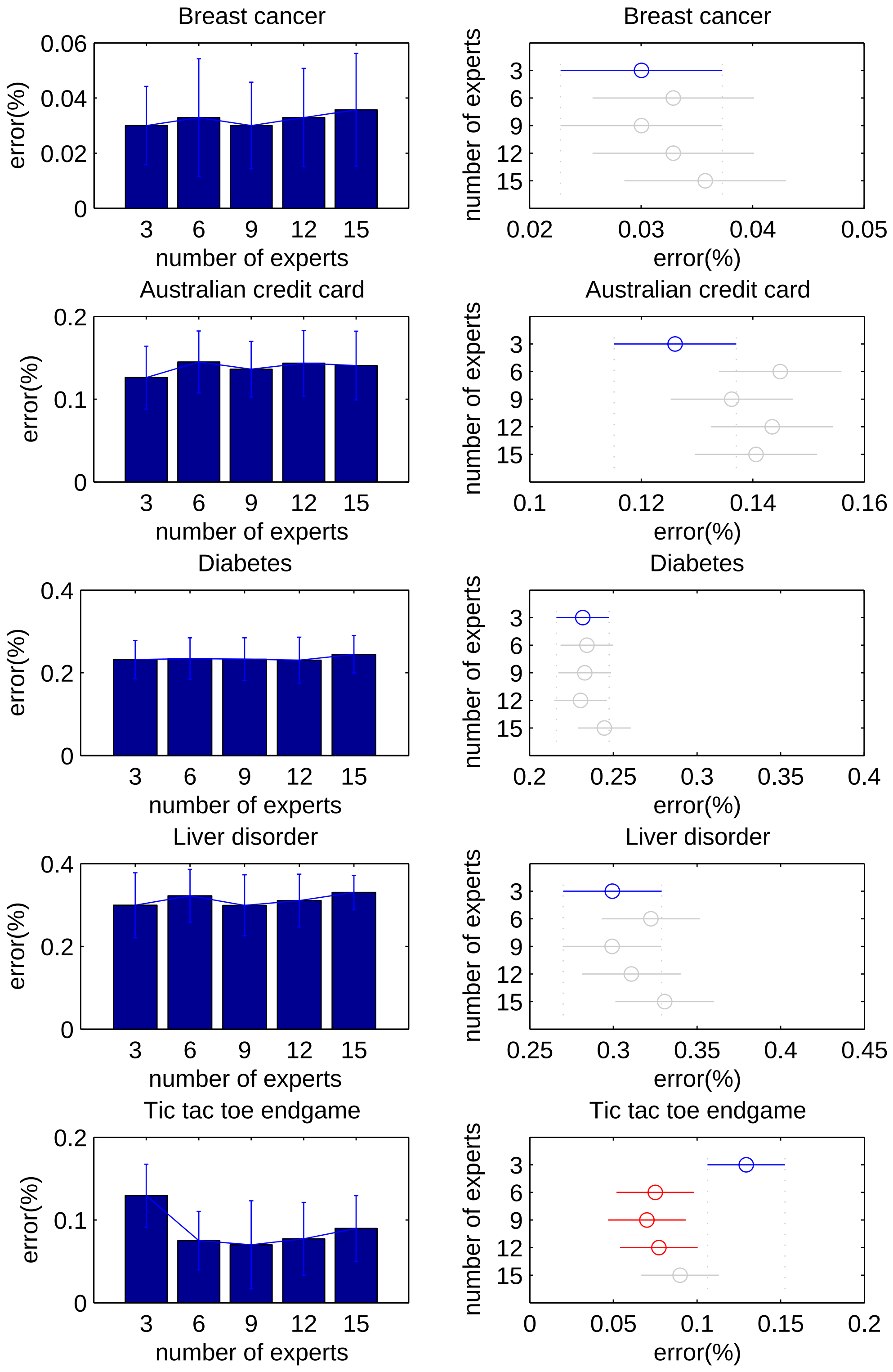


Figure 5.14: Error plots (left) and ANOVA tests (right) for five datasets on different ensemble size: only Tic tac toe dataset show a significant different between ensemble of size 3 and of other sizes. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

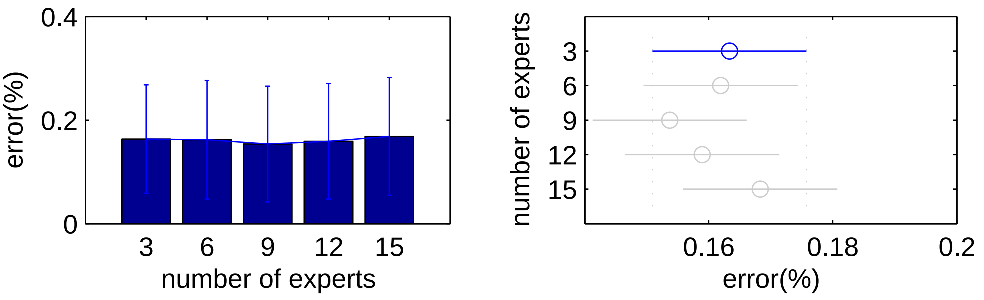


Figure 5.15: ANOVA test on error rates for the ensemble size factor for CCME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

thesis) of the best individual in the generation increases monotonically and converges after a number of generations. The plots of the corresponding validation accuracy of the best individual are as expected, with up and down movements.

Since the validation curves also show the tendency of early stopping, the plots suggest that CCME should stop much earlier in the case of Diabetes than in the other two datasets.

Fitness over generation plots

In this section, I use the usual fitness over generation plots to get insights into how the sub–populations behave over time (units are taken here to means generations). Figure 5.19 shows three types of fitness-time plots: (i) the leftmost displays the fitness of the best ME in each generation, (ii) the middle shows the fitness of the best individuals of the four sub–populations, and (iii) the rightmost shows the average fitness of the four sub–populations. In other words, the first plot in a row corresponds to the evolutionary dynamics of the whole system while the last two correspond to the coevolutionary dynamics of the sub–populations.

The plots of the sub–population's best fitness and average fitness show a small fitness diversity among sub–populations. However, since the fitness is computed as the correction rates of the cooperation between an individual in one sub–population and the best individuals in the other sub–populations, it is not surprising the fitness differences

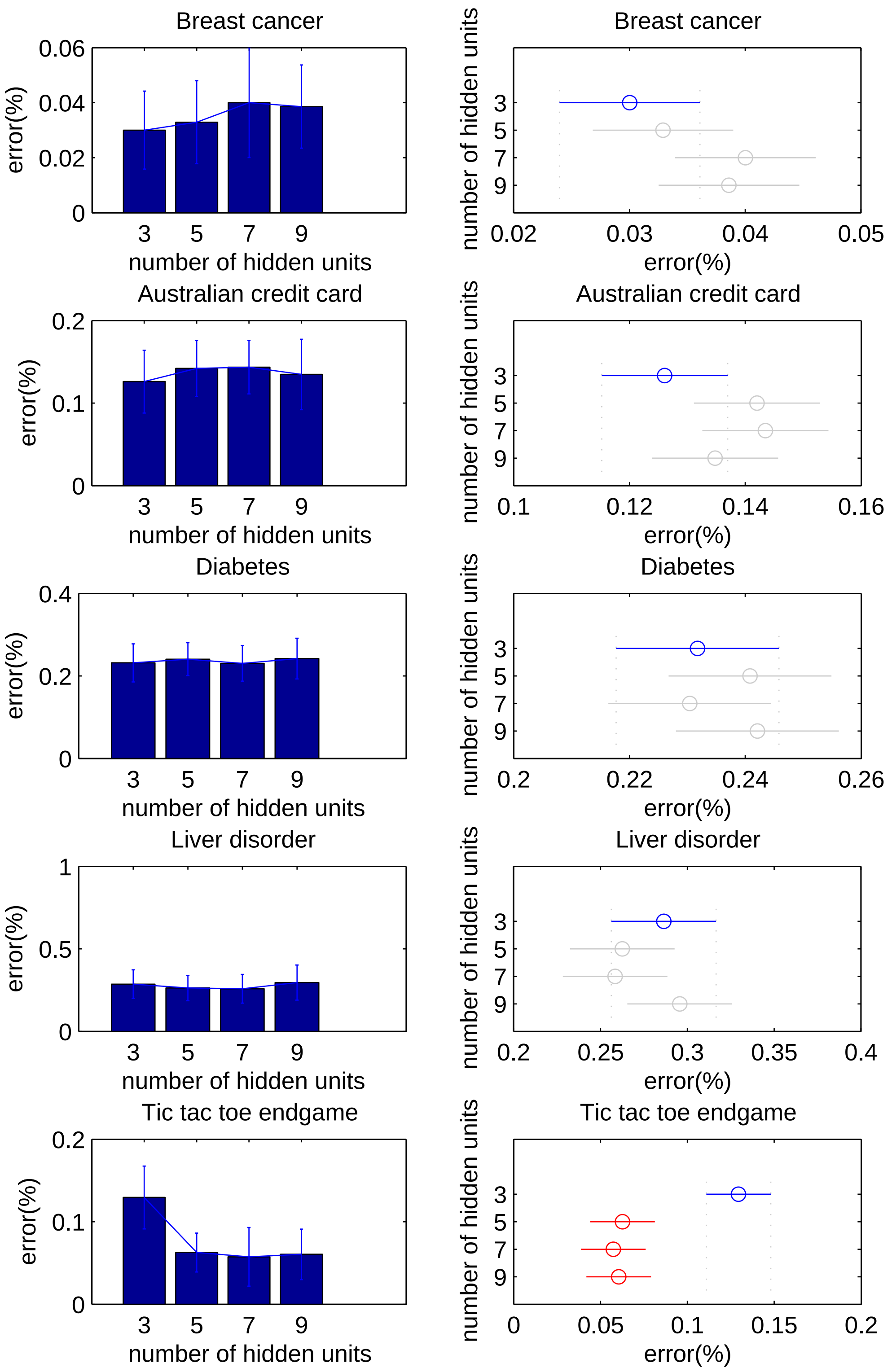


Figure 5.16: Error plots (left) and ANOVA tests (right) for five datasets on different network complexity: only Tic tac toe dataset show a significant different between ensemble of size 3 and of other sizes. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

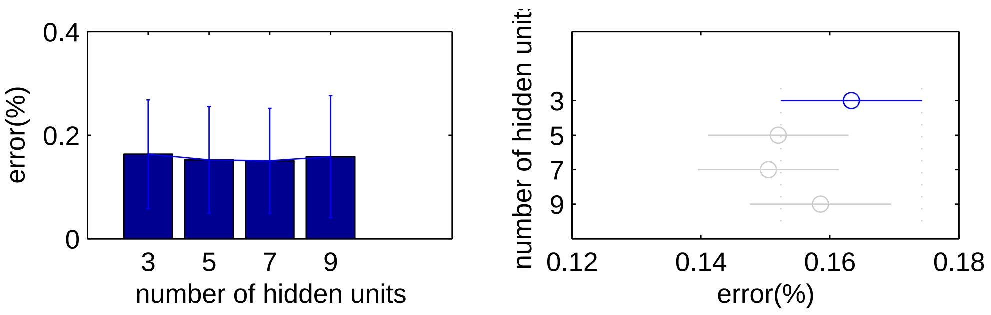


Figure 5.17: ANOVA test on error rates for the network complexity factor for CCME. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

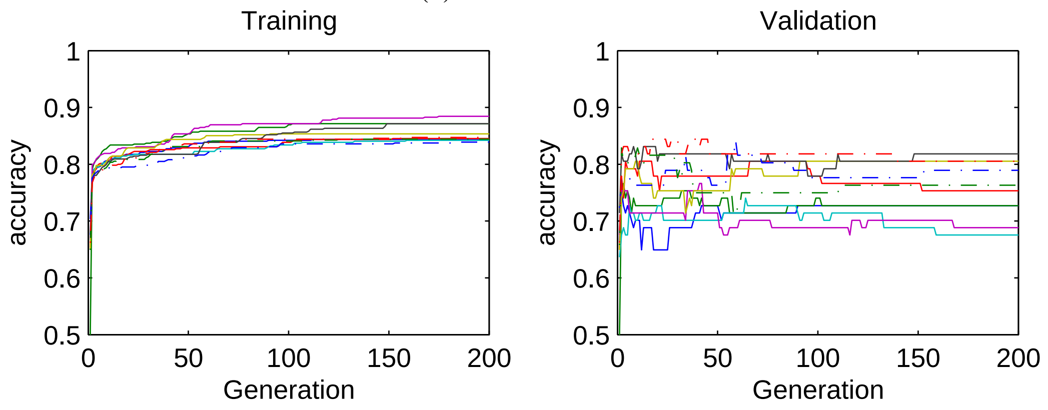
between sub–populations are not high. Moreover, this small diversity in sub–population fitness does not contradict the observation that the system, being a mixture of experts, exerts a much higher genotypic diversity in terms of the different input regions, for which the model's sub–populations are responsible. In chapter 6, I will show that indeed, the sub–populations are diverse in the sense that they are responsible for very different input sub–regions.

Fitness distributions

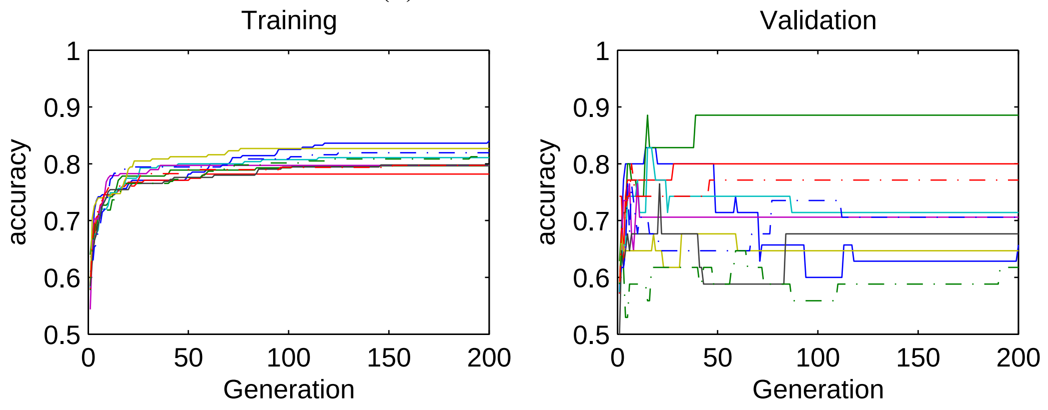
Popovici and De Jong (Popovici and Jong 2003) have proposed a visualization technique to understand how the fitness distribution of an EA population changes over time. In this technique, the fitness data are collected through multiple runs and the fitness distributions of each run are plotted as lines on the same graph. By taking snapshots at some time intervals (i.e. generations) during a complete run, the sequence of plots show how the distributions change over time.

In the following plots, the fitness distributions of ten runs at generation 0 (initialization), 1, 20, 40, 60, 100, 150 and 200 are plotted. Different lines in the same plot represent different runs. A wider spread in the fitness distribution implies higher phenotypic diversity in the population while a narrow distribution implies low population diversity in the fitness space. Figures 5.20,5.22,5.23 present the fitness distributions of the Diabetes, Liver Disorder and Tic Tac Toe Endgames datasets respectively. Over time,

(a) Diabetes



(b) Liver Disorder



(c) Tic Tac Toe Endgames

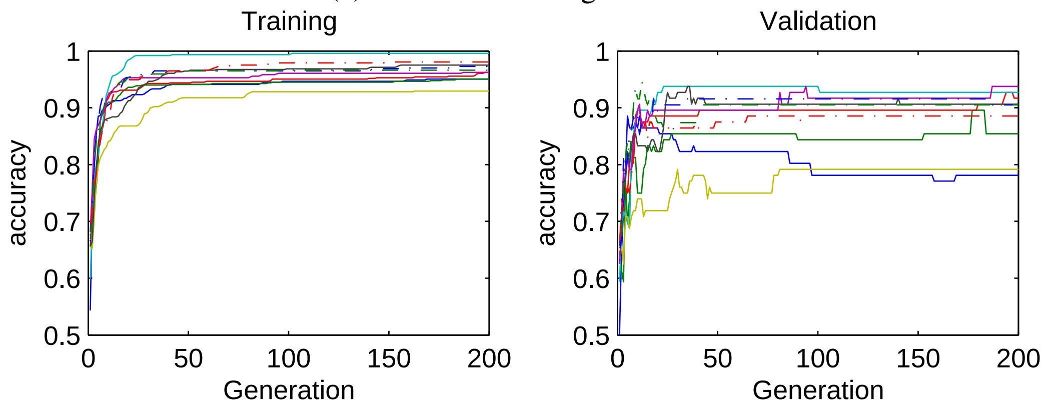


Figure 5.18: CCME: training and validation accuracy of the best individual of each generation vs. generation curves of 10 runs of Diabetes, Liver Disorder and Tic Tac Toe Endgames: different lines represent different runs.

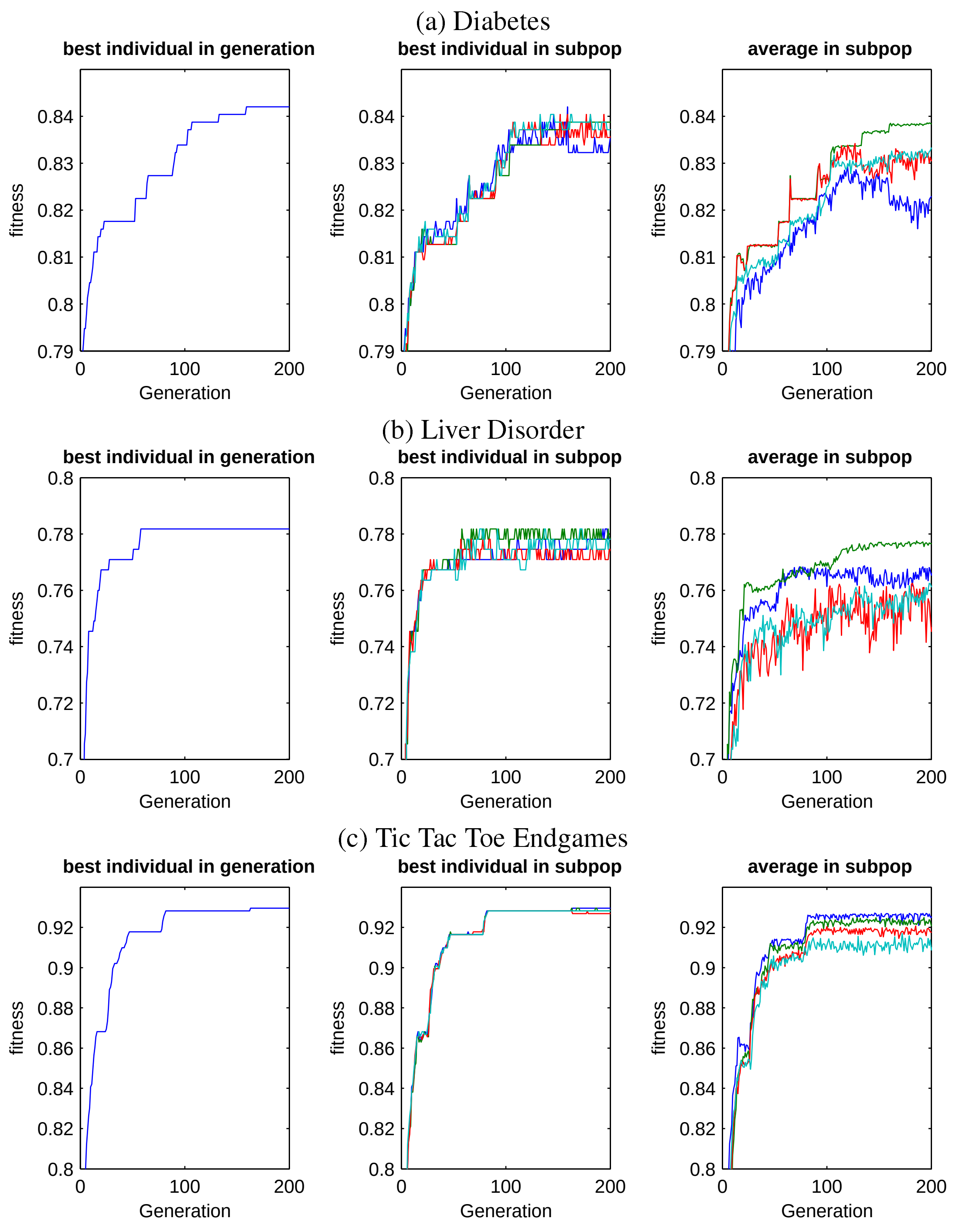


Figure 5.19: CCME: An example of fitness over generation plots for Diabetes, Liver Disorder and Tic Tac Toe:(left) best individual in generation, (middle) best individual in every sub–population, (right) average fitness in every sub–population

i.e. generations, the fitness distribution gradually spreads out and is skewed to the left (i.e the frequency peak lies at the higher fitness). This phenomenon implies that the populations gradually lose their phenotypic diversity over time. However even at generation 200, there is still some spread in the fitness distribution, which suggests CCME is able to explore the lower fitness range after a long evolutionary run. In other words, population diversity is maintained. A detailed example of the fitness distribution over generations for Diabetes is plotted as a 3D graph in Figure 5.21. The plot shows how the distribution of fitness increases over time, as well as how it is gradually skewed to the left.

In the Diabetes and Liver Disorder cases, this change in the fitness distribution occurs much slower than for the Tic Tac Toe dataset, which suggests Tic Tac Toe converges much quicker than the other two sets. This conclusion matches the training accuracy plots in section 5.7.7.

5.8 CCME vs. ME comparisons

In this set of experiments, CCME is compared against the back propagation ME to test whether adding a CC layer improves the generalization performance of the ME model. The experiments are divided into two sets. In the first set of experiments, CCME is verified against a single ME with long training time (i.e. 320,000 epochs). The purpose is to verify CCME against the traditional ME regarding the performance and time complexity, which are reported and compared. To keep the comparison fair, since CCME makes use of 320,000 objective evaluations (i.e. 200 generations x 160 individuals x 10 epochs per generation per individual), I use the same number of epochs (i.e. 320,000) for ME. The second set of experiments is conducted to test whether CC is beneficial in improving the performance of the ME model. To test this, for each fold (in 10 folds), instead of running a single ME for a long period, I run a set of 160 randomized MEs, each of which is trained for 2,000 epochs. The ME with the minimum validation error out of these 160 runs is chosen to be tested against the test data. The reported errors

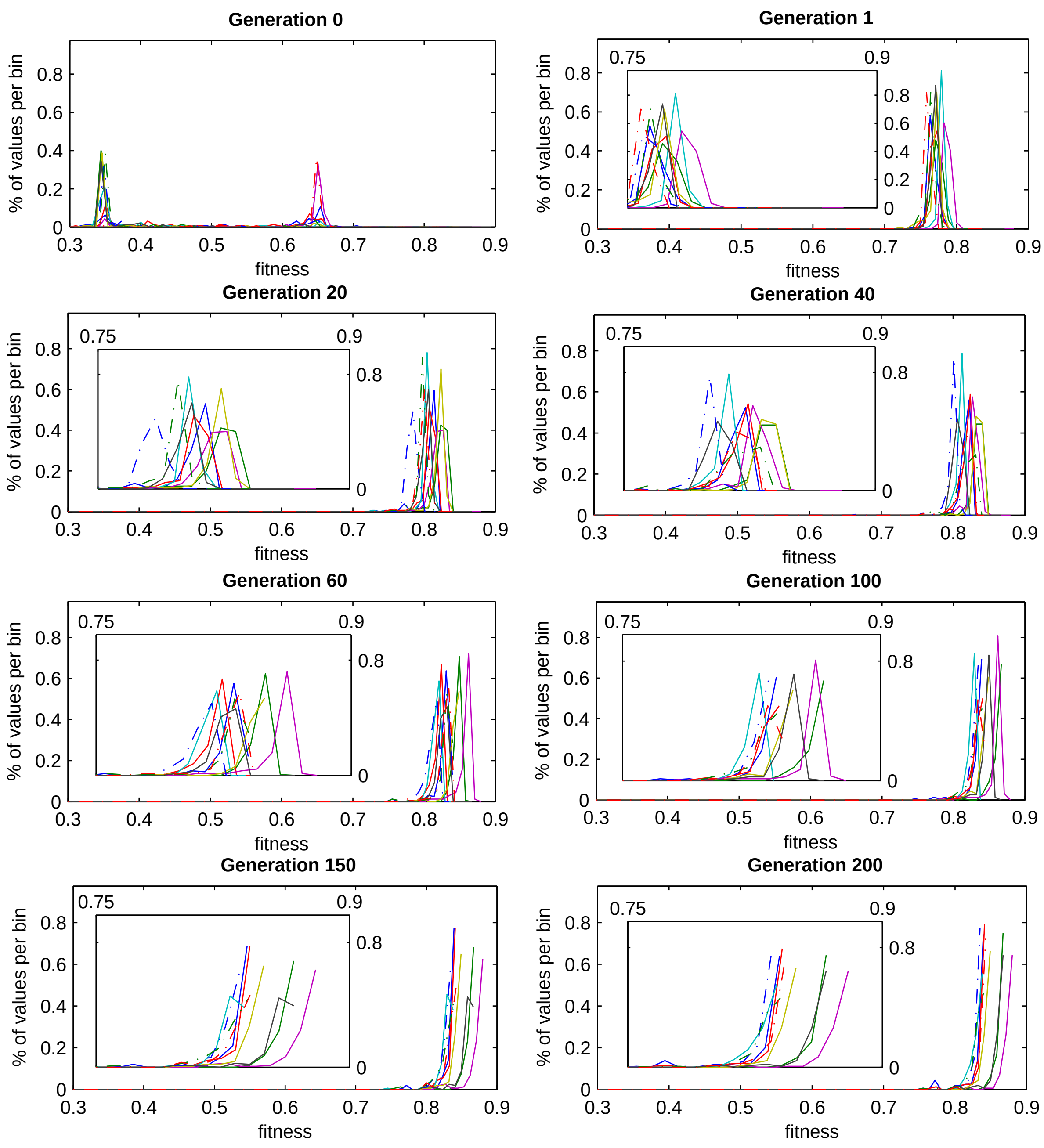


Figure 5.20: 10 fitness distributions of Diabetes taken at generations 0, 1, 20, 40, 60, 100, 150 and 200: different lines represent different runs.

are averaged over ten folds.

Datasets

In this section, I add another ten different datasets. CCME and ME are compared on fifteen benchmark datasets taken from the UCI machine learning repository (Newman, Hettich, Blake, and Merz 1998) and the StatLog database (King, Feng, and Shutherland 1995). The datasets are summarized in Table 5.9. Since the number of inputs greatly affects the networks' complexity, and consequently the system performance, the datasets will be categorized into small size (number of attributes < 20) and medium size (number of attributes ≥ 20). It is undoubtable that larger datasets require more complex systems, i.e. more experts and more hidden nodes per expert; however, it is outside the scope of this thesis to investigate the relationship between data complexity and network complexity. Therefore, the comparisons will be limited to a fixed level of complexity to demonstrate that even with a small system, the method is still compatible/competitive to the simple back propagation ME.

5.8.1 CCME vs ME on performance

CCME is compared against the back propagation ME across different learning rates to avoid possible bias of different learning rates toward different methods. The results (Tables 5.10 and 5.11) show a superior performance of CCME over the original ME, regardless of different learning rates and different stopping criteria. Figure 5.24 shows the significance test (ANOVA) of CCME vs ME for the fifteen datasets, considering all different learning rates with and without early stopping. The tests show that CCME significantly outperforms ME in the small datasets and is better than ME in the medium datasets, although the latter difference is insignificant. These results provide strong evidence of the advantage gained by using CC in conjunction with the ME model.

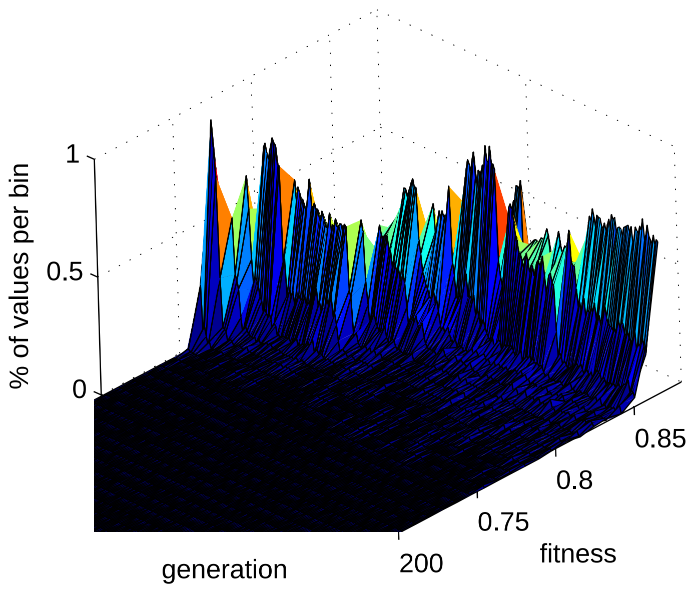


Figure 5.21: A 3D plot of fitness distributions over time of Diabetes

	Number of Instances	Number of Attributes	Continuous Attributes	Discrete Attributes
(i) small datasets				
Breast Cancer	699	9	9	
Cleveland Heart	303	13	7	6
Australian credit card	690	14	6	8
Diabetes	768	8	8	
StatLog Heart	270	13	7	6
Hepatitis	155	19	6	13
Liver Disorder	345	6	6	
Ljubljana Breast Cancer	286	9		9
Tic Tac Toe	958	9		9
House voting 84	435	16		16
(ii) medium datasets				
German credit card	1000	24	7	17
Ionosphere	351	33	33	
King Rook vs King Pawn	3196	36		36
E.coli Promoters	106	57		57
Thyroid sickness	2800/972	27	7	20

Table 5.9: Fifteen datasets from UCI machine learning repository and Statlog database

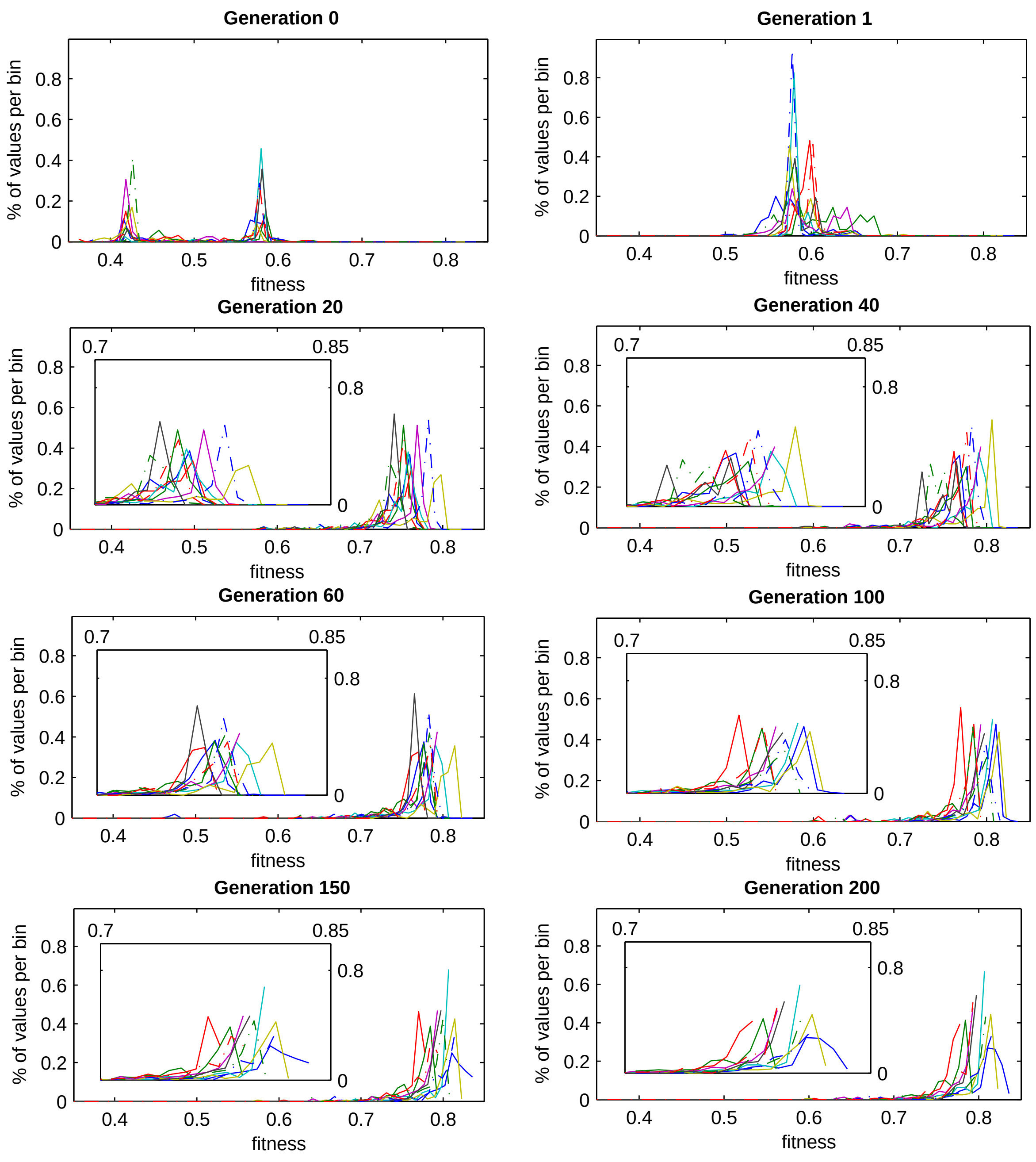


Figure 5.22: 10 fitness distributions of Liver Disorder taking at generations 0, 1, 20, 40, 60, 100, 150 and 200: different lines represent different runs.

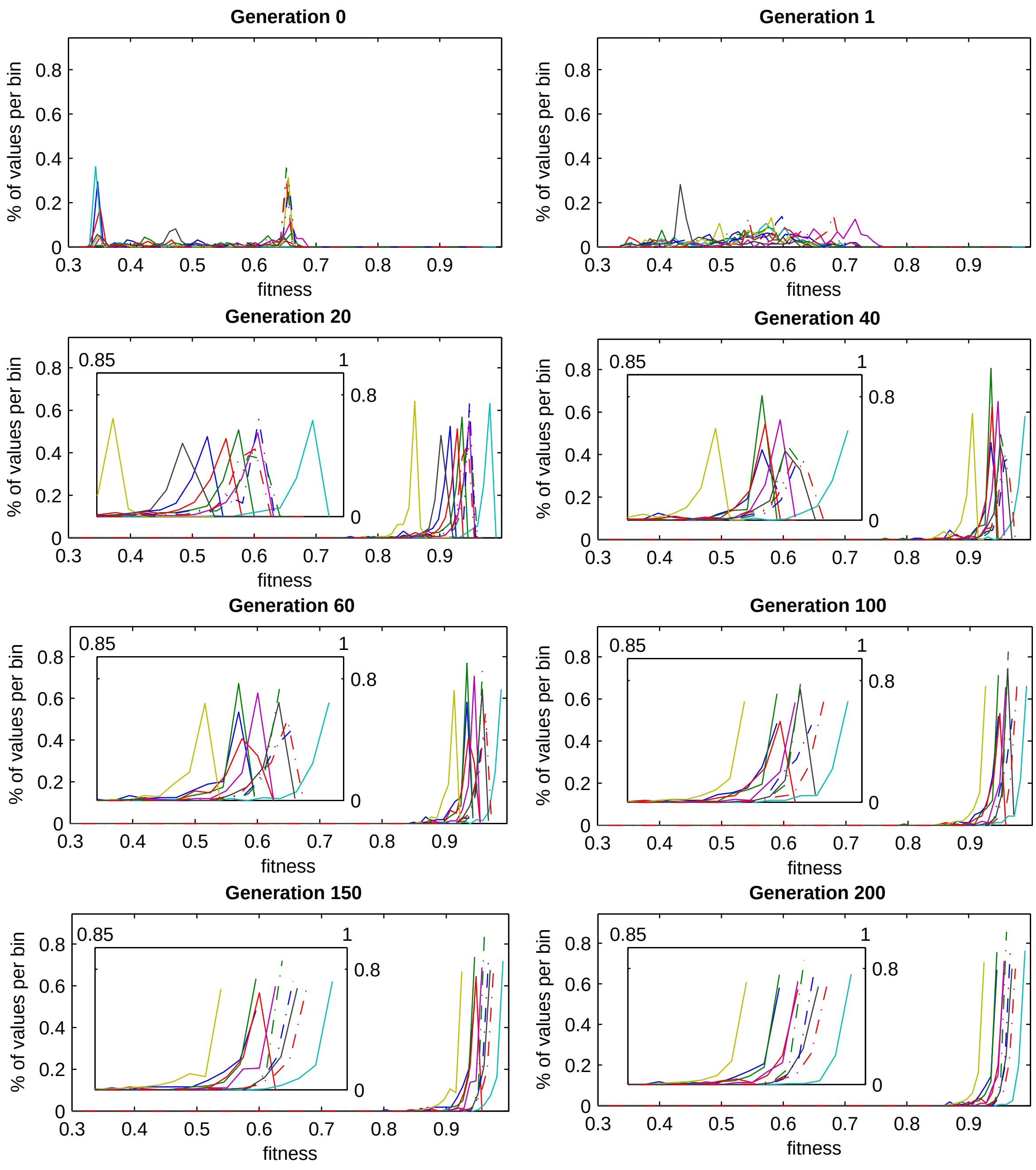


Figure 5.23: 10 fitness distributions of Tic Tac Toe Endgames taking at generations 0, 1, 20, 40, 60, 100, 150 and 200: different lines represent different runs.

