NEURAL NETWORKS TRAINING AND APPLICATIONS USING BIOLOGICAL DATA

A thesis submitted for the degree of Doctor of Philosophy
for the University of London

By
Aristoklis. D. Anastasiadis
Supervisor: Dr. G. D. Magoulas
School of Computer Science and Information Systems
December 2005

To my parents, and Dimitra.

Abstract

Training neural networks in classification problems, especially when biological data are involved, is a very challenging task. Many training algorithms have been proposed so far to improve the performance of neural networks. A popular approach is to use batch learning that employs a different adaptive learning rate for each weight. Most of the existing algorithms of this class are based on the use of heuristics, and they don't guarantee convergence to a local minimiser from any initial weights set. In addition, they converge frequently to local minima, particularly when training starts far away from a minimiser. To alleviate this situation, this PhD thesis proposes new methods that overcome these problems.

It proposes a new class of sign–based schemes with adaptive learning rates that are based on the composite nonlinear Jacobi process. It develops an adaptation strategy that ensures the search direction is a descent one and the decrease of the batch error is guaranteed. Moreover, it equips the new algorithms with the global convergence property; i.e. it proves convergence to a local minimiser from any remote starting point.

The problem of occasional convergence to local minima is then dealt within the context of global search methods by proposing a hybrid-learning scheme that combines deterministic and stochastic search, and adaptive learning rates. Stochastic search is explored in the context of Nonextensive Statistical Mechanics, by modifying the error surface during training using perturbations generated by the q-nonextensive entropic index. The proposed algorithms are applied to train feed-forward neural networks and diverse neural ensembles in biological and bioinformatics datasets.

Acknowledgements

The completion of this thesis came about as the result of invaluable support and friendship from numerous people.

First and foremost, my utmost gratitude goes out to my family for supporting me throughout the years. Without them, I would never have made it this far. I am also deeply grateful to my supervisor, Dr. Magoulas, who has had the most direct influence on my work over the past three years. He was a real friend and a mentor throughout these years. Through him, I have learnt to become more self-critical and meticulous in my work. I am extremely grateful for his encouragement, support and understanding over the years.

I cannot forget to acknowledge my undergraduate teacher Stavros Koubias at University of Patras, without whose support and assistance I would have never started this PhD. I would also like to address special thanks to Prof. Constantino Tsallis for our very useful discussions, during my stay at Santa Fe Institute where I was visiting researcher.

Special appreciation also goes out to my dependable friends Dionisis and George and many others in the Computer Science Department. I wish them all the best in their future endeavours.

My final words go to my family. Particularly, I would like to thank Dimitris, who helped me in my first steps of learning Matlab. I will be forever indebted to my parents, and Dimitra, without whose unconditional support, love, encouragement and reassurance I would have never finished this thesis. This thesis is dedicated to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Beauty is truth, truth beauty, – that is all Ye know on earth, and all ye need to know. John Keats, May 1819.

1.1 Introduction

Neural networks are very sophisticated modelling techniques capable of modelling extremely complex functions. Nowadays, they are being successfully applied across a wide range of problem domains, in areas such as finance, medicine, engineering, geology and physics.

Indeed, anywhere that there are problems of prediction or classification, neural networks are being introduced. However, neural network error surfaces are much more complex, and are characterised by a number of unhelpful features, such as local minima, flat-spots and plateaus, saddle-points, and long narrow ravines. These specific characteristics make the training of the neural network particularly difficult and constitute a crucial factor for the performance of neural networks [91, 87, 64, 49].

Many training algorithms have been proposed so far to improve the neural network's performance [49, 64, 124]. Batch learning that employs a different adaptive learning rate for each weight is very popular. Most of the existing algorithms of this class don't guarantee convergence to a local minimiser from any initial weights set. Predominantly, they converge frequently to local minima when training starts far away from a minimiser.

A variety of approaches inspired from the unconstrained optimisation theory has also been applied, in order to use second derivative related information to accelerate the learning process [13, 65, 71, 135]. Nevertheless, it is not certain that the extra computational cost these methods require leads to speed ups of the minimisation process for nonconvex functions when far from a minimiser [75]. This problem can be overcome through the use of global optimisation. The drawback of this class of methods is that they are very computationally expensive.

This PhD thesis proposes new methods that overcome these problems. This thesis investigates Optimisation methods for Artificial Neural Network training. It proposes sign–based schemes with adaptive learning rates that are based on the composite nonlinear Jacobi process. In addition, it equips the new algorithms with the global convergence property; i.e. it proves convergence to a local minimiser from any remote starting point.

Moreover, the problem of occasional convergence to local minima is also dealt within the context of Statistical Mechanics. A new hybrid-learning scheme with adaptive learning rates that combines deterministic and stochastic search, is presented. Stochastic search is explored in the context of Tsallis Statistical Mechanics, by modifying the error surface during training using the q-nonextensive entropic index.

Lastly, the proposed algorithms are applied for training feed-forward neural networks and diverse neural ensembles. Emphasis is given on classification problems

that use biological data. Also, in depth survey has been carried out for the prediction of the proteins' localisation sites. Two of the most thoroughly studied single-cell organisms, namely the bacterium Escherichia coli and the eukaryote Saccharomyces cerevisiae, also called Yeast [15, 58] were studied.

1.1.1 Aims

In this PhD, the development and implementation of novel appropriate training schemes is investigated, in order to overcome drawbacks of the existing training algorithms. The new proposed methods are tested on benchmark problems so as to be fine-tuned, and are then applied in problems that use biological data, as well as bioinformatics problems that cannot be solved with the existing approaches. Improvement of the learning speed, the classification accuracy and convergence success are important targets.

1.1.2 Objectives

The main objectives of the PhD are given below:

- Develop and implement new gradient descent based training algorithms for supervised learning.
- Propose sign–based schemes with adaptive learning rates that are based on the composite nonlinear Jacobi process.
- Develop a methodology to build globally convergent algorithms.
- Apply Nonextensive Statistical Mechanics in training Neural networks.
- Tested the performance of the proposed schemes using biological data. Also apply these algorithms to predict the localisation sites of the E.coli and Yeast proteins.

1.1.3 Methodology

Most of the existing batch learning algorithms converge frequently to local minima, particularly when training starts far away from a minimiser. Moreover, the error surface may have troughs, valleys, canyons, and a host of shapes. Thus, in presence of many plateaus, and valleys, training gets slow.

To alleviate this situation, this PhD thesis proposes a methodology inspired from the theory of linear and nonlinear iterative methods. For large systems containing thousands of equations, iterative methods often have decisive advantages over direct methods in terms of speed and demands on computer memory. Sometimes, if the accuracy requirements are not stringent, a modest number of iterations will suffice to produce an acceptable solution. Also, iterative methods are often very efficient for sparse systems problems. One of the most widely used class of nonlinear methods is the nonlinear Jacobi methods, which is used for the solution of a system of nonlinear equations[35]. In this PhD, Jacobi methods are applied in the optimisation context. The proposed algorithm of this class follows the Composite Jacobi procedure [79, 141].

Furthermore, this PhD equips these learning schemes with the global convergence property [75]. The issue of making these globally convergent algorithms, is treated with principles from unconstrained minimisation theory. These new globally convergent algorithms converge from a remote starting point. They do not seek global minimisers of the error function E, but ensure convergence to a local minimiser with certainty, despite the use of heuristics.

Finally, the problem of occasional convergence to local minimum is dealt within the context of global search methods. One of these widely used methods is the Simulated Annealing (SA). In the numerical optimisation framework, Simulated Annealing is a procedure that has the ability to move out of regions near local minima [24, 119]. In this PhD work, the methodology that is applied in the

construction of the new learning scheme is based on introducing additive noise in neural network learning by formulating the *perturbed* energy function. In the proposed method, noise is generated by a noise source that is characterised by the nonextensive entropic index q, inspired from Nonextensive Statistical Mechanics [125]. This scheme modifies the error surface during the training. Thus, as the energy landscape is modified during training, the search method is allowed to explore regions of the energy surface that were previously unavailable.

The last methodology that is applied in this thesis is the diverse neural ensembles. The ensemble approach has been justified both theoretically [42] and empirically [78]. It enables an increase in generalisation performance, by combining several individual neural networks trained on the same task. The proposed algorithms are applied to training feed-forward neural networks and diverse neural ensembles. The tested datasets are bioinformatics problems.

1.1.4 Structure of the PhD Thesis

The thesis contains seven chapters. Chapter 2 gives an overview of the basic neural network definitions, network architectures and well known training methods. In Chapter 3 a review of relevant previous research concerning first order training methods for feedforward neural networks is presented. This chapter introduces a new class of sign–based schemes that are based on the composite nonlinear Jacobi process. An algorithm of this class that applies the bisection method is further explored and detailed results of the experimental study are discussed. The new heuristic algorithm, the JRprop, is built on a theoretical basis.

In Chapter 4 a new class of first order globally convergent batch training algorithms, which employ local learning rates is proposed. The new learning rates adaptation strategy ensures the search direction is a descent one and the decrease of the batch error is guaranteed. Finally, convergence to a local minimiser of the

batch error function is obtained from any remote initial weights. In this chapter globally convergent modification of the Rprop [91] (an efficient modification of Backpropagation that is a sign–based learning scheme) and JRprop algorithms are introduced, named GRprop and GJRprop respectively. A theoretical justification for their development is provided, as well as reported comparative results in well studied benchmark problems are reported.

Chapter 5 deals with the application of the Statistical Mechanics in Neural Network training. A brief review of the basic concepts of the Statistical mechanics is presented and the relationship with the neural networks is highlighted. Emphasis is given on the Nonextensive Statistical Mechanics and how ideas based on this theory are applied in neural networks. A hybrid learning scheme that combines deterministic and stochastic search by employing a different adaptive stepsize for each weight, and a form of noise that is characterised by the nonextensive entropic index q that is regulated by a weight decay term, is proposed. Finally, an experimental study is conducted using biological data.

In Chapter 6, we investigate the use of the methods proposed in the previous chapters in training neural ensembles in bioinformatics problems. Effective training of neural ensembles is a subject of active research. It enables an increase in generalisation performance, by combining several individual neural networks trained on the same task. Furthermore, the generation of ensemble with diversity Feedforward Neural Networks (FNNs) provides significant improvement compared to other approaches in the literature for the two tested bionformatics problems. In particular the nonextensive hybrid algorithm proposed in this work generates different classifications, which is an important feature when creating efficient ensembles of neural nets.