	$\eta_{exp} = \eta_{gate} = 0.1$		$\eta_{exp} = 0.01, \eta_{gate} = 0.1$		$\eta_{exp} = 0.1, \eta_{gate} = 0.01$	
Dataset	ME	CCME	ME	CCME	ME	CCME
(i) small datasets						
Breast Cancer	0.037(0.019)	**0.030(0.014)**	0.040(0.016)	**0.033(0.015)**	0.039(0.017)	**0.033(0.015)**
Cleveland Heart	0.208(0.052)	0.225(0.073)	0.258(0.098)	0.228(0.059)	0.241(0.077)	0.215(0.047)
Australian credit card	0.158(0.034)	**0.126(0.038)**	0.158(0.044)	**0.136(0.036)**	0.146(0.034)	0.146(0.030)
Diabetes	0.254(0.057)	0.232(0.046)	0.230(0.065)	0.230(0.048)	0.240(0.060)	0.227(0.038)
StatLog Heart	0.211(0.080)	**0.174(0.089)**	0.185(0.087)	0.159(0.092)	0.219(0.120)	0.181(0.069)
Hepatitis	0.200(0.099)	0.168(0.061)	0.180(0.064)	0.188(0.090)	0.200(0.090)	0.194(0.053)
Liver Disorder	0.285(0.079)	0.299(0.079)	**0.282(0.059)**	0.335(0.106)	0.317(0.067)	0.302(0.081)
Ljubljana Breast Cancer	0.263(0.040)	0.297(0.112)	0.283(0.033)	**0.245(0.049)**	0.256(0.031)	0.276(0.112)
Tic Tac Toe	0.185(0.050)	**0.129(0.038)**	**0.192(0.055)**	0.224(0.070)	0.146(0.034)	0.122(0.056)
House voting 84	0.064(0.035)	**0.037(0.034)**	0.055(0.039)	**0.046(0.032)**	0.071(0.052)	0.062(0.054)
(i) medium datasets						
German credit card	0.261(0.048)	0.242(0.027)	0.262(0.040)	**0.243(0.029)**	0.274(0.038)	**0.243(0.032)**
Ionosphere	0.262(0.069)	**0.222(0.098)**	0.265(0.095)	**0.217(0.088)**	0.268(0.087)	**0.228(0.091)**
King Rook vs King Pawn	0.018(0.008)	**0.010(0.008)**	0.018(0.010)	0.019(0.009)	0.010(0.007)	0.009(0.006)
E.coli Promoters	**0.180(0.182)**	0.254(0.129)	0.215(0.140)	0.194(0.216)	**0.169(0.150)**	0.244(0.157)
Thyroid sickness	0.107(0.004)	**0.098(0.016)**	0.108(0.005)	**0.077(0.018)**	0.107(0.006)	**0.102(0.000)**

Table 5.10: Mean (and standard deviation) error rates of CCME vs single ME running 320,000 epochs using three different sets of learning rates and early stopping. A bold face indicates a significantly better performance in a row.

	$\eta_{exp} = \eta_{gate} = 0.1$		$\eta_{exp} = 0.01, \eta_{gate} = 0.1$		$\eta_{exp} = 0.1, \eta_{gate} = 0.01$	
Dataset	ME	CCME	ME	CCME	ME	CCME
(i) small datasets						
Breast Cancer	0.064(0.022)	**0.033(0.014)**	0.050(0.026)	**0.030(0.014)**	**0.034(0.017)**	0.041(0.020)
Cleveland Heart	0.244(0.076)	0.222(0.065)	0.251(0.074)	0.228(0.045)	0.251(0.088)	**0.192(0.056)**
Australian credit card	0.177(0.030)	**0.154(0.035)**	0.188(0.042)	**0.135(0.036)**	0.201(0.045)	**0.149(0.031)**
Diabetes	0.289(0.068)	**0.247(0.061)**	0.290(0.064)	**0.243(0.052)**	0.264(0.074)	**0.242(0.062)**
StatLog Heart	0.248(0.100)	**0.185(0.076)**	0.233(0.096)	**0.170(0.101)**	0.237(0.068)	**0.174(0.103)**
Hepatitis	0.195(0.126)	0.180(0.073)	0.161(0.076)	0.207(0.088)	0.175(0.096)	0.226(0.074)
Liver Disorder	0.342(0.061)	**0.285(0.086)**	0.337(0.096)	**0.294(0.091)**	0.360(0.072)	**0.299(0.074)**
Ljubljana Breast Cancer	0.301(0.077)	0.266(0.066)	0.269(0.094)	0.273(0.053)	0.287(0.072)	**0.263(0.049)**
Tic Tac Toe	0.223(0.039)	**0.123(0.043)**	0.183(0.041)	0.198(0.065)	0.175(0.042)	**0.118(0.043)**
House voting 84	0.062(0.052)	0.064(0.045)	0.048(0.035)	0.041(0.026)	0.053(0.041)	0.062(0.040)
(i) medium datasets						
German credit card	0.280(0.060)	0.264(0.043)	0.302(0.056)	**0.246(0.046)**	0.325(0.054)	**0.264(0.031)**
Ionosphere	0.242(0.081)	**0.217(0.072)**	0.308(0.073)	**0.225(0.078)**	0.279(0.063)	**0.217(0.089)**
King Rook vs King Pawn	0.018(0.011)	**0.009(0.007)**	**0.011(0.005)**	0.018(0.008)	0.010(0.005)	0.010(0.004)
E.coli Promoters	0.161(0.149)	0.234(0.174)	0.226(0.161)	0.218(0.109)	**0.198(0.166)**	0.281(0.188)
Thyroid sickness	0.109(0.008)	0.176(0.233)	0.106(0.011)	**0.084(0.018)**	0.107(0.005)	**0.102(0.000)**

Table 5.11: Mean (and standard deviation) error rates of CCME vs single ME running 320,000 epochs using three different sets of learning rates and without early stopping. A bold face indicates a significantly better performance in a row.

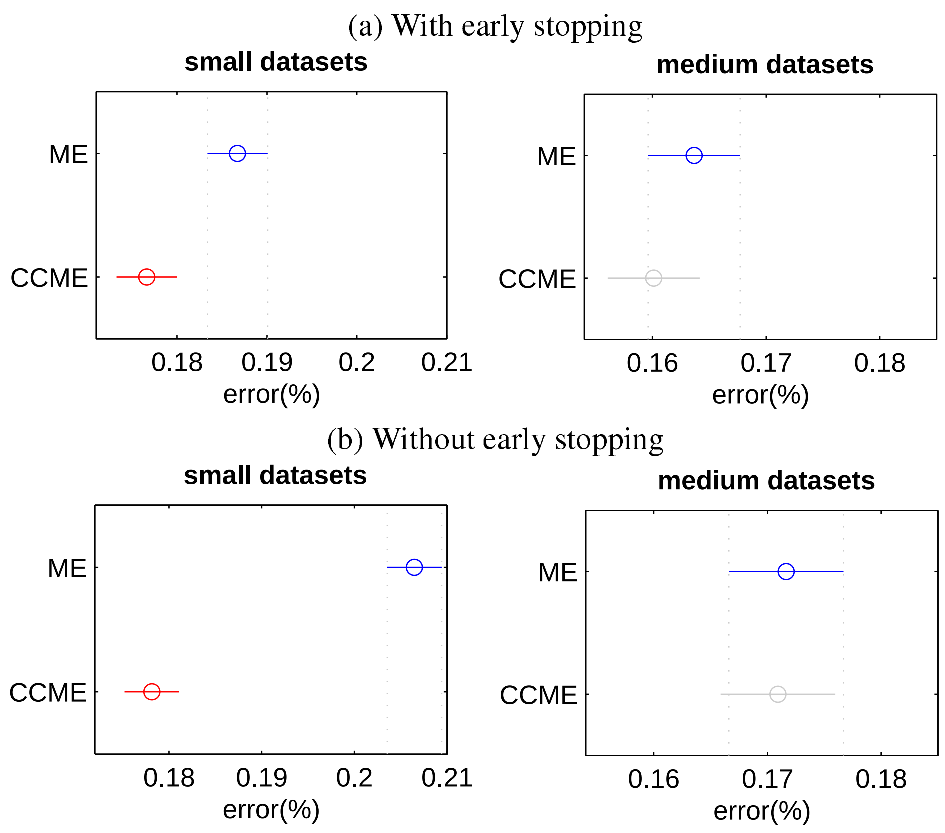


Figure 5.24: ANOVA test on error rates for the CCME vs single ME running 320,000 epochs for small and medium datasets accounting all three learning rates and (a) early stopping, (b) without early stopping. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

5.8.2 CCME vs randomized ME on performance

I have shown above that CCME, on average, performs better than a single ME running for 320,000 epochs. However, someone may argue that a long running time is disadvantageous to back propagation, even though early stopping is used. In this section, for each fold, instead of running a single ME for a long period, a set of 160 randomized MEs are run for 2,000 epochs. This mimics the evolutionary setup, which consists of 160 individuals running for 200 generations x 10 local search epochs. The ME with the lowest validation error is selected, out of 160 randomized MEs, as the representative of the run. This individual is then tested against the test set and its test error is reported as the performance error of the run. The average test error of the ten runs is reported as the generalization error of the randomized back propagation ME.

CCME is compared against the randomized back propagation ME across different learning rates. The results (Tables 5.12 and 5.13) show that on small datasets, CCME performs significantly better than back propagation ME, regardless of different learning rates. Figure 5.25 shows the significance test (ANOVA) of CCME vs ME for the fifteen datasets considering all different learning rates. The test confirms the above conclusion, that CCME significantly out-performs ME on small datasets. For the medium datasets, however, there is not enough evidence to reject the null hypothesis, that CCME is no different from ME. The results in this section, together with those in section 5.8.1, imply that adding a CC layer enhances the generalization ability of ME for some classification problems.

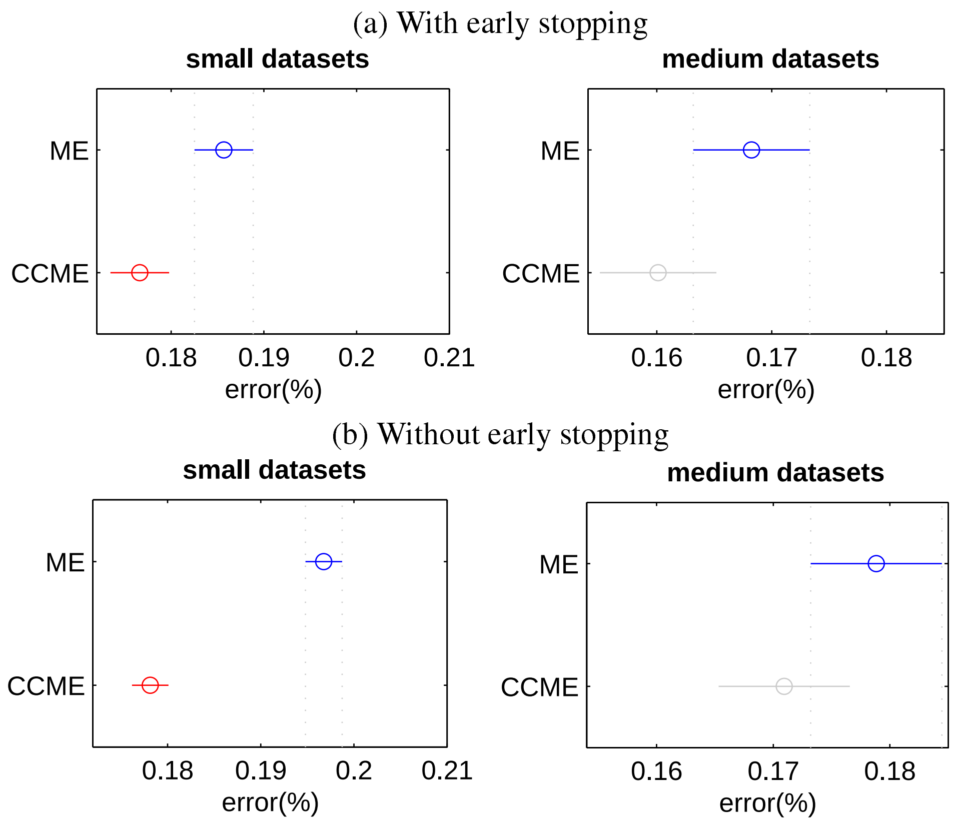


Figure 5.25: ANOVA test on error rates for the CCME vs 160 randomized MEs running 2,000 epochs for small and medium datasets accounting all three learning rates. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

	$\eta_{exp} = \eta_{gate} = 0.1$		$\eta_{exp} = 0.01, \eta_{gate} = 0.1$		$\eta_{exp} = 0.1, \eta_{gate} = 0.01$	
Dataset	ME	CCME	ME	CCME	ME	CCME
(i) small datasets						
Breast Cancer	0.043(0.019)	**0.030(0.014)**	0.046(0.022)	**0.033(0.015)**	0.049(0.018)	**0.033(0.015)**
Cleveland Heart	0.244(0.087)	0.225(0.073)	0.248(0.083)	0.228(0.059)	0.225(0.044)	0.215(0.047)
Australian credit card	0.133(0.046)	0.126(0.038)	0.132(0.039)	0.136(0.036)	0.157(0.041)	0.146(0.030)
Diabetes	0.243(0.055)	0.232(0.046)	0.236(0.055)	0.230(0.048)	0.246(0.055)	**0.227(0.038)**
StatLog Heart	0.222(0.068)	**0.174(0.089)**	0.237(0.088)	**0.159(0.092)**	0.185(0.099)	0.181(0.069)
Hepatitis	0.194(0.062)	0.168(0.061)	0.193(0.080)	0.188(0.090)	**0.161(0.073)**	0.194(0.053)
Liver Disorder	0.287(0.039)	0.299(0.079)	**0.297(0.081)**	0.335(0.106)	0.319(0.071)	0.302(0.081)
Ljubljana Breast Cancer	0.322(0.082)	0.297(0.112)	0.283(0.064)	**0.245(0.049)**	0.245(0.072)	0.276(0.112)
Tic Tac Toe	0.111(0.033)	0.129(0.038)	**0.185(0.065)**	0.224(0.070)	0.130(0.044)	0.122(0.056)
House voting 84	0.067(0.048)	**0.037(0.034)**	0.080(0.053)	**0.046(0.032)**	0.050(0.042)	0.062(0.054)
(i) medium datasets						
German credit card	0.268(0.033)	**0.242(0.027)**	0.273(0.039)	**0.243(0.029)**	0.258(0.040)	0.243(0.032)
Ionosphere	0.239(0.072)	0.222(0.098)	0.197(0.095)	0.217(0.088)	0.228(0.074)	0.228(0.091)
King Root vs King Prawn	0.010(0.006)	0.010(0.008)	**0.011(0.005)**	0.019(0.009)	0.007(0.005)	0.009(0.006)
E.coli Promoters	**0.189(0.123)**	0.254(0.129)	0.214(0.188)	0.194(0.216)	0.239(0.146)	0.244(0.157)
Thyroid sickness	0.112(0.011)	**0.098(0.016)**	0.105(0.005)	**0.077(0.018)**	0.106(0.004)	**0.102(0.000)**

Table 5.12: Mean (and standard deviation) error rates of CCME vs 160 randomized MEs running 2,000 epochs using three different sets of learning rates and early stopping. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

	$\eta_{exp} = \eta_{gate} = 0.1$		$\eta_{exp} = 0.01, \eta_{gate} = 0.1$		$\eta_{exp} = 0.1, \eta_{gate} = 0.01$	
Dataset	ME	CCME	ME	CCME	ME	CCME
(i) small datasets						
Breast Cancer	0.044(0.016)	**0.033(0.014)**	0.038(0.018)	**0.030(0.014)**	0.039(0.014)	0.041(0.020)
Cleveland Heart	0.258(0.043)	**0.222(0.065)**	0.251(0.043)	0.228(0.045)	0.255(0.040)	**0.192(0.056)**
Australian credit card	0.171(0.022)	**0.154(0.035)**	0.150(0.029)	**0.135(0.036)**	0.162(0.024)	**0.149(0.031)**
Diabetes	0.263(0.046)	**0.247(0.061)**	0.246(0.054)	0.243(0.052)	0.247(0.051)	0.242(0.062)
StatLog Heart	0.231(0.076)	**0.185(0.076)**	0.211(0.066)	**0.170(0.101)**	0.225(0.075)	**0.174(0.103)**
Hepatitis	0.197(0.056)	0.180(0.073)	0.189(0.050)	0.207(0.088)	**0.186(0.046)**	0.226(0.074)
Liver Disorder	0.305(0.041)	0.285(0.086)	0.279(0.063)	0.294(0.091)	0.302(0.043)	0.299(0.074)
Ljubljana Breast Cancer	0.306(0.047)	**0.266(0.066)**	0.278(0.049)	0.273(0.053)	0.292(0.045)	**0.263(0.049)**
Tic Tac Toe	0.195(0.016)	**0.123(0.043)**	0.233(0.019)	**0.198(0.065)**	0.175(0.016)	**0.118(0.043)**
House voting 84	0.059(0.038)	0.064(0.045)	0.058(0.038)	**0.041(0.026)**	0.058(0.037)	0.062(0.040)
(i) medium datasets						
German credit card	0.293(0.019)	**0.264(0.043)**	0.284(0.021)	**0.246(0.046)**	0.288(0.021)	**0.264(0.031)**
Ionosphere	0.254(0.048)	**0.217(0.072)**	0.241(0.050)	0.225(0.078)	0.238(0.053)	0.217(0.089)
King Root vs King Prawn	0.017(0.005)	**0.009(0.007)**	0.023(0.006)	**0.018(0.008)**	0.012(0.004)	**0.010(0.004)**
E.coli Promoters	0.204(0.138)	0.234(0.174)	0.197(0.140)	0.218(0.109)	**0.198(0.134)**	0.281(0.188)
Thyroid sickness	0.109(0.002)	0.176(0.233)	0.107(0.003)	**0.084(0.018)**	0.111(0.006)	**0.102(0.000)**

Table 5.13: Mean (and standard deviation) error rates of CCME vs 160 randomized MEs running 2,000 epochs using three different sets of learning rates and without early stopping. A bold face indicates a significantly better performance in a row while an italic font indicates a significantly inferior performance compared to the best.

5.8.3 CCME vs ME on time complexity

To justify the time complexity of CCME, the running time of CCME is compared against the running time of the back propagation ME on fifteen datasets. The running time is counted over 200 generations for CCME and 320,000 epochs for ME,so as to equalize the number of objective evaluations for both methods [320,000 = 160(individuals)x 200(generations) x 10(epochs in local search)]. The results of three learning rates over ten folds are averaged and reported in Figure 5.26 and Table 5.14. The results show that CCME is slightly slower than ME due to the evolutionary overhead. However, the running time of CCME is still of the same magnitude as the running time of ME. In other words, CC does not add significant computation time to the overall running time of the system.

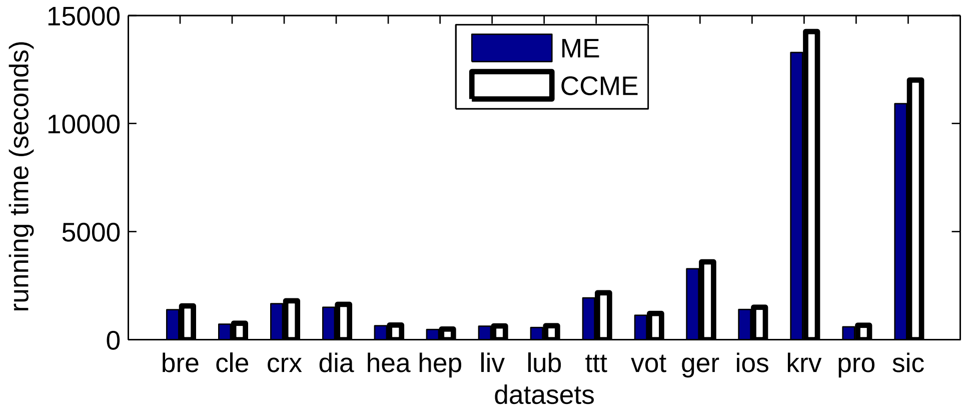


Figure 5.26: Running time (seconds) for ME and CCME on fifteen datasets

5.9 Conclusion

In this chapter, I have first introduced and investigated different aspects of the Mixture of Experts model. The key results of the experiments can be summarized as follows: (i) different error functions do not significantly affect the performance of the ME, (ii) the ME model is robust to various learning rates, ensemble complexity measured by the number of individual ANNs, and ANN complexity in terms of the number of hidden units, and (iii) early stopping significantly improves the generalization performance of the ME model, which confirms the conclusion of chapter 4.

In addition, I have introduced and investigated a novel method based on the principle of cooperative coevolution and the mixture of experts model. The key results of the experiments can be summarized as follows: (i) different error functions do not significantly affect the performance of CCME, (ii) the CCME model is robust against various learning rates, ensemble complexity in terms of the number of individual ANNs and ANN complexity in terms of the number of hidden units, (iii) CCME is robust against over–fitting. These conclusions should be interpreted with appropriate caution regarding extrapolation to other datasets, as our experiments are limited to binary classification.

Finally, the performance of CCME is validated against the traditional ME in a number of ways. The results suggest that (i) adding CC improves the performance of ME on a number of classification problems, (ii) the time complexity of CCME and ME is of the same magnitude, or in other words, CCME is equivalent to ME in terms of time complexity.

Dataset	ME (seconds)	CCME (seconds)
(i) small datasets		
Breast cancert (bre)	1385 (13)	1560 (5)
Cleveland heart (cle)	717 (2)	759 (3)
Australian credit card (crx)	1662 (7)	1801 (7)
Diabetes (dia)	1494 (8)	1632 (7)
Statlog heart (hea)	641 (2)	676 (4)
Hepatitis (hep)	465 (6)	492 (5)
Lover disorder (liv)	619 (3)	634 (5)
Ljubljana breast cancer (lub)	563 (4)	642 (6)
Tic tac toe (ttt)	1927 (12)	2169 (7)
House voting 84 (vot)	1125 (8)	1208 (6)
(i) medium datasets		
German credit card (ger)	3280 (35)	3603 (31)
Ionosphere (ios)	1395 (46)	1501 (9)
King rook vs king pawn (krv)	13290 (134)	14255 (91)
E.coli promoters (pro)	590 (12)	667 (8)
Thyrois sickness (sic)	10913 (93)	12014 (69)

Table 5.14: Mean (and standard deviation) of the running time of ME and CCME on fifteen datasets

Chapter 6

Automatic Problem Decomposition

As mentioned in chapter 3, often real world problems are often too complicated to solve with a single ANN. One efficient way to solve these complex problems is to divide them into a set of simpler solvable sub-problems, and design a set of ANNs to suit these sub-problems. This scheme is often described as Divide-and-Conquer. In fact, problem decomposition is desirable not only in overly-complex problems but also in simple problems. Firstly, decomposability implies that the system has the potential to run in parallel, which can speed up the system if conducted correctly. Secondly, with potentially modular problems, a modular solution better reflects the problem structure and therefore is more desirable than a non-modular solution. Since problem decomposition is a natural way to discover such modular structures, in the solution as well as in the problem, it is beneficial to study problem decomposition.

The key questions in automatic problem decomposition are: (i) how to divide the problem into simpler tasks, (ii) how to assign individuals to solve these sub–tasks and (iii) how to synthesize the whole system back together. If the problem has a clear decomposition nature, it may be possible to derive such a decomposed system by hand. However in most real-world problems, we either know little about the problems, or the problems are too complex to have a clear vision for a hand designed decomposition. Thus it is desirable to have a method to automatically decompose a complex problem into a set of overlapping or disjoint sub problems, and to assign one or more specialists to each of these subproblems.

ME has proven to be a good Divide-and-Conquer method (Waterhouse 1997). By cleverly training a gate component together with the experts, ME answers all three key questions of the automatic problem decomposition. Firstly, by applying a competitive error function, ME can let the experts compete with each other for the input cases. This property addresses the first two issues of APD. Secondly, by training the gate which produces the combination weights for the experts, ME ensures an appropriate integration recipe for the experts. This addresses the last requirement of an APD system.

Although, in the literature of ME, while it is often stated that ME possesses the APD property, it is not often carefully analyzed. In this chapter, I introduce a number of visualization tools to analyze the APD characteristics of the ME and CCME models. I show that, by visualizing the experts' responsibilities vs. the input space, one can see how these models decompose a problem. Moreover, with the help of an artificial 2D problem, I will show that ME and CCME not only decompose the input space, but decomposes it in such a way that each subproblem is simple enough such that the assigned experts can solve it with greater accuracy.

6.1 Visualization tools to analyze APD

In this thesis, I will propose two different mechanisms for visualizing the automatic problem decomposition in high-dimensional feature spaces. The first mechanism is a simple one where data are grouped based on the specialization of each expert and a color–map of the data records is visualized. The second mechanism relies on principal component analysis to project the feature space onto lower dimensions, whereby decision boundaries generated by each expert are visualized through convex approximations.

6.1.1 Output vs input image plot (imgpl)

The simplest way to visualize automatic problem decomposition is to plot the output of an individual ANN together with its inputs. As discuss in the previous chapters, the gate component of a Mixture of Experts acts as a soft switch, selecting an expert (or experts) to be responsible for each data record. Thus the outputs of this gate can be considered as a responsibility measure for the experts for a specific data instance. To visualize the responsibility of each expert versus the data, instead of using the actual weights produced by the gate, a value '1' is assigned to the expert with the maximum responsibility and '0' to the rest. However, since the gate is a 'soft' switch in the sense that it might select more than one expert for certain patterns, in those cases, 0's are assigned to all the experts. In other words, a value of '1' indicates strong responsibility (i.e. the corresponding weight is distinctly higher than other experts' weights).

The outputs of the individual experts are recorded in a matrix as follows. For the sake of explanation, I assume an ME with three experts (i.e. ensemble size = 3) and 10 data records. Figure 6.1 displays such a matrix, where each row corresponds to an expert and each column corresponds to a data pattern. Each entry of the matrix represents the output of network i on data pattern j. A value '1', for example in row 1 and column 4 (highlighted in Figure 6.1), indicates that expert 1 is assigned to pattern 4.

$$\left(\begin{array}{ccc|c|cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Figure 6.1: Matrix of individual experts' responses

To visualize an expert's responsibility for different regions of the input space, a matrix representing the input values for each corresponding record is required. For example, a hypothetical dataset of ten records and 4 attributes is presented in the following matrix:

$$\begin{pmatrix} 0.2 & 1 & 0.33 & 0.25 & 0.1 & 0 & 0.25 & 0.6 & 0.37 & 0 \\ 0.1 & 0 & 0.33 & 0 & 0.1 & 0 & 0.25 & 0.4 & 0.12 & 0 \\ 0.3 & 0 & 0 & 0.25 & 0.6 & 1 & 0.25 & 0 & 0.22 & 1 \\ 0.4 & 0 & 0.33 & 0.5 & 0.2 & 0 & 0.25 & 0 & 0.29 & 0 \end{pmatrix}$$

To make it easier to interpret the plot, the two matrices are sorted according to the experts' responsibilities (i.e. rows are clustered in the Hamming space, with the number of clusters equals to the number of experts):

$$\begin{pmatrix} 0 & 0.33 & 1 & 0.25 & 0.1 & 0.25 & 0.6 & 0.2 & 0.25 & 0.37 \\ 0 & 0.33 & 0 & 0 & 0.1 & 0.25 & 0.4 & 0.1 & 0 & 0.12 \\ 1 & 0 & 0 & 0.25 & 0.6 & 0.25 & 0 & 0.3 & 0.25 & 0.22 \\ 0 & 0.33 & 0 & 0.25 & 0.2 & 0.25 & 0 & 0.4 & 0.5 & 0.29 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Finally, a color map is used to plot the two matrices. An example for the Breast cancer dataset with 3 experts is shown in Figure 6.2

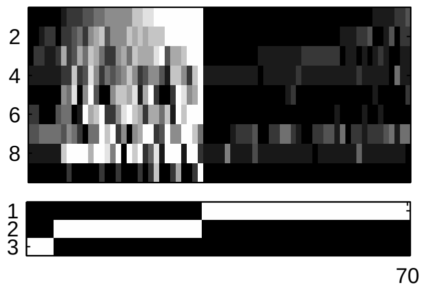


Figure 6.2: Individuals' output vs. inputs of the Breast cancer dataset with ensemble size = 3

The plots of experts' responsibilities against the inputs show that the Mixture of

Experts model can decompose the problem and assign individual experts to each region of the input space. In each run, the white bars in the lower box represent the experts' responsibilities, while the rows in the upper box display a color–map of the values of each input attribute. The x-axis is the pattern index. Figure 6.2 shows that the white bars (in the lower plot) distinctively divide the upper plot into regions with dark and light shade. This means the experts are distinctively responsible for different regions of the input space.

This figure is visually useful for simple datasets. For more complex datasets, it will be too complicated to allow us to analyze the relationship between the experts' responses and the inputs. Figure 6.3 is an example of a more complex dataset. Unlike the Breast cancer dataset, the Australian credit card dataset makes use of most of its components, as shown in most of the runs. The relationship between the experts' responsibilities and the input space becomes extremely difficult to detect in this kind of plot, because (i) the correlation between the inputs is not simple, and thus the image map is no longer useful, (ii) each expert is responsible for a small number of cases, and thus to detect the transition between the white bars and the corresponding transition in the input space is visually hard, and (iii) the relationship between the experts and the cases may not be a simple function.

6.1.2 Output vs principal components plot (pcapl)

In this thesis, I introduce a second type of plot based on *Principal Component Analysis* (PCA).

> The central idea of principal component analysis is to reduce the dimensionality of a data set in which there are a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This reduction is achieved by transforming to a new set of variables, the *principal components* (PC), which are uncorrelated, and which are ordered

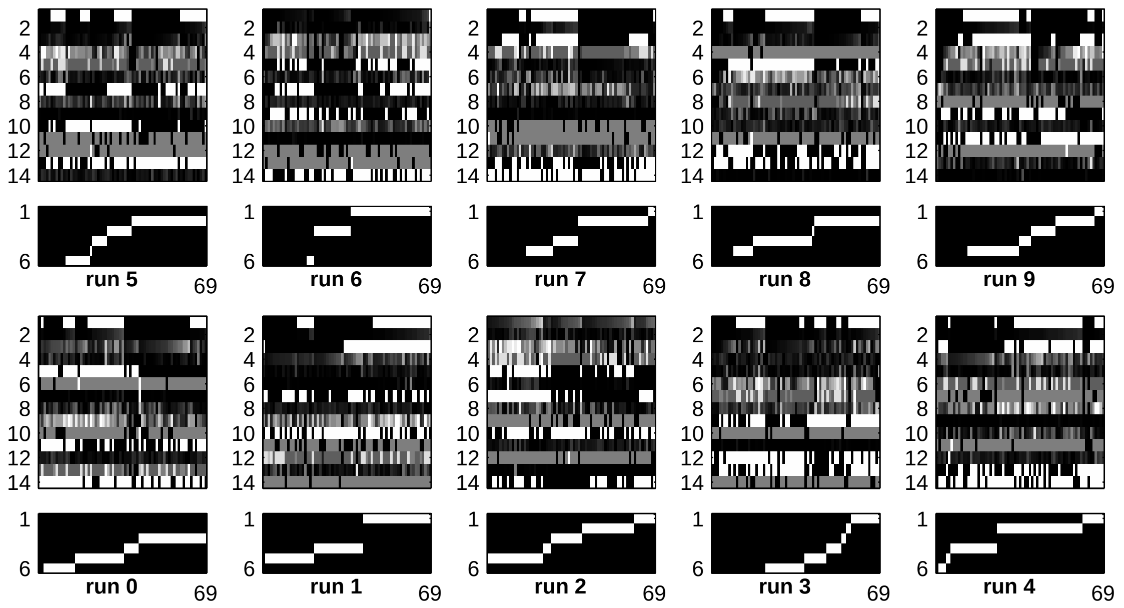


Figure 6.3: Individuals' output vs. inputs in 10 runs of the Australian credit card dataset with ensemble size = 6

> so that the first *few* retain most of the variation present in *all* of the original variables (Jolliffe 1986).

For a detailed derivation, and procedures for using PCA, readers are referred to (Jackson 1991; Jolliffe 1986). Here, I present a simple introduction. Suppose I have a set of vectors $\vec{x} = \{x_1, x_2, ..., x_p\}$ of p variables. What I am interested in, is to analyze the variances, covariances and correlations between these p variables in order to discover the interrelationships between these variables. If $p \gg 1$ then obviously it is very hard to find such relationships. However, in many cases, these variables are correlated, sometimes even highly correlated. The PCA method tries to remove these redundancies, and thus reduce the number of variables, while preserving the underlying information from the relationships between these variables.

The outputs of a PCA procedure are a new coordinate systems of p principal components, and the transformed data from the original to the new coordinate systems. This new coordinate systems is ordered so that the first variable accounts for most of the variation in the original data, and the last variable accounts for the least variation (Figure

6.4). In other words, only the first few k ($k \ll p$) variables, called principal components, are required to account for most of the information in the data. The obvious question is, how many principal components are sufficient to explain the variability in the data. The literature suggests a number of ways to decide on this number. One way is to look at the SCREE plot- a simple line segment plot that shows the fraction of total variance in the data as explained or represented by each principal component (Jolliffe 1986) - such as in Figure 6.4 and find the place where the bar curve levels off to the right of the plot (e.g. 2 in the left plot and 3 or 4 in the right plot). A cruder way is to find enough principal components to account for 60%-80% of the variability, depending on the problem. In many cases, between one and three principal components are sufficient.

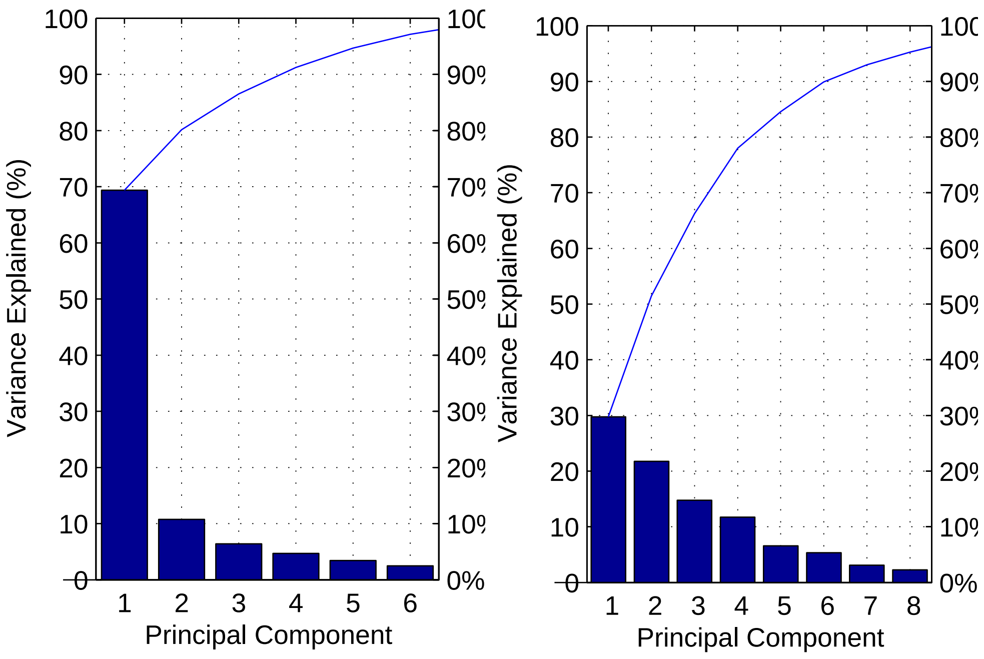


Figure 6.4: SCREE plots of the percent variability explained by each principal component. Left: Breast cancer, right: Diabetes

To assist our analysis of problem decomposition, PCA is first applied to reduce the number of inputs, and then a plot of the expert's responsibility based on the categorical value of the data against the first few principal components can be used to analyze the automatic problem decomposition property of ME. A maximum of three principal components is used in our analysis. They accounted in all our examples for at least 70%

of the variance in the data. With three principal components, one can visualize the data in 3D space, but here, to simplify the visualization, I plot three 2D views of the data against different pairs of principal components (i.e. PC1 vs PC2, PC1 vs PC3, and PC2 vs PC3).

Figure 6.5 shows an example of our pcapl plot, where different colors of the markers correspond to different experts' responsibilities; for example, red corresponds to expert 1 and blue to expert 2. The black points correspond to the cases where no expert is solely responsible for the data. The boundaries, using the convex hull algorithm in Algorithm 15, are shown together with the data, to give a clustering view of the data. "A convex hull of a set of points is the smallest convex set that contains the points" (Barber, Dobkin, and Huhdanpaa 1996).

The percentage displayed in the label of the subplot shows the percentage of variance of the inputs retained by the first three principal components. From the pcapl plot, one can see how the experts decompose the input space using the most important principal components.

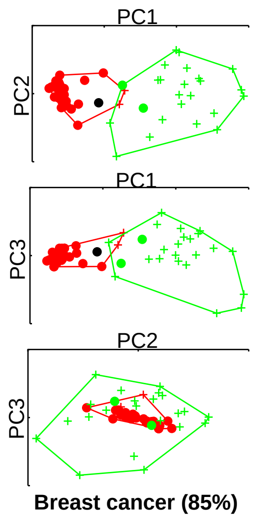


Figure 6.5: pcapl of experts'responsibility in principal component analysis for the Breast cancer dataset. Percentage shows how much variance of the original inputs is retained in the principal components. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes.

Algorithm 15 Quickhull algorithm for the convex hull in $\Re^d$
(Barber, Dobkin, and Huhdanpaa 1996)

```
1: create a simplex of $d + 1$ points
2: for each facet $F$ do
3:     for each unassigned point $p$ do
4:         if $p$ is above $F$ then
5:             assign $p$ to $F$'s outside set
6:         end if
7:     end for
8: end for
9: for each facet $F$ with a non-empty outside set do
10:     select the furthest point $p$ of $F$'s outside set
11:     initialize the visible set $V$ to $F$
12:     for all unvisited neighbors $N$ of facets in $V$ do
13:         if $p$ is above $N$ then
14:             add $N$ to $V$
15:         end if
16:         the set of horizon ridges $H$ is the boundary of $V$
17:         for each ridge $R$ in $H$ do
18:             create a new facet from $R$ and $p$
19:             link the new facet to its neighbors
20:         end for
21:         for each new facet $F'$ do
22:             for each unassigned point $q$ in an outside set of a facet in $V$ do
23:                 if $q$ is above $F'$ then
24:                     assign $q$ to the outside set of $F'$
25:                 end if
26:             end for
27:         end for
28:     end for
29: end for
```

6.2 Analysis of ME and CCME on UCI datasets using the proposed visualization tools

In this section, I apply the two proposed visualization tools, imgpl and pcapl, to analyze how ME and CCME automatically decompose real binary classification problems. The analysis is conducted on the experimental results from Chapter 5 for the Breast Cancer dataset, which is a relatively easy binary classification problem, and the Australian credit card dataset, which is a complex binary classification. These datasets are taken from the UCI Machine Learning Repository (Newman, Hettich, Blake, and Merz 1998).

6.2.1 Application of imgpl on the Breast cancer dataset with different ensemble sizes

ME

Figures 6.6 and 6.7 show the plots for the Breast cancer dataset with ensemble sizes of 3 and 9. The plots show a couple of interesting results. With larger ensemble, only a fraction of the ensemble is used. In fact, at most three experts are used for most of the runs. This is understandable, because Breast cancer is an easy dataset, therefore, it is not necessary to use a large ensemble with many experts to classify the dataset. With larger ensemble size, the split seems to be softer. In runs 0 and 2 of Figure 6.7, some experts are assigned to the same distinct regions of the input space; in the rest of the data, none of the experts has a high enough weight to be solely responsible for the pattern. Probably, ME finds a way to make use of the redundancy of experts, by jointly assigning them to the same patterns. Nevertheless, the plots give a good visual demonstration of the automatic problem decomposition characteristic of ME.

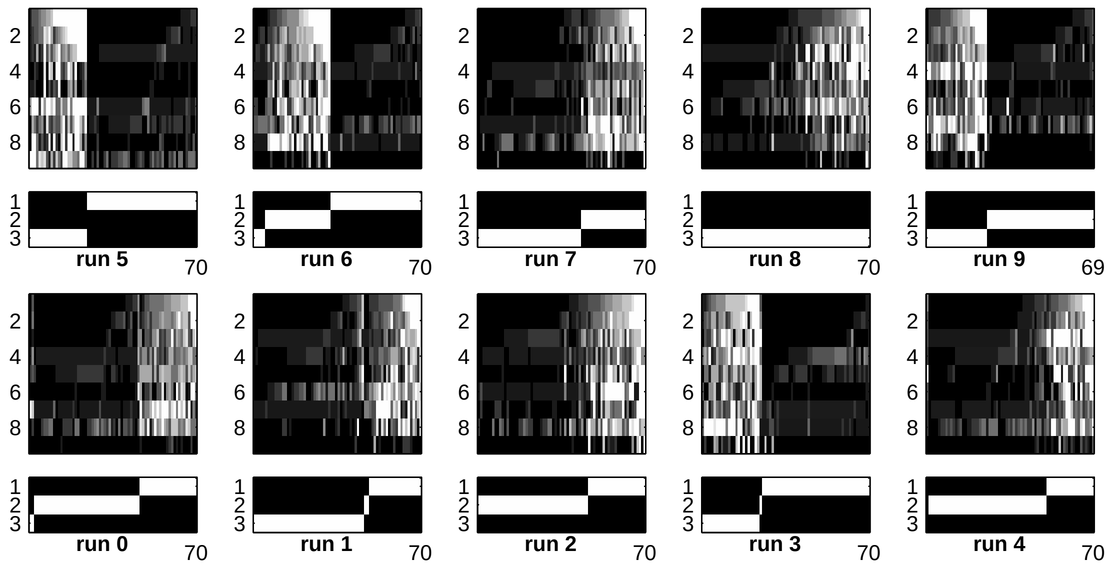


Figure 6.6: ME: imgpl of expert's responsibility vs. inputs in 10 runs of the Breast cancer dataset with ensemble size = 3

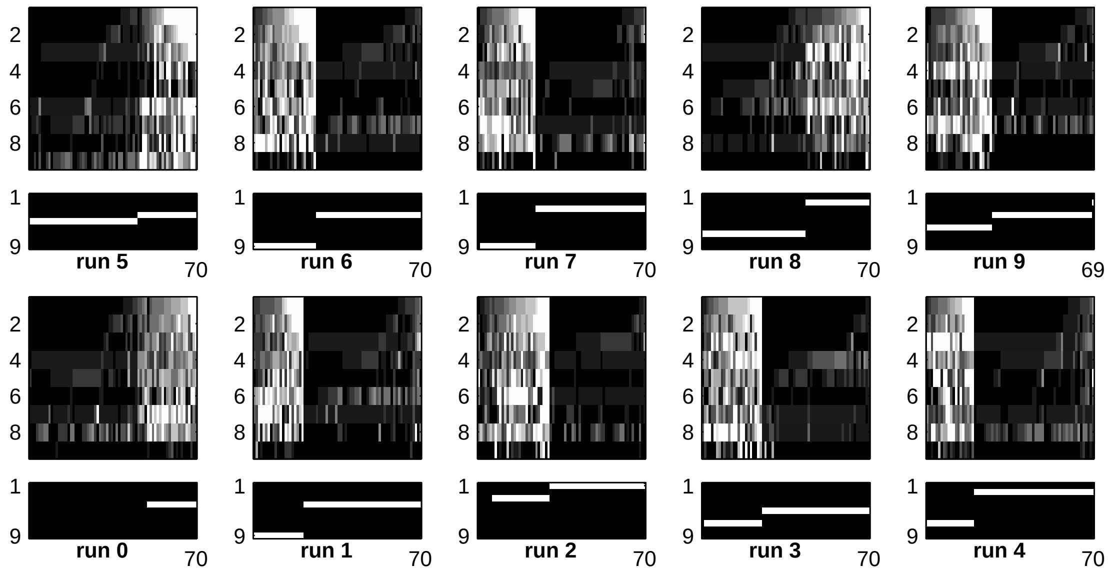


Figure 6.7: ME: imgpl of expert's responsibility vs. inputs in 10 runs of the Breast cancer dataset with ensemble size = 9

CCME

Similarly, the imgpl plots in figures 6.8 and 6.9 show how CCME, with ensemble sizes of 3 and 9, decomposes the Breast cancer dataset. Again, different experts are assigned to distinct regions of the input space. Since the Breast cancer dataset is simple enough, the model does not use all of the experts, as seen in figure 6.9. In fact, approximately two to three experts are enough to solve the problem.

6.2.2 Application of imgpl on a more complex classification problem: the case of the Australian credit card dataset

Figures 6.10 and 6.11 show the imgpl plots for ME and CCME on the Australian credit card dataset. Although Australian credit card is a difficult dataset, to some extent the plots still reveal how the models decompose the problems. Throughout the plots, it is obvious that some experts particularly respond to some inputs. For example, in run 0 of Figure 6.10, expert 1 is obviously responsible for inputs 1 and 5; in run 1, expert 3 is responsible for input 3.

6.2.3 Application of pcapl on the classification problems

Breast cancer dataset

Figures 6.12 and 6.13 display the pcapl of ME and CCME on the Breast cancer dataset respectively. The plots show that both ME and CCME are able to decompose the input space, represented by three principal components, into separate regions and assign different experts to these regions (recall different colors corresponding to different experts). One particular observation from the plot of the Breast cancer dataset is that, by reducing the input dimensions to three principal components, the dataset turns out to be modular: the cases of class ‘o’ mostly group together in one cluster while the cases of ‘+’ group in another cluster. Both ME and CCME are able to find this modularity and assign experts correctly to different clusters.

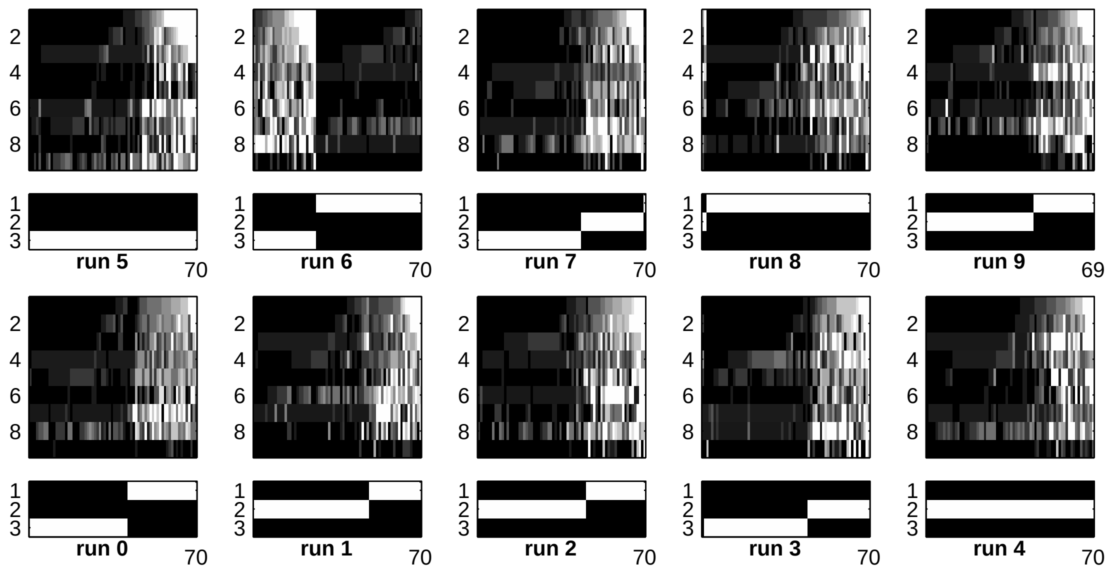


Figure 6.8: CCME: imgpl of expert's responsibility vs. inputs in 10 runs of Breast cancer dataset with ensemble size = 3

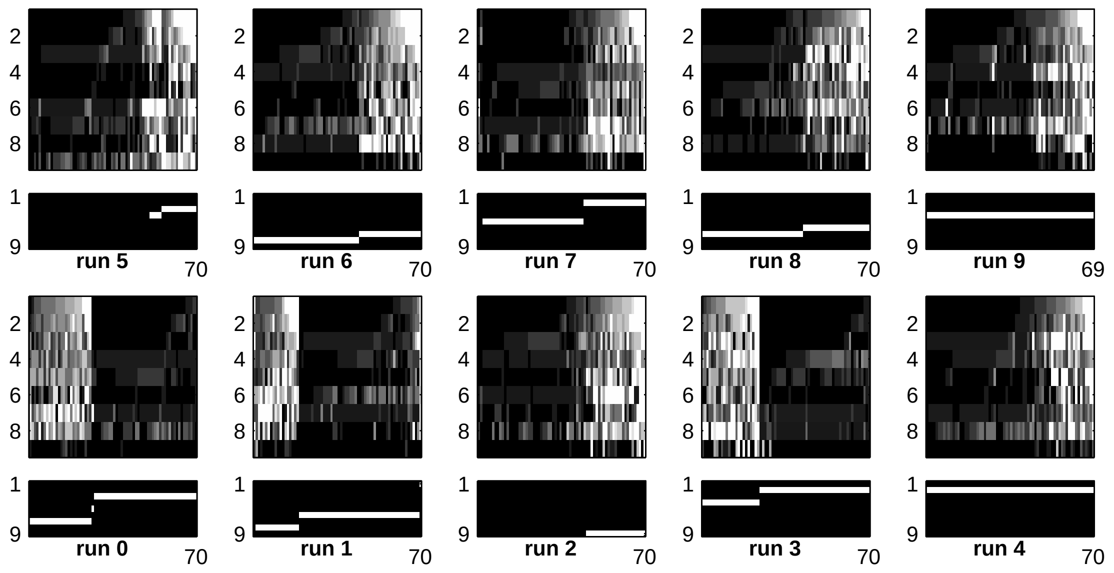


Figure 6.9: CCME: imgpl of expert's responsibility vs. inputs in 10 runs of the Breast cancer dataset with ensemble size = 9

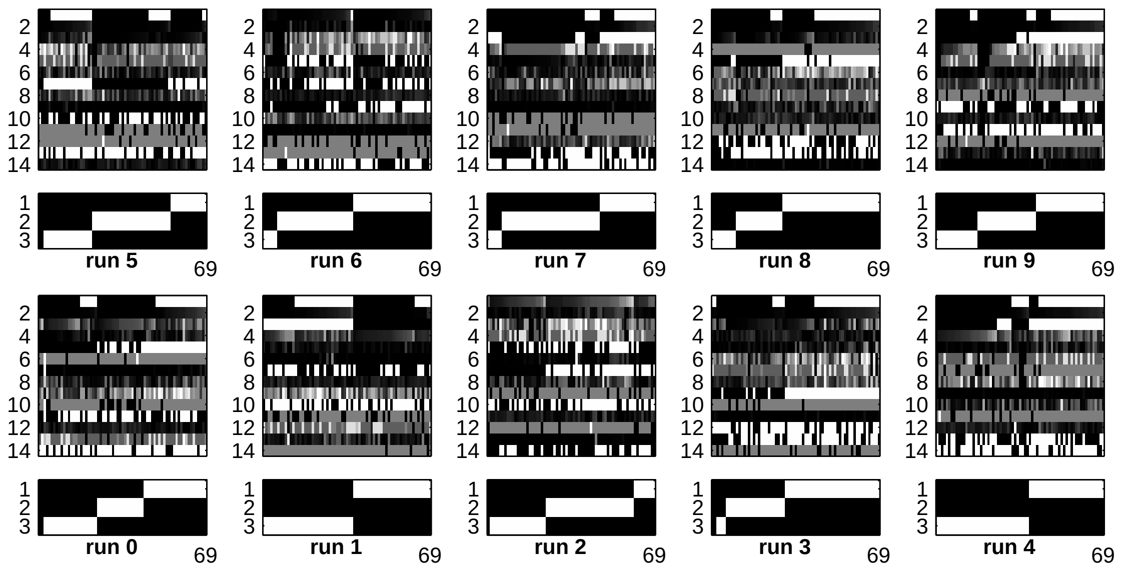


Figure 6.10: ME: imgpl of expert's responsibility vs. inputs in 10 runs of the Australian credit card dataset with ensemble size = 3

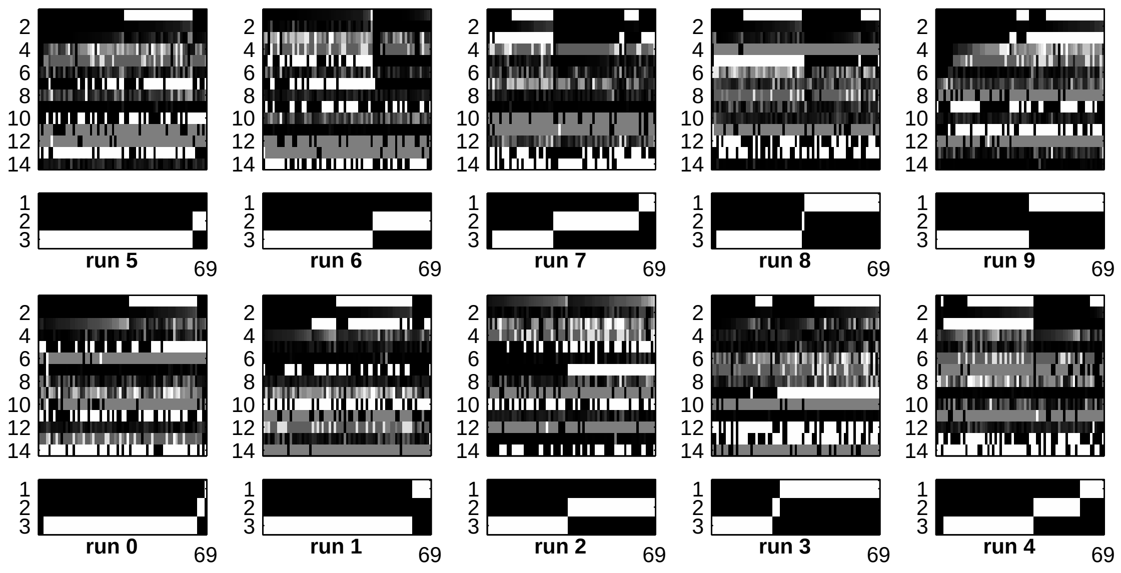


Figure 6.11: CCME: imgpl of expert's responsibility vs. inputs in 10 runs of the Australian credit card dataset with ensemble size = 3

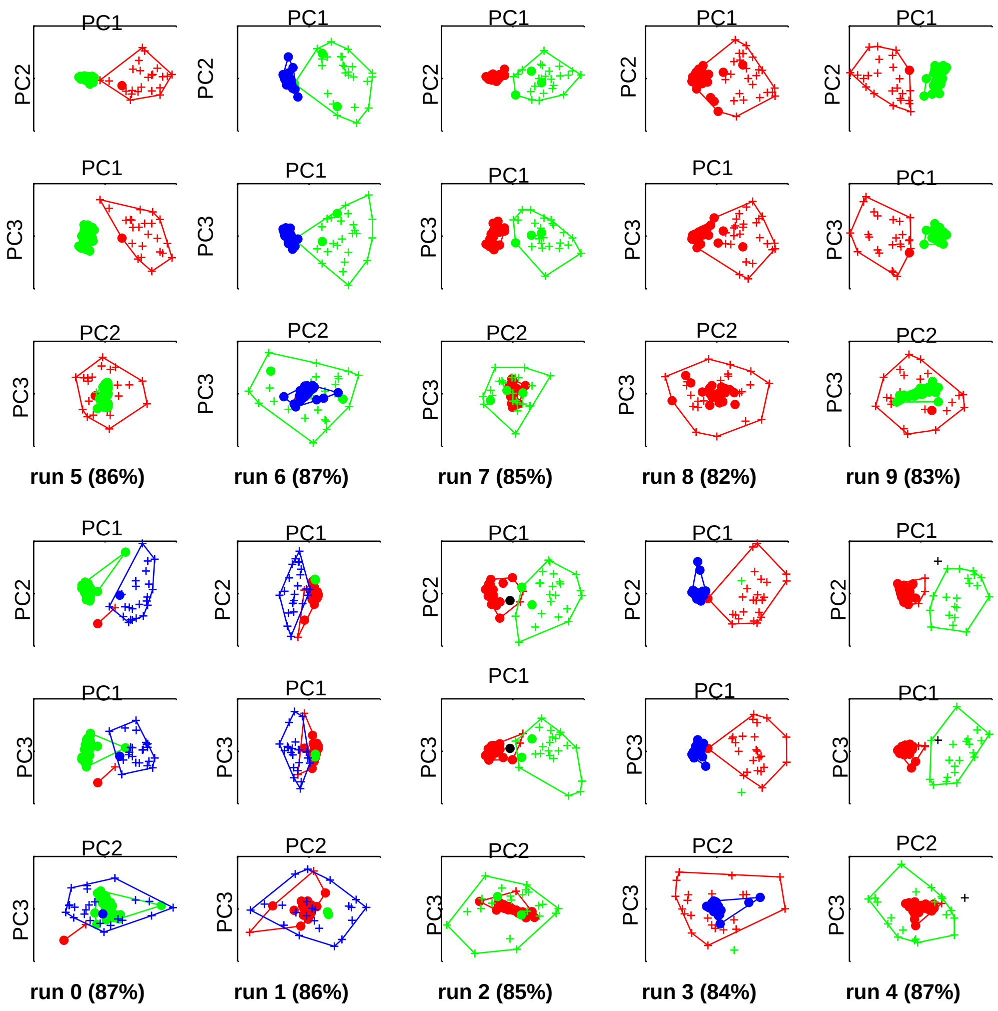


Figure 6.12: ME: pcapl plots of experts' responsibilities in 10 runs of the Breast cancer dataset. Percentage shows how much variance of the original inputs is retained in the principal components. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes.

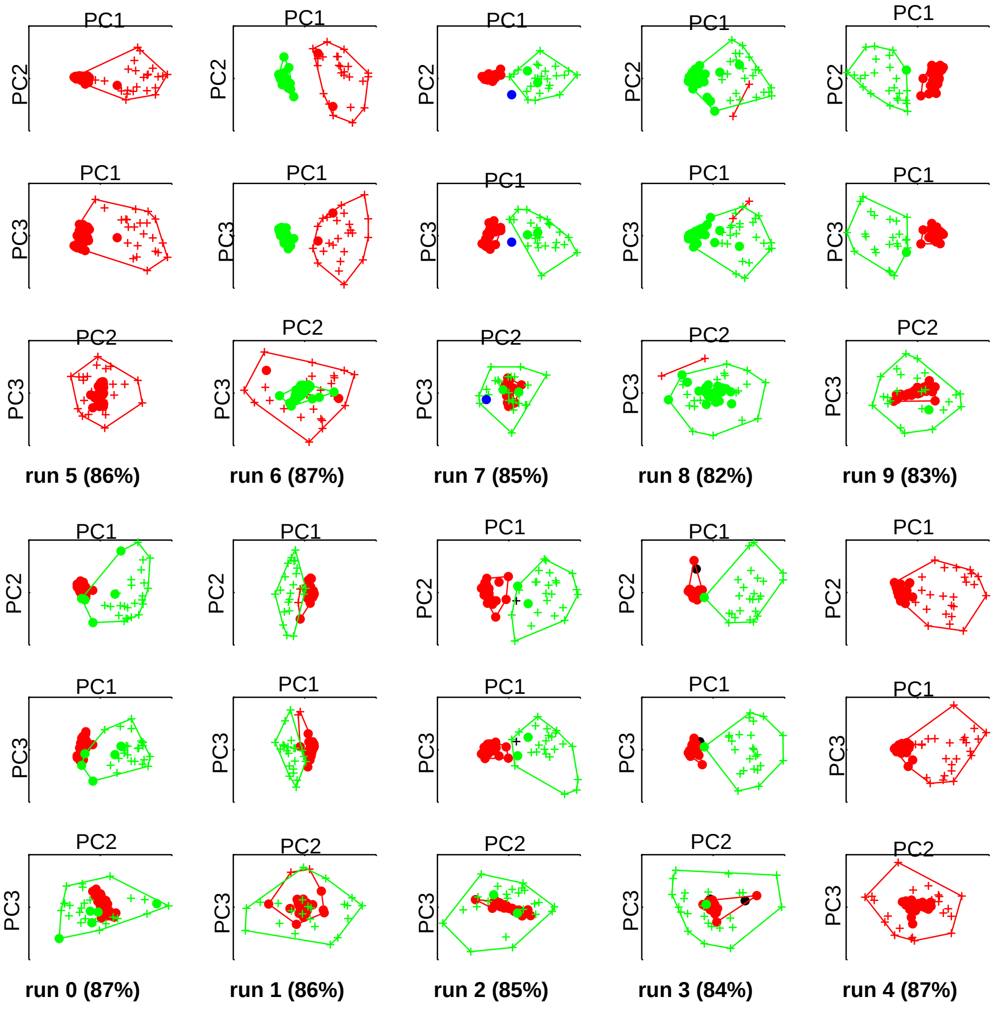


Figure 6.13: CCME: pcapl plots of experts' responsibilities in 10 runs of the Breast cancer dataset. Percentage shows how much variance of the original inputs is retained in the principal components. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes.

Australian credit card dataset

Figures 6.14 and 6.15 display the pcapl of ME and CCME on the Australian credit card dataset, one of the more complex classification problems. Although it is harder to view the APD property of the dataset, the plots still show that both ME and CCME are able to decompose the input space, represented by three principal components, into separate regions and assign different experts to these regions.

6.3 Analysis of CCME on artificial problems

In the previous section, I examined the value of the visualization tools in visualizing the APD properties of ME and CCME on real datasets. However, to gain a deeper understanding of the working mechanisms of CCME, it is desirable to work with a problem of lower dimensionality (in my case, 2D), while maintaining the key features of real datasets (i.e. reasonable difficulty). In this section, I use a 2D artificial problem to analyze the functioning mechanism of CCME. Although this artificial dataset has lower dimensionality, it remains a difficult problem because of the discontinuity in the class distribution.

6.3.1 Artificial data description

To analyze CCME, an artificial binary classification problem, as in Figure 6.16, is used. This artificial dataset has two attributes x_1 and x_2 where $-1 < x_1 < 1$ and $-1 < x_2 < 1$ and binary class labels. The eight class boundaries are defined by the following set of equations:

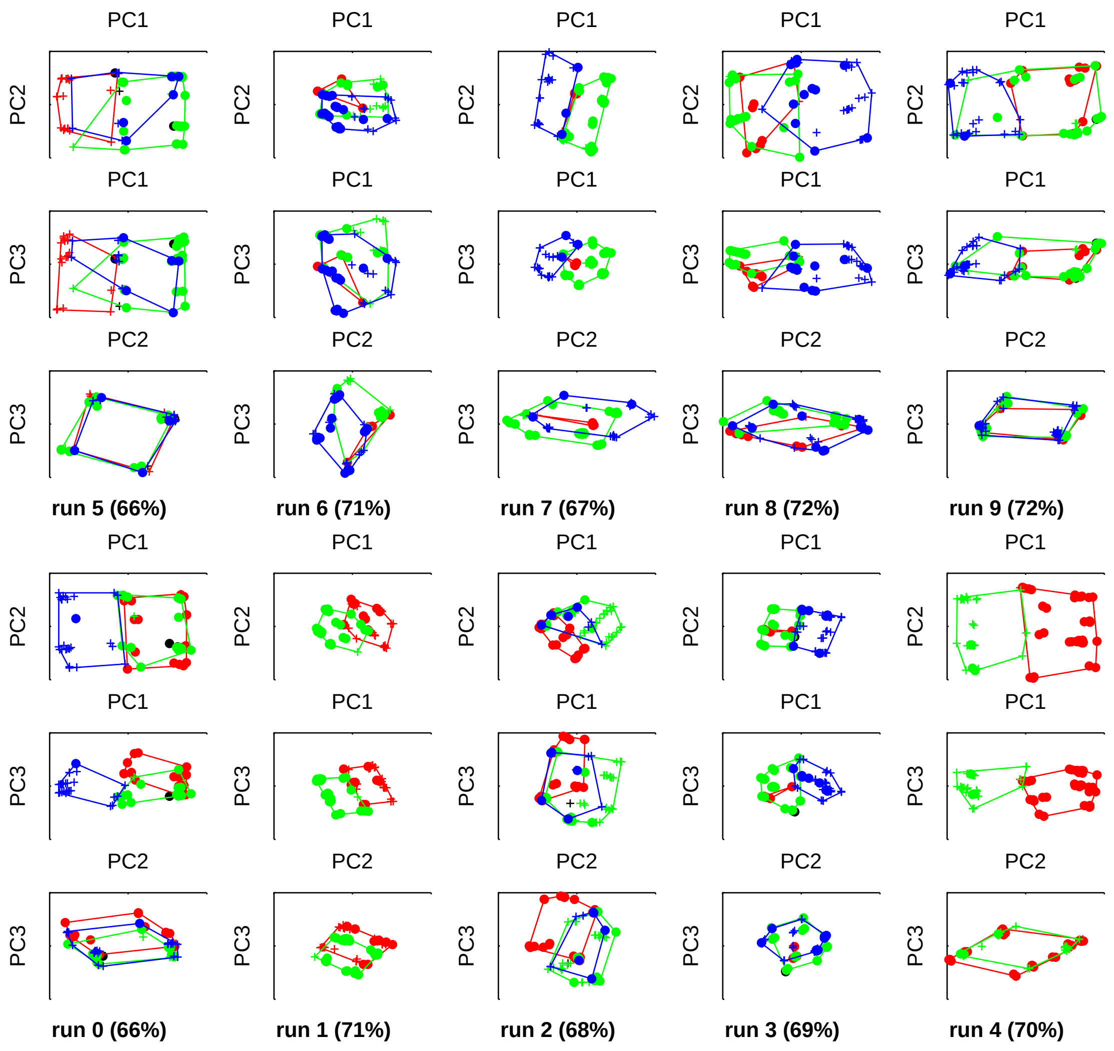


Figure 6.14: ME: pcapl plots of experts' responsibilities in 10 runs of the Australian credit card dataset. Percentage shows how much variance of the original inputs is retained in the principal components. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes.

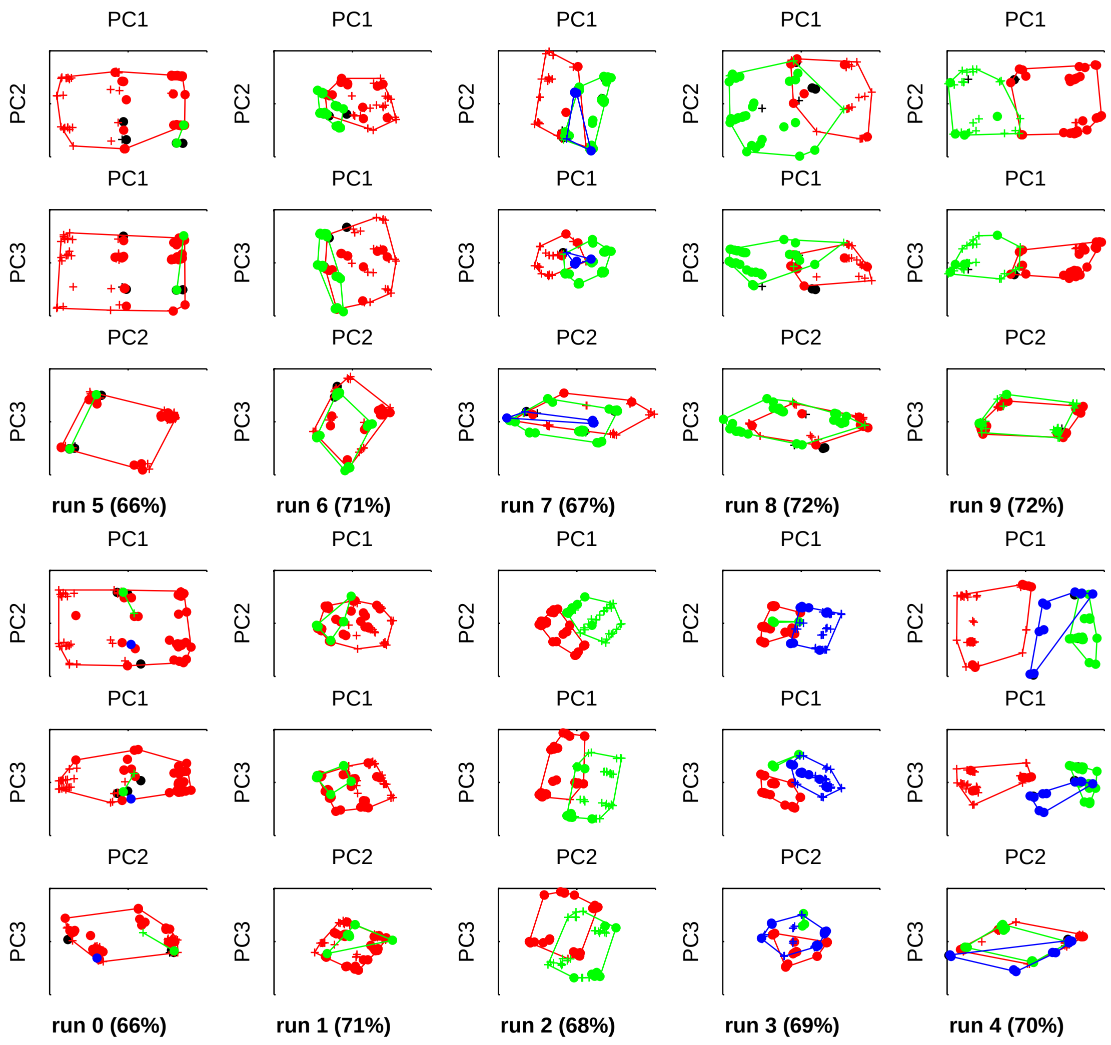


Figure 6.15: CCME: pcapl plots of experts' responsibilities in 10 runs of the Australian credit card dataset. Percentage shows how much variance of the original inputs is retained in the principal components. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes.

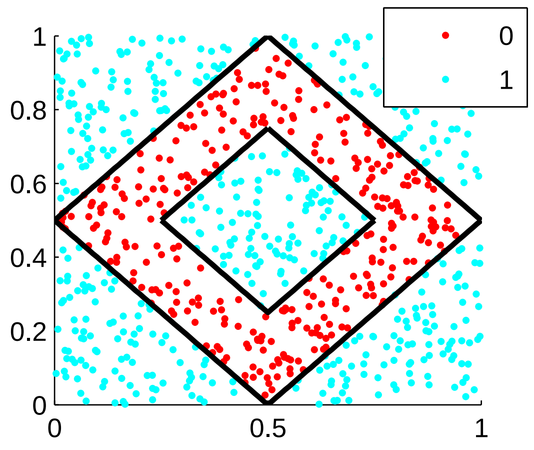


Figure 6.16: Artificial dataset with class boundaries

$$\begin{cases} x_1 - x_2 + 0.5 = 0 & x_1, x_2 \in [0,1] \\ x_1 + x_2 - 0.5 = 0 & x_1, x_2 \in [0,1] \\ x_1 - x_2 - 0.5 = 0 & x_1, x_2 \in [0,1] \\ x_1 + x_2 - 1.5 = 0 & x_1, x_2 \in [0,1] \\ x_1 - x_2 + 0.25 = 0 & x_1 \in [0.5, 0.75], \quad x_2 \in [0.25, 0.5] \\ x_1 + x_2 - 1.25 = 0 & x_1 \in [0.5, 0.75], \quad x_2 \in [0.5, 0.75] \\ x_1 - x_2 - 0.25 = 0 & x_1 \in [0.25, 0.5], \quad x_2 \in [0.5, 0.75] \\ x_1 + x_2 - 0.75 = 0 & x_1 \in [0.25, 0.5], \quad x_2 \in [0.25, 0.5] \end{cases}$$

The innermost and outermost regions contain records of class ‘1’ while the middle ring contains all records of class ‘0’. As mentioned before, the reasons for this particular setup are: (i) the problem can be viewed in two dimension and (ii) the problem is not easy in the sense that the class distribution is discontinuous.

6.3.2 Analysis of CCME on artificial data

As mentioned in (Sharkey 1998), the Mixture of Experts model not only decomposes the input space but also learns the relationship between the input and labels of the data. It is useful to plot and analyze these relationship in the same clustering view, to see how CCME takes into account the input-output relationship. Figure 6.17 shows the PCA plots of the CCME model on the artificial dataset. The artificial problem is 2D, which could be analyzed in its original form. To be consistent with the analysis throughout the chapter, I use PCA instead. Henceforth, the readers may notice discrepancies between the PCA-based images of problems and their original forms; for example, the positions of the decision boundaries may be rotated according to their principal components.

Again, the circle markers represent class '0' while the cross markers represent class '1'. The black markers are cases where no experts are dominantly responsible for the case. In extension to the proposed pcapl plot, described in section 6.1.2, I plot the predicted outputs instead of the true class labels to see how the experts perform in their designated regions. Moreover, to distinguish between the right and wrong classifications, yellow markers are used when the predicted labels are wrong (i.e. an error). For example, if the true label is '1', i.e. supposed to be a colored '+', and the predicted label is '0', a yellow 'o' is plotted instead of a colored '+'. In short, apart from the black and yellow markers, the color markers correspond to different experts in the ensemble.

To view how the whole system classifies the dataset, the predicted class boundaries are plotted as thick black lines. To plot the class boundary, a contour curve at level 0.5 is plotted. Figure 6.18 shows an example of how contour curves are located. A contour curve (or level curve) "of a function u: $\Re^2 \rightarrow \Re$ is a curve $g : [a, b] \rightarrow \Re^2$ such that $u(g(t)) = c$ for $t \in [a, b]$ where c is a constant" (Eriksson, Estep, and Johnson 2004). In this thesis, a contour curve at the threshold of the classification is plotted. Since there are two classes: 0 and 1, the threshold is selected at $c = 0.5$.

It is obvious that CCME learns to decompose the problem into distinct sub–regions of the input space. The joint responsibility of experts only occurs in the shared spaces between two different experts' regions (e.g. run 4 of Figure 6.17). In other words, the experts display strong confidence in their primary regions, and only join with each other in a few (possibly hard) shared cases.

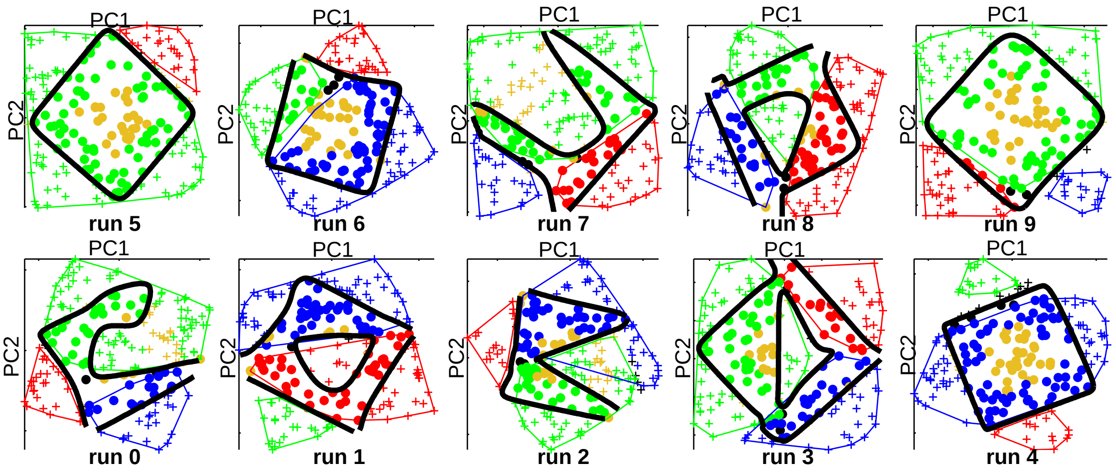


Figure 6.17: CCME with 3 experts and 3 hidden nodes on artificial data. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes. Yellow markers correspond to wrong outputs, thick black lines represent the predicted class boundaries.