Finally, Chapter 7 summarises the contents of the present PhD thesis and discusses its contribution. It also identifies areas for future work and makes suggestions on how to take this work forward.

1.1.5 Contribution

In Chapter 3, a gradient descent based heuristic scheme that uses one-step of the bisection method to locate an approximation of the subminimiser along each weight direction is suggested. The use of the bisection method helps the new learning scheme to converge more times than the other tested algorithms. This training scheme, the JRprop algorithm, helps to reduce the computational effort for high dimensional non convex functions when they are far from a minimiser, as is common in neural network training. Finally, by taking into account the evolution of the error in order to update the weights, the JRprop avoids convergence to local minima in some cases.

In Chapter 4 two globally convergent first order algorithms that possess the global convergence property to a local minimiser are proposed. The GRprop and GJRprop improve the learning speed as well as ensure subminimisation of the error function along each weight direction.

The Hybrid Learning Scheme (HLS) is another new stochastic learning algorithm for neural networks. It combines deterministic and stochastic search steps by employing a different adaptive stepsize for each network weight, and applies a form of noise that is characterised by the nonextensive entropic index q, regulated by a weight decay term. An experimental study verifies that there are indeed improvements in the convergence speed and classification success of this new learning algorithm.

The proposed training algorithms are developed and compared to well known existing algorithms. It was found that the new developed learning schemes achieve superior performance in terms of learning speed, and convergence success. Additionally, the aforementioned characteristics make these algorithms attractive to be used in wide range of biological and bioinformatics applications, such as the classification of the proteins into localisation sites and protein folding, as well as

the prediction of cancer, diabetes and thyroid.

Finally, this PhD attempts to produce classifiers that will generate different classifications. It also contributes to the process of building ensembles of neural nets. These ensembles are shown to improve the overall performance compared to the single classification methods in difficult biological problems.

Chapter 2

Supervised Training in an Optimization context

2.1 Formulation of the Training Problem

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. It is a network of interconnected elements inspired by studies of biological nervous systems. Artificial Neural Networks attempt to create machines that work in a similar way to the human brain by building them using components that behave like biological neurons. However, the operation of artificial neural networks and artificial neuron is far more simplified that the operation of the human brain; an abridged figure of the biological neuron is given in Figure 2.1. The brain consists of millions of these neurons, which may be specialised in some task or not. By modifying the synaptic connections new dendrites grow lengthening the axon. This behaviour of the brain inspired McCulloch and Pitts (1943) [70] to devise an artificial neuron called the perceptron, which is the basis of all neural network models. Neural networks' basic function is to produce an output pattern when presented with an input pattern. They

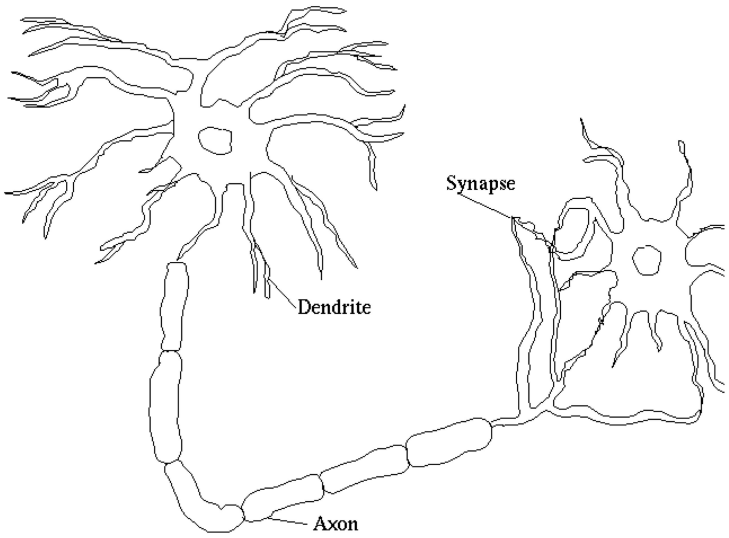


Figure 2.1: The biological neuron

have the specific ability to learn and generalise well.

In the 1950's, Rosenblatt (1958) [96] suggested a two-layer network, which was capable of learning certain classifications by adjusting connection weights. Although the perceptron was successful in classifying certain patterns, it had a number of limitations, which led to the decline of the field of neural networks. However, the perceptron had laid the foundations for later work in neural computing. In the early 1980's, researchers showed renewed interest in neural networks. Recent neural models include Boltzmann machines, Hopfield nets, competitive learning models, multilayer networks, and adaptive resonance theory models.

Nowadays, artificial neural networks are used in many systems. It has been widely known that neural networks can serve as a powerful tool for classification. Learning is essential to most of the neural network models. Learning can be supervised, when the network is provided with the correct answer for the output during training, or unsupervised, when no external teacher is present. Figure 2.2

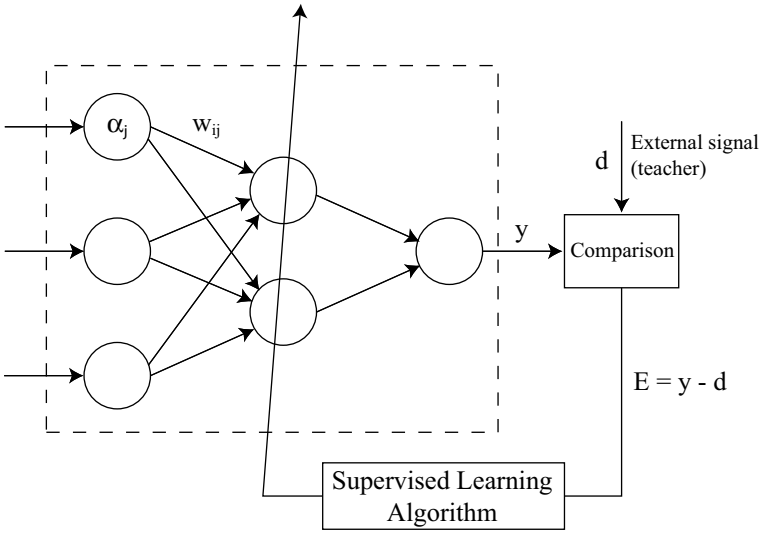


Figure 2.2: Supervised Learning

presents briefly the supervised learning procedure, where E is the *error function*, y is the actual output, d is the external signal, which is the desired output. The goal of the Neural Network learning is to iteratively adjust the weights, in order to globally minimise a measure of the difference between the actual output of the network and the desired output, as specified by a teacher, for all examples in a training set [43]. The vast majority of artificial neural network solutions have been trained with supervision. Therefore, emphasis in this work is given on the development of well-performing supervised learning schemes to apply on classification problems.

A widely used class of supervised neural network models, is the multilayer feedforward neural network (FNN). FNNs are nonlinear systems modelled on the general features of biological systems that exhibit emergent behavior [97]. Statisticians have studied the properties of this general class and have found that many results from the statistical theory of nonlinear models can be easily applied directly to feedforward nets [94]. Methods that are commonly used for fitting nonlinear models, such as various Levenberg-Marquardt and conjugate gradient algorithms, can also be used to train feedforward nets [14]. The FNNs can compute predicted

values simpler and faster than other neural network models. Finally, FNNs are better at learning moderately functions, which have discontinuities, than many other methods do with stronger smoothness assumptions [14].

Thus, at this point it is useful to describe briefly the operation of an FNN. FNN is usually based on the following equations:

$$net_j^l = \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1,l} y_i^{l-1}, \qquad y_j^l = f(net_j^l), \tag{2.1}$$

where l is the number of the layers in the neural network, net_j^l is for the j-th node in the l-th layer ($j = 1, \ldots, n_l$), the sum of its weighted inputs. The weights from the i-th node at the $(l-1)$ layer to the j-th node at the l-th layer are denoted by $w_{ij}^{l-1,l}$, y_j^l is the output of the j-th node that belongs to the l-th layer, and $f(net_j^l)$ is the j-th's node activation function.

If there is a fixed, finite set of input–output examples, the squared error over the training set, which contains P representative examples, is:

$$E(w) = \sum_{p=1}^{P} \sum_{j=1}^{n_L} \big(y_{j,p}^L - t_{j,p}\big)^2 = \sum_{p=1}^{P} \sum_{j=1}^{n_L} \big[\sigma^L\big(\, net_j^L + \theta_j^L\big) - t_{j,p}\big]^2. \tag{2.2}$$

This equation formulates the energy function, called *error function*, to be minimised, in which $t_{j,p}$ specifies the desired response at the j–th output node for the example p and $y_{j,p}^L$ is the output of the j–th node at layer L that depends on the weights of the network, and σ is a nonlinear activation function, such as the well known logistic function $\sigma(x) = (1 + e^{-x})^{-1}$. The weights in the network can be expressed using vector notation $w \in R^n$, as:

$$w = \Big(\ldots, w_{ij}^{l-1,l}, w_{i+1\ j}^{l-1,l}, \ldots, w_{N_{l-1}\ j,}^{l-1,l}, \theta_j^l, w_{i\ j+1}^{l-1,l}, w_{i+1\ j+1}^{l-1,l}, \ldots\Big)^\top, \tag{2.3}$$

where θ_j^l denotes the bias of the j–th node ($j = 1, \ldots, N_l$) at the l–th layer ($l = 2, \ldots, L$), and n denotes the total number of weights and biases in the network.

2.2 Optimisation Approaches

The crucial target of the supervised neural networks is to find the optimal solution. This solution is known as global minimum, which is the lowest possible error, and it is therefore the acceptable solution. It is well known in the neural networks field [43, 97] that the rapid computation of such a global minimum is a difficult task because the dimensionality of the weights space is high, and the corresponding nonconvex multimodal objective function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

First–order gradient based methods are the most widely used class of algorithms for supervised learning of neural networks [13]. First–order methods are linear approximators and more practical computationally [62]. Adaptive stepsize algorithms is a popular class of first order training algorithms that try to overcome the inherent difficulty of choosing the right stepsize for each problem [66]. The stepsize is used in the same way as the learning rate in the backpropagation. The step size determines how fast the algorithm moves down the gradient towards the optimal value of the parameters or weights. Too large a step size and the algorithm may diverges, too small and it will take a long time to reach the optimal solution. Determining the best step size is highly dependent upon the data in the specific problem and it is largely a trial and error effort to set the step size appropriately. Adaptive step size algorithms adjust the step size each weight update, essentially searching for the best step size. They work by controlling the magnitude of the changes in the weight states during learning in an attempt to avoid oscillations and, at the same time, maximising the length of the minimisation step [91].