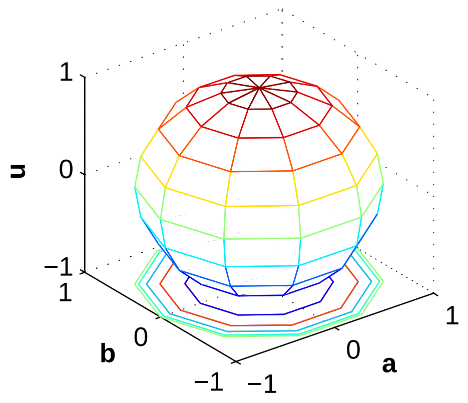


Figure 6.18: An example of the surface and contour plots of the sphere function. The contour curves $g(a, b)$ on the flat surface present different levels $u(g(a, b)) = c$ where c is a scalar.

As seen in Figure 6.17, CCME splits data into clusters in such a way that each cluster is easier to classify. The emergent class boundaries are definitely simpler than the original class boundaries. For example, looking closely at run 4 and run 7 (Figure 6.19(a)), the predicted class boundaries (running black lines) are obviously simpler than the true boundaries (both running and breaking black lines). A possible reason is that the classifiers, with only three hidden units, are insufficient to find better boundaries for complex clusters. However, looking at run 1 and run 8 (Figure 6.19(b)), the gate has split the data into easier clusters and thus the existing classifiers are able to classify the data with much higher accuracy.

These observations of the correlation between the ease of the sub–problems (i.e. clusters) and the accuracy of the system conform very well to the automatic problem decomposition principle: the system decomposes a complex problem into easier sub problems, which the available modules are able to handle with higher accuracy.

In the following sections, I investigate this APD nature of CCME by varying the network complexity and the model complexity. The network complexity is measured by the number of hidden units, and the model complexity by the number of experts. Intuitively, with sufficient complexity, the system should perform better. Sufficiency here means two possible things. First of all, providing that the system can reasonably divide the data into clusters, more complex experts should help improve the performance of the system because each expert can solve the data in its area of responsibility with a higher degree of accuracy. Secondly, equipped with reasonably simple classifiers, it is probable that the system with more components can find a better decomposition. The word “reasonably” is stressed here, because larger does not always mean better. Recalling the analysis in chapter 5, increasing ensemble size and different network complexity may not always significantly enhance the system’s performance.

6.3.2.1 CCME with various number of hidden units

To validate the correlation between the difficulty of the sub–tasks and the accuracy of the system, the same experiments are conducted with different network complexities, in terms of the number of hidden units per expert. The error rates and ANOVA tests are recorded in Table 6.1 and Figure 6.20. The best configurations lie between ten and fifteen hidden units per expert. Again, too small ANNs (e.g. 3 hidden units) and too large ANNs (e.g. 25 hidden units) are not desirable.

Figure 6.21 presents the pcapl of ensemble of 3 experts, consisting of 3, 10 and 25 hidden nodes. The plots support the observation that ANNs which are either too small or too large are not desirable. With 3 hidden nodes, the class boundaries are too simple to correctly classify the problem. With 25 hidden nodes, the class boundaries in most of the runs are quite close to the true boundaries. However, in some runs (i.e. run 0 and 6), CCME fails to generate more complex boundaries, so that the average error rate (out of ten runs) increases slightly. It is obvious that the best configuration lies somewhere between these two extreme cases. With 10 hidden nodes, CCME performs generally well in all of the runs, with the predicted class boundaries closely matching the true locations.

Moreover, the visualization motivates an important observation: more complex experts can generate more complex class boundaries, and hence, can classify the data in their clusters with higher accuracy. In the top set of plots (i.e. 3 hidden units), the individual experts can only produce simple class boundaries with continuous class distributions. For example, in run 7 of the 3 hidden units case, the green expert cannot produce the required U-shape for its cluster. Instead, it opens up a bridge by assuming a part of the data as class '+', which is wrong in this case. With larger ANN such as 10 hidden units, the experts can produce much more complex boundaries. As seen in most of the runs with 10 hidden units, at least one of the available three experts can classify discontinuous class distributions with high accuracy.

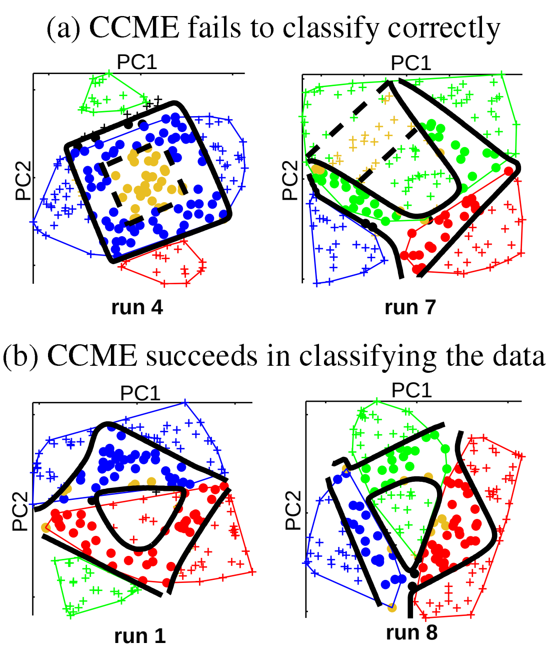


Figure 6.19: CCME on artificial datasets: failure and success. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes. Yellow markers correspond to wrong outputs, thick black lines represent the predicted class boundaries.

number of hidden units	error rate
3	0.126(0.042)
5	0.069(0.037)
7	0.052(0.034)
10	0.045(0.017)
15	0.048(0.017)
20	0.065(0.044)
25	0.076(0.045)
35	0.096(0.063)

Table 6.1: Mean (and standard deviation) of error rates of CCME with different numbers of hidden units on artificial data

6.3.2.2 CCME with various number of experts

In this section, I examine CCME with varying ensemble sizes of 3, 6, 10 and 15 experts, each expert consisting of three hidden nodes. Intuitively, with increasing ensemble sizes, the system can generate more clusters, so that each cluster is easier for a simple 3-hidden-unit classifier to classify. The error rates and ANOVA tests are recorded in Table 6.2 and Figure 6.22. The best configuration is six experts; adding more experts does not help the model to perform better. These results support the findings in chapter 5.

number of experts	error rate
3	0.126(0.042)
6	0.094(0.050)
10	0.098(0.043)
15	0.096(0.053)

Table 6.2: Mean (and standard deviation) of error rates of CCME with different number of experts on artificial data

Figure 6.23 presents the pcapl of ensemble of 3 and 6 experts, whose number of hidden units are fixed at 3. The plots show that by adding more experts, the model can generate more clusters. For example, with 6 experts, the model can generate up to 4 clusters, e.g. run 3 of 6 experts. However, when the number of experts grows too large, some of them either become redundant, or are used jointly to predict clusters which one alone could handle. An example is seen in the 15-expert experiments. The total number of clusters is limited to five clusters (5 different colors), so the other 10 experts are either redundant, or subordinately assist the five main ones.

One interesting observation from these plots is that the model can find very good ways to divide the data into better clusters. For example in run 3 of the middle plots, or in runs 1, 2, 5 and 8 of the bottom plots, the model can find the optimum division scheme to divide the data, and therefore the generated class boundaries are very close to perfect. However, in some runs, the model cannot find a good division scheme at all, and therefore classifies the data badly. The reason for this bad performance is unclear.

Nevertheless, these results raise a couple of interesting questions: (1) can a measure of the difficulty of a cluster be defined?, (2) can this kind of measurement of the quality of a cluster be used to guide the model towards dividing the data in a better way, and (3) can this measure be used to explain the differences in performance between methods of automatic problem decomposition? These questions remain open and serve as possible future directions.

6.4 Conclusion

In this chapter, I have proposed a number of tools to analyze CCME as an automatic problem decomposition method. I have also tested CCME on a complex artificial dataset to see how CCME performs in 2D problems. The key results can be summarized as follow. Firstly, the tools help to view how the model decomposes the data into regions of simpler input-output relationship. The tools show how ME and CCME can discover the modularity of the problems (if there is any). Moreover, with more complex problems, the tools still provide a sensible view of how the models apply the principle of divide-and-conquer, in which a complex problem is divided into simpler sub–tasks (clusters) to suit the available components (i.e. experts). Thirdly, there is a possible correlation between the input-output relationship in the clusters, and the system's performance, which may be useful to define clustering metrics based on input-output relationship as opposed to the normal purity measurement used in existing clustering techniques. I hypothesize that such a metric could be used as a guide, forcing CCME toward producing better clusters in terms of ease classification.

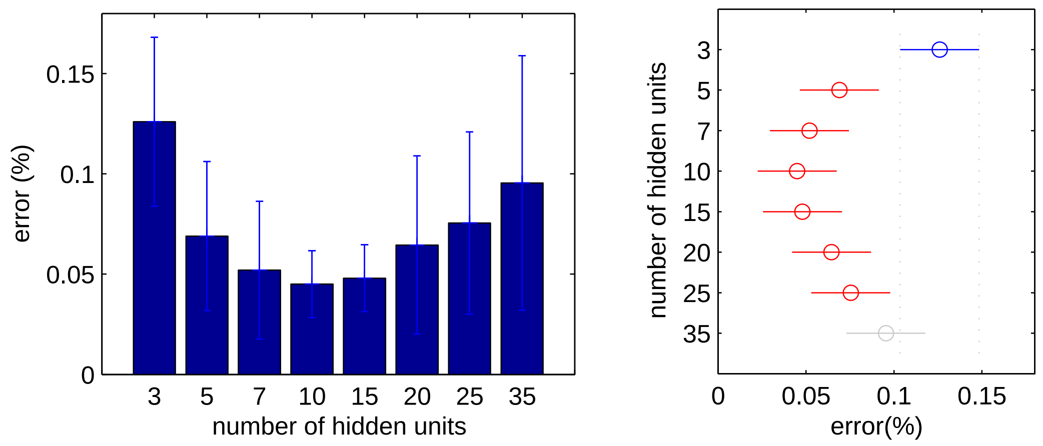


Figure 6.20: Error rate and ANOVA plots of CCME with different numbers of hidden units on artificial data. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

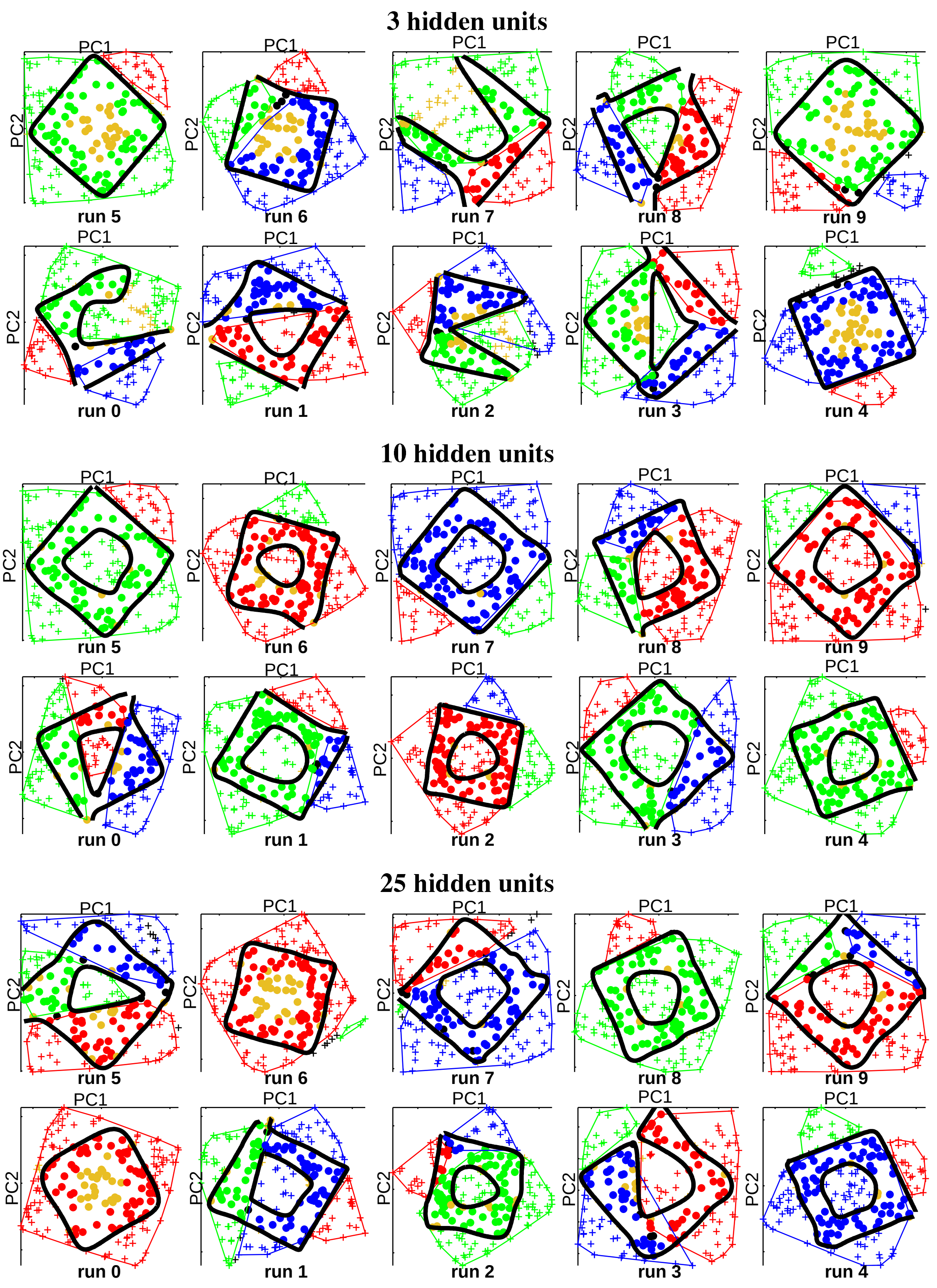


Figure 6.21: CCME on the artificial dataset with 3,10 and 25 hidden units. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes. Yellow markers correspond to wrong outputs, thick black lines represent the predicted class boundaries.

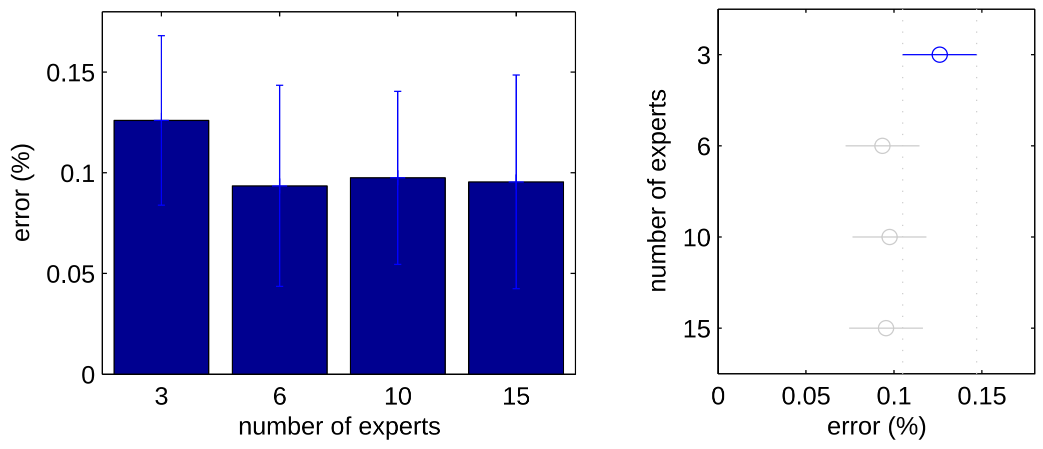


Figure 6.22: Error rate and ANOVA plots of CCME with different number of experts on artificial data. Overlapping intervals of groups' averages imply the groups are not significantly different; while disjoint intervals imply they are significantly different.

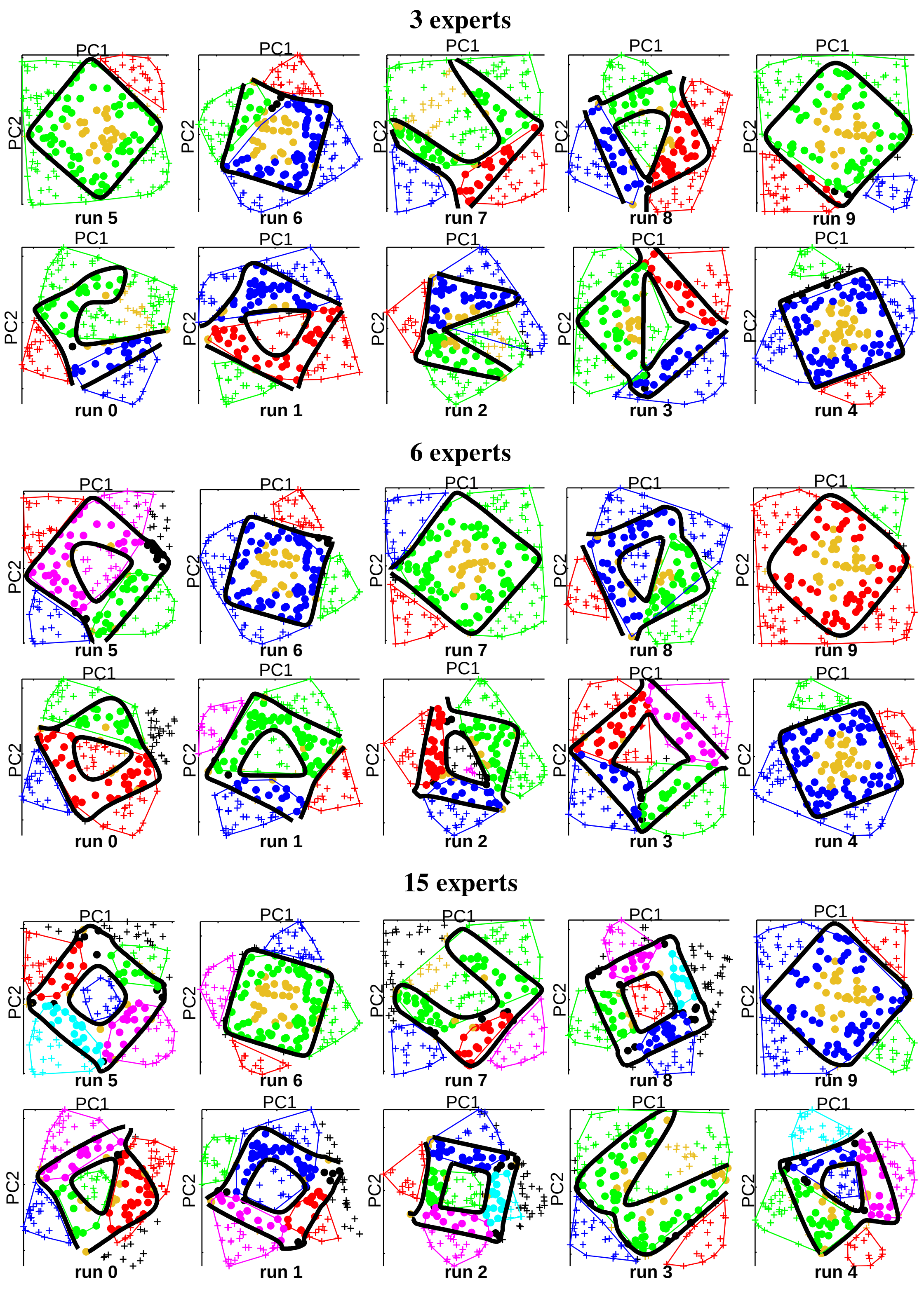


Figure 6.23: CCME on the artificial dataset with 3, 6 and 15 experts. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Marker types ('+' and 'o') represent different classes. Yellow markers correspond to wrong outputs, thick black lines represent the predicted class boundaries.

Chapter 7

Effects of Regularization on ME and CCME

In chapter 2, I discussed a number of regularization methods commonly applied to ANN. Among them are the methods of weight elimination (Weigend, Rumelhart, and Huberman 1990) and structural learning by forgetting of Ishikawa et al (Ishikawa 1996; Ishikawa and Yoshino 1993). In chapter 3, I examined some of the ANN regularization literature from the perspective of problem decomposition. In this chapter, I investigate the effects of the above two regularization methods on ME and CCME, to see if regularization (1) can reduce the structural complexity of ME and CCME while maintaining the generalization accuracy, and (2) is beneficial in enhancing the problem decomposition power of the models. A number of visualization and analysis tools are introduced, to analyze the effects of regularization on the system architecture, especially the magnitude and distribution of the weights.

7.1 Regularization revisited

A complex fitting with a high number of free coefficients tends to generate a mapping with high curvature and complex structure, as a result of over–fitting the noise in the training data (Bishop 1995). Figure 7.1 shows different boundaries obtained through using different networks with varying complexity for a classification problem (Bishop 1995). A too-simple structure results in a poor performance, since it is insufficient for the task; but a too-complex structure results in over–fitting the noise in the data. From a generalization point of view, therefore, it is more preferable to obtain an intermediate level of complexity sufficient for the problem. In fact, according to the well-known Occam's razor principle, a simpler model is preferable to a complex one, to the extent that one may trade off complexity against the fit of the model to the data (Bishop 1995).

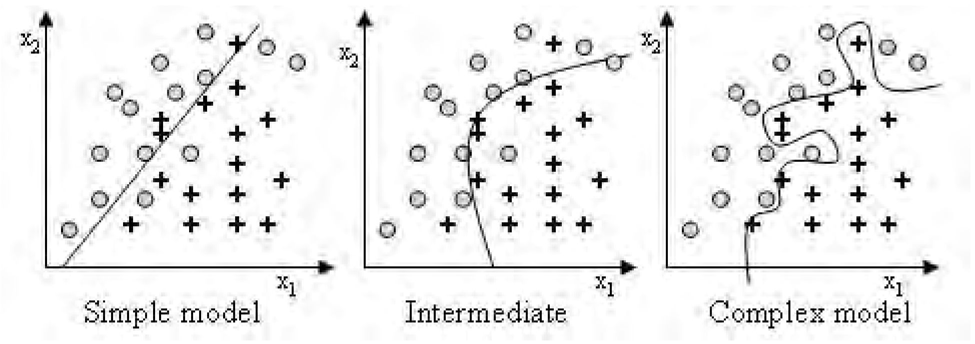


Figure 7.1: Different network complexity results in different class boundaries. The 'o' and '+' represent different classes while the unbroken lines/curves represent the predicted class boundaries.

One way to simplify an ANN is to add a penalty term, called the regularization term, to the cost function. This technique is generally known as regularization. The purpose of this regularization term is to encourage a smoother mapping, by penalizing a complex network, and thus to force the redundant weights towards zero. From another perspective, regularization removes redundant connections, and thus thins out the structure, resulting in minimal but effective ANNs.

In a regularized ANN, the cost function is usually formulated as follows

$$E = E_D + \epsilon E_R \tag{7.1}$$

where E_D is the usual error function, e.g. sum-square-error, and E_R is the regularization term. $\epsilon = \eta\epsilon'$ is the regularization parameter, which controls the degree of regularization, and η is the learning rate. The connection weights w_k are updated correspondingly as follows

$$\triangle w_k = -\eta \frac{\partial E_D}{\partial w_k} - \epsilon \frac{\partial E_R}{\partial w_k} \tag{7.2}$$

7.1.1 Learning by forgetting (LF)

LF was proposed in (Ishikawa and Yoshino 1993), in which the regularization term was defined as $E_R = \sum_{k=1}^{W} |w_k|$, where W was the number of weights w_k to be regularized. In fact, LF is analogous to the Laplace prior regularization method in the Bayesian framework, in that both use a Laplace distribution as the weight distribution. The regularization parameters can be estimated using the Bayesian framework (Goutte and Hansen 1997; Williams 1995; Williams 1993) or by other means such as adaptive, trial-and-error or cross-validation (Ishikawa 1996; Ishikawa and Yoshino 1993; Kozma, Kitamura, Malinowski, and Zurada 1995; Miller and Zurada 1997).

In this chapter, I use the original LF (Ishikawa 1996). The weight update is done by the following equation, where $\frac{\partial E_R}{\partial w_k} = sgn(w_k) = 1$ if $w_k \geq 0$ and -1 otherwise:

$$\triangle w_k = -\eta \frac{\partial E_D}{\partial w_k} - \epsilon sgn(w_k) \tag{7.3}$$

In LF, the regularization parameter ϵ can be thought of as the rate of forgetting. It is expected that a higher forgetting rate often causes more weights to be removed (i.e. forgotten) (Ishikawa and Yoshino 1993).

7.1.2 Weight elimination (WE)

Another popular regularization term in the literature of ANN is the modified weight decay, namely weight elimination (Weigend, Rumelhart, and Huberman 1990). In the weight elimination scheme, the regularization term is defined as $E_R = \sum_{k=1}^{W} \frac{w_k^2}{w_0^2 + w_k^2}$ where w_0^2 is a scaling parameter, often chosen as 1. This penalty term "tends to favor a few large weights rather than many small ones, so is more likely to eliminate weights from the network than is the simple weight-decay" (Bishop 1995). With $w_0^2 = 1$, the weight update is done by the following equation where $\frac{\partial E_R}{\partial w_k} = 2\frac{w_k}{(1+w_k^2)^2}$

$$\triangle w_k = -\eta \frac{\partial E_D}{\partial w_k} - \epsilon \frac{w_k}{(1+w_k^2)^2} \tag{7.4}$$

7.2 Regularization on ME and CCME

The aim of this chapter is to investigate the benefits of using regularization on ME and CCME. In the ME model, over–fitting can be considered at different levels. At the simplest level, the system, viewed as a very large ANN, can over–fit the data and thus not generalize well. This perspective considers all connection weights in the ensemble equally, regardless of whether they belong to the experts or the gate. At the second level, the gate network can be overtrained so as to poorly divide the problem, while each individual expert in the ensemble can be overtrained so that it memorizes the noise in its data region. This level deals with the experts and the gate as distinguished entities of ME. Different regularization terms and different regularization parameters may be used for the different components, i.e. experts and gate, of the system. At the highest level, a special regularization scheme involving parameters from both the experts and the gate may be used in a cooperative fashion on the whole system, instead of treating them separately as on the lower level. The reason is that the components of the ME function as inseparable parts of a whole system. Therefore the joint effects between them may be taken into account when considering regularization.

Although regularization can be applied at different levels of the hierarchy, I choose to experiment with the simplest level, in which the individual connection weights are updated by equation 7.1. In this case, the only difference between the regularized and un-regularized update lies in the extra regularization term, which is added to whatever weight update function was used in the previous chapter.

7.2.1 Regularization classes

Different priors are suitable for different types of weights found in FFNNs. Generally, the connection weights of a simple FFNN, without direct input-to-output connections, can be classified as (1) weights with hidden nodes as destinations (i.e. input-to-hidden and hidden-to-hidden weights), (2) weights with outputs as destinations (i.e. hidden-to-output connections) and (3) bias weights. For bias weights, it is generally argued that the prior should have constant density, so that biases should be excluded from regularization (Williams 1993; Williams 1995). In a FFNN, the hidden-to-output weights act purely as scaling factors for the outputs of the hidden units. In other words, an inactive hidden unit, due to its pruned inputs, should emit a very small output value, compared to the active ones. Thus with the same order of magnitude of the scaling weights, its contribution is insignificant. Therefore, it is not necessary to regularize these weights. Our networks have only a single hidden layer. Consequently, the only weights that need to be subjected to regularization in the following experiments are the input-to-hidden weights.

7.3 An experimental study of regularization on ME and CCME

I investigate the effects of two different regularization schemes, LF and WE, on ME and CCME. The purpose is not to find the best regularization scheme, but to test whether regularization of any kind is beneficial for ME and CCME.

7.3.1 Experimental setups

In the experiments, the same experimental setups as in chapters 5 and 6 are applied. The ensemble size is fixed to 3 experts, which are simple sigmoidal FFNNs with one hidden layer, consisting of three hidden nodes. The gating FFNN has no hidden layer, and consists of as many linear outputs as the number of experts. The back propagation algorithm with learning rate of 0.1 is employed to train the models. For CCME, the system is run for 200 generations; local search is performed for 10 epochs per generation. The same early stopping scheme, based on the minimum ensemble error on the validation set, is applied in this set of experiments. The hypothesis is verified on the same fifteen benchmark datasets as described in chapter 5.

For each regularization scheme, i.e. LF or WE, three different values, 10^{-5}, 10^{-4}, 10^{-3}, of the regularization parameter are used. These values of epsilon are consistent with the literature of Learning by Forgetting (Ishikawa 1996; Ishikawa and Yoshino 1993).

7.3.2 ME vs. regularized ME

7.3.2.1 On performance in terms of error rate

Figure 7.2 and tables 7.1 and 7.2 show the testing error rates, computed on unseen test data, of ME with LF and WE on fifteen ML datasets. The confidence columns in the tables show the statistical significance of the difference between the regularized and unregularized (i.e. $\epsilon = 0.0$) systems. Boldface indicates cases where regularization statistically significantly deteriorates the performance of the system, with confidence level of 90% and above. It is clear that, for most datasets (excepting LF with $\epsilon = 10^{-3}$ on Liver Disorder, Tic Tac Toe and King-rook-versus-king-prawn datasets), regularization does not significantly deteriorate the classification performance, even with high regularization coefficients.

7.3.2.2 On complexity in terms of the number of active connections

Since a minimal network often has better generalization ability than a complex one, regularization, especially LF, is designed to obtain this desired skeletal architecture. This is done by pushing the irrelevant connection weights towards zero, and maintaining only the relevant weights (Ishikawa and Yoshino 1993). Hence, an important measure

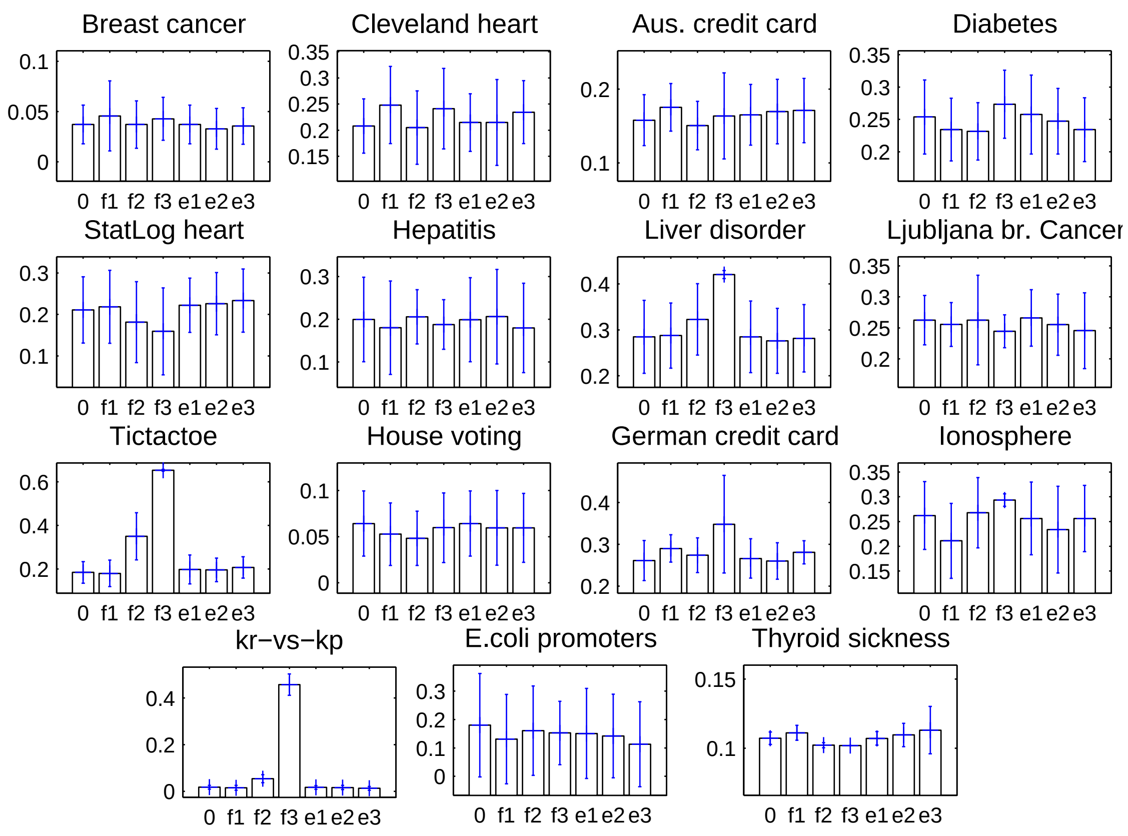


Figure 7.2: Error rate: ME with LF (denoted ‘f’) and weight elimination (denoted ‘e’) with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

Dataset	$\epsilon = 0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	0.037(0.019)	0.046(0.035)	0.037(0.023)	0.043(0.021)	75%	50%	87%
Cleveland Heart	0.208(0.052)	**0.248(0.074)**	0.205(0.070)	0.241(0.077)	**95%**	57%	84%
Australian credit card	0.158(0.034)	0.175(0.032)	0.151(0.033)	0.164(0.058)	88%	78%	59%
Diabetes	0.254(0.057)	0.234(0.048)	*0.232(0.044)*	0.273(0.052)	84%	*92%*	81%
StatLog Heart	0.211(0.080)	0.219(0.088)	0.181(0.098)	*0.159(0.105)*	64%	88%	*96%*
Hepatitis	0.200(0.099)	*0.180(0.109)*	0.206(0.064)	0.188(0.058)	*96%*	59%	63%
Liver Disorder	0.285(0.079)	0.288(0.071)	**0.323(0.077)**	**0.420(0.009)**	57%	**98%**	**100%**
Ljubljana Breast Cancer	0.263(0.040)	0.256(0.035)	0.263(0.072)	*0.245(0.026)*	66%	50%	*90%*
Tic Tac Toe	0.185(0.050)	0.180(0.061)	**0.351(0.108)**	**0.653(0.004)**	60%	100%	100%
House voting 84	0.064(0.035)	0.053(0.034)	*0.048(0.029)*	0.060(0.038)	86%	*97%*	65%
(ii) medium dataset							
German credit card	0.261(0.048)	**0.290(0.033)**	0.274(0.041)	**0.348(0.117)**	**98%**	75%	**96%**
Ionosphere	0.262(0.069)	*0.211(0.076)*	0.268(0.071)	**0.293(0.013)**	*100%*	58%	**93%**
King Root vs King Prawn	0.018(0.008)	0.015(0.011)	**0.054(0.017)**	**0.457(0.046)**	73%	**100%**	**100%**
E.coli Promoters	0.180(0.182)	*0.131(0.157)*	0.161(0.157)	0.153(0.112)	*93%*	75%	77%
Thyroid sickness	0.107(0.004)	**0.111(0.005)**	*0.102(0.002)*	*0.102(0.000)*	**92%**	*100%*	*100%*

Table 7.1: Error rates of LF with different values of ϵ for ME. The confidence columns show the statistical significance of differences between LF and the unregularized system. Italic and boldface indicate LF is significantly (90% and above) better or worse than the unregularized system respectively.

for regularization is the percentage of remaining (i.e. active) connections. It is noted that the weights are often not driven to absolute zero. In fact, with LF, tiny weights

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	0.037(0.019)	0.037(0.019)	*0.033(0.020)*	0.036(0.018)	0%	*90%*	66%
Cleveland Heart	0.208(0.052)	0.215(0.055)	0.215(0.082)	**0.235(0.060)**	83%	61%	**90%**
Australian credit card	0.158(0.034)	0.165(0.041)	**0.170(0.044)**	0.171(0.044)	84%	**93%**	87%
Diabetes	0.254(0.057)	0.258(0.061)	*0.247(0.051)*	*0.234(0.049)*	67%	*91%*	*98%*
StatLog Heart	0.211(0.080)	0.222(0.065)	0.226(0.075)	**0.233(0.076)**	83%	76%	**92%**
Hepatitis	0.200(0.099)	0.199(0.098)	0.206(0.110)	*0.180(0.105)*	52%	66%	*96%*
Liver Disorder	0.285(0.079)	0.285(0.078)	0.276(0.071)	0.282(0.073)	51%	73%	56%
Ljubljana Breast Cancer	0.263(0.040)	0.266(0.045)	0.255(0.049)	0.246(0.061)	83%	62%	83%
Tic Tac Toe	0.185(0.050)	0.198(0.066)	0.196(0.054)	**0.208(0.049)**	82%	76%	**97%**
House voting 84	0.064(0.035)	0.064(0.035)	0.060(0.040)	0.060(0.037)	0%	83%	70%
(ii) medium dataset							
German credit card	0.261(0.048)	0.266(0.047)	0.260(0.044)	0.281(0.028)	89%	55%	85%
Ionosphere	0.262(0.069)	0.256(0.073)	0.233(0.088)	0.256(0.067)	71%	89%	63%
King Root vs King Prawn	0.018(0.008)	0.017(0.008)	0.017(0.009)	*0.013(0.008)*	77%	75%	*94%*
E.coli Promoters	0.180(0.182)	*0.151(0.159)*	0.142(0.147)	*0.113(0.149)*	*90%*	89%	*98%*
Thyroid sickness	0.107(0.004)	0.107(0.005)	0.110(0.008)	0.113(0.017)	54%	83%	87%

Table 7.2: Error rates of WE with different values of ϵ for ME. The confidence columns show the statistically significance of the difference between WE and the unregularized system. Italic and boldface indicate WE is significantly (90% and above) better or worse than the unregularized system respectively.

often fluctuate between $(-\epsilon, +\epsilon)$ instead of zero. However, for normalized data, if ϵ is small enough, the weights whose magnitude is bound by $|\epsilon|$ do not have any effect on the overall performance, and thus, the corresponding connections can be physically removed. In this thesis, the connections are not removed physically. Instead, a threshold of 10^{-3} is used to differentiate the active and inactive weights, i.e. a weight is said to be active if its absolute value is greater than this threshold.

Figures 7.3 and 7.4 respectively show the averaged percentage of active weights for the experts, and for the gate, of ME. The results and the statistical tests in tables 7.3 and 7.4 indicate that LF significantly reduces the number of weights. The results and the statistical tests in tables 7.5 and 7.6 show that WE can also push the connection weights towards zero, although it is not as effective as LF. In fact, only with a high regularization coefficient ($\epsilon = 10^{-3}$), is WE able to thin out the network architecture (with significance levels of 90% and above). Smaller regularization parameters do not significantly affect the structural complexity of the ANN, in terms of the number of connection weights.

Section 7.3.2.1 shows that any effect of regularization on ME's performance(in terms of error rates) is not statistically significant. This section shows that regularization does affect the structural complexity of ME by removing the redundant connection weights. These observations strongly suggest that regularization is beneficial to ME, by finding a minimal architecture - hence, less training time if the redundant connections are removed - while maintaining the generalization performance. This finding fits well into the literature of ANN, in which regularization is often found to be beneficial to the generalization ability by reducing the complexity of the ANN (Bishop 1995).

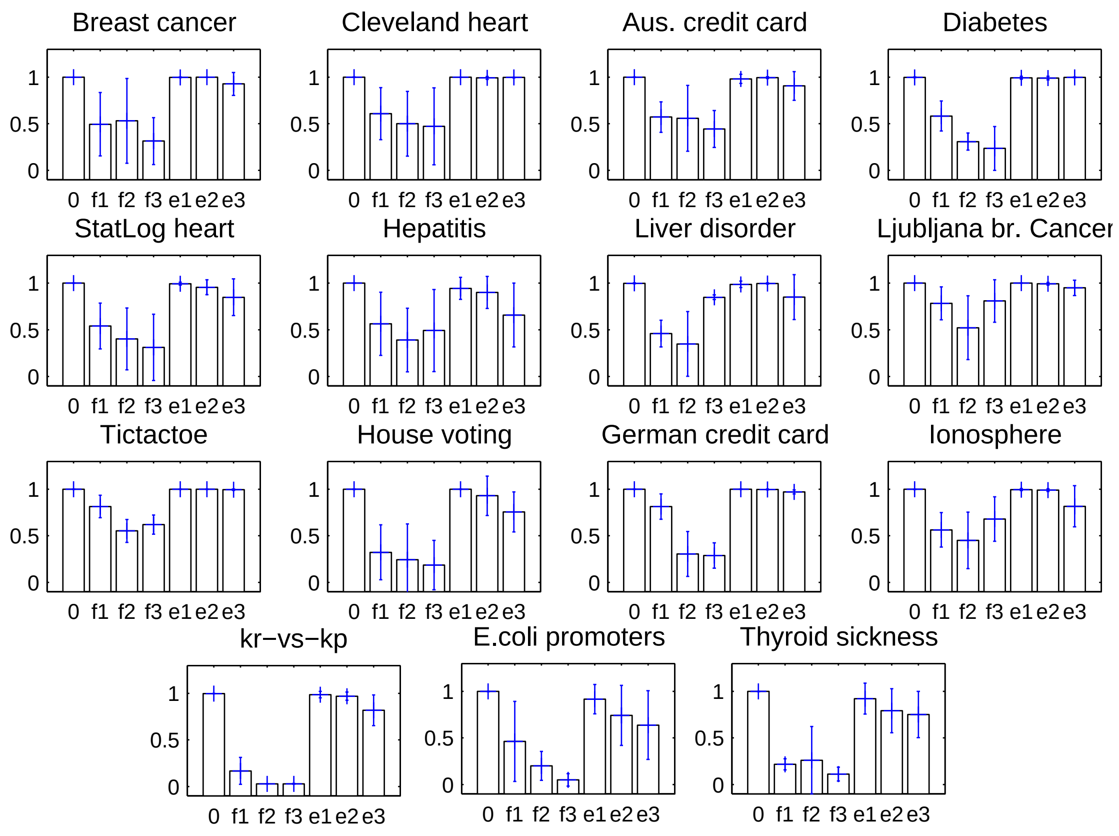


Figure 7.3: Percentage of active weights for the experts of ME with LF (denoted ‘f’) and weight elimination (denoted ‘e’) with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

7.3.2.3 On comparison between LF and WE

Figure 7.5 plots the percentage of active weights (i.e. structural complexity) versus the generalization error rates (i.e. performance) for six different sets of regularization – 3 regularization parameters for LF and 3 for WE – on fifteen datasets. The scatter plot shows that with the current set up and parameters, LF is more effective than WE in

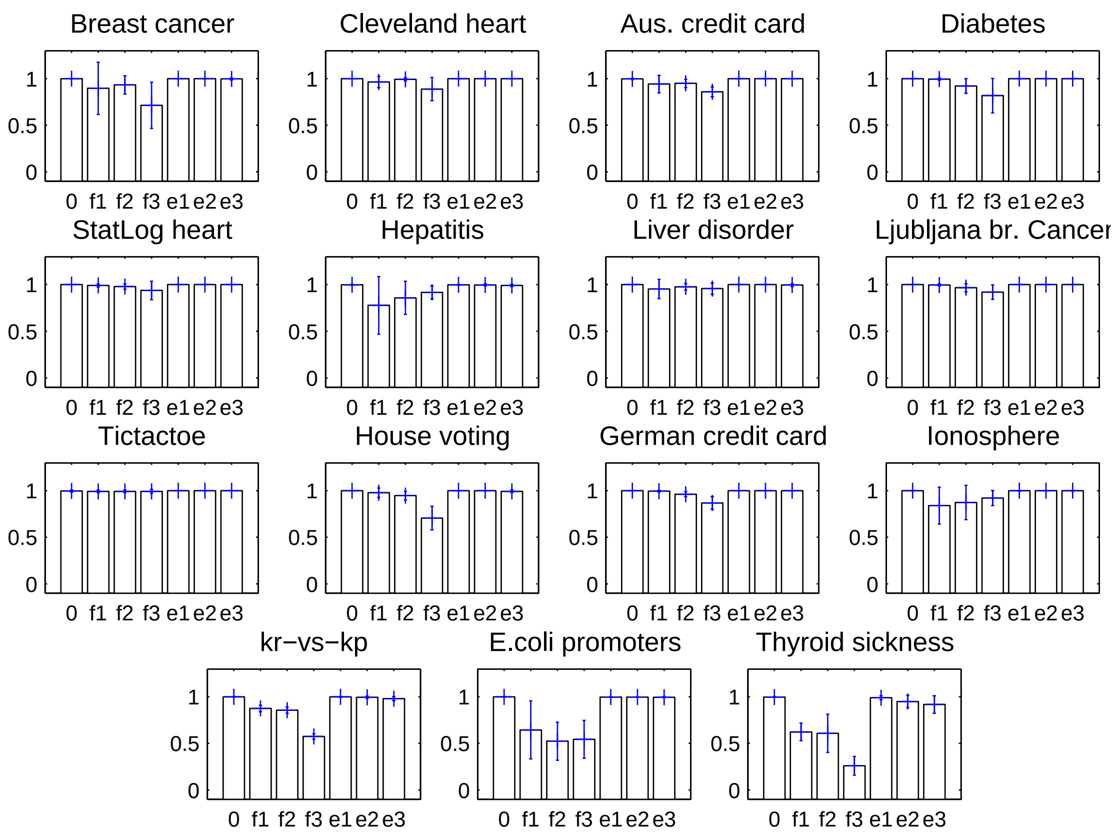


Figure 7.4: Percentage of active weights for the gate of ME with LF (denoted ‘f’) and weight elimination (denoted ‘e’)with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	0.494(0.341)	0.531(0.455)	0.313(0.253)	100%	100%	100%
Cleveland Heart	0.999(0.003)	0.608(0.279)	0.500(0.347)	0.472(0.413)	100%	100%	100%
Australian credit card	0.999(0.002)	0.572(0.163)	0.559(0.353)	0.444(0.198)	100%	100%	100%
Diabetes	0.999(0.004)	0.583(0.162)	0.307(0.092)	0.235(0.234)	100%	100%	100%
StatLog Heart	1.000(0.000)	0.541(0.246)	0.402(0.331)	0.311(0.355)	100%	100%	100%
Hepatitis	1.000(0.000)	0.564(0.339)	0.390(0.341)	0.492(0.440)	100%	100%	100%
Liver Disorder	0.998(0.005)	0.459(0.143)	0.348(0.347)	0.849(0.025)	100%	100%	100%
Ljubljana Breast Cancer	1.000(0.000)	0.783(0.176)	0.522(0.342)	0.809(0.228)	100%	100%	99%
Tic Tac Toe	1.000(0.000)	0.814(0.120)	0.552(0.124)	0.621(0.102)	100%	100%	100%
House voting 84	0.999(0.002)	0.322(0.295)	0.242(0.386)	0.186(0.265)	100%	100%	100%
(ii) medium dataset							
German credit card	1.000(0.000)	0.814(0.136)	0.306(0.241)	0.288(0.136)	100%	100%	100%
Ionosphere	0.999(0.002)	0.564(0.185)	0.450(0.303)	0.680(0.239)	100%	100%	100%
King Root vs King Prawn	0.998(0.002)	0.168(0.145)	0.027(0.000)	0.028(0.004)	100%	100%	100%
E.coli Promoters	1.000(0.001)	0.463(0.430)	0.200(0.155)	0.049(0.066)	100%	100%	100%
Thyroid sickness	1.000(0.001)	0.217(0.060)	0.261(0.362)	0.112(0.075)	100%	100%	100%

Table 7.3: Percentage of active weights of LF with different values of ϵ for experts of ME. The confidence columns show how statistically significantly LF is different from the unregularized system.

reducing the structural complexity, while maintaining a similar level of accuracy. In the next section, I analyze different effects of LF on the weights of ME.

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	0.897(0.281)	0.933(0.097)	0.713(0.248)	86%	97%	100%
Cleveland Heart	1.000(0.000)	0.964(0.059)	0.993(0.016)	0.888(0.123)	96%	90%	99%
Australian credit card	0.998(0.007)	0.942(0.095)	0.949(0.043)	0.860(0.052)	95%	100%	100%
Diabetes	1.000(0.000)	0.996(0.012)	0.922(0.079)	0.819(0.185)	83%	99%	99%
StatLog Heart	1.000(0.000)	0.990(0.017)	0.979(0.029)	0.938(0.099)	95%	98%	96%
Hepatitis	0.998(0.005)	0.778(0.309)	0.858(0.178)	0.918(0.069)	98%	98%	100%
Liver Disorder	1.000(0.000)	0.952(0.103)	0.976(0.034)	0.957(0.061)	91%	97%	97%
Ljubljana Breast Cancer	1.000(0.000)	0.997(0.011)	0.967(0.044)	0.920(0.076)	83%	98%	100%
Tic Tac Toe	0.997(0.011)	0.993(0.014)	0.993(0.014)	0.993(0.021)	70%	70%	66%
House voting 84	1.000(0.000)	0.978(0.049)	0.947(0.046)	0.706(0.126)	90%	100%	100%
(ii) medium dataset							
German credit card	1.000(0.000)	0.996(0.009)	0.963(0.027)	0.869(0.067)	90%	100%	100%
Ionosphere	1.000(0.000)	0.839(0.198)	0.874(0.183)	0.922(0.082)	98%	97%	99%
King Root vs King Prawn	1.000(0.000)	0.875(0.035)	0.857(0.033)	0.574(0.029)	100%	100%	100%
E.coli Promoters	1.000(0.000)	0.645(0.312)	0.524(0.204)	0.544(0.203)	100%	100%	100%
Thyroid sickness	0.999(0.004)	0.624(0.094)	0.608(0.205)	0.260(0.101)	100%	100%	100%

Table 7.4: Percentage of active weights of LF with different values of ϵ for the gate network of ME. The confidence columns show how statistically significantly LF is different from the unregularized system.

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	0.999(0.004)	1.000(0.000)	0.928(0.123)	83%	0%	95%
Cleveland Heart	0.999(0.003)	1.000(0.000)	0.992(0.015)	0.997(0.004)	83%	94%	96%
Australian credit card	0.999(0.002)	0.982(0.049)	0.995(0.009)	0.906(0.154)	85%	92%	96%
Diabetes	0.999(0.004)	0.994(0.016)	0.991(0.019)	0.999(0.004)	81%	86%	50%
StatLog Heart	1.000(0.000)	0.994(0.015)	0.956(0.080)	0.848(0.197)	86%	94%	98%
Hepatitis	1.000(0.000)	0.944(0.118)	0.901(0.172)	0.658(0.342)	92%	95%	99%
Liver Disorder	0.998(0.005)	0.987(0.035)	0.997(0.007)	0.851(0.242)	82%	70%	96%
Ljubljana Breast Cancer	1.000(0.000)	1.000(0.000)	0.993(0.014)	0.950(0.082)	0%	92%	96%
Tic Tac Toe	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.996(0.008)	0%	0%	95%
House voting 84	0.999(0.002)	1.000(0.000)	0.930(0.212)	0.758(0.215)	83%	84%	100%
(ii) medium dataset							
German credit card	1.000(0.000)	1.000(0.000)	0.998(0.004)	0.972(0.021)	0%	89%	100%
Ionosphere	0.999(0.002)	0.996(0.011)	0.991(0.014)	0.817(0.220)	75%	94%	99%
King Root vs King Prawn	0.998(0.002)	0.987(0.035)	0.970(0.044)	0.819(0.165)	83%	96%	100%
E.coli Promoters	1.000(0.001)	0.917(0.157)	0.742(0.323)	0.638(0.369)	94%	98%	99%
Thyroid sickness	1.000(0.001)	0.922(0.165)	0.792(0.237)	0.752(0.249)	91%	99%	99%

Table 7.5: Percentage of active weights of WE with different values of ϵ for experts of ME. The confidence columns show how statistically significantly WE is different from the unregularized system.

7.3.2.4 Analyzing LF on ME

As seen above, LF has the effect of pushing irrelevant weights towards zero. In this section, I use the Diabetes dataset to analyze the effects of LF on ME.

7.3.2.4.1 Effect of the regularization parameters on the magnitude of weights Figure 7.6 shows a typical effect of LF on the weights' magnitude. The weights' magnitude is plotted on a log_{10} scale. The solid line at 10^{-3} represents the potential threshold for physical weight pruning (this is not actually implemented as it is viewed as beyond the

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.997(0.011)	0%	0%	83%
Cleveland Heart	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Australian credit card	0.998(0.007)	1.000(0.000)	1.000(0.000)	1.000(0.000)	83%	83%	83%
Diabetes	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
StatLog Heart	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Hepatitis	0.998(0.005)	0.998(0.005)	0.997(0.011)	0.992(0.012)	50%	66%	92%
Liver Disorder	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.995(0.015)	0%	0%	83%
Ljubljana Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Tic Tac Toe	0.997(0.011)	1.000(0.000)	1.000(0.000)	1.000(0.000)	83%	83%	83%
House voting 84	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.994(0.013)	0%	0%	90%
(ii) medium dataset							
German credit card	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Ionosphere	1.000(0.000)	0.999(0.003)	1.000(0.000)	0.999(0.003)	83%	0%	83%
King Root vs King Prawn	1.000(0.000)	0.999(0.003)	0.995(0.012)	0.981(0.021)	83%	90%	99%
E.coli Promoters	1.000(0.000)	0.998(0.004)	0.999(0.002)	0.997(0.006)	90%	92%	96%
Thyroid sickness	0.999(0.004)	0.990(0.021)	0.951(0.066)	0.918(0.093)	87%	97%	99%

Table 7.6: Percentage of active weights of WE with different values of ϵ for the gate network of ME. The confidence columns show how statistically significantly WE is different from the unregularized system.

scope of this thesis). In each subplot, different colors, in separate blocks, represent different experts of ME. I show here the weights of the experts (lower plots) and of the gate (upper plots). As expected, with higher forgetting rates, more weights are pushed towards zero. As seen in Figure 7.6, in the case of no regularization (i.e. $\epsilon = 0.0$), the weights of both the experts and the gate have magnitudes of order 1 (i.e. 10^0). In the intermediate subplots, an increase in the regularization parameters corresponds to a decrease in the overall weights' magnitudes, for both the experts and the gate. With a heavy regularization of $\epsilon = 10^{-3}$, most of the experts' weights are pushed below the threshold (10^{-3}) and those of the gate reduce at least an order of magnitude (i.e. $10^{-2} - 10^{-1}$)

Figure 7.7 plots the error rates and weight distributions, in a histogram, of the experts and gate versus different forgetting parameters for the Diabetes dataset.

The plots show some very interesting observations. First, the plots confirm the findings in the literature of ANN regularization, that higher regularization parameters tend to push more weights toward lower magnitude (Bishop 1995; Ishikawa 1996; Ishikawa and Yoshino 1993; Kozma, Kitamura, Malinowski, and Zurada 1995; Miller and Zurada 1997). For both the gate and the experts, the weight magnitudes with high ϵ, are at least one order smaller than the unregularized case.

Secondly, the plots also confirm the findings in the original structural learning

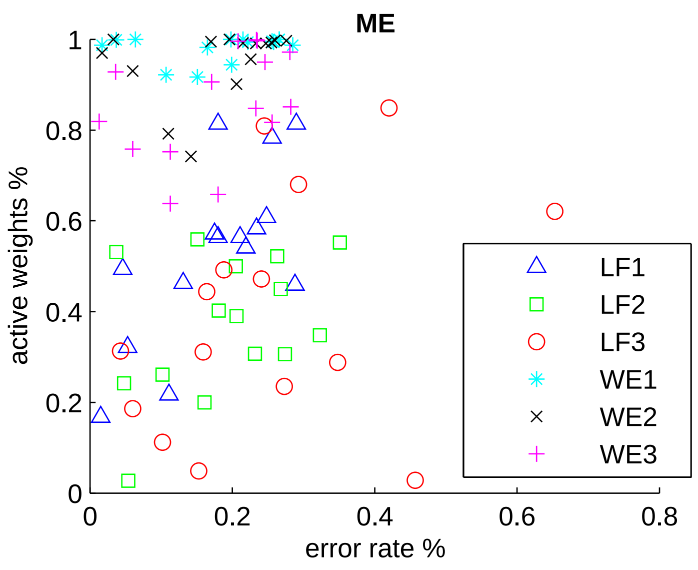


Figure 7.5: Percentage of active weights vs. error rate for ME with LF and WE and three different regularization values $\{1, 2, 3\} \equiv \{10^{-5}, 10^{-4}, 10^{-3}\}$

by forgetting studies, that irrelevant weights are pushed toward zero, while the rest are responsible for error minimization (Ishikawa 1996; Ishikawa and Yoshino 1993; Kozma, Kitamura, Malinowski, and Zurada 1995; Lozowski, Miller, and Zurada 1996; Miller and Zurada 1997). This can be seen clearly from the plot of the experts' weights. For $\epsilon > 10^{-5}$, the weight histogram has two peaks: one at the order of $10^0 = 1$, the other at a much lower order, for example, with $\epsilon = 10^{-5}$, the two peaks are at 10^0 and 10^{-5}.

Thirdly, looking at the weight distribution plots, one can distinguish three different phases: (1) when ϵ is small enough, $\epsilon \leq 10^{-6}$, the weight distributions follow a sort of bell shape around a median value of 10^0, (2) when ϵ increases to a certain range, $10^{-6} \leq \epsilon \leq 10^{-2}$, the weight distributions split into two peaks, one peak stays at 10^0, while the other monotonically decreases, and (3) when ϵ is high enough, $\epsilon \geq 10^{-2}$, the weight magnitude starts to increase again. The second phase can be divided further into two sub phases: (2a) both the weights of the experts and gate decrease, and (2b) the experts' weights increase while the gate's weights still decrease.

In terms of the error rates, in phase (1), the error rate curve is relatively flat with a low value. In phase (3), where ϵ is high, the error curve is also relatively flat with

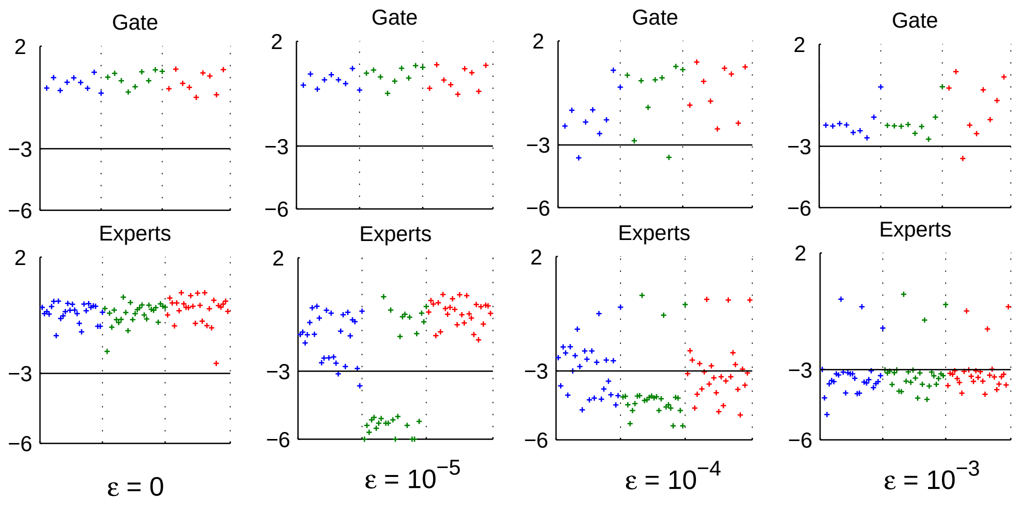


Figure 7.6: An example of LF on ME with different regularization parameters ϵ. y-axis: $log_{10}|w|$. x-axis: weight number. 10^{-3}: cutoff value. Different colors refer to different experts in ME.

a high value. The interesting phase is the intermediate one, where the curve displays a phase transition. In the Diabetes dataset, it is interesting to see that the starting and ending points of each phase exactly correspond to the changes in the error rates.

In phase (1), the regularization parameters are too small for the regularization term to have much impact on the weights of the system. Therefore, these weights are similar to the weights of the unregularized case. From phase (2a) onwards, the penalty term starts to have an effect on the weight distribution, and also on the generalization error. As in figure 7.7, with increasing ϵ, the error rate slightly decreases while the weight magnitudes of both the gate and the experts are quickly reduced.

In phase (2b), the majority of the weights of the experts are low, and most likely do not contribute to the functioning of the system. Meanwhile, some of the gate weights have a large magnitude. Thus, this observation and the discussion about feature selection below suggest that in this phase, the gate is in charge of the system performance.

Finally, the only difference between phases (2b) and (3), in terms of the weight distribution, is a phase change in the weight magnitude of the gate, from mostly in the higher peak (phase (2b)) towards the lower peak (phase (3)). This observation suggests that with high values of ϵ, i.e. phase (3), the gate is over regularized and therefore works

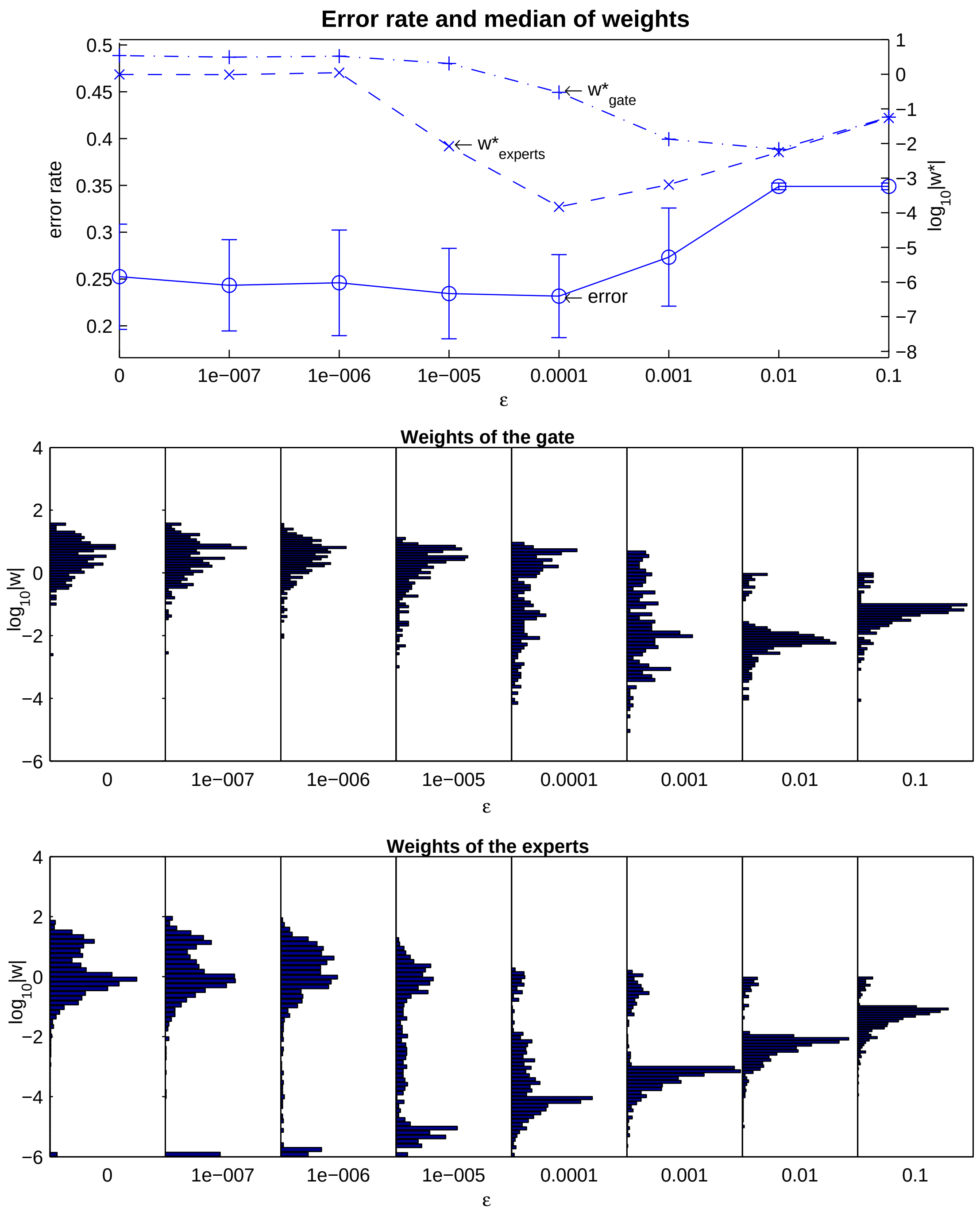


Figure 7.7: ME: weight distribution versus regularization parameter ϵ for Diabetes dataset. Top plot: error rate and median ($w*$) of the weight distributions of ME experts and gate, middle: weight distribution of ME experts, bottom: weight distribution of ME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

inefficiently with the error function, consequently causing a high error rate.

To confirm these observations, similar experiments and plots are conducted on the House Voting 84 dataset (Figure 7.8) and the Ionosphere dataset (Figure 7.9). Although the phase changes in the error rate and the weight distribution plots are not as clear as in the Diabetes dataset, they are still observable in these two datasets. Furthermore, the weight histograms for these datasets show similar characteristics as in the Diabetes dataset.

The discussion in this section opens a number of future research questions: (1) is there a relationship between the weight distribution and the generalization error?, (2) can phase changes in the weight distribution be used as proxies for phase changes in the generalization errors? and (3) if there is such a relationship, can one find an optimized weight distribution which produces an optimized neuro ensemble, in terms of its generalization ability?

Figure 7.10 shows a typical architecture of the ensemble with different forgetting rates. It is expected that with higher forgetting rates (e.g $\epsilon = 10^{-3}$), the network becomes more sparse, with many connections being removed.

7.3.2.4.2 The effect of the regularization parameters on relevant input features

Figure 7.11 shows the ensemble architectures with LF ($\epsilon = 10^{-4}$) on ME. With $\epsilon = 10^{-4}$, LF reduces the error rate from 25.4% to 23.2% (significant level = 92%).

First, it is interesting to note that some input attributes do not participate at all in the ensemble, as shown by the absence of links from those inputs to any hidden units of the three experts. For example in runs 0,4, and 5 in Figure 7.11, none of the inputs are connected to any hidden units of any expert, except for the bias node (the last node in the input layer of each expert). In other words, in these runs, the regularized ME depends upon the gate to cluster the data into self-similar clusters (in terms of class purity), in which the responsible expert will always give the same class. That is, experts do not classify but simply propose a constant class. Looking closely at the cluster plots, - based

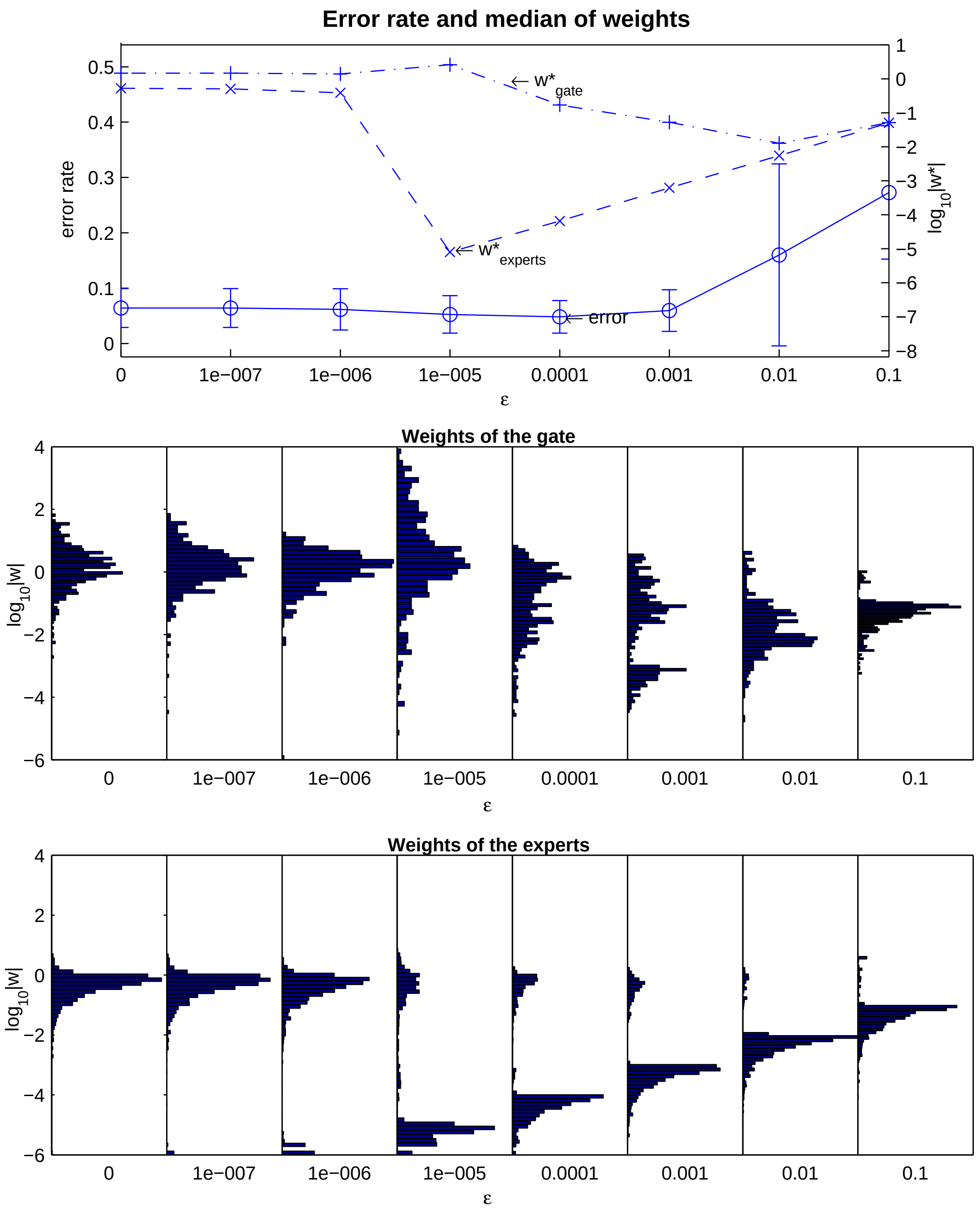


Figure 7.8: ME: weight distribution versus regularization parameter ϵ for House Voting 84 dataset. Top plot: error rate and median $(w*)$ of the weight distributions of ME experts and gate, middle: weight distribution of ME experts, bottom: weight distribution of ME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

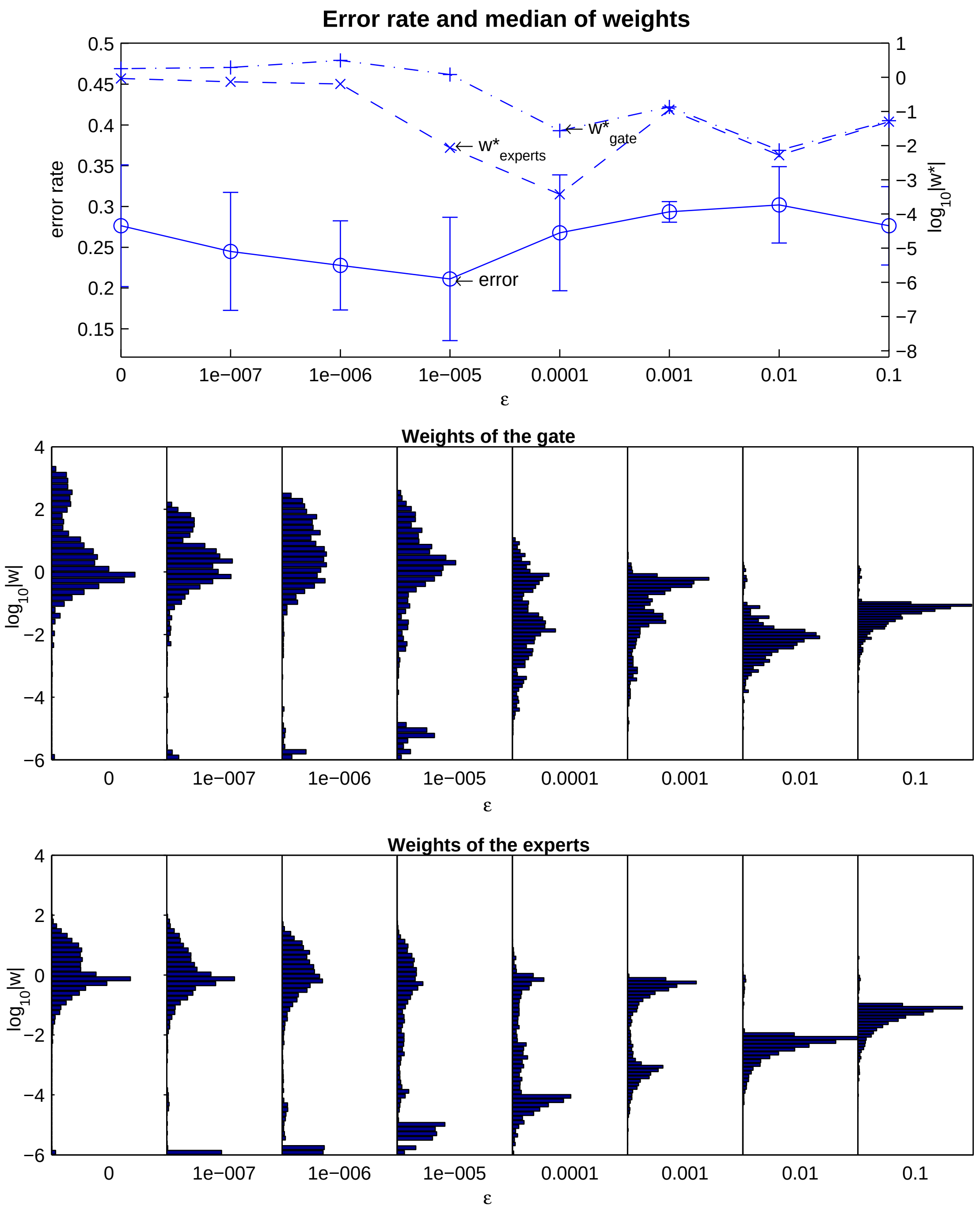


Figure 7.9: ME: weight distribution versus regularization parameter ϵ for Ionosphere dataset. Top plot: error rate and median $(w*)$ of the weight distributions of ME experts and gate, middle: weight distribution of ME experts, bottom: weight distribution of ME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

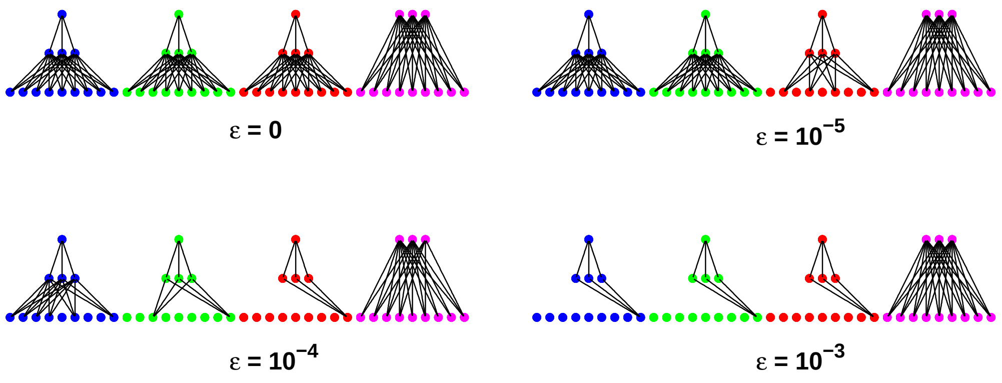


Figure 7.10: A typical ME architecture with LF and different regularization parameters ϵ. Different colors refer to different experts in ME. The last ANN in each case is the gate.

on Principle Component Analysis - of one of these runs (Figure 7.12), one can see the evidence of this behavior, in that there is only one class in each cluster. This strongly suggests that by applying LF, ME can be emergently turned into a clustering system.