A variety of approaches inspired by unconstrained optimisation theory has also been applied, in order to use second derivative related information to accelerate the learning process [13, 65, 71, 135]. Methods such as Conjugate Gradients [71], the Levenberg-Marquardt algorithm [31, 41], which is based on the

model-trust region approach, a popular alternative to the long-established line search methods [106, p.69], and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [37] are considered popular choices for training feedforward neural networks. Nevertheless, it is not certain that the extra computational cost these methods require leads to speed ups of the minimisation process for nonconvex functions when far from a minimiser [75]; this is usually the case with neural network training problems [13]. Although, the capacity of modern computers has been improved considerably the last few years, there are still problems that hamper the use of these powerful second order algorithms in some problems. For example, a large number of weights often makes the direct application of second order methods impractical. Lastly, these methods use approximations of the Hessian matrix which at some point during training may come close to singular, or badly scaled, and as a consequence they might produce inaccurate results.

An inherent difficulty with first–order and second–order learning schemes is convergence to local minima. While some local minima can provide acceptable solutions, they often result in poor network performance. This problem can be overcome through the use of global optimisation [21, 86, 87, 124]. The drawback of this class of the training algorithms is that they are very computationally expensive, particularly for large networks.

The following chapters discuss the class of first order training schemes that are based on the gradient descent. In the next chapter emphasis is given on the development of an efficient supervised learning algorithms, [7, 8]. Analytical theoretical and experimental study is provided and future directions are also given.

Chapter 3

Gradient Descent based Training Schemes

3.1 Introduction

Gradient descent is the most widely used class of algorithms for supervised learning of neural networks. The most popular training algorithm of this category is the batch Back-Propagation (BP) [97]. It is a first order method that minimises the error function by updating the weights using the steepest descent method [13]:

$$w(t+1) = w(t) - \eta \bigtriangledown E\left(w(t)\right) \tag{3.1}$$

where E is the batch error measure defined as the Sum of Squared differences Error function (SSE) over the entire training set, and t indicates iterations (time steps). The $\nabla(E)$ is the gradient vector, which is computed by applying the chain rule on the layers of the FNN[97]. The parameter η is a heuristic, called learning rate. The optimal value of η depends on the shape of the error function. The learning rate values help to avoid convergence to a saddle point or a maximum. In order to secure the convergence of the BP training algorithm and

avoid oscillations in a steep direction of the error surface a small learning rate is chosen ($0 < \eta < 1$).

One of the most common problems with the BP algorithm is that it leads to slow training, and often yields suboptimal solutions [38]. Convergence to a local minimum prevents the neural network from learning the entire training set and results in poor performance. Once the backpropagation based learning algorithms settle into one of these local minima, it is very difficult for the algorithms to continue their search to find the global minimum. This is a result of the insufficient number of the hidden nodes, as well as an improper initial weight vector. Furthermore, BP based algorithms tend to get stuck when the error surface is flat (the gradient is close to zero) or when the surface is rugged. Generally, the neural networks surfaces are very complex. These nonconvex error surfaces result to multiple minima. This peculiar neural network surface makes the choice of the learning rate for each weight direction very crucial. It is possible that the learning rate in one weight direction is not appropriate for the other weight directions or for all the portions of the error surface [50].

Many attempts have also been made to improve the performance of the BP algorithm by modifying the way that the learning rate is chosen. Some of them try to adapt dynamically the learning rate during the training [13, 134], or keeping constant learning rates and constant momentum [97]. On the other hand, the use of a constant learning rate introduces difficulties in obtaining convergence. Trying to train a neural network using a constant learning rate is usually a tedious process requiring much trial [59]. For backpropagation based algorithms with constant learning rate, there are also theoretical results that show convergence when the learning rate is constant and proportional to the inverse of the Lipschtz constant, which in practice is not easily available [64].

Other important efforts to improve the performance of the Back-propagation

algorithm include the application of BP using learning rate adaptation methods. This can be implemented by using a common adaptive learning rate for all the weights or an individual adaptive learning for each weight and apply the Goldstein/Armijo line search [66]. Adaptive training schemes try to overcome the problems of the BP algorithm by applying a weight specific learning rate as suggested by Jacobs [50]. A class of adaptive learning schemes is the local adaptation techniques, which use only weight-specific information (e.g the partial derivative) to adapt weight specific parameters. Well known examples of this category are the Delta-Bar-Delta and the SuperSAB [123]. Both of them are based on the idea of the sign-dependent learning rate adaptation. They perform a modification of the weight specific learning rate according to the observed behaviour of the error function. The adapted learning rate is eventually used to calculate the weight-step [91].

A different approach is that of the Quickprop as proposed by Fahlman [29]. Quickprop method belongs to the class of quasi-Newton methods. A modified globally convergent modification of the Quickprop has been proposed recently by Vrahatis et al achieving improved convergence speed and stability [139]. Other important gradient descent based training schemes that observe the sign of the gradients are the Silva and Almeida's method [113] and the Resilient Backpropagation (Rprop) [91, 92]. Finally, another class of adaptive learning rate algorithms is to adapt the learning rate by subminimisation in each weight direction [67].

This chapter focuses on first order algorithms which use adaptive learning rates as the simulations and the evaluation of these methods showed improvement learning speed and good convergence behavior [64, 66, 67]. Emphasis is given on the performance of the Adaptive gradient–based algorithms with individual step–sizes, and more specifically on one of the best algorithms of this class [91, 82, 83, 49, 7, 8], in terms of convergence speed, accuracy and robustness with respect to its learning parameters, the Resilient Backpropagation (Rprop) algorithm,

introduced by Riedmiller and Braun [91].

Relevant literature [82, 83, 64, 66, 67, 49, 7, 8] shows that Rprop–based learning schemes exhibit fast convergence in empirical evaluations, but usually require introducing or even fine tuning additional heuristics. Moreover, literature shows a lack of theoretical results underpinning the development of Rprop modifications, particularly in the case of hybrid schemes. This is not surprising as heuristics may not be able to guarantee convergence to a local minimiser of the error function when adaptive learning rates for each weight are used in the calculation weight updates [49, 66, 81, 91]. Nevertheless, no guarantee is provided that the network error will monotonically decrease at each iteration and that the weight sequence will converge to a minimiser of the sum-of-squared-differences error function E.

3.2 The Resilient Propagation Algorithm

A widely used example of Adaptive gradient–based algorithms with individual step–sizes, is the Rprop algorithm. The Resilient backpropagation approach is considered eminently suitable for applications where the gradient is numerically estimated or the error is noisy [49]; it is easy to implement it in hardware and it is not susceptible to numerical problems [80]. The basic principle of Rprop is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. The size of the weight change is exclusively determined by a weight-specific "update–value"

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t), & \text{if} \quad \dfrac{\partial E(t)}{\partial w_{ij}} > 0, \\ +\Delta_{ij}(t), & \text{if} \quad \dfrac{\partial E(t)}{\partial w_{ij}} < 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $\partial E(t)/\partial w_{ij}$ denotes the true gradient, which is summed over all patterns of the training set. The second step of Rprop learning is to determine the new update-values.

$$\Delta_{ij}(t) = \begin{cases} \eta^+ \cdot \Delta_{ij}(t-1), & \text{if } \dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} > 0, \\ \eta^- \cdot \Delta_{ij}(t-1), & \text{if } \dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} < 0, \\ \Delta_{ij}(t-1), & \text{otherwise,} \end{cases}$$

where $0 < \eta^- < 1 < \eta^+$.

Thus, every time the partial derivative of the corresponding weight $w_{ij}(t)$ changes its sign, which indicates that the last update was too big and the algorithm has jumped over the local minimum, the update-value $\Delta_{ij}(t)$ is decreased by the factor η^-. If the derivative retains its sign, the update value is slightly increased in order to accelerate convergence in shallow regions. Additionally, in case of a change in sign, there should be no adaptation in the succeeding learning step. In practice this can be achieved by setting $\partial E(t)/\partial w_{ij} = 0$ in the adaptation rule. This strategy helps to speed up convergence when the derivative is negative but may be inefficient when the two derivatives are positive, as in this case the weight updates may lead the weight trajectory far away from the minimum or in regions with higher error function values. In an attempt to alleviate these

situations Rprop employs a heuristic parameter $\Delta_{\max}$, which constrains the size of the update step.

In fact the Rprop algorithm requires setting the following parameters: (i) the increase factor is set to $\eta^+ = 1.2$; (ii) the decrease factor is set to $\eta^- = 0.5$; (iii) the initial update-value is set to $\Delta_0 = 0.1$; (iv) the maximum weight step, which is used in order to prevent the weights from becoming too large, is $\Delta_{\max} = 50$, and the minimum step size is constantly fixed to $\Delta_{\min} = 10^{-6}$ [91, 92].

A high level description of the weight update procedure, which shows the kernel of the Rprop adaptation and learning process, is described below. The minimum/maximum operator is supposed to deliver the minimum/maximum of two numbers; the sign operator returns $+1$, when the argument is positive; -1, when the argument is negative and 0 otherwise, as suggested by Riedmiller and Braun [91, 92].

Rprop weight update procedure:

repeat

compute the gradient vector $\nabla E(t)$

for all weights and biases

if $\dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} > 0$ **then**

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

$\Delta w_{ij}(t) = -\text{sign}\dfrac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

$\dfrac{\partial E(t-1)}{\partial w_{ij}} = \dfrac{\partial E(t)}{\partial w_{ij}}$

else if $\dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} < 0$ **then**

$$\Delta_{ij}(t) = \max\{\Delta_{ij}(t-1) \cdot \eta^{-},\ \Delta_{\min}\}$$
$$\frac{\partial E(t-1)}{\partial w_{ij}} = 0$$

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

$$\Delta w_{ij}(t) = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$$
$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$
$$\frac{\partial E(t-1)}{\partial w_{ij}} = \frac{\partial E(t)}{\partial w_{ij}}$$

end if

until termination criterion is met

where the termination criterion will be a predefined number of epochs or specific error goal.

The effectiveness of Rprop in practical applications has motivated the development of several variants aiming at improving the convergence behavior and effectiveness of the original method. These variants can be roughly categorised into: (i) hybrid learning schemes that equip Rprop with second derivative related information, such as the *QRprop* algorithm, which approximates the second derivative by one–dimensional secant steps, and the *Diagonal Estimation Rprop–DERprop* [81], which directly computes the diagonal elements of the Hessian matrix; (ii) approaches inspired by global optimisation theory, such as the *Simulated Annealing Rprop–SARprop* and the *Restart mode Simulated Annealing Rprop–ReSARprop* [124]. Recently, the *Improved Rprop–IRprop* algorithm [49] has shown improved convergence speed when compared against existing Rprop variants, as well as the conjugate gradients and the BFGS training methods.