Second, different experts are responsible for different sets of input feature, although these sets are not necessarily disjoint. For example, in run 1, the first expert is responsible for inputs $\{1, 2, 3, 4, 6, 8\}$, the second is responsible for $\{1, 2, 3, 8\}$ and the third is responsible for none of the inputs.

This automatic feature selection is an emergent property of the LF algorithm. According to Ishikawa (Ishikawa 1996; Ishikawa and Yoshino 1993), LF possesses the automatic problem decomposition property in the sense that the single ANN, in their experiment, can reorganize itself into different regions of the input space by skeletonizing the architecture, and thus selecting the appropriate input features for each hidden unit. Our experiments support this claim and extend it to the ME context. The experiments suggest that by applying LF to the ME model, the system can select relevant input features for each expert in the model.

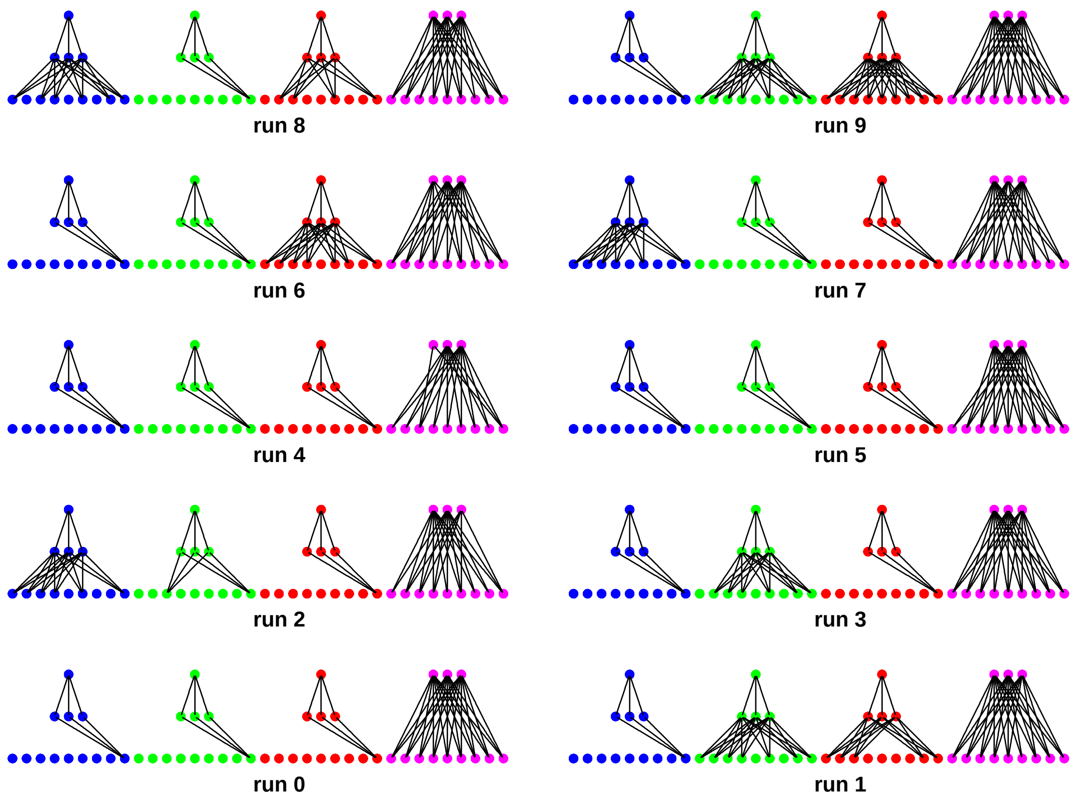


Figure 7.11: ME: a typical ensemble architecture with LF (error rate = 23.2% compared to 25.4% of the unregularized ME). Different colors refer to different experts in ME. The last ANN in each case is the gate.

7.3.3 CCME vs. regularized CCME

7.3.3.1 On performance in terms of error rate

Figure 7.13, and tables 7.7 and 7.8 show the error rates of CCME with LF and WE for fifteen datasets. The results in table 7.7, for seven out of the fifteen datasets, show a statistically significant preference for the unregularized case, suggesting that a high forgetting rate ($\epsilon = 10^{-3}$) is not suitable for CCME. The best forgetting rate is an intermediate value ($\epsilon = 10^{-4}$), for which only two out of fifteen datasets show a statistically significant (significance level of 98%) deterioration compared with the unregularized system.

On the other hand, the results in table 7.8 show that with and without WE, the generalization performance is generally comparable. Two, four and four out of fifteen

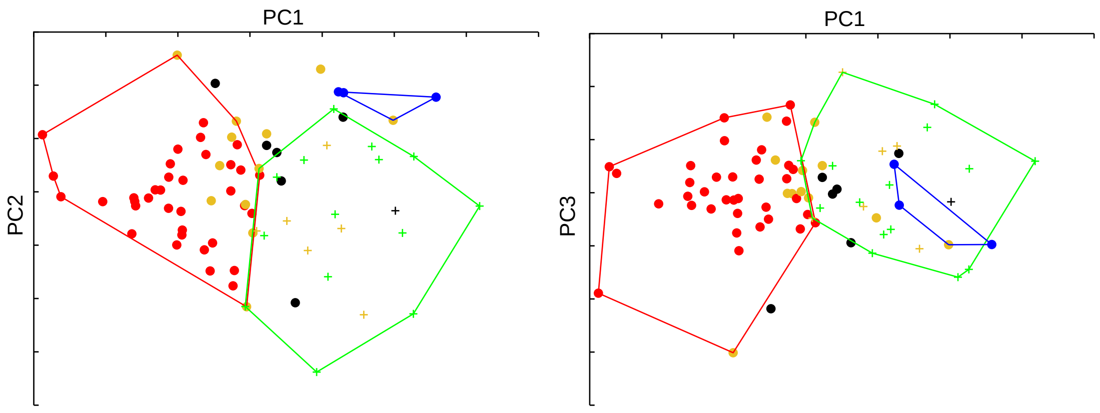


Figure 7.12: Class purity when using high LF on ME. Different colors correspond to different experts' responsibilities. Black points correspond to the cases where no expert is solely responsible for the data. Yellow points correpond to wrong classification. Marker types ('+' and 'o') represent different classes.

datasets (boldface) show a statistically significant worse performance for WE with $\epsilon = 10^{-5}, 10^{-4}, 10^{-3}$ respectively. Hence, the best value for WE is $\epsilon = 10^{-5}$.

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	0.030(0.014)	**0.037(0.014)**	0.034(0.010)	**0.039(0.021)**	**95%**	86%	**93%**
Cleveland Heart	0.225(0.073)	0.224(0.049)	0.205(0.030)	0.208(0.062)	50%	85%	72%
Australian credit card	0.126(0.038)	**0.133(0.040)**	**0.145(0.033)**	**0.143(0.048)**	**93%**	**98%**	**96%**
Diabetes	0.232(0.046)	0.238(0.049)	0.236(0.041)	0.249(0.046)	72%	68%	87%
StatLog Heart	0.174(0.089)	0.170(0.099)	0.189(0.073)	0.181(0.086)	58%	89%	75%
Hepatitis	0.168(0.061)	0.174(0.060)	0.168(0.034)	**0.220(0.087)**	66%	50%	**99%**
Liver Disorder	0.299(0.079)	0.308(0.060)	0.317(0.098)	**0.337(0.076)**	71%	70%	**94%**
Ljubljana Breast Cancer	0.297(0.112)	0.287(0.123)	0.290(0.128)	0.286(0.120)	71%	61%	69%
Tic Tac Toe	0.129(0.038)	0.118(0.047)	0.116(0.038)	**0.242(0.039)**	67%	88%	**100%**
House voting 84	0.037(0.034)	**0.050(0.037)**	0.041(0.043)	0.046(0.036)	**94%**	75%	87%
(ii) medium dataset							
German credit card	0.242(0.027)	**0.264(0.025)**	0.237(0.033)	**0.271(0.043)**	**98%**	66%	**100%**
Ionosphere	0.222(0.098)	0.233(0.095)	0.222(0.076)	0.259(0.089)	79%	50%	86%
King Root vs King Prawn	0.010(0.008)	0.009(0.009)	0.009(0.005)	**0.051(0.012)**	64%	71%	**100%**
E.coli Promoters	0.254(0.129)	0.245(0.153)	**0.302(0.095)**	0.228(0.183)	62%	**98%**	71%
Thyroid sickness	0.098(0.016)	0.140(0.125)	0.097(0.015)	0.102(0.004)	84%	52%	83%

Table 7.7: Error rates of LF with different values of ϵ for CCME. The confidence columns show how statistically significantly LF is different from the unregularized system. Italic and boldface indicate LF is significantly (90% and above) better and worse than the unregularized system respectively.

7.3.3.2 On complexity in term of the number of active connections

Figures 7.14 and 7.15 show the averaged percentage of active weights for the experts and the gate of CCME respectively. The figures suggest that LF, especially with

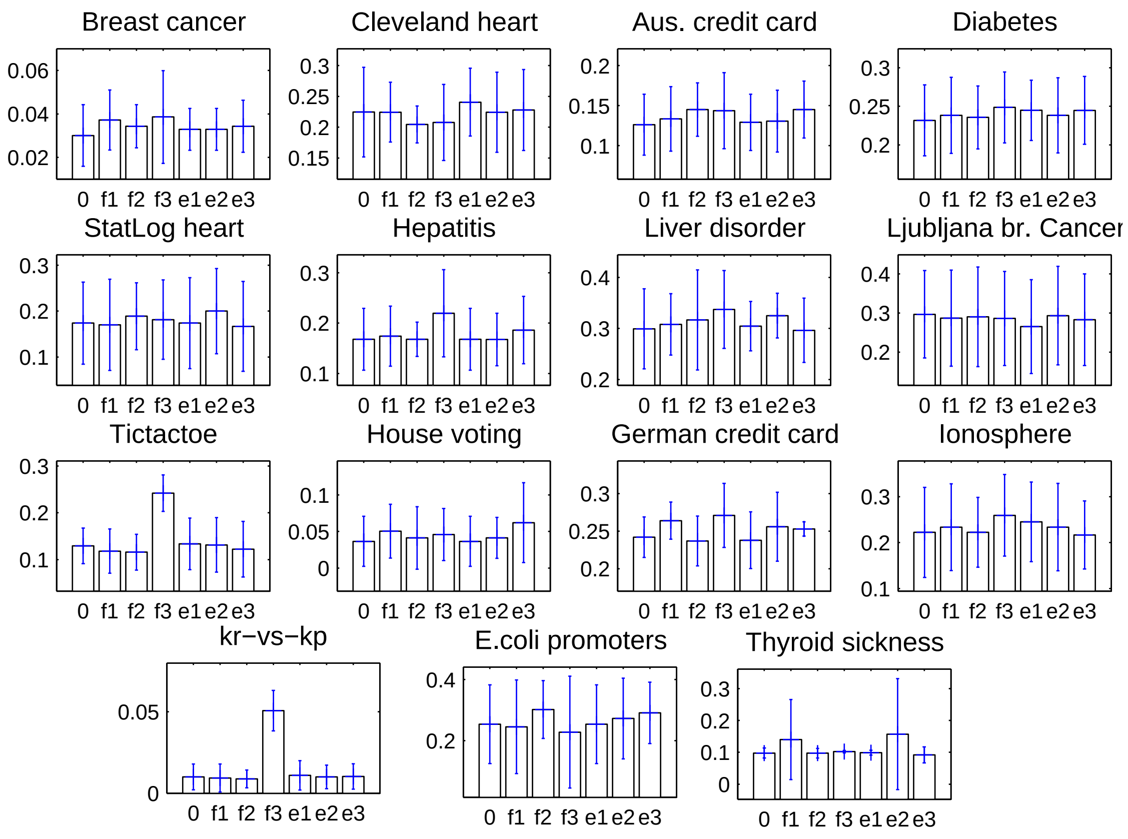


Figure 7.13: Error rate: CCME with LF (denoted ‘f’) and WE (denoted ‘e’) with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	0.030(0.014)	0.033(0.010)	**0.033(0.010)**	**0.034(0.012)**	83%	**92%**	**90%**
Cleveland Heart	0.225(0.073)	0.241(0.055)	0.224(0.065)	0.228(0.066)	70%	50%	55%
Australian credit card	0.126(0.038)	0.129(0.035)	0.130(0.039)	**0.145(0.035)**	66%	72%	**98%**
Diabetes	0.232(0.046)	**0.245(0.039)**	0.238(0.049)	**0.245(0.044)**	**99%**	82%	**96%**
StatLog Heart	0.174(0.089)	0.174(0.099)	**0.200(0.093)**	0.167(0.098)	50%	**93%**	70%
Hepatitis	0.168(0.061)	0.168(0.061)	0.167(0.052)	0.186(0.067)	0%	52%	75%
Liver Disorder	0.299(0.079)	0.304(0.049)	**0.325(0.044)**	0.296(0.063)	58%	**94%**	56%
Ljubljana Breast Cancer	0.297(0.112)	*0.265(0.120)*	0.293(0.126)	0.283(0.117)	*97%*	56%	74%
Tic Tac Toe	0.129(0.038)	0.134(0.055)	0.131(0.058)	0.122(0.060)	61%	56%	64%
House voting 84	0.037(0.034)	0.037(0.034)	0.041(0.028)	**0.062(0.055)**	0%	71%	**90%**
(ii) medium dataset							
German credit card	0.242(0.027)	0.238(0.038)	0.256(0.046)	0.253(0.009)	68%	88%	83%
Ionosphere	0.222(0.098)	**0.245(0.087)**	0.234(0.095)	0.217(0.074)	**98%**	78%	61%
King Root vs King Prawn	0.010(0.008)	0.011(0.009)	0.010(0.007)	0.010(0.008)	80%	50%	56%
E.coli Promoters	0.254(0.129)	0.254(0.129)	**0.273(0.132)**	**0.291(0.100)**	0%	**92%**	**92%**
Thyroid sickness	0.098(0.016)	0.099(0.008)	0.157(0.174)	0.092(0.025)	58%	85%	71%

Table 7.8: Error rates of WE with different values of ϵ for CCME. The confidence columns show the statistical significance of differences between WE and the unregularized system. Italic and boldface indicate WE is significantly (90% and above) better and worse than the unregularized system respectively.

higher forgetting rates, does have positive effect on the structure of the ANN, in terms of the number of pruned weights for both the experts and the gate. This is confirmed by the t-test results (i.e. confidence columns in tables 7.9 and 7.10). On the other hand, it seems that WE is not effective in pruning the connection weights. The t-test in tables 7.11 and 7.12, in which most of the values are not statistically significant, support this observation.

In the previous section, I have shown that LF with forgetting rate of $\epsilon = 10^{-4}$ is the best regularization scheme for CCME. The results in this section indicate that the network complexity with LF at $\epsilon = 10^{-4}$ is still statistically significantly less than that without regularization. In other words, LF is beneficial to CCME providing that a suitable forgetting rate is chosen.

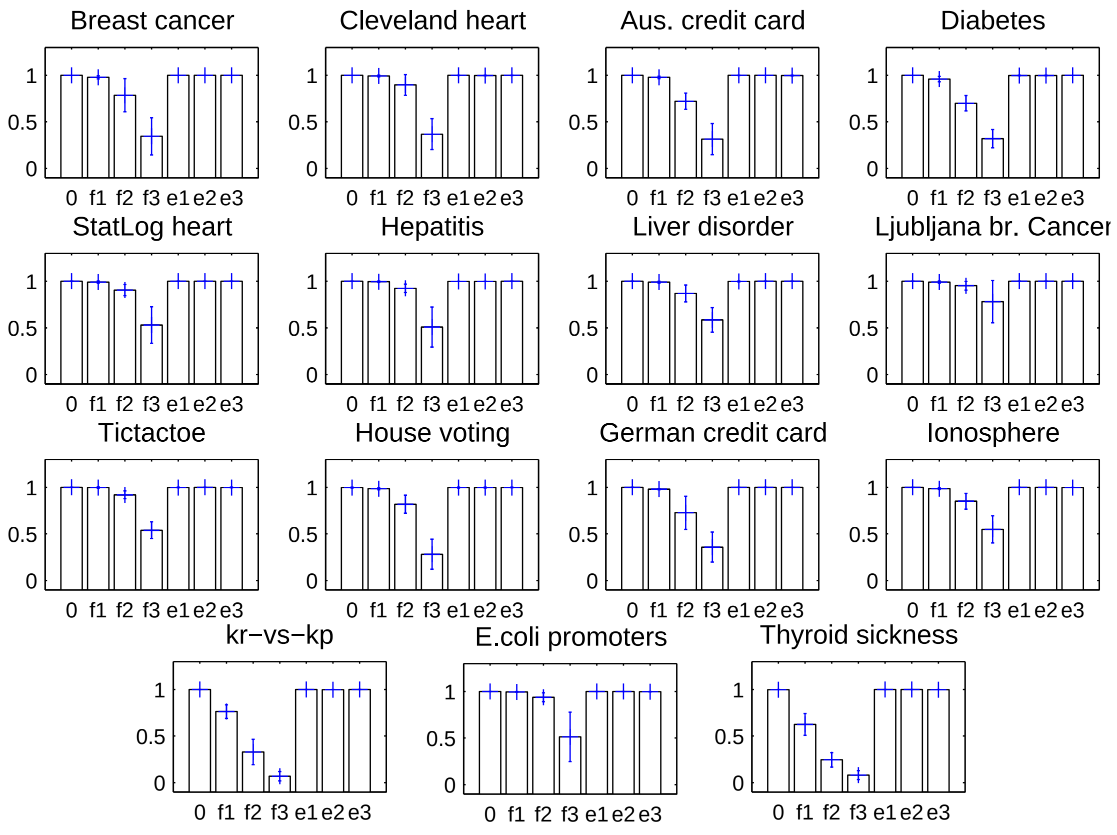


Figure 7.14: Percentage of active weights for the experts of CCME with LF (denoted ‘f’) and weight elimination (denoted ‘e’) with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

7.3.3.3 On comparing LF and WE

Figure 7.16 plots the averaged percentage of active weights (i.e. structural complexity) versus the error rates (i.e. performance) for six different sets of regulariza-

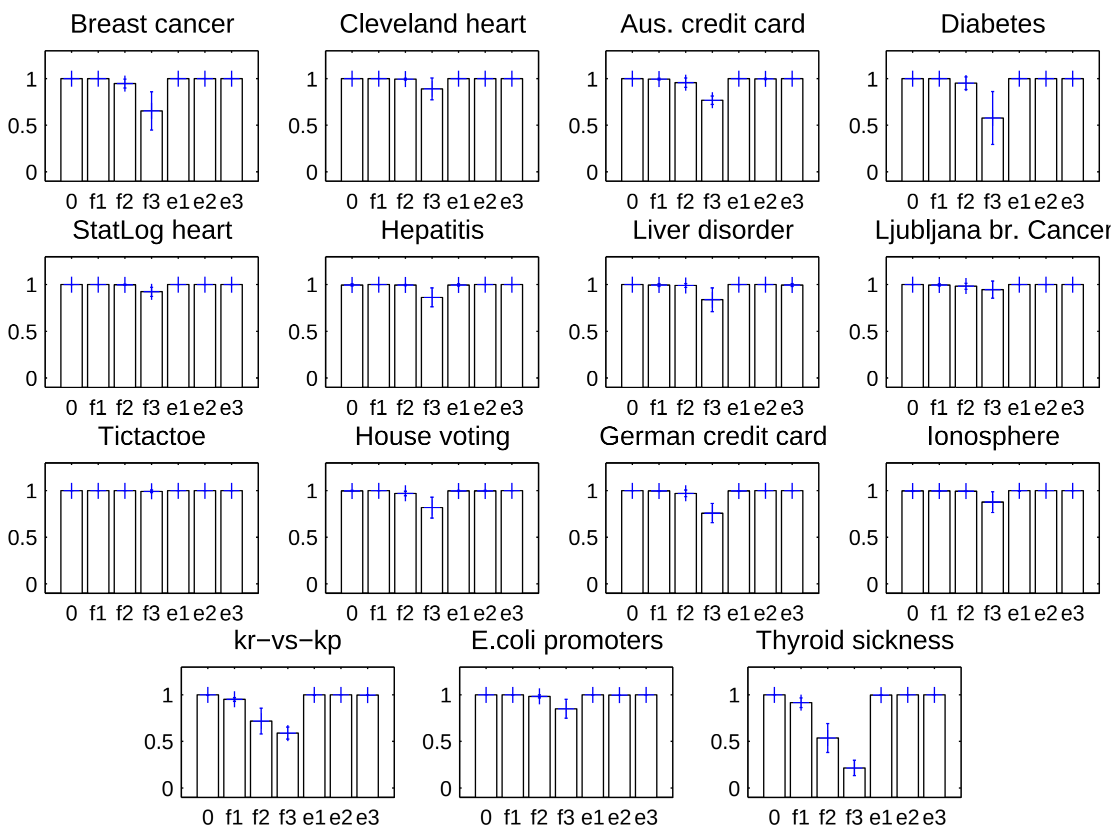


Figure 7.15: Percentage of active weights for the gate of CCME with LF (denoted 'f') and weight elimination (denoted 'e')with different regularization parameters $\{0, 1, 2, 3\} \equiv \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	0.979(0.016)	0.786(0.178)	0.344(0.199)	100%	100%	100%
Cleveland Heart	1.000(0.000)	0.993(0.010)	0.897(0.111)	0.368(0.167)	97%	99%	100%
Australian credit card	1.000(0.000)	0.978(0.014)	0.721(0.087)	0.314(0.168)	100%	100%	100%
Diabetes	1.000(0.000)	0.959(0.029)	0.700(0.083)	0.320(0.098)	100%	100%	100%
StatLog Heart	1.000(0.000)	0.990(0.012)	0.906(0.060)	0.530(0.195)	98%	100%	100%
Hepatitis	1.000(0.000)	0.995(0.009)	0.924(0.045)	0.509(0.214)	95%	100%	100%
Liver Disorder	1.000(0.000)	0.990(0.011)	0.870(0.091)	0.586(0.132)	99%	100%	100%
Ljubljana Breast Cancer	1.000(0.000)	0.991(0.011)	0.952(0.046)	0.781(0.227)	98%	100%	99%
Tic Tac Toe	1.000(0.000)	0.998(0.007)	0.920(0.042)	0.540(0.089)	83%	100%	100%
House voting 84	0.998(0.004)	0.985(0.011)	0.819(0.097)	0.282(0.161)	100%	100%	100%
(ii) medium dataset							
German credit card	1.000(0.000)	0.981(0.007)	0.727(0.177)	0.358(0.161)	100%	100%	100%
Ionosphere	0.999(0.002)	0.987(0.011)	0.851(0.086)	0.548(0.146)	100%	100%	100%
King Root vs King Prawn	0.999(0.001)	0.763(0.073)	0.329(0.136)	0.068(0.049)	100%	100%	100%
E.coli Promoters	0.999(0.001)	0.997(0.004)	0.938(0.048)	0.512(0.265)	96%	100%	100%
Thyroid sickness	0.999(0.002)	0.626(0.117)	0.244(0.078)	0.081(0.048)	100%	100%	100%

Table 7.9: Percentage of active weights of LF with different values of ϵ for experts of CCME. The confidence columns show the statistical significance of differences between LF and the unregularized system.

tion values: 3 for LF and 3 for WE, on fifteen datasets. The scatter plot shows that with the current set up and parameters LF is more effective than WE in reducing the structural

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	1.000(0.000)	0.947(0.048)	0.653(0.206)	0%	100%	100%
Cleveland Heart	1.000(0.000)	1.000(0.000)	0.995(0.010)	0.890(0.117)	0%	92%	99%
Australian credit card	1.000(0.000)	0.996(0.009)	0.958(0.051)	0.769(0.045)	92%	99%	100%
Diabetes	1.000(0.000)	1.000(0.000)	0.952(0.068)	0.578(0.284)	0%	97%	100%
StatLog Heart	1.000(0.000)	1.000(0.000)	0.998(0.008)	0.924(0.049)	0%	83%	100%
Hepatitis	0.997(0.011)	1.000(0.000)	0.997(0.007)	0.863(0.101)	83%	50%	100%
Liver Disorder	1.000(0.000)	0.995(0.015)	0.990(0.020)	0.838(0.127)	83%	92%	100%
Ljubljana Breast Cancer	1.000(0.000)	0.997(0.011)	0.983(0.032)	0.947(0.092)	83%	93%	95%
Tic Tac Toe	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.993(0.014)	0%	0%	92%
House voting 84	0.998(0.006)	1.000(0.000)	0.971(0.025)	0.820(0.112)	83%	100%	100%
(ii) medium dataset							
German credit card	1.000(0.000)	0.997(0.006)	0.971(0.036)	0.760(0.105)	92%	99%	100%
Ionosphere	0.998(0.004)	0.998(0.004)	0.995(0.005)	0.877(0.112)	50%	90%	100%
King Root vs King Prawn	0.999(0.003)	0.952(0.020)	0.718(0.138)	0.589(0.064)	100%	100%	100%
E.coli Promoters	1.000(0.000)	0.999(0.002)	0.983(0.017)	0.850(0.101)	83%	99%	100%
Thyroid sickness	1.000(0.000)	0.915(0.051)	0.537(0.155)	0.214(0.082)	100%	100%	100%

Table 7.10: Percentage of active weights of LF with different values of ϵ for the gate network of CCME. The confidence columns show the statistical significance of differences between LF and the unregularized system.

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Cleveland Heart	1.000(0.000)	0.999(0.003)	0.998(0.003)	0.999(0.003)	83%	92%	83%
Australian credit card	1.000(0.000)	0.999(0.002)	1.000(0.000)	0.998(0.005)	83%	0%	90%
Diabetes	1.000(0.000)	0.999(0.004)	0.999(0.004)	1.000(0.000)	83%	83%	0%
StatLog Heart	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Hepatitis	1.000(0.000)	0.999(0.002)	0.999(0.002)	1.000(0.000)	83%	83%	0%
Liver Disorder	1.000(0.000)	0.998(0.005)	1.000(0.000)	1.000(0.000)	83%	0%	0%
Ljubljana Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Tic Tac Toe	1.000(0.000)	0.999(0.004)	1.000(0.000)	0.999(0.004)	83%	0%	83%
House voting 84	0.998(0.004)	0.998(0.004)	0.998(0.004)	0.999(0.004)	0%	0%	83%
(ii) medium dataset							
German credit card	1.000(0.000)	1.000(0.001)	1.000(0.000)	1.000(0.001)	83%	0%	83%
Ionosphere	0.999(0.002)	0.999(0.001)	1.000(0.001)	0.998(0.002)	66%	83%	78%
King Root vs King Prawn	0.999(0.001)	0.999(0.001)	0.998(0.002)	1.000(0.000)	50%	83%	96%
E.coli Promoters	0.999(0.001)	0.999(0.001)	0.999(0.001)	0.999(0.002)	0%	70%	70%
Thyroid sickness	0.999(0.002)	1.000(0.001)	1.000(0.001)	0.999(0.002)	92%	92%	50%

Table 7.11: Percentage of active weights of WE with different values of ϵ for experts of CCME. The confidence columns show the statistical significance of differences between WE and the unregularized system.

complexity of CCME while maintaining similar level of accuracy. In the next section, the effects of LF on the weights of CCME are analyzed.

7.3.3.4 Analysis of LF on the CCME model

As seen above, LF has the effect of pushing irrelevant weights towards zero. In this section, the Diabetes dataset is used to analyze the use of LF on the CCME model.

Dataset	$\epsilon = 0.0$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-3}$	Confidence		
(i) small dataset							
Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Cleveland Heart	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Australian credit card	1.000(0.000)	1.000(0.000)	0.998(0.007)	1.000(0.000)	0%	83%	0%
Diabetes	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
StatLog Heart	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Hepatitis	0.997(0.011)	0.997(0.011)	1.000(0.000)	1.000(0.000)	0%	83%	83%
Liver Disorder	1.000(0.000)	1.000(0.000)	1.000(0.000)	0.995(0.015)	0%	0%	83%
Ljubljana Breast Cancer	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
Tic Tac Toe	1.000(0.000)	1.000(0.000)	1.000(0.000)	1.000(0.000)	0%	0%	0%
House voting 84	0.998(0.006)	0.998(0.006)	0.998(0.006)	1.000(0.000)	0%	0%	83%
(ii) medium dataset							
German credit card	1.000(0.000)	0.997(0.006)	1.000(0.000)	1.000(0.000)	92%	0%	0%
Ionosphere	0.998(0.004)	0.999(0.003)	1.000(0.000)	1.000(0.000)	70%	92%	92%
King Root vs King Prawn	0.999(0.003)	0.999(0.003)	1.000(0.000)	0.998(0.004)	0%	83%	70%
E.coli Promoters	1.000(0.000)	0.999(0.002)	0.999(0.002)	0.999(0.002)	83%	92%	83%
Thyroid sickness	1.000(0.000)	0.999(0.004)	1.000(0.000)	1.000(0.000)	83%	0%	0%

Table 7.12: Percentage of active weights of WE with different values of ϵ for the gate network of CCME. The confidence columns show the statistical significance of differences between WE and the unregularized system.

7.3.3.4.1 Effect of regularization parameters on the weight magnitude

Similar to the analysis for the ME model, figure 7.17 shows a typical effect of LF on the weight magnitude of CCME. As expected, with higher forgetting rates, more weights are pushed towards zero. As seen in Figure 7.17, in the unregularized case (i.e. $\epsilon = 0.0$), the weights of both the experts and the gate have a magnitude of order 1 (i.e. 10^0). In the subsequent plots, an increase in regularization parameters corresponds to a decrease in the overall weight magnitudes of both the experts and the gate. With a high value of $\epsilon = 10^{-3}$, most of the experts' weights, and some weights of the gate, are pushed below the threshold of 10^{-3}.

Figures 7.18, 7.20 and 7.19 show the detailed weight distributions of the gate and experts of CCME for three datasets: Diabetes, House Voting 84 and Ionosphere. As in the ME case, there is an interesting observation about certain relationships between the phase transition of the weight distributions and the generalization errors of the system. From the plots, it is obvious that the interesting phase transition, in terms of the error rate, happens when ϵ lies in the range $[10^{-4}, 10^{-1}]$. This corresponds exactly to a phase transition in the weight distributions of the gate and the experts. This observation strengthens my belief that there exists a relationship between the weight distribution and the generalization performance of the ME family.

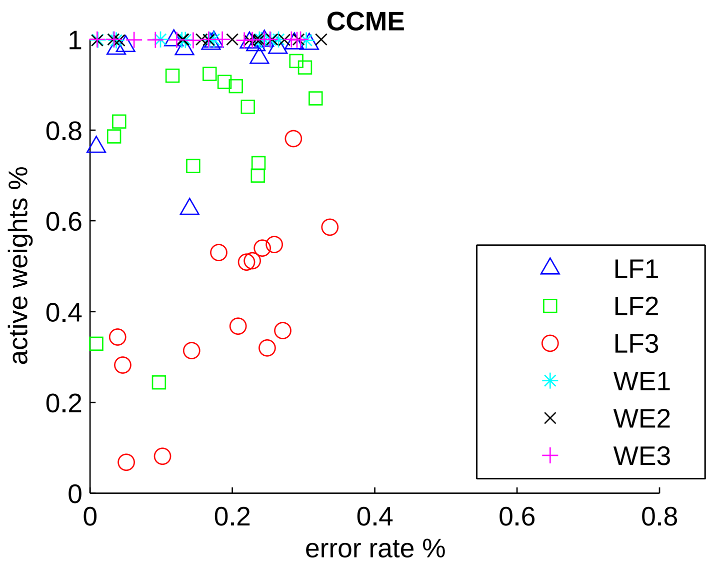


Figure 7.16: Percentage of active weights vs. error rate for CCME with LF and WE and three different regularization values $\{1, 2, 3\} \equiv \{10^{-5}, 10^{-4}, 10^{-3}\}$

Moreover, the minimum generalization error for all three datasets occurs at $\epsilon \leq 10^{-4}$. At $\epsilon = 10^{-4}$, a number of weights in both the gate and experts are still in the lower magnitude region. The plot confirms my previous findings that regularization with a suitable control parameter has the effect of reducing the structural complexity of the system while maintaining the level of accuracy compared to the unregularized system.

Figure 7.21 shows a typical architecture of the ensemble with different forgetting rates. As expected, with higher forgetting rate (e.g $\epsilon = 10^{-3}$), the network becomes sparser, with many connections being removed. However, comparing this to the same plot in the ME model (section 7.3.2.4.1), it appears that LF may have less effect on the CCME model.

7.3.3.4.2 Effect of the regularization parameters on relevant input features

Figure 7.22 shows the ensemble architectures for LF ($\epsilon = 10^{-4}$) on CCME. In this case, LF has a similar error rate to the unregularized case (23.6% compared to 23.2%).

As in ME, different experts are responsible for different sets of input features. For example, in run 2, the first expert works on $\{4, 5\}$, the second on $\{5, 6, 8\}$ and the third on all inputs. Again, this automatic feature selection is an emergent property of the

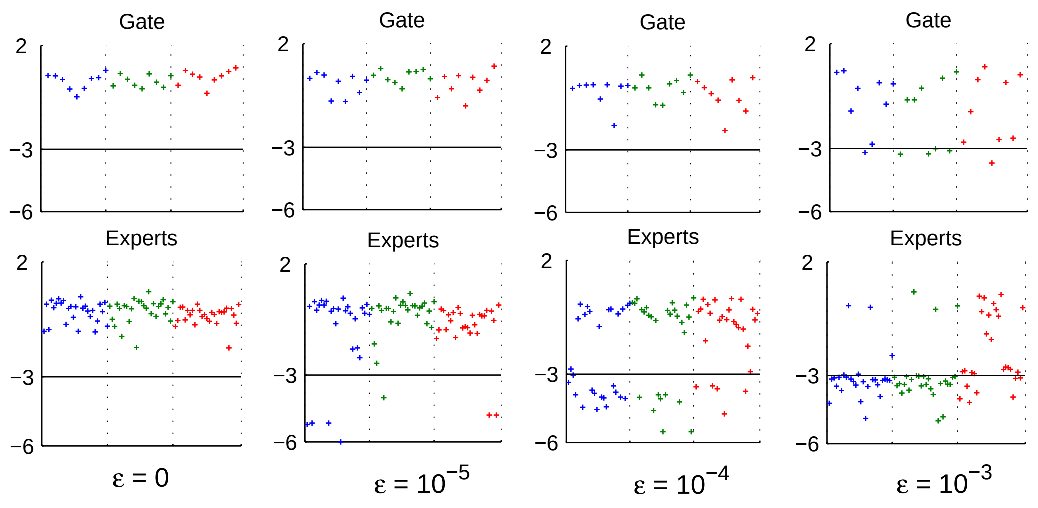


Figure 7.17: An example of LF on CCME with different regularization parameters ϵ. y-axis: $log_{10}|w|$. x-axis: weight number. 10^{-3}: cutoff value. Different colors refer to different experts in CCME.

LF algorithm. The results suggest that by applying LF to the CCME model, the system automatically selects relevant input features for each expert in the model.

7.3.4 Effects of regularization on CCME and ME

Figure 7.23 contrasts the scatter plots of the regularization effects on ME and CCME. The plots clearly show that regularization has different effects on ME and CCME.

In terms of percentage of active weights, although LF clearly outperforms WE for both ME and CCME, the effect of LF on ME is stronger than on CCME. This difference is best seen in the cases of LF1 and LF2. The scatter plots show LF2 (square markers) can pull the percentage of active weights of ME down below 60% while in CCME, LF2 markers concentrate around 80% in terms of percentage of active weights. Similarly, LF1 (triangle markers) is still effective in reducing the percentage of active weights of ME, while it almost has no effect on CCME. Moreover WE has no effect on CCME while WE3 is effective in reducing the percentage of active weights of ME to about 60%-80%. In conclusion, the scatter plots suggest CCME is less responsive to regularization than ME.