3.2.1 Simulated Annealing Rprop–SARprop

SARprop introduced by Treadgold and Gedeon [124] is a method that tries to solve the problem of poor local minima. It attempts to address this problem by combining the method of Simulated Annealing (SA) [52, 10] and the Rprop algorithm. SA involves the addition of noise to the parameters undergoing optimization. In SARprop, noise is added to the standard Rprop weight update value when both the error gradient changes sign in successive epochs and the magnitude of the update value is less than a value proportional to the SA term. The reason for adding noise to the update value only when both the error gradient changes sign and the update value is below a given setting, is to minimize the disturbance to the normal adaptation of the update value. This means that the update value is only modified by noise when it has a relatively small value. This can allow the weight to jump out of local minima while minimizing the disturbance to the adaptation process.

The only parameter requiring setting prior to training is the temperature T, a part of the SA term which affects the speed by which the noise is reduced. The optimal setting of this parameter is problem dependent, although good values were found in the range of 0.01 to 0.05 [124]. In general it was found the more complex the problem, the lower the temperature value required and hence the slower the annealing process.

Finally, a restart mode Simulated Annealing Rprop–ReSARprop by Treadgold and Gedeon (1998) [124], is a modification of SARprop. It uses SARprop in a restart mode. This is done by restarting training whenever SARprop converges. ReSARprop tries to solve the problem of selecting a good value for the temperature parameter. The temperature can be initially set to give fast annealing. If a good solution has not been reached this temperature can be reset to allow for slower annealing when the network is restarted. The removal of the temperature parameter in ReSARprop results in ReSARprop being parameter free.

3.2.2 Hybrid Resilient Propagation Algorithms

The next two Hybrid Algorithms QRprop and DERprop , as suggested by Pfister and Rojas (1994), include second order information into an adaptive step method [82, 83]. They combine the fast convergence speed of Rprop with the good local properties of second order methods. These algorithms are hybrid methods using a local strategy i.e. the adaptive inclusion of second order information takes place for each weight independently. QRprop and DERprop are batch algorithms to which no additional parameters have been introduced. For very large and redundant data sets, there is a waste of computational time in training the neural networks properly [83].

QRprop: General Description

This algorithm is obtained by adaptively switching between Rprop and local one dimensional secant steps. The idea of QRprop is the following one: the Rprop is used if two subsequent error function gradient components $\partial E(t)/\partial w_{ij}$ have the same sign which guarantees a fast approach to minimum regions. If the sign of the gradient changes, we know that we have over jumped a local minimum in this specific weight direction. Then neither weights nor learning rates are changed, which is done in the next step using a constrained secant approximation. It is important to determine the new update-values $\Delta_{ij}(t)$ that are used in QRprop algorithm:

$$\Delta_{ij}(t) = \begin{cases} \eta^+ \cdot \Delta_{ij}(t), & \text{if } \dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} > 0, \\ \Delta_{ij}(t-1), & \text{if } \dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} < 0, \\ \mathrm{quad}_{ij} \cdot \Delta_{ij}(t-1), & \text{otherwise,} \end{cases}$$

where $0 < \eta^- < 1 < \eta^+$, $\mathrm{quad}_{ij} = \max\{\eta^-, \min \dfrac{1}{\eta^+}, \mathrm{q}_{ij}\}$. The factor q_{ij} is the

quadratic approximation, which is defined as follows:
$q_{ij} = |\frac{\partial E(t-1)}{\partial w_{ij}} / (\frac{\partial E(t)}{\partial w_{ij}} - \frac{\partial E(t-2)}{\partial w_{ij}})|$

DERprop: General Description

DERprop is similar to QRprop. The main difference is that second order information is not approximated by secant steps but directly computed by evaluating the diagonal terms of $\nabla^2 E(w^t)$, hence the name of the algorithm which stands for Diagonal Estimation Rprop (DERprop). Using the diagonal terms of the Hessian matrix has the advantage that quadratic local minimum regions are not just recognised after they have been jumped over but eventually earlier. DERprop is divided into two 'subalgorithms'. In the first one ($DERprop_A$), second order information is only used to minimise local minima as in QRprop and in the second one ($DERProp_B$), second order information is only used to avoid to overshoot local minima. For $DERProp_B$ step, it should not be larger than an Rprop step to avoid too large steps and oscillations and it should not be smaller than the previous step so that, in case of a suboptimal step, the learning rate is at least not worsened. For $DERprop_A$ the same constraints for the second order steps as for QRprop are applied.

In the $DERProp_B$ algorithm the learning rate changes following the procedure:

$$\Delta_{ij}(t) = \begin{cases} \max\{\min(quad_{ij}, \Delta_{\max}), \Delta_{\min}\}, & \text{if } \frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} > 0, \\ \max\{\Delta_{ij}(t-1) \cdot \eta^-,\ \Delta_{\min}\}, & \text{if } \frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} < 0, \\ \Delta_{ij}(t-1), & \text{otherwise,} \end{cases}$$

where $0 < \eta^- < 1 < \eta^+$, $\text{quad}_{ij} = \max\{\eta^-, \min \frac{1}{\eta^+}, \text{q}_{ij}\}$. The factor q_{ij} is the quadratic approximation, which is defined as follows:

$$q_{ij} = |\frac{\partial E(t-1)}{\partial w_{ij}} / (\frac{\partial E(t)}{\partial w_{ij}} - \frac{\partial E(t-2)}{\partial w_{ij}})|$$

3.2.3 The IRprop algorithm: Improved Rprop

The IRprop algorithm [49] is a recently proposed modification of the resilient propagation. The decision in Rprop to undo a step is somewhat arbitrary. Thus, the idea of IRprop is to make the step reversal depend on the evolution of the error. It suggests that weight updates that have caused changes to the signs of the corresponding partial derivatives are reverted, but only in case of an error increase. It is a backtracking strategy to Rprop update for some or all of the weights in order to decide for each weight individually whether or not to revert a step.

IRprop weight update procedure:

repeat

compute the gradient vector $\nabla E(t)$

for all weights and biases

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

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

$\Delta w_{ij}(t) = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

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

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

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

if $E(t) < E(t-1)$ **then**

$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij}(t-1)$

end if

$$\frac{\partial E(t-1)}{\partial w_{ij}} = 0$$

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

$$\Delta w_{ij}(t) = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

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

end if

until termination criterion is met

where sign defines the well known triple valued sign function and the termination criterion will be a predefined number of epochs or specific error goal.

In this algorithm, the previous error $E(t-1)$ has to be stored, in order to check the evaluation of the error and decide how the weight update will be done. Compared to the original Resilient Propagation algorithm (Rprop) only one additional variable is estimated (i.e the previous error $E(t-1)$). The IRprop performs weight-backtracking in the few cases where the overall error increases, and secondly it always sets the derivative to zero when the sign of the product of the derivatives is negative. Finally, all the parameters are set at the same values as suggested by Riedmiller and Braun for the Rprop algorithm [91, 92].

3.3 An Approach Based on the Nonlinear Jacobi Methods

In this section, first–order algorithms with an adaptive learning rate for each weight are analysed as composite *nonlinear Jacobi* methods applied to the gradient of the error function. The class of nonlinear Jacobi methods is widely used for the solution of a system of nonlinear equations:

$$F(x_1, x_2, \ldots, x_n) = \Theta^n \equiv (0, 0, \ldots, 0), \tag{3.2}$$

where $F = (f_1, f_2, \ldots, f_n) : \mathcal{D} \subset R^n \to R^n$.

Along this line, in a function minimisation problem all local minima $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ of a continuous differentiable function f should satisfy the necessary conditions:

$$\nabla f(x^*) = \Theta^n. \tag{3.3}$$

Eq. (3.3) represents a set of n nonlinear equations, which must be solved to obtain x^*. Therefore, one approach to the minimisation of the function f is to seek the solutions of the set of Eq. (3.3) by including a provision to ensure that the solution found does, indeed, correspond to a local minimiser. Solving Eq. (3.3) is equivalent to solving the following system of equations:

$$\begin{aligned} \partial_1 f(x_1, x_2, \ldots, x_n) &= 0, \\ \partial_2 f(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ \partial_n f(x_1, x_2, \ldots, x_n) &= 0, \end{aligned} \tag{3.4}$$

where $\partial_i f(x_1, \ldots, x_i, \ldots, x_n)$ denotes the partial derivative of f with respect to the ith coordinate.

3.3.1 The Composite nonlinear Jacobi

The nonlinear Jacobi process applies a parallel update of the variables [79]. Starting from an arbitrary initial vector $x^0 \in \mathcal{D}$, one can subminimise at the kth iteration the function:

$$f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_n^k), \tag{3.5}$$

along the ith direction and obtain the corresponding subminimiser $\hat{x}_i$. Obviously for the subminimiser $\hat{x}_i$

$$\partial_i f(x_1^k, \ldots, x_{i-1}^k, \hat{x}_i, x_{i+1}^k, \ldots, x_n^k) = 0. \tag{3.6}$$

This is a one–dimensional subminimisation because all the components of the vector x^k, except from the ith component, are kept constant. Then the ith component is updated according to the equation:

$$x_i^{k+1} = x_i^k + \tau_k(\hat{x}_i - x_i^k), \tag{3.7}$$

for some relaxation factor τ_k. The objective function in (3.5) is subminimised in parallel for all i. When exact one–dimensional submimisation is applied and $\tau_k = 1$ for all k the following result is available for strictly convex functions.

Theorem 1 [19]: Suppose that the objective function $f : \mathcal{D} \subset R^n \to R$ is twice continuously differentiable on a convex domain $\mathcal{D}$ and that f is a strictly convex function. Assume that there exists $\gamma \in R$ such that $\mathcal{S}_\gamma = \{x \in \mathcal{D} \colon f(x) \leqslant \gamma\}$ is nonempty and compact and that $\partial_{ii}^2 f(y) \neq 0$ for $i = 1, 2, \ldots, n$ and $y \in \mathcal{S}_\gamma$, unless y is the point at which f attains its minimum, where $\partial_{ij}^2 f(y)$ denotes the h_{ij} element of the Hessian matrix of f at y, $H = [h_{ij}]$. Suppose further, that from any point $x^0 = (x_1^0, x_2^0, \ldots, x_n^0) \in \mathcal{S}_\gamma$ a sequence $\{x^k\}_{k=0}^\infty$ is generated:

$$\begin{aligned}
&x_j^{k+1} = x_j^k, \quad j \neq i_k \quad \text{and} \\
&x_{i_k}^{k+1} \quad \text{is the solution of} \\
&\partial_{i_k} f(x_1^k, \ldots, x_{i_k-1}^k, x_{i_k}, x_{i_k+1}^k, \ldots, x_n^k) = 0,
\end{aligned}$$

where i_k is any one of the integers $1, 2, \ldots, n$. Such a sequence $\{x^k\}_{k=0}^{\infty}$ is uniquely defined and converges to x^*, the unique global minimiser of f, provided that in the above iterative process every coordinate direction i is chosen an infinite number of times.

Various composite nonlinear Jacobi training algorithms can be obtained depending on the one–dimensional minimisation method applied [141]. In case an inexact one–dimensional subminimisation is applied, the number of the iterations or steps of the subminimisation method is related to the requested accuracy in obtaining the subminimiser approximations. Thus, significant computational effort is needed in order to find accurate approximations of the subminimiser along each variable direction at each iteration.

Moreover, this computational effort is increased for problems with a high number of variables, as for example when training neural networks with several hundred weights. Taking into account that training neural networks involves minimising non convex functions, it is a computationally expensive process without guaranteeing the convergence to a global minimiser. Similar situations can also occur in the iterative solution of nonlinear equations [79].

3.3.2 The Jacobi–bisection method

In this section, we synthesize a composite Jacobi method that is inspired by the Rprop algorithm. The method, named JRprop, combines "individual" information about the error surface, i.e. the sign of the partial derivative of the error function with respect to each one of the weights, with more "global" information, i.e. the magnitude of the network's learning error at each epoch t, in order to decide for each weight individually whether or not to revert/reduce a step.

Following the nonlinear Jacobi prescription, one–dimensional subminimisation is

applied along each weight direction. Let us assume that along a weight's direction an interval is known which brackets a local minimum $\hat{w}_{ij}$. When the gradient of the error function is available at the endpoints of the interval of uncertainty along this weight direction, it is necessary to evaluate function information at an interior point in order to reduce this interval. This is because it is possible to decide if between two successive epochs (t) and $(t-1)$ the corresponding interval brackets a local minimum simply by looking the function values $E(t-1)$, $E(t)$ and gradient values $\partial E(t-1)/\partial w_{ij}$, $\partial E(t)/\partial w_{ij}$ at the endpoints of the considered interval (see [101] for a general discussion on the problem).

The conditions that have to be satisfied, [101], are:

$$\begin{aligned}
&\frac{\partial E(V1)}{\partial w_{ij}} < 0 \ \text{ and } \ \frac{\partial E(V2)}{\partial w_{ij}} \ > \ 0, \\
&\frac{\partial E(V1)}{\partial w_{ij}} < 0 \ \text{ and } \ E(V1) \ < \ E(V2), \\
&\frac{\partial E(V1)}{\partial w_{ij}} > 0 \ \text{ and } \ \frac{\partial E(V2)}{\partial w_{ij}} \ > \ 0 \ \text{ and } \ E(V1) \ > \ E(V2),
\end{aligned} \tag{3.8}$$

where $V1$ and $V2$ determine the sets of weights for which the coordinate that corresponds to the weight w_{ij} is replaced by $a_i = \min\{w_{ij}^{(t-1)},\ w_{ij}^{(t)}\}$, and $b_i = \max\{w_{ij}^{(t-1)},\ w_{ij}^{(t)}\}$ correspondingly. Notice that, at this instance, between two successive epochs $(t-1)$ and (t) all the other coordinates remain the same (this is because we follow the nonlinear Jacobi prescription). The above three conditions lead to the conclusion that the interval $[a_i, b_i]$ includes a local subminimiser along the direction of weight w_{ij}. A robust method of interval reduction, such as the bisection could now be used. By computing the midpoint $m_i = \frac{1}{2}(a_i + b_i)$ of the interval $[a_i, b_i]$ we take as the next interval whichever of $[a_i, m_i]$ and $[m_i, b_i]$ that still brackets a minimiser according to the criteria mentioned above.

For the case of the first condition of (3.8) we will consider here the bisection method, which has been modified to the following version described in [137, 138]:

$$w_{ij}^{p+1} = w_{ij}^{p} + h_i \operatorname{sign} \partial_i E(w^p) \, / 2^{p+1}, \tag{3.9}$$

where $p = 0, 1, \ldots$ is the number of subminimisation steps and $w_i^0 = a_i$; $h_i = \operatorname{sign} \partial_i E(w^0)\,(b_i - a_i)$; w^0 determines the set of weights at the $(t-1)$ epoch while w^p is obtained by replacing the coordinate of w^0 that corresponds to the weight w_{ij} by w_i^p. Of course, the iterations (3.9) converge to $\hat{w}_i \in (a_i, b_i)$ if for some $w_i^p, \quad p = 1, 2, \ldots,$ the first one of the conditions (3.8) holds. In this case, the bisection method always converges with certainty within the given interval $[a_i, b_i]$.

Also, the number of steps of the bisection method that are required for the attainment of an approximate minimiser $\hat{w}_i$ of (3.7) within the interval $[a_i, b_i]$ to a predetermined accuracy ε is known beforehand and is given by

$$\nu = \left\lceil \log_2[(b_i - a_i)\varepsilon^{-1}] \right\rceil . \tag{3.10}$$

Moreover, it has a great advantage since it is worst–case optimal, i.e. it possesses asymptotically the best possible rate of convergence in the worst–case [110, 111]. This means that it is guaranteed to converge within the predefined number of iterations, a property no other method has. Therefore, using the Relation (3.10) it is easy to pre-determine the number of iterations needed. Finally, it requires the algebraic signs of the values of the gradient to be computed.

Theoretical approach of the JRprop

Next, a theoretical result is given that ensures that the composite Jacobi method that uses a multi–step bisection method for reducing the intervals of uncertainty converges to a solution. In particular, this result shows that there is a neighborhood of a minimiser of the objective function for which convergence to the local minimiser can be guaranteed. Notice that this result does not require exact one–dimensional subminimisation, but only an approximation of the local minimiser.

Corollary 1. Let $E\colon \mathcal{D} \subset R^n \to R$ be twice continuously differentiable in an open neighborhood $\mathcal{S}_0 \subset \mathcal{D}$ of a point $w^* \in \mathcal{D}$ for which $\nabla E(w^*) = \Theta^n$ and the

Hessian, $H(w^*)$ is positive definite with the property A^π. Then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ (where $\mathcal{S}(w^*, r)$ denotes the open ball centered at w^* with radius r), such that any sequence $\{w^k\}_{k=0}^{\infty}$ generated by the nonlinear Jacobi process converges to w^* which minimises E.

The proof of the corollary originates from the application of Theorem 1, derived in [141], on the error function. As mentioned above, Corollary 1 guarantees only local convergence, and naturally imposes some conditions on the Hessian matrix. Clearly, the necessary and sufficient conditions for the point w^* to be a local minimiser of the function E are satisfied by the hypothesis $\nabla E(w^*) = \Theta^n$ and the assumption of positive definitiveness of the Hessian at w^*. Finding such a point is equivalent to minimising the nonlinear function (3.5) by applying the nonlinear Jacobi process and employing the multi-step bisection of Relation (3.9).

Proof. Consider the decomposition of $H(w^*)$ into its diagonal, strictly lower–triangular and strictly upper–triangular parts:

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*). \tag{3.11}$$

Since, $H(w^*)$ is symmetric and positive definite, then $D(w^*)$ is positive definite [132]. Moreover, since $H(w^*)$ has the property A^π, the eigenvalues of

$$\Phi(w^*) = D(w^*)^{-1}\left[L(w^*) + L^\top(w^*)\right], \tag{3.12}$$

are real and $\rho(\Phi(w^*)) < 1$ [11] (where $\rho(A)$ indicates the spectral radius of the matrix A); then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any initial vector $w^0 \in \mathcal{S}$, there is a sequence $\{w^k\}_{k=0}^{\infty} \subset \mathcal{S}$ which satisfies the nonlinear Jacobi process such that $\lim_{k\to\infty} w^k = w^*$ [79].

Remark 1. The Property A: Young [147] has discovered a class of matrices described as having property A that can be partitioned into block tridiagonal form, possibly after a suitable permutation [11]. An algorithmic procedure for transforming a symmetric matrix to a tridiagonal form is presented in [112].

Implementation of the JRprop

Based on the above theoretical discussion we will below consider obtaining $\hat{w}_i$ by minimising the function (3.5) with one-step of a subminimisation method. In particular, an Rprop-based heuristic scheme is proposed that uses one-step of the bisection method to locate an approximation of the subminimiser $\hat{w}_{ij}$ along each weight direction. The one step of the subminimisation method helps to reduce the computational effort for high dimensional non convex functions when far from a minimiser, as it usually happens in neural network training [141].

Next, a high level description is presented of the proposed algorithm that implements a heuristic version of the JRprop. It is based on the idea of *function comparison methods*, [101], taking into account $E(t-1) < E(t)$, and exploits the signs of the gradient values. The parameter q is a reduction factor that is used to update the midpoint of the considered interval; choice of q has an influence on the number of error function evaluations required to obtain an acceptable weight vector [66].

JRprop algorithm:

repeat

compute the gradient vector $\nabla E(t)$

if $E(t) \leqslant E(t-1)$ **then**

for all weights and biases

if $\dfrac{\partial E(t-1)}{\partial w_{ij}} \cdot \dfrac{\partial E(t)}{\partial w_{ij}} > 0$ **then**

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

$\Delta w_{ij}(t) = -\text{sign}\dfrac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

$\dfrac{\partial E(t-1)}{\partial w_{ij}} = \dfrac{\partial E(t)}{\partial w_{ij}}$

$\Delta w_{ij}(t-1) = \Delta w_{ij}(t)$

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

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

$\frac{\partial E(t-1)}{\partial w_{ij}} = 0$

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

$\Delta w_{ij}(t) = -\text{sign}\frac{\partial E(t)}{\partial w_{ij}} \cdot \Delta_{ij}(t)$

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

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

$\Delta w_{ij}(t-1) = \Delta w_{ij}(t)$

end if

$q = 1$

end if

if $E(t) > E(t-1)$ **then**

$w_{ij}(t+1) = w_{ij}(t) + \frac{1}{m^q}\Delta w_{ij}(t-1)$

$q = q + 1$

end if

until termination criterion is met

The particular implementation of the JRprop does not take a special consideration for the third condition of (3.8). This condition requires special treatment as it may lead the algorithm to converge to an undesired extremum. As shown in the pseudocode description, in JRprop this is handled by the standard Rprop. Moreover when the bisection is applied, we introduce the parameter m, which takes values $m > 1$. In our experiments we fixed m at the value of five ($m = 5$). Finally, for all the learning parameters are set at the same values as suggested by Riedmiller and Braun for the Rprop algorithm [91, 92].