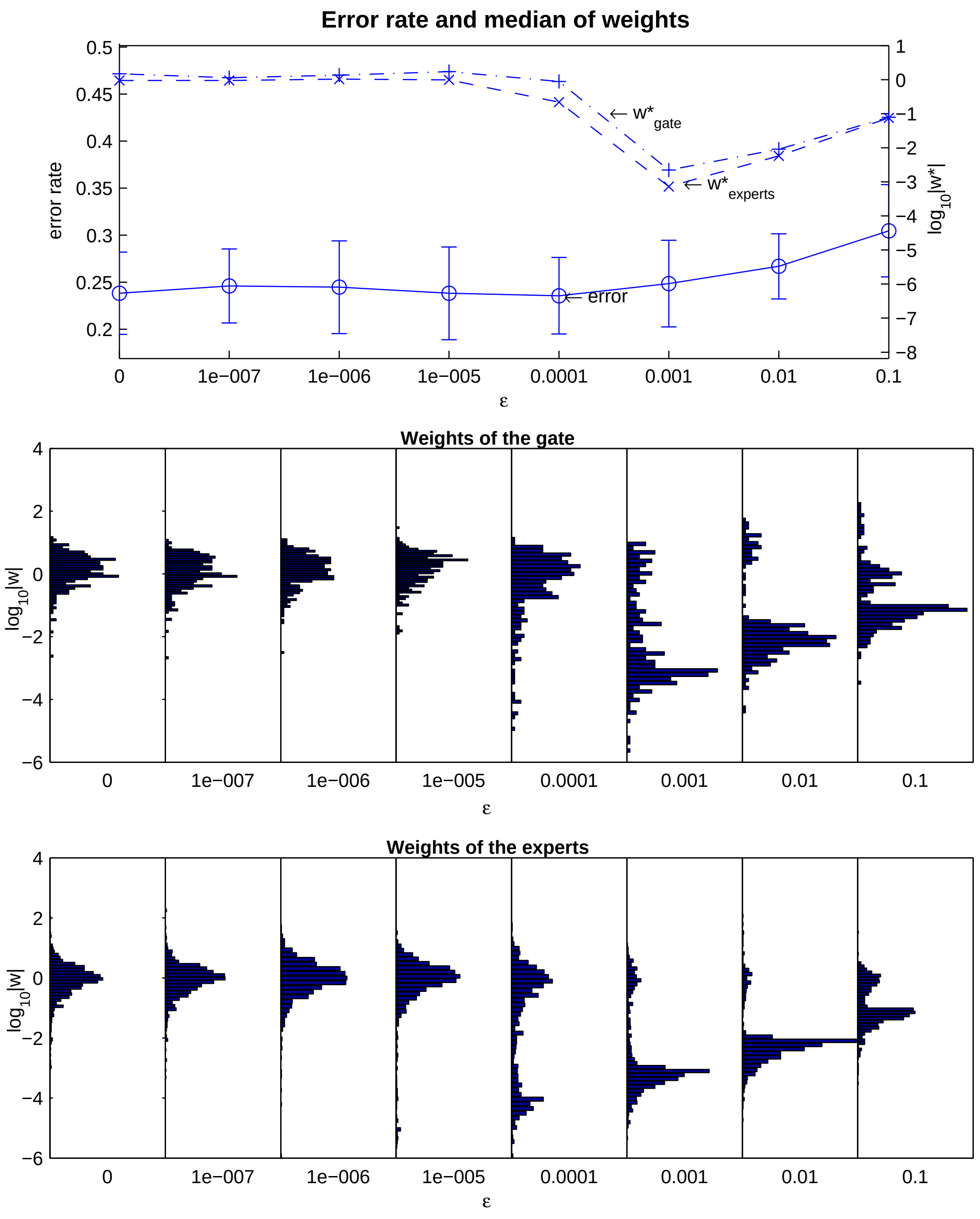


Figure 7.18: CCME: weight distribution versus regularization parameter ϵ for Diabetes dataset. Top plot: error rate and median $(w*)$ of the weight distributions of CCME experts and gate, middle: weight distribution of CCME experts, bottom: weight distribution of CCME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

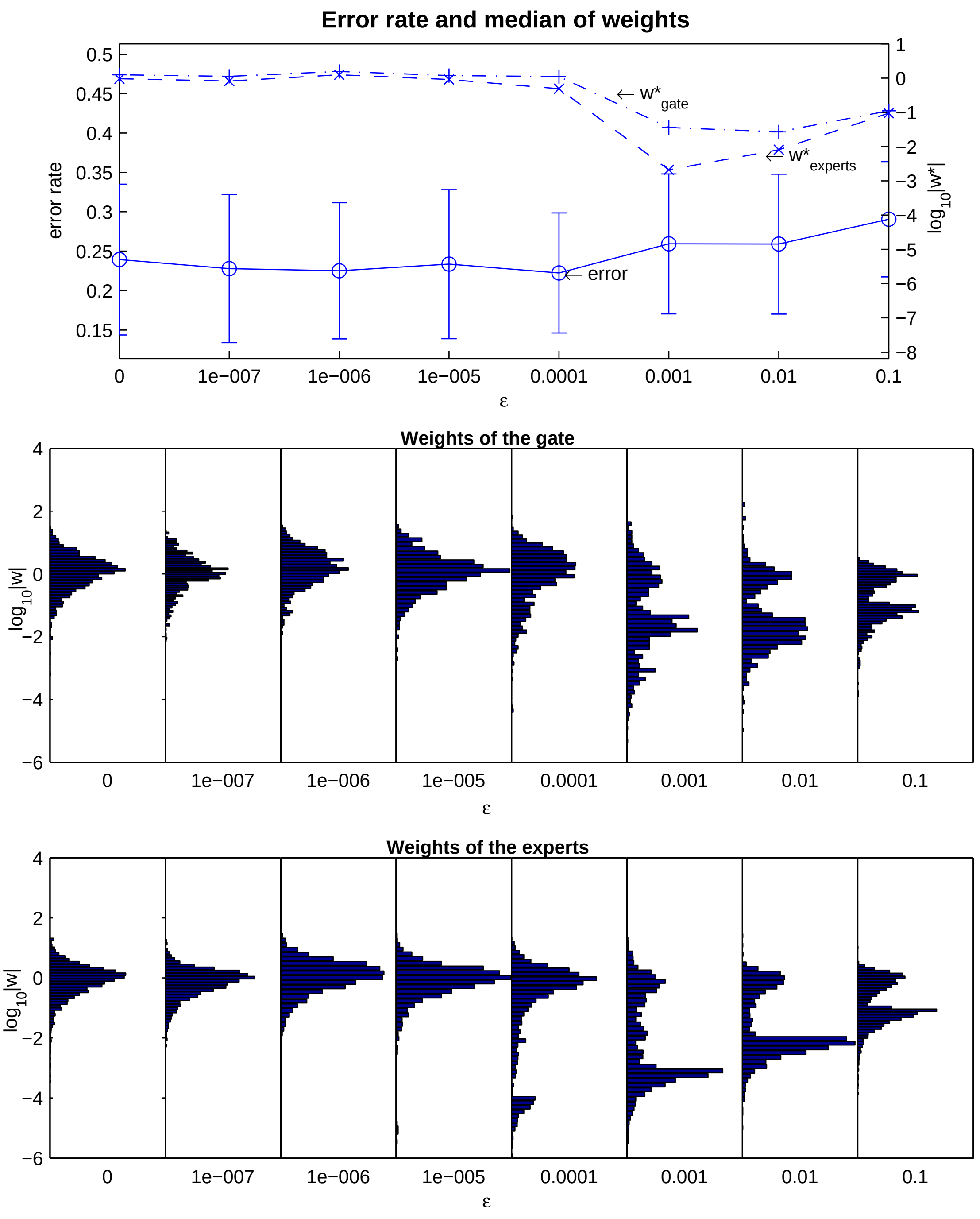


Figure 7.19: CCME: weight distribution versus regularization parameter ϵ for House Voting 84 dataset. Top plot: error rate and median ($w*$) of the weight distributions of CCME experts and gate, middle: weight distribution of CCME experts, bottom: weight distribution of CCME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

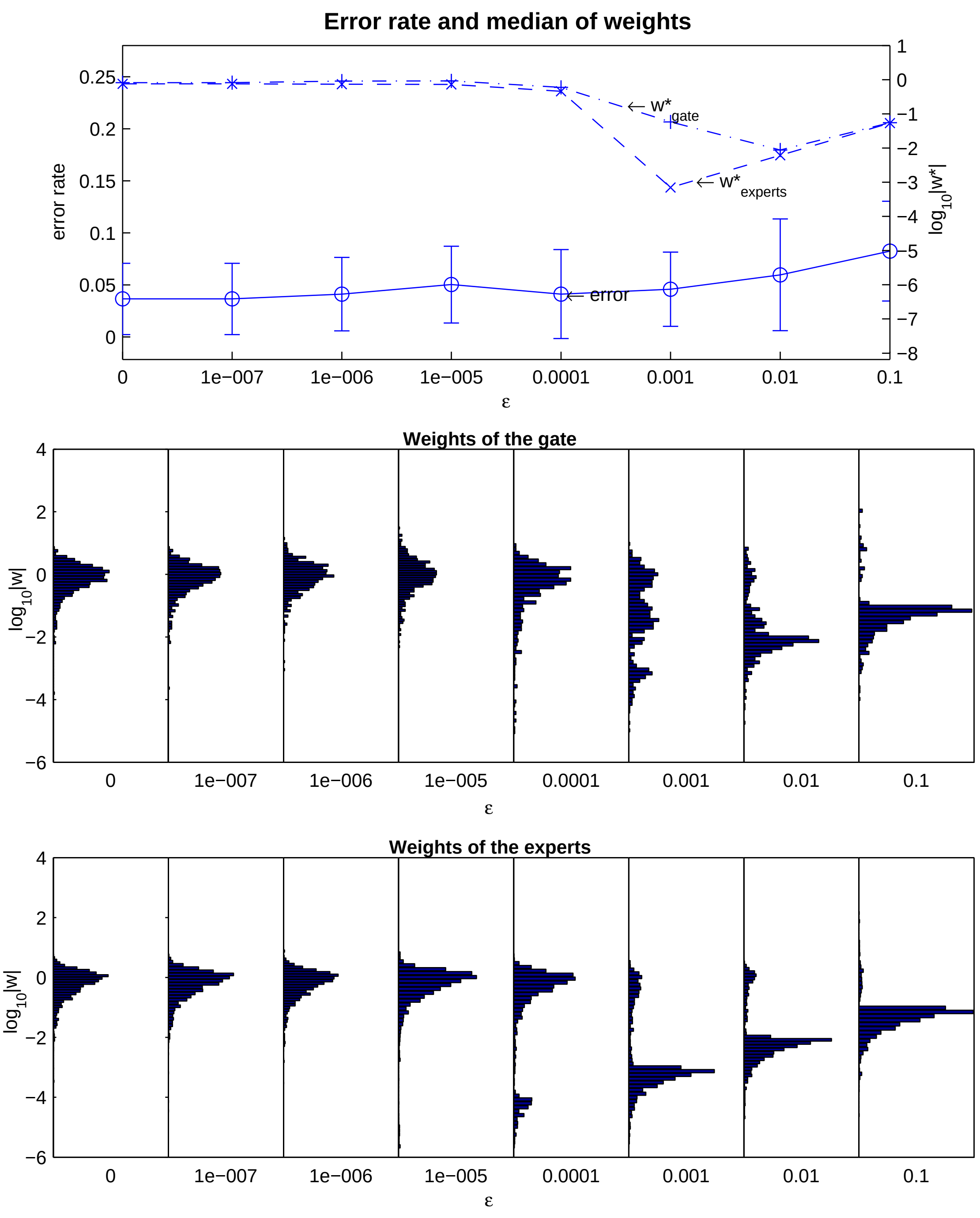


Figure 7.20: CCME: weight distribution versus regularization parameter ϵ for Ionosphere dataset. Top plot: error rate and median $(w*)$ of the weight distributions of CCME experts and gate, middle: weight distribution of CCME experts, bottom: weight distribution of CCME gate. Histogram: the scale of x-axis in each box are equalized for the whole subplot.

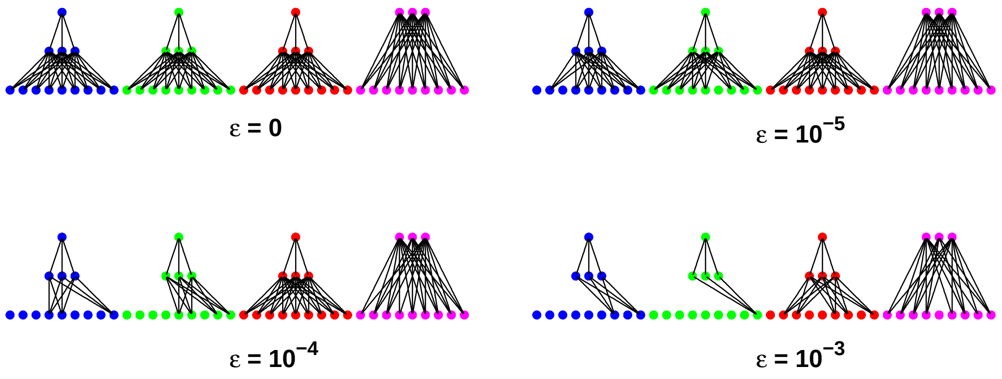


Figure 7.21: A typical CCME architecture with LF and different regularization parameters ϵ. Different colors refer to different experts in CCME. The last ANN in each case is the gate.

In terms of the error rates, except for three outliers in ME with LF3 where the error rates jump to 40%-70%, both regularization mechanisms are able to keep the error rates of ME and CCME below 40%. This confirms the hypothesis, namely that regularization does not, on average, deteriorate the generalization performance of ME and CCME.

7.4 Conclusion

In this chapter, I have investigated two popular regularization schemes: learning by forgetting and weight elimination. The purpose is to see if regularization is beneficial to the mixture of experts and cooperative coevolutionary mixture of experts models. The results suggest the following conclusions. First, in terms of accuracy, regularization does not significantly affect the performance of the models. Second, in terms of network complexity, learning by forgetting is effective in pushing irrelevant weights towards zero while maintaining the same level of accuracy. Comparing learning by forgetting and weight elimination, the latter is not as effective as the former although it still can prune a number of weights. Third, because of its structural modularization ability (Ishikawa 1996; Ishikawa and Yoshino 1993), learning by forgetting can manipulate the models

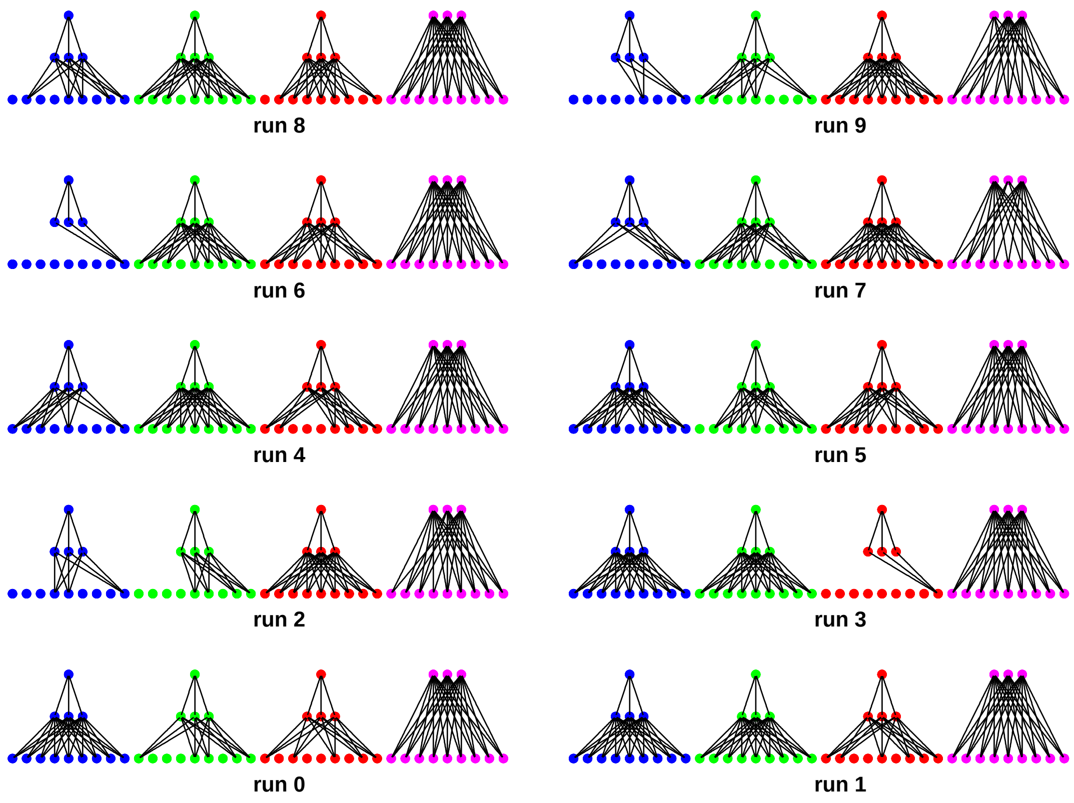


Figure 7.22: CCME: a typical ensemble architecture with LF, $\epsilon = 10^{-4}$, error rate = 23.6% compared to 23.2%(without LF). Different colors refer to different experts in CCME. The last ANN in each case is the gate.

into doing different types of tasks. In some cases, it can turn the ME model into a traditional clustering system, by pruning all the input-to-hidden connections of the experts and letting the gate divide the data into clusters of the same class. Another benefit of learning by forgetting lies in allowing different experts of the models to select different, but not necessarily disjoint, sets of input features. Finally, the experiments and analysis of this chapter suggest that there is a relationship between the weight distribution, in terms of magnitude, and the generalization ability of the ME and CCME model. This serves as an open research question for future works.

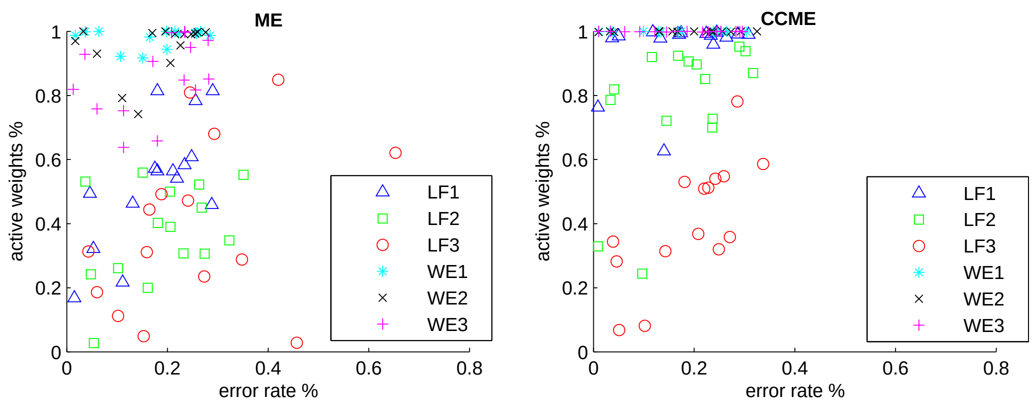


Figure 7.23: Percentage of active weights vs. error rate for ME (left) and CCME (right) with LF and WE and different regularization parameters $\{1, 2, 3\} \equiv \{10^{-5}, 10^{-4}, 10^{-3}\}$

Chapter 8

Conclusions and Future Work

In this thesis, I have presented a systematic study of neuro ensembles on binary classification problems. First, I have investigated the relationship between diversity and accuracy of a number of state-of-the-art neuro ensembles, aiming to generate insights into how to design a good neuro ensemble. Applying these insights, I have proposed a method to co-evolve the gate and experts of a mixture of experts (ME) using a cooperative coevolution (CC) framework, which I name the cooperative coevolutionary mixture of experts or CCME. The proposed method blends concepts from mixture of experts and cooperative coevolution to generate a modularized neuro ensemble, which solves both the diversity problem - by forcing the experts to work on different regions of input (ME) and to be diverse in terms of species diversity (CC) - and the generalization problem - the whole system must work cooperatively. The more conventional ME is then compared against CCME in terms of the generalization performance and running time. I have also investigated and presented the traditional back propagation ME and the proposed method in the light of automatic problem decomposition, using a newly-derived set of visualization tools. Finally, I have shown the effects of regularization, especially the so-called learning by forgetting, on the structural complexity and the behaviors of ME and CCME.

The main findings from the research work in this thesis can be summarized as follow:

1. This thesis has investigated different aspects of a number of state-of-the-art neuro ensemble methods (Chapter 4): namely the Simple Ensemble, the Negative Correlation Learning Ensemble, the Island Ensemble and the Multi Objective Ensemble. The results verify a number of points raised in the literature:
   (a) Combining individual networks into an ensemble improves the performance of the system.
   (b) Different combination gates have little effect on the average performance of the ensemble
   (c) The diversity level maintained by negative correlation learning is poor.
   (d) Local search helps evolution to find better solutions
   (e) Noise injection shows interesting effects on performance enhancement, though the improvement is not yet clear.
   (f) Early stopping enhances generalization
   (g) The connection between diversity and performance of the ensemble remains a hypothesis with limited or no verification
2. The proposed Cooperative Coevolutionary Mixture of Experts (CCME) method is validated against a set of benchmark binary classification problems. I have analyzed different aspects of both the traditional back propagation ME and the novel CCME in chapter 5. The key findings are:
   (a) In terms of performance, CCME is better on average than the traditional ME in classification problems.
   (b) CCME is comparable to ME in terms of the running time.
   (c) Early stopping is useful in enhancing the generalization of both back propagation ME and CCME.

   (d) CCME and ME are robust to different error functions, learning rates, number of experts and network complexity in a number of binary classification problems.

3. The proposed visualization tools are applied to analyze the problem decomposition behaviors of ME and CCME (Chapter 6). The key results are:

   (a) Both ME and CCME can decompose the input space into less complex regions (i.e. sub–tasks) in such a way that the available experts are able to classify the data in their clusters with higher accuracy.

   (b) The tools show how ME and CCME can discover the modularity of the problem if there is any.

   (c) Increasing network complexity, in terms of number of hidden units, helps the experts to classify their clusters with better accuracy. However, there is a phase transition, beyond which an increase in complexity deteriorates the system performance.

   (d) Increasing the number of experts also enhances the system performance by dividing the data into more clusters. Again, however, there is a critical region, beyond which adding more experts does not improve the system performance.

4. The thesis has extended both ME and CCME models by adding a regularization term - based on learning by forgetting and weight elimination - during training (Chapter 7). Another contribution of this chapter is that it introduces and uses a number of novel visualization tools to visualize the weight distribution and the architecture of the model. The key results are:

   (a) In terms of accuracy, regularization does not significantly affect the performance of the models.

   (b) In terms of network complexity, learning by forgetting is effective in pushing irrelevant weights toward zero, while maintaining the same level of accuracy.

(c) weight elimination is not as effective as learning by forgetting in pruning out the irrelevant weights

(d) Because of its structural modularization ability (Ishikawa 1996; Ishikawa and Yoshino 1993), learning by forgetting can manipulate the models into doing different types of tasks. In some cases, it can turn the ME model into a traditional clustering system by pruning all the input-to-hidden connections of the experts, and letting the gate divide the data into clusters of the same class. Another benefit of learning by forgetting is to allow different experts of the models to select different, but not necessarily disjoint, sets of input features.

(e) There is a possible relationship between the weight distribution, in terms of magnitude, and the generalization ability of the ME and CCME model.

In conclusion, the experiments and findings answer the research question: artificial evolution can produce neuro ensembles that automatically decompose complex classification problems by dividing the data to sub–regions where the input–output relationship is easier to learn and assigning different experts to these sub–regions. The higher performance of CCME over the conventional ME implies that by adding a cooperative co–evolution layer, the system can explore the search space more efficiently (CCME outperforms the random search for ME) and thus find more fitting neuro ensembles.

8.1 Future Work

Numerous directions for further explorations and investigations have emerged from the work of this thesis. Some open research questions have already been highlighted in the respective chapters where they directly arise from the experiments and analysis. These can be summarized as follows:

1. In chapter 6, the automatic problem decomposition analysis of CCME shows that the model can find a very good way to divide the data into better clusters, where

"better" measures the ease with which an expert can classify the data in the clusters. The results suggest the following questions: (1) can we define a measure for the ease with which a cluster may be classified?, and if yes, (2) can this measurement be used to guide the model toward dividing the data in a better way, or can it at least explain the differences in performance between methods of automatic problem decomposition?

2. In section 7.3.2.4.1, the correlation between the generalization error curves and the weight distribution plots suggests the following open research directions: (1) is there a relationship between the weight distribution and the generalization error?, (2) in the case of regularization, can the phase changes in the weight distribution be used as an estimate for the phase change in the generalization errors? and (3) if there is such a relationship, can one find an optimized weight distribution which will produce an optimized neuro ensemble in terms of the generalization ability?

Besides the above questions, a number of research directions arising from the philosophical issues underlying the thesis are outlined here:

First, a critical question, that still remains open in the ensemble literature, is how to decide on the optimum ensemble size. As I discussed in the thesis, cooperative coevolution is a powerful framework allowing a suitable number of sub–populations to emerge, based on the fitness of the whole system. The principle is to add and remove sub–populations when the overall fitness stagnates for a number of generations (Potter 1997; Potter and De Jong 2000). The proposed framework allows the ensemble size to emerge as the system adjusts itself to the best fitness.

Second, in this thesis, I have shown how ME and CCME automatically decompose a hard problem into sub–regions of the input with the property that the sub–regions are easier, in terms of input-output relationship, for the available classifiers to solve. However, the decomposition mechanism underlying the models remains an open question. As suggested, a measure of the ease-of-classification of the sub–regions could be very valuable in guiding the system towards better decomposition schemes. If such a measure exists, it could be cooperated into the performance fitness of the system, for example serving as an objective in a multi–objective optimization method. It would force the gate to divide the data into optimum regions, in terms of simplest input-output relationship, and at the same time evolve accurate experts to solve these sub–regions with minimum error.

Finally, although learning by forgetting is applied and analyzed in this thesis, it is applied in a simple form in which the regularization parameters are the same for both the gate and the experts. As suggested in the experiments and analysis in chapter 7, the relationship between these regularization parameters and the weight distributions of the gate and the experts is not simple. It is therefore interesting to further investigate the effect of LF with different parameters for these components. Also, it will be valuable to study the relationship between the weight distribution and the error rate, since such relationship might give valuable insights into the optimum range and shape of the weight distribution to generate neuro ensembles that generalize well. If such an optimum weight distribution can be found, then a mixture of experts can be quickly designed to solve the problem. Besides learning by forgetting, there are other successful regularization schemes in the literature of ANN. It would be interesting to study the effects of these schemes on both ME and CCME.

Appendix A

Derivations of error functions and their derivatives

A.1 Back–propagation for feedforward neural networks

Given a dataset consisting of N records: $\{\vec{x^i}, \vec{d^i}\}$ where $\vec{x^i} = \{x_1, x_2, ..., x_P\}$ and $\vec{d^i} = \{d_1, d_2, ..., d_Q\}$ are the attributes and targets of case i respectively; given a FFNN as shown in figure A.1; the output of each layer k is denoted as $o_j^{(k)}$ where the subscript j refers to the position of the node in its layer. Since the input node has no transformation function, $o_j^{(0)} \equiv \vec{x}$. Each hidden node often has two parts: a summation function $a_j^{(k)} = \sum_l w_{lj}^{(k)} o_l^{(k-1)}$, and an activation function $\varphi(a_j^{(k)})$.

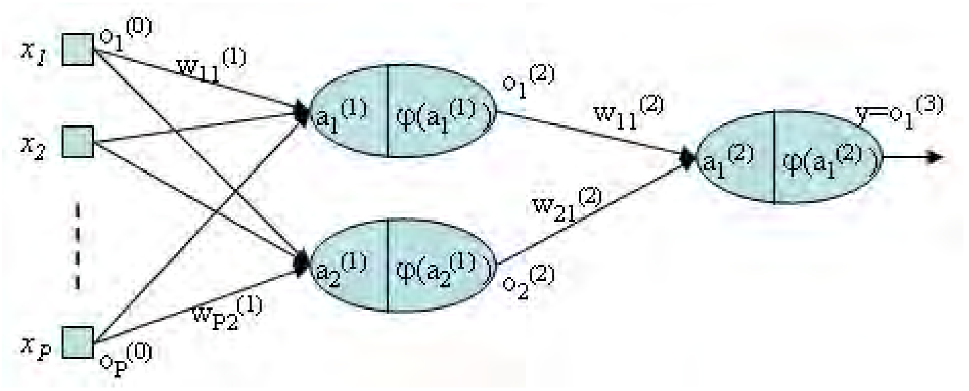


Figure A.1: FFNN architecture with denoted signals

Often, to train the system, a cost function is given as a measurement of the ANN performance. The aim of the training process is to minimize this cost function.

$$E = \frac{1}{N}\sum_{i}^{N} E^i \tag{A.1}$$

The instantaneous value E^i of the total energy of pattern i is obtained by summing the errors e_q over all the neurons q in the output layer.

$$E^i = \frac{1}{2}\sum_{q} e_q^2 \tag{A.2}$$

where $e_q = d_q - o_q^{(2)}$

A.1.1 Derivatives

The aim of the training process is to adjust the free parameters (i.e. $\vec{w}$ to minimize the overall cost function). Back propagation uses gradient descent to adjust these weights in the direction of minimizing the gradients of each weight. The update is done as follow:

$$w_{lj}^{(k)} = w_{lj}^{(k)} + \triangle w_{lj}^{(k)} \tag{A.3}$$

$$\triangle w_{lj}^{(k)} = -\eta \frac{\partial E}{\partial w_{lj}^{(k)}} = -\eta \frac{\partial E}{\partial a_j^{(k)}} \frac{\partial a_j^{(k)}}{\partial w_{lj}^{(k)}} \tag{A.4}$$

Since $a_j^{(k)} = \sum_l w_{lj}^{(k)} o_l^{(k-1)}$, $\frac{\partial a_j^{(k)}}{\partial w_{lj}^{(k)}} = o_l^{(k-1)}$. Define an error term (also known as the local gradient at neuron j) $\delta_j^{(k)} \equiv \frac{\partial E}{\partial a_j^{(k)}}$, the updated term for the weight is therefore

$$\triangle w_{lj}^{(k)} = -\eta \delta_j^{(k)} o_l^{(k-1)} \tag{A.5}$$

A.1.1.1 Error terms for the output layer

For the node q in the output layer, the error term is expanded as follow

$$\delta_q^{(k)} \equiv \frac{\partial E}{\partial a_j^{(k)}} = \frac{\partial E}{\partial E^i}\frac{\partial E^i}{\partial e_q}\frac{\partial e_q}{\partial o_q^{(k)}}\frac{\partial o_q^{(k)}}{\partial a_q^{(k)}} = 1 \times e_q \times (-1) \times \varphi'(a_q^{(k)}) \tag{A.6}$$

Recall with sigmoid function, $\varphi'(x) = \varphi(x)[1 - \varphi(x)]$, which leads to

$$\delta_q^{(k)} = -(d_q - o_q^{(k)})o_q^{(k)}(1 - o_q^{(k)}) \tag{A.7}$$

A.1.1.2 Error terms for the hidden layers

Each node j in hidden layer k is connected to nodes in the next layer $k+1$ through a set of weights. Therefore, the error term at hidden node j is the sum of the back propagated errors from all the nodes in the next layer to which j is connected.

$$\delta_j^{(k)} = -\sum_q w_{jq}^{(k+1)}\delta_q^{(k+1)}o_j^{(k)}(1 - o_j^{(k)}) \tag{A.8}$$

A.2 Back–propagation for mixture of experts on binary classification

Notation: since in this section, I introduce only the binary classification, the output of expert m is simplified as y_m, and output m of the gate is denoted as z_m, g_m. Also, this section deals with the uppermost level of the derivatives. To derive the complete derivative for each weight in the experts and the gate, one just need to apply the same chain rule on the error function in this section and the derivatives derived in section A.1.1. Therefore to update the weights, chain rules can be applied here in the same manner as above. For example, to update weight r of expert m, given $\frac{\partial E^i}{\partial y_m}$, which is derived

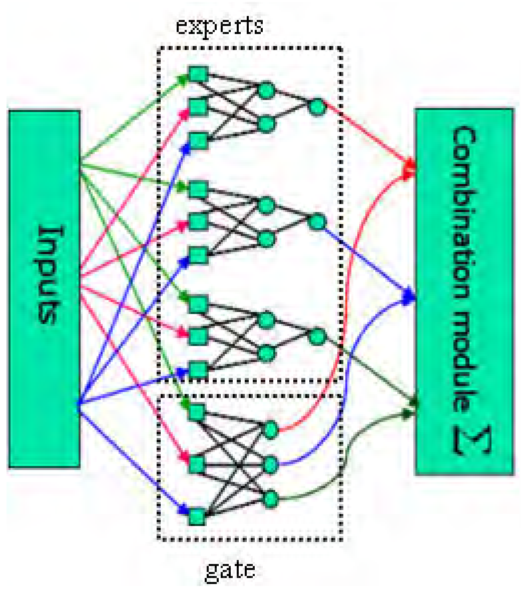


Figure A.2: Mixture of experts architecture

in this section, the chain rule can be applied as follow:

$$\triangle w_{r,m} = -\eta \frac{\partial E^i}{\partial w_{r,m}} = -\eta \frac{\partial E^i}{\partial y_m} \frac{\partial y_m}{\partial a_m} \frac{\partial a_m}{\partial w_{r,m}} \tag{A.9}$$

Let the weights of each expert m be $\vec{w}_m$, and the weights, corresponding to expert m, of the gate be $\vec{v}_m$. Since the gate has no hidden units, the raw output z_m of output node m of the gate is computed as the weighted sum of the inputs: $z_m = \vec{v}_m \vec{x}$. To maintain the probabilistic property of the gate output, this raw output is passed through a softmax function, resulted in

$$g_m = \frac{exp(z_m)}{\sum_j exp(z_j)} \tag{A.10}$$

Some useful derivatives for the gate:

$$\frac{\partial g_m}{\partial z_m} = g_m(1 - g_m) \tag{A.11}$$

$$\frac{\partial g_m}{\partial z_{j,j \neq m}} = -g_m g_j \tag{A.12}$$

In chapter 5, I has defined four error functions to be investigated in this thesis. Here, I will provide more derivations and derivatives of these error functions.

A.2.1 Residual cancelling error function

For each pattern i, the error function is defined as

$$E^i = \|d^i - \sum_m g^i_m y^i_m\|^2 \tag{A.13}$$

- For expert m

$$\frac{\partial E^i}{\partial y^i_m} = g^i_m(\sum_j g^i_j y^i_j - d^i) = g^i_m(y^i_{me} - d^i) \tag{A.14}$$

- For output m of the gate, which corresponds to expert m

$$\begin{aligned}\frac{\partial E^i}{\partial z^i_m} &= \frac{\partial E^i}{\partial g^i_m}\frac{\partial g^i_m}{\partial z^i_m} + \sum_{j,j\neq m}\frac{\partial E^i}{\partial g^i_j}\frac{\partial g^i_j}{\partial z^i_m} \\ &= g^i_m(1-g^i_m)y^i_m(y^i_{me} - d^i) - \sum_{j,j\neq m} g^i_m g^i_j y^i_j (y^i_{me} - d^i) \\ &= g^i_m(y^i_{me} - d^i)(y^i_m - y^i_{ens})\end{aligned} \tag{A.15}$$

A.2.2 Competitive error function

For each pattern i, the error function is defined as

$$E^i = \sum_m g^i_m (y^i_m - d^i)^2 \tag{A.16}$$

- For expert m

$$\frac{\partial E^i}{\partial y^i_m} = g^i_m(y^i_m - d^i) \tag{A.17}$$

- For output m of the gate, which corresponds to expert m

$$\begin{aligned}\frac{\partial E^i}{\partial z^i_m} &= \frac{\partial E^i}{\partial g^i_m}\frac{\partial g^i_m}{\partial z^i_m} + \sum_{j,j\neq m}\frac{\partial E^i}{\partial g^i_j}\frac{\partial g^i_j}{\partial z^i_m} \\ &= g^i_m(1-g^i_m)(y^i_m - d^i)^2 - \sum_{j,j\neq m} g^i_m g^i_j (y^i_j - d^i)^2 \\ &= g^i_m[(y^i_m - d^i)^2 - \sum_m g^i_m(y^i_m - d^i)^2]\end{aligned} \tag{A.18}$$

A.2.3 Gaussian error function

For each pattern i, the error function is defined as

$$E^i = -log \sum_m g^i_m exp(-\frac{1}{2}\|y^i_m - d^i\|^2) \tag{A.19}$$

Let $\Psi_m = exp(-\frac{1}{2}\|y^i_m - d^i\|^2)$

- For expert m

$$\frac{\partial E^i}{\partial y^i_m} = 2h^i_m(y^i_m - d^i) \tag{A.20}$$

where

$$h^i_m = [\frac{g^i_m \Psi_m}{\sum_j g^i_j \Psi_j}] \tag{A.21}$$

is the posterior probability.

- For output m of the gate, which corresponds to expert m, since

$$\frac{\partial E^i}{\partial g^i_m} = -\frac{\Psi_m}{\sum_j g^i_j \Psi_j} \tag{A.22}$$

$$\frac{\partial E^i}{\partial z^i_m} = \frac{\partial E^i}{\partial g^i_m}\frac{\partial g^i_m}{\partial z^i_m} + \sum_{j,j\neq m} \frac{\partial E^i}{\partial g^i_j}\frac{\partial g^i_j}{\partial z^i_m} = g^i_m - h^i_m \tag{A.23}$$

A.2.4 Multi–nomial error function on binary classification

For each pattern i, the error function is defined as

$$E^i = -log \sum_m g^i_m (y^i_m)^{d^i}(1 - y^i_m)^{(1-d^i)} \tag{A.24}$$

Let $\Psi_m = (y^i_m)^{d^i}(1 - y^i_m)^{(1-d^i)}$

The posterior probability is defined as

$$h^i_m = \frac{g^i_m \Psi_m}{\sum_j g^i_j \Psi_j} \tag{A.25}$$

- For expert m

$$\frac{\partial E^i}{\partial y^i_m} = 2h^i_m(y^i_m - d^i) \tag{A.26}$$

- For output m of the gate, which corresponds to expert m

$$\frac{\partial E^i}{\partial z^i_m} = \frac{\partial E^i}{\partial g^i_m}\frac{\partial g^i_m}{\partial z^i_m} + \sum_{j,j\neq m}\frac{\partial E^i}{\partial g^i_j}\frac{\partial g^i_j}{\partial z^i_m} = g^i_m - h^i_m \tag{A.27}$$

Appendix B

Dataset Descriptions

The following fifteen benchmark datasets are taken from the UCI machine learning repository (Newman, Hettich, Blake, and Merz 1998) and the StatLog database (King, Feng, and Shutherland 1995). The datasets are categorized as small and medium types based on the number of attributes: small datasets have at most 20 attributes while medium ones have more than 20 attributes.

B.1 Small datasets

The breast cancer dataset was originally obtained from W. HG. Wolberg at the University of Wisconsin Hospitals, Madison. The set is divided into two classes: benign or malignant. The set has 699 instances (as of 15 July 1992) represented by 9 categorical attributes. 458 instances are benign and 241 are malignant.