Various termination criteria can be used in the weight update procedure, e.g.

meeting a predefined training error goal $E(t)$, reaching the maximum number of epochs t, or having the norm of the gradient vector close to zero.

In case the initial weights are far from the neighbourhood of a local minimiser, then it is possible to equip the algorithm with a strategy for adapting the direction of search to a descent one. In this way, a decrease of the function value can be ensured at each iteration and convergence to a local minimiser of the objective function from remote initial points can be achieved as shown in [68]. This case will be considered in another chapter.

3.4 Experimental study using biological datasets

IRprop is a recently proposed algorithm that claims improved performance over the Rprop algorithm and achieves similar results compared to well known second order techniques [49]. Comparative study shows that Rprop outperforms other well known training schemes, such as Silva-Almeida, Quickprop and the Super-SAB [93]. Also, this experimental survey contains two well known second order training algorithms, namely the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [37], and the Scaled conjugate gradient backpropagation (SCG) proposed by Moller [71]. Both are tested in four benchmark classification problems.

The BFGS and SCG algorithms have been applied in an attempt to use second derivative related information to accelerate the learning process as suggested in [13, 71]. The BFGS quasi-Newton method is an alternative to the direct application of the second order Newton's method, which is considered of limited applicability in neural network training, particularly when the networks have a large number of weights [71]. In the experimental study the standard BFGS method is applied following the guidelines of Matlab 6.5 neural network toolbox. The tested BFGS algorithm trains any network as long as its weight, net input, and transfer functions have derivative functions. For a more detailed discussion

of the BFGS quasi-Newton method, which is applied in the experiments see [37, p.119].

Another reliable approach that performs well is the SCG method. This method has superlinear convergence rate [71]. It is a variation of the well known conjugate gradient method, which is based on conjugate directions. This algorithm does not perform a line search at each iteration. See [71, p.525-533] for a more detailed discussion of the scaled conjugate gradient algorithm.

Below, results from 1000 independent trials are reported for six neural network classification problems, namely cancer1, diabetes1, thyroid1, genes2, E.coli, Yeast [88]. These 1000 random weight initialisations are the same for all the learning algorithms. In all cases, except genes, we have used Feed-forward neural networks (FNN) with sigmoid hidden and output nodes. A FNN with tansig hidden nodes was used in the genes2 problem. The notation I-H-O is adopted to denote a network architecture with I inputs, H hidden layer nodes and O outputs nodes. Finally the symbol $\pm$, which is presented in the comparative tables, denotes the corresponding value of standard deviation.

The data sets for the cancer1, diabetes1, thyroid1, and genes2 problems were used as supplied on the PROBEN1 website. PROBEN1 provides explicit instructions for creating training and testing sets and choosing network architectures for many problems [88]. These well-studied problems from the UCI Repository of Machine Learning Databases [72] were used in an attempt to reduce as much as possible biases introduced by the size of the weights space. We decided not to enhance the algorithms tested with add-on techniques to improve the classification success in the testing phase (i.e. generalisation ability of the trained neural network), as this would require introducing, and fine tuning or optimising additional heuristics depending on the learning task. The data sets for the E.coli and Yeast problems were used as supplied on the UCI repository and the sets for training and testing were created following guidelines published by Horton [47].

There are no standard neural architectures for these two problems so we have conducted our own preliminary experiments as will be described in the problem subsections.

In all experiments the parameters have been set as follows: $\eta^+ = 1.2$; $\eta^- = 0.5$; $\Delta_{ij}^0 = \eta^0 = 0.1$; $\Delta_{\max} = 50$ [91]. We remark that the selection of m, q is not critical for successful learning, however it influences the number of error function evaluations to meet the error target. In all experiments a value of $m = 5$ is applied.

Next, the performance of the new method is evaluated in these benchmarks and compared to the original Rprop [91] and the IRprop [49]. The implementation of the algorithms has been done in Matlab 6.5.

Learning speed or equivalent training speed or convergence speed is represented in our tables by the "Epochs" and "Time". It is a critical factor when deciding which algorithm to use. Classification success in testing (representing in tables by "Generalisation") is another crucial factor. Experimental results are presented below for both factors. The results for these pattern classification problems are summarised in the following tables (a "+" indicates statistical significance of the results of JRprop over the other methods). The Wilcoxon Rank Test [114], which is given in *Appendix A*, justifies the statistical significance of the experimental results.

The Cancer1 problem

This is a breast cancer diagnosis problem, which is described analytically in *Appendix* A. A feed-forward neural network is used with 9-4-2-2 nodes as suggested in the PROBEN1 benchmark collection and in [49]. The error goal in training was $E < 0.02$ to harmonise with the training errors obtained in [49]. The other termination criterion was set at 2000 epochs. The parameter m of the JRprop is

set at a value of five ($m = 5$).

The results for this pattern classification problem are summarised in Table 3.1. The average time ("Time", measured in secs) is represented, the number of epochs that are needed to meet the error target ("Epochs"), the classification success with the testing set ("Generalisation", measured by the percentage of testing patterns that were classified correctly), and the convergence success in the training phase ("Convergence", measured by the percentage of simulation runs that converged to the error goal) for an algorithm. The new algorithm performs significantly better than the other two methods. The differences between IRprop and Rprop do not seem to be important. For each comparison we apply the Wilcoxon rank test to calculate the significance of the results of the JRprop with respect to the other methods. Statistically significant cases are marked with (+).

The percentage of improvement that the new algorithm achieved over Rprop and IRprop in terms of learning speed (Time) is 12.1% and 15.2% respectively. The results of JRprop show slightly less classification success compared to Rprop and IRprop and significant improvement over the BFGS and SCG algorithms.

Table 3.2 presents the number of times each algorithm outperforms the other methods in terms of training speed and generalisation within 1000 independent runs. It yields that the new learning scheme is frequently faster and achieves better generalisation than the other two members of the Rprop family.

Table 3.1: Comparison of algorithms performance in the Cancer problem for the runs which converged

Cancer				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	280 (+)	1.85 ± 1.30 (+)	97.0 (−)	94 (+)
IRprop	285 (+)	1.90 ± 1.35 (+)	97.0 (−)	94 (+)
BFGS	632 (+)	10.7 ± 6.5 (+)	94.00 (+)	72 (+)
SCG	258 (−)	2.20 ± 1.5 (+)	95.99 (+)	88 (+)
JRprop	255	1.65 ± 0.70	96.9	97

Table 3.2 shows that the second order algorithms BFGS and SCG meet the error goal 720 and 880 times out of the 1000 runs respectively. The SCG algorithm converges faster than the Rprop and IRprop in epochs but needs more time as it requires on average more time in each iteration for the computations. Moreover, the BFGS algorithm has the worst performance, as many times during training the Hessian matrix is close to singular or is badly scaled. Finally, Table 3.3 gives in more details the performance of the BFGS and SCG compared to the new proposed scheme JRprop.

It yields that the new learning scheme is frequently faster and achieves better generalisation more times than the other tested algorithms. The Jacobi Rprop training scheme converges faster more than 500 times compared to Rprop, IRprop, SCG and more than 900 times than BFGS. Although the generalisation of the JRprop is slightly less than the other first order algorithms, it achieves better generalisation in more runs.

Table 3.2: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Cancer problem with respect to training speed and generalisation.

Cancer	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	IRprop	JRprop	Time	Rprop	IRprop	JRprop
Rprop	–	198	366	2.2 ± 2.51	–	10	231
IRprop	504	–	371	2.1 ± 2.50	7	–	228
BFGS	2	2	0	15.7± 12.5	2	3	2
SCG	450	454	370	2.50 ± 2.6	57	55	22
JRprop	506	511	–	1.5 ± 0.90	241	241	–

The Diabetes1 problem

The aim of this real-world classification task is to decide when a Pima Indian individual is diabetes positive or not. More details are given in *Appendix* A. The

Table 3.3: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Cancer problem with respect to training speed and generalisation.

Cancer	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	BFGS	SCG	JRprop	Time	BFGS	SCG	JRprop
BFGS	–	20	0	15.7± 12.5	–	90	12
SCG	860	–	372	2.50 ± 2.6	107	–	42
JRprop	970	598	–	1.6 ± 0.85	541	470	–

PROBEN1 collection proposes several architectures for this problem, including one with 8-2-2-2 nodes. This architecture was decided to be used, as it was also suggested by others [49]. The error goal in this case was set at 0.14 to conform to the training error obtained in [49]. The other termination criterion is the reach of 2000 epochs. The experimental results of the JRprop for this problem are taken by setting $m = 5$.

Table 3.4 summarises the performance of the tested algorithms. The algorithm exhibits increased training speed (Time, in secs) and generalisation performance. In the diabetes classification problem both Rprop and IRprop behave similarly. Table 3.4 gives the average convergence speed and success of the tested algorithms. The table also includes the results of the Wilcoxon Rank test. It is worth noting that the standard deviation value of the JRprop is significantly less than the corresponding Rprop and IRprop values, which means JRprop performance is closer to the average value.

Table 3.5 gives an analytic view of the comparative results in the 1000 trials. There is an increase in the training time of the Rprop and IRprop as both converge less times than the JRprop algorithms. The convergence success of the JRprop is 94%. It is also important to highlight the number of runs that the JRprop is faster than the other tested algorithms (725 and 644 times better learning speed than IRprop and Rprop). Finally, Table 3.6 shows the performance of

Table 3.4: Comparison of algorithms performance in the Diabetes problem for the runs which converged

Diabetes				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	455 (+)	2.40 ± 2.1 (+)	75.8 (+)	86 (+)
IRprop	460 (+)	2.50 ± 2.2 (+)	75.8 (+)	85 (+)
BFGS	462 (+)	5.2± 4.2 (+)	74.68 (+)	64 (+)
SCG	475 (+)	4.3± 3.2 (+)	74.64 (+)	76 (+)
JRprop	410	2.15 ± 1.5	76.1	94

the second order training algorithms compared to JRprop. The new proposed scheme outperforms BFGS and SCG algorithms in terms of learning speed and generalisation in most of the 1000 runs.

Table 3.5: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Diabetes problem with respect to training speed and generalisation.

Diabetes	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	IRprop	JRprop	Time	Rprop	IRprop	JRprop
Rprop	–	477	190	3.5 ± 3.4	–	107	108
IRprop	401	–	144	3.7 ± 3.8	99	–	77
BFGS	144	125	14	7.7 ± 5.1	30	29	20
SCG	201	130	20	6.7 ± 4.8	49	49	24
JRprop	644	725	–	2.7 ± 2.1	600	590	–

The Thyroid problem

This dataset consists of patient query data and patient examination data. Appendix A describes in details this interesting problem. An architecture with 21-4-3 nodes is used, as suggested by Treadgold and Gedeon [124]. The termination criterion is $E < 0.0036$ within 2000 epochs as suggested in [124], and we set the parameter $m = 5$.

Table 3.6: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Diabetes problem with respect to training speed and generalisation.

Diabetes	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	BFGS	SCG	JRprop	Time	BFGS	SCG	JRprop
BFGS	–	205	14	7.7 ± 5.1	–	179	39
SCG	301	–	20	6.7 ± 4.8	169	–	34
JRprop	926	920	–	2.7 ± 2.1	700	690	–

Results are given in Table 3.7. The JRprop outperforms the other algorithms, particularly in training speed. The results in Table 3.8 show that average training time and the corresponding standard deviation of the Rprop and IRprop increase when the results of all the runs (1000 cases) are taken into account. This happens because the two algorithms converge only 870 and 910 times out of the 1000 runs (see Table 3.7). Nevertheless, the performance of the JRprop appears significantly better: JRprop converges faster in 525 and 538 runs compared to the Rprop and the IRprop, respectively. Moreover, the value of the deviation of the new algorithm is significantly lower than the standard deviation of the other two methods (see Table 3.8). Rprop and IRprop achieve better generalisation than JRprop in 350 and 340 runs, while JRprop outperforms in 423 and 431 runs respectively.

It is important in this stage to discuss the experimental results of the BFGS and SCG. In these tests, the SCG didn't converge in any of the 1000 runs; this is indicated with a D in the Table 3.7. Generally the training in the SCG stops when any of these conditions occur: (i) The maximum number of epochs (repetitions) is reached, (ii) The maximum amount of time has been exceeded, (iii) Performance has been minimised to the goal, (iv) The performance gradient falls below a predefined value. In these experiments we change the value of the gradient in order to stop the training only in the case the error goal was met or the number of the maximum epochs is reached. When the error goal is very

small (e.g. 0.0036) the SCG cannot converge. If we increase the error to 0.01 then the SCG has a 90% convergence success and an average about 500 epochs to meet the error target, while the Rprop and JRprop need 80 and 70 epochs respectively.

The BFGS outperforms in terms of learning speed and classification success the Rprop and IRprop but exhibits a small percentage of convergence (only 360 out the 1000 runs); this is because the approximations of the Hessian matrix were close to singular. JRprop outperforms other algorithms both in learning speed and classification success.

Table 3.7: Comparison of algorithms performance in the Thyroid problem for the runs which converged

Thyroid				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	770 (+)	24.20 ± 12.1 (+)	97.9 (+)	80 (+)
IRprop	760 (+)	23.50 ± 12.4 (+)	97.9 (+)	80 (+)
BFGS	586 (−)	21.2± 10.4 (−)	98.12 (−)	36 (+)
SCG	*D*	*D*	*D*	0
JRprop	700	22.00 ± 9.5	98.0	88

Table 3.8: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Thyroid problem with respect to training speed and generalisation.

Thyroid	**Times faster algorithm**					**Times better Generalisation**			
Algorithm	Rprop	IRprop	BFGS	JRprop	Time	Rprop	IRprop	BFGS	JRprop
Rprop	–	181	340	171	28.7 ± 17.9	–	20	12	35
IRprop	185	–	300	161	29.5 ± 18.6	13	–	10	34
BFGS	325	338	–	310	43.7 ± 30.5	123	131	–	100
SCG	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*
JRprop	479	538	400	–	23.7 ± 10.5	129	121	80	–

The Genes2 problem

This benchmark is known as the *genes2* problem *Appendix* A. A FNN with hyperbolic tan activation function (tansig symbolised in Matlab)(120-4-2-3 nodes

network) is used, and the testing and training data were created as suggested in PROBEN1 [88]. The error goal was set at 10^{-5} in an attempt to explore the effectiveness of the algorithms in reaching minimisers with high degree of accuracy. The parameter m was set at a value of five for this difficult problem ($m = 5$).

Table 3.9 shows the performance of each algorithm in terms of: average number of epochs (Epochs), average training time (Time, in secs) to reach the error goal $\pm$ the corresponding value of standard deviation, average generalisation (Generalisation, percentage of correctly classified test patterns) and convergence success out of the 1000 runs (Convergence, percentage). The improvement of JRprop is statistically significant when compared to Rprop and IRprop with respect to both learning speed and classification success. Furthermore, it is important to note that the testing classification success of JRprop is 100% in this binary problem. This is particularly important as the Wilcoxon Rank Test is satisfied. However it is clear that the second order algorithms achieve significantly improved learning speed in this problem with respect to the first order methods with the SCG outperforming all the other algorithms.

Table 3.9: Comparison of algorithms performance in the Genes2 problem for the runs which converged

Genes				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	4860 (+)	73.5 $\pm$ 36.4 (+)	97.1 (+)	85 (+)
IRprop	4900 (+)	73.8 $\pm$ 36.9 (+)	97.0 (+)	85 (+)
BFGS	3790 (−)	54.6 $\pm$ 26.9 (−)	96.70 (+)	82 (+)
SCG	2170 (−)	47.6 $\pm$ 24.9 (−)	96.20 (+)	80 (+)
JRprop	4800	71.8 $\pm$ 34.0	100	92

Table 3.10 presents comparative results in terms of training speed (in secs) and generalisation for all of the 1000 runs. Table 3.10 highlights the number of runs an algorithm is better than the other methods. It also shows the average training time and the corresponding standard deviation for each algorithm calculated over

the 1000 runs. The JRprop algorithm achieves the best performance compared to other tested first order training algorithms Rprop and IRprop. Nevertheless, both of the BFGS and SCG display better learning speed than JRprop. Therefore, Table 3.11 contains a detailed description of the JRprop, BFGS and SCG within 1000 runs.

Table 3.10: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Genes2 problem with respect to training speed and generalisation.

Genes	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	IRprop	JRprop	Time	Rprop	IRprop	JRprop
Rprop	–	40	426	84.3 ± 44.6	–	2	34
IRprop	39	–	390	84.9 ±45.0	0	–	33
BFGS	674	650	600	77.0 ±44.0	4	5	0
SCG	700	690	650	74.9 ±42.0	10	12	0
JRprop	574	590	–	78.0 ± 40.0	44	45	–

Table 3.11: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Genes2 problem with respect to training speed and generalisation.

Genes	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	BFGS	SCG	JRprop	Time	BFGS	SCG	JRprop
BFGS	–	150	600	77.0 ±44.0	–	55	0
SCG	200	–	650	74.9 ±42.0	40	–	0
JRprop	274	190	–	78.0 ± 40.0	920	920	–

3.4.1 Classification of protein Localisations sites

In this section the new learning scheme, the JRprop, is applied in two bioinformatics datasets. A comparative study with Rprop and IRprop is presented. Preliminary results (200 runs) of BFGS and SCG show that these two algorithms cannot meet the target error in almost all tested runs. Therefore, the

experimental results with solely the JRprop, Rprop and IRprop algorithms will follow.

The Ecoli problem

This problem concerns the classification of protein localisation patterns into eight classes. The Escherichia dataset consists of 336 different proteins. It is a drastically imbalanced data set of 336 patterns, which means that there are classes with 140 patterns and other ones with only 2 and 5 patterns. For this problem we have used seven different attributes as in [46, 47, 73, 74]. Finally, the parameter m is set at value of five, $m = 5$.

Literature suggests no standard architecture for the E.coli problem. To get an understanding of the requirements of this problem a set of preliminary experiments is conducted to find the most suitable FNN architecture in terms of training speed. In these experiments the neural networks were tested using 4–fold cross validation, as this approach has been used before in the literature for training probabilistic and nearest neighbor classifiers in this problem [47]. Several networks are trained with one and two hidden layers using the Rprop algorithm. In particular, various combinations of hidden nodes were tried, i.e. 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes. Each FNN architecture was trained 10 times with different initial weights. The best available architectures found was a 7-16-8 FNN. Rprop–trained FNNs of this architecture achieved better generalisation than the best results reported in the literature [47], when the training error goal was $E < 0.02$ [6].

Results from 1000 runs for three algorithms using the same architecture are given in Table 3.12. A detailed account of the algorithms' performance is exhibited in Table 3.13. The new learning scheme is faster than Rprop and IRprop 561 and 577 times respectively. It is clear that there are no significant differences in the generalisation performance of the three algorithms.

Table 3.12: Comparison of algorithms performance in the Ecoli problem for the runs which converged

Ecoli				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	157 (+)	1.36 ± 0.35 (+)	90.4 (−)	99 (−)
IRprop	158 (+)	1.37 ± 0.40 (+)	90.4 (−)	99 (−)
JRprop	144	1.24 ± 0.22	90.6	100

Table 3.13 presents comparative results in terms of time (in secs) and generalisation for the 1000 runs. Table 3.13 highlights the number of runs an algorithm is better than the other methods. It also shows the average training time and the corresponding standard deviation for each algorithm calculated over the 1000 runs;

Table 3.13: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Ecoli problem with respect to training speed and generalisation.

Ecoli	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	IRprop	JRprop	Time	Rprop	IRprop	JRprop
Rprop	–	493	419	1.45 ± 2.7	–	0	390
IRprop	414	–	398	1.50 ± 2.9	0	–	390
JRprop	561	577	–	1.24 ± 0.22	381	381	–

The Yeast problem

In the Yeast problem a pattern consists of 8 attributes [47, 73, 74]. The data set is drastically imbalanced like the Ecoli data. We worked in a similar way to the E.coli problem to find a suitable architecture, as there is no specific advice in the literature. A set of preliminary experiments are conducted training several networks with one and two hidden layers using the Rprop algorithm. Combinations of 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes were tried. Each FNN architecture

was trained 10 times with different initial weights. The best available architecture found was an 8-16-10 FNN, and this was used in the rest of the experiments. In these experiments 10–fold cross validation was applied, as this approach has been reported to produce higher generalisation performance [47, 133]. Rprop–trained FNNs that reached training errors $E < 0.05$ produced better generalisation on the average, [6], than the best available classifier [47]. For this problem the tuning of the parameter m does not play a significant role in the performance of the JRprop. Similar results can be achieved with values of m greater than one. Specifically, in the following tables, results are presented with the value of $m = 5$.