The diabetes dataset was donated by Vincent Sigillito from Johns Hopkins University and was constructed by constrained selection from a larger database by the National Institute of Diabetes and Digestive and Kidney Diseases. All patients here are females at least 21 years old of Pima Indian heritage. The set consists of 768 instances of 8 attributes. 500 examples tested negative for diabetes, and 268 are tested positive.

The liver disorder dataset was donated by Richard S. Forsyth from the collected data by BUPA Medical Research Ltd. The set has 345 instances with 6 attributes. The class distribution is 145 (class 0) and 200 (class 1).

The Australian credit card assessment dataset contains 690 instances of 15 attributes. There are two classes: + with 307 instances and – with 383 instances.

The Tic Tac Toe endgame dataset was created and donated by David W. Aha on 19 August 1991. This database encodes the complete set of possible board configurations at the end of Tic Tac Toe end games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). There are 958 instances of 9 attributes (each corresponding to one Tic Tac Toe square). There are two classes: negative (332 instances) and positive (625 instances).

The Cleveland heart disease dataset was donated by David W. Aha on July, 1988. The purpose is to distinguish the presence and absence of heart disease in the patient. This database originally contains 76 attributes, but only a subset of 7 categorical and 6 continuous attributes is used in all published experiments. There are two classes: presence (139 instances) and absence (164 instances).

The StatLog heart disease dataset was obtained from the StatLog database (King, Feng, and Shutherland 1995). The purpose us to distinguish the presence and absence of heart disease in the patient. There are two classes: presence (120 instances) and absence (150 instances) of heart disease.

The hepatitis dataset was donated by G.Gong (Carnegie-Mellon University). The dataset is divided into two classes: die (32 instances) and live (123 instances). There are 6 continuous and 13 categorical attributes in the dataset.

The Ljubljana breast cancer dataset was obtained from the University Medical Center, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. This data set includes 201 instances of the no-recurrence-events class and 85 instances of the recurrence-events class. The instances are described by 9 attributes.

The house voting 84 dataset was donated by Jeff Schlimmer on 27 April 1987. The data were collected from the 98th Congress Congressional Quarterly Almanac, 2nd session 1984, Volume XL: Congressional Quarterly Inc., Washington, D.C., 1985. This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition). There are two classes: democrats (267 instances) and republicans (168 instances).

B.2 Medium datasets

The German credit card dataset (King, Feng, and Shutherland 1995) was donated by Professor Dr. Hans Hofmann Institut f"ur Statistik und "Okonometrie Universit" at Hamburg. For algorithms that need numerical attributes, Strathclyde University produced the file "german.data-numeric", which has been edited and several indicator variables added to make it suitable for algorithms which cannot cope with categorical variables. Several attributes that are ordered categorical (such as attribute 17) have been coded as integer. There are totally 24 attributes in the edited file. There are two classes: good credit (700 instances) and bad credit (300 instances).

The ionosphere dataset was donated by Vince Sigillito in 1989. "This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. Good radar returns are those showing evidence of some type of structure in the ionosphere. Bad returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this databse are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal" (Blake and Merz 1998). There are two classes: good (225 instances) and bad (126 instances)

The King Rook vs King Pawn dataset was donored by Rob Holte on 1 August 1989. The dataset describes a chess game of King Rook versus King Pawn on a7 (usually abbreviated KRKPA7). The pawn on a7 means it is one square away from queening. There are a total of 36 categorical attributes. It is the King Rook's side (white) to move. There are two classes: White-can-win (1669 instances) and White-cannot-win (1527 instances).

The E. coli promoter gene sequences (DNA) with associated imperfect domain theory dataset was donated by M. Noordewier and J. Shavlik on 30 June 1990. There are 57 attributes corresponding to 57 sequential nucleotide (”base-pair”) positions. There are two classes: positive (53 instances) and negative (53 instances).

The Thyroid disease dataset was supplied by the Garavan Institute and J. Ross Quinlan, New South Wales Institute, Sydney, Australia in 1987. The dataset records an archive of thyroid diagnoses obtained from the Garvan Institute. There are 2800 training (data) instances and 972 test instances in total with 27 attributes. The are two classes: sick (171 training instances and 60 testing instances) and negative (2629 training instances and 912 testing instances).

Bibliography

Abbass, H. A. (2002a). An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine 25*(3), 265–281.

Abbass, H. A. (2002b). Self-adaptive pareto differential evolution. In *Proceedings of the 2002 Congress on Evolutionary Computation*, Volume 1, pp. 831–836. IEEE Press.

Abbass, H. A. (2003a). Pareto neuro ensembles. In *Proceedings of Australian Conference on Artificial Intelligence*, pp. 554–566. Springer.

Abbass, H. A. (2003b). Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In *The IEEE Congress on Evolutionary Computation*, Volume 3, pp. 2074–2080. IEEE-Press.

Abbass, H. A. and K. Deb (2003, April). Searching under Multi-evolutionary Pressures. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele (Eds.), *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, Faro, Portugal, pp. 391–404. Springer. Lecture Notes in Computer Science. Volume 2632.

Abbass, H. A. and R. Sarker (2002). The pareto differential evolution algorithm. *International Journal on Artificial Intelligence Tools 11*(4), 531–552.

Adamidis, P. (1994, Nov). Review of parallel genetic algorithms bibliography. Internal tech. report, Aristotle University of Thessaloniki, Greece, Automation and Robotics Lab., Dept. of Electrical and Computer Eng.

Aksela, M. (2003, June). Comparison of classifier selection methods for improving commitee performance. In T. Windeatt and F. Roli (Eds.), *Multiple Classifier Systems*, Volume 2709 of *Lecture Notes in Computer Science*, Guilford, UK, pp. 84–93. Springer-Verlag.

Alba, E., J. F. Aldana, and J. M. Troya (1993). Fully automatic ANN design: a genetic approach. In *Proc. of International Workshop on Artificial Neural Networks*, Volume 686 of *Lecture Notes in Computer Science*, pp. 399–404. Springer-Verlag.

Alexandre, L. A., A. Campilho, and M. Kamel (2004). Bounds for the average generalization error of the mixture of experts neural network. In A. Fred (Ed.), *SSPR/SPR*, pp. 618–625. Springer-Verlag.

Angeline, P. J., G. M. Sauders, and J. B. Pollack (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transaction on Neural Networks 5*(1), 54–65.

Avnimelech, R. and N. Intrator (1999). Boosted mixture of experts: An ensemble learning scheme. *Neural Computation 11*(2), 483 – 497.

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolution Programming, Genetic Algorithms*. Oxford University Press, Inc., USA.

Balakrishnan, K. and V. Honavar (1995). Evolutionary Design of Neural Architectures: A Preliminary Taxonomy and Guide to Literature. Technical report, Department of Computer Science, Iowa State University, Ames, Iowa.

Banfield, R. E., L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer (2003, June). A new ensemble diversity measure applied to thinning ensembles. In T. Windeatt and F. Roli (Eds.), *Multiple Classifier Systems*, Volume 2709 of *Lecture Notes in Computer Science*, Guilford, UK, pp. 306–316. Springer-Verlag.

Barber, B. C., D. P. Dobkin, and H. Huhdanpaa (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22*(4), 469–483.

Barlett, P. and T. Downs (1990, January). Training a neural network with a genetic algorithm. Technical report, Department of Electrical Engineering, University of Queensland.

Baxter, J. (1992). The evolution of learning algorithms for artificial neural networks. In D. Green and T.Bossomaier (Eds.), *Complex Systems*, pp. 313–326. IOS Press, Amsterdam.

Beer, R. D. and J. C. Gallagher (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior 1*, 91–122.

Bengio, Y. and S. Bengio (1990). Learning a synaptic learning rule. Technical Report 751, Deparment d'Informatique et de Recherche Operationelle, Universite de Motreal, Canada.

Binner, J. M. and G. Kendall (2002). Co-evolving neural networks with evolutionary strategies: A new application to divisia money. In *IC-AI*, pp. 884–889.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, New York.

Blake, C. L. and C. J. Merz (1998). UCI repository of machine learning databases, html://www.ics.uci.edu/ mlearn/mlrepository.html.

Bongard, J. C. and R. Pfeifer (2003). Evolving complete agents using artificial ontogeny. In F. Hara and R. Pfeifer (Eds.), *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, pp. 237–258. Springer-Verlag.

Bornholdt, S. and D. Graudens (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks 5*, 327–334.

Bouckaert, R. (2002). Accuracy bounds for ensembles under 0 - 1 loss. citeseer.ist.psu.edu/bouckaert02accuracy.html.

Breiman, L. (2000). Randomizing output to increase prediction accuracy. *Machine Learning 40*(3), 229–242.

Brown, G. (2004, January). *Diversity in Neural Network Ensemble*. Ph. D. thesis, The University of Birmingham.

Brown, G. and J. Wyatt (2003, June). Negative correlation learning and the ambiguity family of ensemble methods. In T. Windeatt and F. Roli (Eds.), *Proceedings of the 4th International Workshop on Multiple Classifier Systems*, Volume 2709 of *Lecture Notes in Computer Science*, Guilford, UK, pp. 266–275. Springer-Verlag.

Brown, G., J. Wyatt, R. Harris, and X. Yao (2004). Diversity creation methods: a survey and catergorisation. *Information Fusion 6*(1), 5–20.

Campbell, C. (1997). Constructive learning techniques for designing neural network systems. In *Neural Network Systems Technologies and Applications*. San Diego: Academic Press.

Cantu-Paz, E. and C. Kamath (2005, October). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B 35*(5), 915 – 927.

Caudell, T. P. and C. P. Dolan (1989). Parametric connectivity: Training of constrained networks using genetic algorithms. In J. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, pp. 370–374. Morgan Kaufmann, San Mateo, CA.

Chalmers, D. (1990). The evolution of learning: An experiment in genetic connectionism. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (Eds.), *Proceedings of the 1990 Connectionist Summer School Workshop*, pp. 81–90. Morgan Kaufmann.

Chandra, A. and X. Yao (2004). DIVACE: Diverse and accurate ensemble learning algorithm. In *Proc. of the Fifth International Conference on Intelligent Data Engineering and Automated Learning*, Volume 3177 of *Lecture Notes in Computer Science*, pp. 619–625. Springer.

Cho, S.-B. and J.-H. Ahn (2001). Speciated neural networks evolved with fitness sharing technique. In *Proceedings of the 2001 Congress on Evolutionary Computation*, Volume 1, Korea, pp. 390–396. IEEE.

Cho, S.-B., J.-H. Ahn, and S.-I. Lee (2001). Exploiting diversity of neural ensembles with speciated evolution. In *Proceedings. IJCNN '01. International Joint Conference on Neural Networks*, Volume 2, pp. 808–813. IEEE.

Christensen, S. W. (2003, June). Ensemble construction via designed output distortion. In T. Windeatt and F. Roli (Eds.), *Proceedings of the 4th International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, Guilford, UK, pp. 286–295. Springer-Verlag.

Coello Coello, C. A. (2002, February). Evolutionary Multi-Objective Optimization: A Critical Review. In R. Sarker, M. Mohammadian, and X. Yao (Eds.), *Evolutionary Optimization*, pp. 117–146. New York: Kluwer Academic Publishers. ISBN 0-7923-7654-4.

Crosher, D. (1993). The artificial evolution of a generalized class of adaptive processes. In X. Yao (Ed.), *Preprints of AI'93 Workshop on Evolutionary Computation*, pp. 18–36.

Cunningham, P. and J. Carney (2000, May 31 - June 2). Diversity versus quality in classification ensembles based on feature selection. In *11th European Conference on Machine Learning*, Volume 1810, Barcelona, Catalonia, Spain, pp. 109–116. Springer, Berlin.

De Garis, H. (1991). Using the genetic algorithm to train time dependent behaviors in neural networks. In R. Michalski and G. Tecuci (Eds.), *Proc. of the First International Workshop on Multistrategy Learning*, pp. 273–280. Center for Artificial Intelligence, Fairfax, VA, USA.

Dellaert, F. and R. D. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. In R. M. Maes and P. (Eds.), *Artificial Life IV*. Cambridge: MIT Press.

Di Ferdinando, A., R. Calabretta, and D. Parisi (2001). Evolving modular architectures for neural networks. In R. French and J. Sougn (Eds.), *Proceedings of the Sixth Neural Computation and Psychology Workshop Evolution, Learning, and Development*, pp. 253–262. London: Springer Verlag.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine 18*, 97136.

Dill, F. A. and B. C. Deer (1991). An exploration of genetic algorithms for the selection of connection weights in dynamical neural networks. In *Proc. of IEEE 1991 National Aerospace and Electronics Conference*, Volume 3, pp. 1111–1115. IEEE Press, New York, NY.

Dodd, N. (1991). Optimisation of neural network structure using genetic techniques. In G. Rzevski and R. Adey (Eds.), *Proc. of the Conference on Applocations of Artificial Intelligence in Engineering VI*, pp. 939–944. Elsevier Applied Science, London, UK.

Efron, B. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans*. Philadelphia: Society for Industrial and Applied Mathematics.

Eggenberger, P. (1997). Creation of neural networks based on developmental and evolutionary principles. In *ICANN*, pp. 337–342.

Eggenberger, P. (2000). Evolving neural network structures using axonal growth mechanisms. In S.-I. Amari, C. Giles, M. Gori, and V. Piuri (Eds.), *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000. IJCNN 2000*, Volume 6, Como, Italy, pp. 591–595. IEEE.

Eggenberger, P. (2001). Suitable evolutionary strategies for large scale neural networks. In *Proceedings of The Sixth Int. Symp. on Artificial Life and Robotics (AROB 6th'01)*, pp. 152–153.

Eriksson, K., D. Estep, and C. Johnson (2004). *Applied Mathematics: Body and Soul*, Volume 3. Springer.

Fahlman, S. and C. Lebiere (1990). The cascade correlation architecture. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Volume 2, pp. 524–532. Denver: Morgan Kaufmann, San Mateo.

Festing, M. F. and D. G. Altman (2005). Guidelines for the design and statistical analysis of experiments using laboratory animals. *ILAR J/National Research Council, Institute of Laboratory Animal Resources. 46*(3), 244–258.

Fogel, D. B. (1993). Using evolutionary programing to create neural networks that are capable of playing tic-tac-toe. In *IEEE International Conference on Neural Networks*, pp. 875–880. IEEE, San Francisco, CA, USA.

Fogel, D. B., L. J. Fogel, and V. W. Porto (1990). Evolving neural networks. *Biological Cybernetics 63*, 487 – 493.

Fogel, D. B., E. C. Wasson, and E. M. Boughton (1995). Evolving neural networks for detecting breast cancer. *Cancer Letters 96*(1), 49–53.

Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation. *Science 261*, 872–878.

Freund, Y. and R. Schapire (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteen International Conference on Machine Learning*, pp. 149–156.

Freund, Y. and R. E. Schapire (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pp. 23–37.

Fumela, G. and F. Roli (August 6-9, 2002, August 6-9). Performance analysis and comparison of linear combiners for classifier fusion. In T. Caelli, A. Amin, R. P. W. Duin, M. S. Kamel, and D. de Ridder (Eds.), *Structural, Syntactic, and Statistical*

Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Proceedings, Volume 2396 of *Lecture Notes in Computer Science*, Windsor, Ontario, Canada, pp. 424–432. Springer.

Gabryel, M., K. Cpalka, and L. Rutkowski (2005). Evolutionary strategies for learning of neuro-fuzzy systems. In *International Workshop on Genetic Fuzzy Systems*, pp. 119–123.

Gallant, S. I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks 1*(2), 179–191.

Garcia-Pedrajas, N., C. Hervas-Martinez, and D. Ortiz-Boyer (2005). Cooperative coevolution of artificial neural network ensembles for pattern recognition. *IEEE Transaction on Evolutionary Computation 9*(3), 271–302.

Gelman, A. (2005). Analysis of variance why it is more important than ever. *Annals of Statistics 33*(1), 1–33.

Goutte, C. and L. K. Hansen (1997). Regularization with a pruning prior. *Neural Networks 10*(6), 1053–1059.

Greenwood, G. W. (1997). Training partially recurrent neural networks using evolutionary strategies. *IEEE Transactions on Speech and Audio Processing 5*(2), 192 – 194.

Gruau, F. (1994a). Automatic definition of modular neural networks. *Adaptive Behavior 3*, 151–183.

Gruau, F. (1994b). *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. Phd thesis, Ecole Normale Suprerieure de Lyon, France.

Hansen, J. V. (2000). *Combining Predictors: Meta Machine Learning Methods and Bias/Variance and Ambiguity Decompositions*. Ph. D. thesis, Department of Computer Science, University of Aarhus, Denmark.

Hansen, J. V. and R. D. Meservy (1996). Learning experiments with genetic optimization of a generalized regression neural network. *Decision Suppoprt Systems 18*(3-4), 317–325.

Hassibi, B. and D. G. Stork (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, Volume 5, pp. 164–171. Morgan Kaufman.

Haykin, S. S. (1999). *Neural networks : a comprehensive foundation*. Prentice-Hall, Upper Saddle River, New Jersey, USA.

Hayward, R. (1999). *Analytical and Inductive Learning in an Efficient Rule-Based Connectionist Reasoning System*. Phd thesis, Computer Science, QUT.

Heistermann, J. and H. Eckardt (1989). Parallel algorithms for learning in neural networks with evolution strategy. In D. Evans, G. Jourbert, and F. Peters (Eds.), *Proceedings of Parallel Computing 89*, pp. 275–280. Elsevier Science Publishers B.V., Amsterdam.

Hopkins, W. G. (2000). A new view of statistics. http://newstatsi.org. Accessed 15 December 2005.

Horn, J., D. E. Goldberg, and K. Deb (1994). Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation 2*(1), 37–66.

Ishikawa, M. (1996). Structural learning with forgetting. *Neural Networks 9*(3), 509–521.

Ishikawa, M. and K. Yoshino (1993). Automatic task decomposition in modular networks by structural learning with forgetting. In *Proceedings of 1993 International Joint Conference on Neural Networks*, pp. 1345–1348.

Jackson, J. E. (1991). *A User's Guide to Principal Components*. Wiley series in Probability and Mathematical Statistics. John Wiley & Sons, inc, New York.

Jacobs, R. A., M. I. Jordan, and A. G. Barto (1991). Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks. Technical report, University of Massachusetts.

Jacobs, R. A., M. J. Jordan, S. J. Nowlan, and G. E. Hinton (1991). Adaptive mixtures of local experts. *Neural Computation 3*, 79 – 87.

Jimenez, D. and N. Walsh (1998). Dynamically weighted ensemble neural networks for classification. In *Proceedings of the 1998 International Joint Conference on Neural Networks*, pp. 753–756.

Jin, Y., T. Okabe, and B. Sendhoff (2004). Neural network regularization and ensembling using multi-objective evolutionary algorithms. *Congress on Evolutionary Computation 1*, 1–8.

Jolliffe, I. T. (1986). *Principle Component Analysis*. Springer series in Statistics. Springer-Verlag.

Jordan, M. I. and R. A. Jacobs (1992). Hierarchies of adaptive experts. In J. Moody, S. Hanson, and R. Lippmann (Eds.), *Advances in Neural Information Processing Systems*, Volume 4, pp. 985–992. Morgan Kaufmann.

Jordan, M. I. and R. A. Jacobs (1994). Hierachical mixtures of experts and the EM algorithm. *Neural Computation 6*(2), 181–214.

Jordan, M. I. and L. Xu (1993, November). *Convergence Results for the EM Approach to Mixtures of Experts Architectures*. Aritificial Intelligence Laboratory@MIT.

Kang, K. and J.-H. Oh (1997). Statistical mechanics of the mixture of experts. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9, pp. 183. The MIT Press.

Khare, V., X. Yao, B. Sendhoff, Y. Jin, and H. Wersing (2005, September). Coevolutionary modular neural networks for automatic problem decomposition. In *Proc. of The 2005 IEEE Congress on Evolutionary Computation*, pp. 2691–2698. IEEE Press.

Kim, H. B., S. H. Jung, T. G. Kim, and K. H. Part (1996). Fast learning method for backpropagation neural network by evolutionary adaptation of learning rates. *Neurocomputing 11*(1), 101–106.

King, R., C. Feng, and A. Shutherland (1995, May/June). Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence 9*(3), 259–287.

Kinnebrock, W. (1994). Accelerating the standard backpropagation method using a genetic approach. *Neurocomputing 6*(5-6), 583–588.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems 4*, 461–476.

Koeppen, M., M. Teunis, and B. Nickolay (1997). Neural network that uses evolutionary learning. In *Proc. of the 1997 IEEE International Conference on Evolutionary Computation*, pp. 635–639. IEEE Press.

Kondo, N., T. Hatanaka, and K. Uosaki (2005, June 2-5, 2005). Pareto RBF network ensemble using multi-objective evolutionary computation. In *International Conference on Control, Automation and Systems*. Gyeonggi-Do, Korea.

Koza, J. R. and J. P. Rice (1991). Genetic generation of both the weights and architecture for a neural network. In *Proc. of 1991 IEEE International Joint Conference on Neural Networks*, Volume 2, pp. 297–404. IEEE Press, New York, NY.

Kozma, R., M. Kitamura, A. Malinowski, and J. M. Zurada (1995, November). On performance measures of artificial neural networks trained by structural learning algorithms. In *Proc. 2nd New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pp. 22–25. IEEE Computer Soc. Press.

Krogh, A. and J. Vedelsby (1995). Neural network ensembles, cross validation and active learning. In G. Tesauro, D. Touretzky, and T. Len (Eds.), *Advances in Neural Information Processing System*, Volume 7, pp. 231–238. MIT Press.

Kuncheva, L. I. (2003a, June). Error bounds for aggressive and conservative AdaBoost. In T. Windeatt and F. Roli (Eds.), *Proceedings of the 4th International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, Guilford, UK, pp. 25–34. Springer-Verlag.

Kuncheva, L. I. (2003b). That elusive diversity in classifier ensembles. In *Proc IbPRIA 2003, Mallorca, Spain, 2003, Lecture Notes in Computer Science*, pp. 1126–1138. Springer-Verlag.

Kuscu, I. (1995, 10). Evolution of learning rules for supervised tasks I: simple learning problems. Technical Report 394, Falmer, Brighton, Sussex, UK.

Laumanns, M., L. Thiele, K. Deb, and E. Zitzler (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation 10*(3), 263–282.

Lazarevic, A. and Z. Obradovic (2001). Effective pruning of neural network classifier ensembles. In *Proceedings. International Joint Conference on Neural Networks*, Volume 2, pp. 796–801.

Le Cunn, Y., J. S. Denker, and S. A. Solla (1990). Optimal brain damage. In *Advances in Neural Information Processing Systems*, Volume 2, pp. 598–605. Morgan Kaufman.

Likartsis, A., I. Vlachavas, and L. H. Tsoukalas (1997). New hybrid neural-genetic methodology for improving learning. In *Proc. of the 9th IEEE International Conference on Tools with Artificial Intelligence*, pp. 32–36. IEEE Press.

Likothanassis, S. D., E. Georgopoulos, and D. Fotakis (1997). Optimizing the structure of neural networks using evolution techniques. In *Proc. of the 5th International Conference on the Application of High-Performance Computers in Engineering*, pp. 157–168. Computational Mechanics Publisher.

Lin, G., X. Yao, and I. Macleod (1996). Parallel genetic algorithm on PVM. In *Proceedings of the International Conference on Parallel Algorithms*, Volume 1, pp.

605–610. Wuhan, P.R. China.

Liu, Y. and X. Yao (1996). A population-based learning algorithm which learns both architectures and weights of neural networks. *Chinese Journal of Advanced Software Research 3*(1), 54–65.

Liu, Y. and X. Yao (1997). Evolving modular neural networks which generalise well. In *Proc. of 1997 IEEE International Conference on Evolutionary Computation*, Indianapolis, USA, pp. 605–610.

Liu, Y. and X. Yao (1999). Ensemble learning via negative correlation. *Neural Networks 12*, 1399–1404.

Liu, Y., X. Yao, and T. Higuchi (2000). Evolutionary ensembles with negative correlation learning. *IEEE Trans. Evolutionary Computation 4*(4), 380–387.

Lozowski, A., D. A. Miller, and J. M. Zurada (1996, May 12-15). Dynamics of error backpropagation learning with pruning in the weight space. In *IEEE International Symposium on Circuits and Systems*, Volume 3 of *Connecting the World*, pp. 449–452.

Magoulas, G., V. Plagianakos, and M. Vrahatis (2001). Hybrid methods using evolutionary algorithms for on–line training. In *Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN'2001), Washington D.C.*

Maniezzo, V. (1994). Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks 5*(1), 39–53.

Marshall, S. J. and R. F. Harrison (1991). Optimization and training of feedforward neural networks by genetic algorithms. In *Proc. of the Second IEE International Conference on Artificial Neural Networks*, pp. 39–43. IEE Press, London, UK.

MathWorks (1997, December 31st). Matlab helpdesk. http://www-ccs.ucsd.edu/matlab/.

McKay, R. and H. A. Abbass (2001). Anti-correlation: A diversity promoting mechanism in ensemble learning. *Tha Australian Journal of Intelligence Information Processing Systems 7*(3/4), 139–149.

Mezard, M. and J. P. Nadal (1989). Learning in feedforward layered networks: the tiling algorithm. *Journal of Physics A22*, 2191–2203.

Miller, D. A. and J. M. Zurada (1997). Dynamics of structural learning with an adaptive forgetting rate. In *Proc. of the 1997 International Joint Conference on Neural Networks*, pp. 1827–1832.

Miller, G. F., P. M. Todd, and S. U. Hedge (1989). Designing neural networks using genetic algorithms. In J. Schaffer (Ed.), *Proc. of the Third International Conference on Genetic Algorithms and Their Applications*, pp. 379–384. Morgan Kaufmann, San Mateo, CA.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press.

Moerland, P. (1997a). Mixtures of experts estimate a posteriori probabilities. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds.), *Proceedings of the International Conference on Artificial Neural Networks (ICANN'97)*, Number 1327 in Lecture Notes in Computer Science, Berlin, pp. 499–504. Springer-Verlag. (IDIAP-RR 97-07).

Moerland, P. (1997b, November). Some methods for training mixtures of experts. Technical report, Dalle Molle Institute for Perceptive Artificial Intelligence.

Mondada, F. and D. Floreano (1995). Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems 16*(2-4), 183–195.

Montana, D. and L. Davis (1989). Training feedforward neural networks unsing genetic algorithms. In *Proceedings of Eleventh International Joint Conference on Artificial Intelligence*, pp. 762 – 767. Morgan Kaufmann.

Navone, H. D., P. F. Verdes, P. M. Granitto, and H. A. Ceccatto (2000). Selecting diverse members of a neural network ensemble. In *SBRN 2000, VI Brazilian Symposium on Neural Networks*, Rio de Janeiro, Brazil.

Neirotti, J. P. and N. Caticha (2003). Dynamics of the evolution of learning algorithms by selection. In *Physical review E*, Volume 67 of *Statistical, nonlinear, and soft matter physics*. the American Physical Society.

Newman, D. J., S. Hettich, C. L. Blake, and C. J. Merz (1998). UCI repository of machine learning databases.

Nguyen, M.-H., H. A. Abbass, and R. McKay (2004). Diversity and neuro ensemble. In A. Ghosh and L. Jain (Eds.), *Evolutionary Computing in Data Mining*, pp. 125–153. Physica-Verlag, Germany.

Nolfi, S. and D. Floreano (1999). Learning and evolution. *Autonomous Robots 7*(1), 89–113.

Nolfi, S. and D. Parisi (1992). Growing neural networks. Technical report, Institute of Psychology, CNR, Rome,.

Oliveira, L. S., R. Sabourin, F. Bortolozzi, and C. Y. Suen (2003, August 3-6, 2003). Feature selection for ensembles: A hierarchical multi-objective genetic algorithm approach. In *7th International Conference on Document Analysis and Recognition (ICDAR2003)*, Volume 2, Edinburgh-Scotland.

Opitz, D. W. and J. W. Shavlik (1996). Actively searching for an effective neural network ensemble. *Connection Science. Special Issue on Combining Artificial Neural: Ensemble Approaches 8*(3/4), 337–354.

Osmera, P. (1995). Optimization of neural networks by genetic algorithms. *Neural Network World 5*(6), 965–976.

Oza, N. C. (2003, June). Boosting with averaged weight vectors. In T. Windeatt and F. Roli (Eds.), *Proceedings of the 4th International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, Guilford, UK, pp. 15–24.

Springer-Verlag.

Pennock, D. M., P. Maynard-Reid II, C. L. Giles, and E. Horvitz (2000, June). A normative examination of ensemble learning algorithms. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, Stanford, CA, pp. 735–742.

Perrone, M. P. and L. N. Cooper (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone (Ed.), *Neural Networks for Speech and Image Processing*, pp. 126–142. Chapman-Hall.

Popovici, E. and K. A. D. Jong (2003). Understanding ea dynamics via population fitness distributions. In *GECCO*, pp. 1604–1605.

Popovici, E. and K. A. D. Jong (2005). Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In *GECCO*, pp. 1507–514.

Porto, V. W., D. B. Fogel, and L. J. Fogel (1995). Alternative neural network training methods. *IEEE Expert 10*(3), 16–22.

Potter, M. and K. De Jong (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation 8*(1), 1–29.

Potter, M. A. (1997). *The Design and Analysis of a Computational Model of Cooperative Coevolution*. Ph. D. thesis, George Mason University.

Prados, D. L. (1992). New learning algorithm for training multilayer neural networks that uses genetic-algorithm techniques. *Electronics Letters 28*, 1560–1561.

Pujol, J. C. F. and R. Poli (1998). Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence 8*(1), 73–84.

Radi, A. and R. Poli (2003). Discovering efficient learning rules for feedforward neural networks using genetic programming. In *Recent advances in intelligent paradigms and applications*, pp. 133–159. Heidelberg, Germany, Germany: Physica-Verlag GmbH.

Ragg, T. and S.Gutjahr (1997). Automatic determination of optimal network topologies based on information theory and evolution. In *Proc. of the 1997 23rd EUROMICRO Conference*, pp. 549–555. IEEE Computer Society Press.

Ramamurti, V. and J. Gosh (1996). Structural adaptation in mixture of experts. In *IEEE Proceedings of International Conference on Pattern Recognition*, pp. 704–708.

Raviv, Y. and N. Intrator (1999). Variance reduction via noise and bias contraints. In A. Sharkey (Ed.), *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, Chapter 7, pp. 163–175. Springer-Verlag.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Rechenberg, I. (1994). *Evolutionsstrategie '94*, Volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Frommann-Holzboog, Stuttgart.

Renner, R. S. (1999). *Improving generalization of constructive neural networks using ensembles*. Phd, The Florida State University.

Rojas, R. (1995). *Neural Networks A Systematic Introduction*. Springer-Verlag, Berlin, Germany.

Rosen, B. E. (1996). Ensemble learning using decorrelated neural networks. *Connection Science. Special Issue on Combining Artificial Neurals: Ensemble Approaches 8*(3/4), 373–384.

Rosin, C. D. and R. K. Belew (1995). Methods for competitive co-evolution: Finding opponents worth beating. In L. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Francisco, CA, pp. 373–380. Morgan Kaufmann.

Rosin, C. D. and R. K. Belew (1997). New methods for competitive coevolution. *Evolutionary Computation 5*(1), 1–29.

Ruta, D. and B. Gabrys (2001). Analysis of the correlation between majority voting error and the diversity mearsures in multiple classifier systems. In *Proceedings of the 4th International Symposium on Soft Computing*.

Saravanan, N. and D. B. Fogel (1995). Evolving neural control systems. *IEEE Expert 10*(3), 23–27.

Sarkar, M. and B. Yegnanarayana (1997). Feedforward neural networks configuration using evolutionary programming. In *Proc. of the 1997 IEEE International Conference on Neural Networks, Part 1 of 4*, pp. 438–443. IEEE Press.

Schaffer, J. D., R. A. Caruana, and L. J. Eshelman (1990). Using genetic search to exploit the emergent behavior of neural networks. *Physica D 42*, 244–248.

Schapire, R. E. (1990). The strength of weal learnability. *Machine Learning 5*, 197–227.

Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1401–1406.

Schwefehm, H. P. (1981). *Numerical Optimization for Computer Models*. John Willey, Chichester, U.K.

Secton, R. S., R. E. Dorsey, and J. D. Johnson (1998). Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems 22*(2), 171–185.

Sharkey, A. (1998). *Combining Artificial Neural Nets. Ensemble and Modular Multi-Net Systems*. Springer-Verlag New York, Inc.

Sharkey, A. J. C. (1996). On combining artificial neural nets. *Connection Science 8*, 299–313.

Shipp, C. A. and L. I. Kuncheva (2002). Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion 3*(2), 135–148.

Skalak, D. B. (1997). *Combining nearest neighbor classifiers*. Ph. D. thesis, Departtment of Computer Science, University of Massachusetts, Amherst MA.

Spears, W. M., K. A. D. Jong, T. Bäck, D. B. Fogel, and H. de Garis (1993). An overview of evolutionary computation. In P. B. Brazdil (Ed.), *Proceedings of the European Conference on Machine Learning (ECML-93)*, Volume 667, Vienna, Austria, pp. 442–459. Springer Verlag.

Srinivas, M. and L. M. Patnaik (1991). Learning neural network weights using genetic algorithms - imrpvocing performance by search space reduction. In *Proc. of 1991 IEEE International Jount Conference on Neural Networks*, Volume 3, pp. 2331–2336. IEEE Press, New York, NY.

Standish, R. (2002). Diversity evolution. In *Proceedings of the eighth international conference on Artificial life*, pp. 131–137. MIT Press, Cambridge, MA, USA.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evol. Comput. 10*(2), 99–127.

Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. Eighth Ann. Conf. Cognitive Science Soc.*, pp. 823–831.

Tang, B., M. I. Heywood, and M. Shepherd (2002). Input partitioning to mixture of experts. In *International Joint Conference on Neural Networks*, pp. 227–232.

Tang, K. S., C. Y. Chan, K. F. Man, and S. Kwong (1995). Genetic structure for NN topology and weights optimization. In *Proc. of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 250–255. IEE Conference Publication 414.

Thierens, D. (1996). Non-redundant genetic coding of neural networks. In *Proc. of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 571–575. IEEE Press.

Toffolo, A. and E. Benini (2003). Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation 11*(2), 151–167.

Topchy, A. P. and O. A. Lbedko (1997). Neural network training by means of co-operative evolutionary search. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 389*(1-2), 240–241.

Tsakonas, A. D. and G. Dounias (2002). A scheme for the evolution of feedforward neural networks using BNF-grammar driven genetic programming. In *European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, Algarve, Portugal.

Tumer, K. and J. Ghosh (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science 8*(3-4), 385–403.

Ueda, N. and R. Nakano (1996). Generalization error of ensemble estimators. In *IEEE International Conference on Neural Networks*, Volume 1, pp. 90–95.

Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph. D. thesis, The Air Force Institute of Technology, Air University, Wright-Patterson AFB, OH.

Wan, E. and D. Bone (1996). Interpolating earth-science data using RBF networks and mixtures of experts. In *NIPS*, pp. 988–994.

Waterhouse, S., D. MacKay, and T. Robinson (1996). Bayesian methods for mixture of experts. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, pp. 351–357. The MIT Press.

Waterhouse, S. R. (1997, February). *Divide and Conquer: Pattern Recognition using Mixtures of Experts*. Ph. D. thesis, University of Cambridge.

Waterhouse, S. R. and G. Cook (1997). Ensemble methods for phoneme classification. In M. Mozer, J. Jordan, and T. Petsche (Eds.), *Advances in Neural Information*

Processing Systems, Volume 9, pp. 800–806. MIT Press.

Weigend, A. S., D. E. Rumelhart, and B. A. Huberman (1990). Generalization by weight-elimination with application to forecasting. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, San Francisco, CA, USA, pp. 875–882. Morgan Kaufmann Publishers Inc.

White, D. and P. Ligomenides (1993). Gannet: a genetic algorithm for optimizing topology and weights in neural network design. In *Proc. of International Workshop on Artificial Neural Networks*, Volume 686 of *Lecture Notes in Computer Science*, pp. 322–327. Springer-Verlag.

Williams, P. M. (1993). Improved generalization and network pruning using adaptive laplace regularization. In *Third International Conference on Artificial Neural Networks*, pp. 76–80.

Williams, P. M. (1995). Bayesian regularisation and pruning using a laplace prior. *Neural Computation 7*(1), 117–143.

Wu, J.-X., Z.-H. Zhou, and Z.-Q. Chen (2001). Ensemble of GA based selective neural network ensembles. In *Proceedings of the 8th International Conference on Neural Information Processing*, Volume 3, pp. 1477–1482.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE 87*, 1423–1447.

Yao, X. and Y. Liu (1996). Ensemble structure of evolutionary artificial neural networks. In *IEEE International Conference on Evolutionary Computation (ICEC'96)*, Nagoya, Japan, pp. 659–664.

Yao, X. and Y. Liu (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks 8*, 694–713.

Yao, X. and Y. Liu (1998a). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics,* *Part B: Cybernetics 28*, 417–425.

Yao, X. and Y. Liu (1998b). Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation 91*(1), 83–90.

Zenobi, G. and P. Cunningham (2001). Using diversity in preparing ensemble of classifiers based on different subsets to minimize generalization error. In *Lecture Notes in Computer Science*, Volume 2167.

Zhou, Z.-H., J. Wu, and W. Tang (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence 137*(1-2), 239–263.

Zhou, Z.-H., J.-X. Wu, Y. Jiang, and S.-F. Chen (2001). Genetic algorithm based selective neural network ensemble. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pp. 797–802.

Zitzler, E. (1999, November). *Evolutionary Algoirthms for Multiobjective Optimization: Methods and Applications*. Ph. D. thesis, Swiss Federal Institute of Technology, Zurich, Switzer- land.