Comparative performance results are presented in Table 3.14. Table 3.15 shows the results in detail.

Table 3.14: Comparison of algorithms performance in the Yeast problem for the runs which converged

Yeast				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	926 (+)	32.0 ± 7.9 (+)	63.7 (−)	99 (−)
IRprop	930 (+)	33.0 ± 8.5 (+)	63.7 (−)	99 (−)
JRprop	870	30.5 ± 5.7	63.7	100

Table 3.15: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Yeast problem with respect to training speed and generalisation.

Yeast	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	IRprop	JRprop	Time	Rprop	IRprop	JRprop
Rprop	–	451	440	1.45 ± 2.7	–	0	480
IRprop	559	–	423	1.50 ± 2.9	0	–	480
JRprop	547	576	–	1.20 ± 0.2	476	476	–

3.5 Discussion

JRprop is a heuristic scheme that is based on the idea of *function comparison methods*, [101]. It takes into account the evolution of the error and the signs of the gradient values; $\frac{1}{m^q}$ is a reduction factor that is used to update the midpoint of the considered interval. The choice of m, q has an influence on the number of error function evaluations required to obtain an acceptable weight vector [66].

The new learning scheme achieves better performance than the other tested algorithms. In all tested cases the JRprop's learning speed is significantly faster than the Rprop and IRprop. However, the second order training schemes converge faster in the genes problem and the BFGS algorithm also in the thyroid problem. Nevertheless, in most cases the new adaptive learning scheme, the JRprop, generalises better and has an increased convergence success. The influence of the reduction factor $\frac{1}{m^q}$ plays an important role in the learning speed of the JRprop. In these experiments, in all cases, JRprop is applied by setting $m = 5$. The increase of the value of m usually increases the convergence success but there is no guarantee that the learning speed subsequently increases (i.e in cancer, E.coli and Yeast problem by applying $m = 2$ we can improve slightly the learning speed of JRprop).

3.6 Summary and Contribution of the chapter

It is widely accepted that the Rprop algorithm is one of the best performing sign–based learning algorithms for neural networks with arbitrary topology. This chapter introduced a new class of sign–based schemes that are based on the composite nonlinear Jacobi process. An algorithm of this class that applies the bisection method to locate subminimiser approximations along each weight direction was derived, and a simplified version was proposed. This new algorithm

constitutes an efficient improvement of the Rprop algorithm that is built on a theoretical basis. By fine tuning the extra parameter m of the Jacobi-Rprop algorithm, significant improvement in the learning speed can be achieved in all cases without affecting the classification success. Finally, by taking into account the evolution of the error in order to perform the weights update, the JRprop avoids converging in local minima, which are far away from a desired minimiser.

Results were presented on the behavior of the new algorithm in pattern classification problems from the PROBEN1 repository to compare its performance to the Rprop, and the IRprop (a recently introduced modification of the Rprop algorithm).

The JRprop exhibited significantly better convergence speed than the Rprop and IRprop in most cases. 1000 different runs were conducted for each problem in order to obtain a better view of the tested algorithms' performance. Finally, the Wilcoxon test was implemented and comparison tables with the details of the 1000 different runs were given. JRprop still has problems to converge in local minima in some cases, but there is a significant improvement in its convergence ability. The next chapter discusses how the Rprop and the Jacobi-Rprop methods can be equipped with the global convergence property, i.e. convergence to a local minimiser from any initial starting point, so as to overcome the problem of local minima. Also, ideas based on global search methods are investigated in chapter five, in order to improve the convergence success and overcome the problem of the local minima.

Chapter 4

Globally Convergent Training Algorithms

4.1 Introduction

In the previous chapter a modification of the Rprop was presented based on the Jacobi procedure. Despite the fact that improved learning speed was achieved, convergence was not ensured for all initial conditions. This chapter proposes new Rprop based learning schemes that are equipped with the global convergence property. i.e. starting from almost any initial weight vector the sequence of the weights generated by the learning scheme will converge to a local minimiser of the error function.

At this point, it is important to clarify that in our context, the term global convergence is used in the same sense as in Dennis and Schnabel [25], where the authors use it "to denote a method that is designed to converge to a *local* minimiser of a nonlinear function, *from almost any starting point*" [25, p.5]. Dennis and Schnabel also note that " it might be appropriate to call such methods *local* or *locally convergent*, but these descriptions are already reserved by tradition

for another usage". Moreover, Nocedal, [75, p.200], defines a globally convergent algorithm as an algorithm with iterates that converge from a remote starting point. Thus, in this context, global convergence is totally different from global optimisation [124].

In this chapter, results of the experimental evaluation of the new globally convergent algorithms as well as comparisons with the original Rprop and the IRprop are reported. The chapter ends with a short discussion, concluding remarks and the contribution of the proposed algorithms. In order to illustrate the effectiveness of this approach, two globally convergent algorithms are comparatively evaluated.

4.2 The Notion of Global Convergence

As it has already been mentioned, the globally convergent algorithms are different from the global optimisation methods [75, p.200]. In a strict mathematical sense, global optimisation means to find the complete set of the globally optimal solutions (global minimisers) x^* of the objective function f, and the associated global optimum value $f^* = f(x^*)$. In this section, finding global minimisers of the error function E is not the main target, but emphasis is given on the development of algorithms that will converge to a local minimiser with certainty.

The globally convergent schemes are built on the following assumptions from unconstrained minimisation theory: (i) $f : \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is the function to be minimised and f is bounded below in R^n; (ii) f is continuously differentiable in a neighborhood $\mathcal{N}$ of the level set $\mathcal{L} = \{x : f(x) \leqslant f(x^0)\}$, (iii) the gradient of f denoted by g is Lipschitz continuous on R^n that is for any two points x and $y \in R^n$, g satisfies the Lipschitz condition $\|g(x)-g(y)\| \leqslant L\|x-y\|, \quad \forall x, y, \in \mathcal{N}$, where $L > 0$ denotes the Lipschitz constant, and x^0 is the starting point of the

following iterative scheme

$$x^{k+1} = x^k + \tau^k d^k, \qquad k = 0, 1, \ldots \tag{4.1}$$

Convergence of the general iterative scheme (4.1), in which d^k is the search direction and $\tau^k > 0$ is a step–length, requires that the adopted search direction d^k satisfies the condition $g(x^k)^\top d^k < 0$, which guarantees that d^k is a descent direction of $f(x)$ at x^k. The step–length in (4.1) can be defined by means of a number of rules, such as the Armijo's rule [25, 140], the Goldstein's rule [25], or the Wolfe's rule [144, 145], and guarantees the convergence in certain cases. For example, when the step–length is obtained through Wolfe's rule [144, 145]

$$f(x^k + \tau^k d^k) - f(x^k) \leqslant \sigma_1 \tau^k g(x^k)^\top d^k, \tag{4.2}$$

$$g(x^k + \tau^k d^k)^\top d^k \geqslant \sigma_2 g(x^k)^\top d^k, \tag{4.3}$$

where $g(x)$ is the gradient of f at x, and $0 < \sigma_1 < \sigma_2 < 1$, then a theorem by Wolfe [144, 145] is used to obtain convergence results. Moreover, the Wolfe's Theorem [25, 75] suggests that if the cosine of the angle between the search direction d^k and $-g(x^k)$ is positive then $\lim_{k\to\infty} \| g(w^k) \| = 0$, which means that the sequence of gradients converges to zero. For an iterative scheme (4.1), $\lim_{k\to\infty} \| g(w^k) \| = 0$ is the best type of global convergence result that can be obtained (see [75] for a detailed discussion). Evidently, no guarantee is provided that (4.1) will converge to a global minimiser, x^*, but only that it possesses the global convergence property, [25, 75], to a local minimiser.

In general, any batch–type BP training algorithm of the form (4.1) can be made globally convergent if

(i) the adopted search direction d^k is a descent direction and it does not tend to be orthogonal to the gradient direction

(ii) the learning rate τ^k satisfies the two Wolfe conditions (4.2)–(4.3). Notice that, since d^k is a descent direction and E is continuously differentiable

and bounded below along the radius $\{w^k + \tau d^k \mid \tau > 0\}$, then there always exist τ^k satisfying the Wolfe's conditions (4.2) and (4.3) [25, 75].

Next, the Globally Resilient Backpropagation (GRprop) is proposed and its performance is investigated.

4.3 The Globally Resilient Backpropagation Algorithm

In our approach Rprop's convergence to a local minimiser is treated with principles from unconstrained minimisation theory. In batch training, the conditions (i)-(iii) of section 4.2 are fulfilled because E is bounded from below, since $E(w) \geqslant 0$. For a given training set and network architecture, if w^* exists such that $E(w^*) = 0$, then w^* is a global minimiser; otherwise, w with the smallest $E(w)$ value is considered a global minimiser. Also, when using *smooth enough* activations (the derivatives of at least order p are available and continuous), such as the well known hyperbolic tangent, the logistic activation function etc., the error E is also smooth enough.

Theorem 1. Suppose that the assumptions (i)–(iii) are fulfilled, then for any $w^0 \in R^n$ and any sequence $\{w^k\}_{k=0}^{\infty}$ generated by the Rprop scheme

$$w^{k+1} = w^k - \tau^k \operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}\Big(g(w^k)\Big), \qquad k = 0, 1, \ldots \tag{4.4}$$

where $\operatorname{sign}\Big(g(w^k)\Big)$ denotes the column vector of the signs of the components of $g(w^k) = \big(g_1(w^k), g_2(w^k), \ldots, g_n(w^k)\big)$, $\tau^k > 0$, η_m^k $(m = 1, 2, \ldots, i-1, i+1, \ldots, n)$ are small positive real numbers generated by Rprop's learning rates schedule:

$$\text{if} \quad \Big(g_m(w^{k-1}) \cdot g_m(w^k) > 0\Big) \quad \text{then} \quad \eta_m^k = \min\Big(\eta_m^{k-1} \cdot \eta^+,\ \Delta_{\max}\Big), \tag{4.5}$$

$$\text{if} \quad \Big(g_m(w^{k-1}) \cdot g_m(w^k) < 0\Big) \quad \text{then} \quad \eta_m^k = \max\Big(\eta_m^{k-1} \cdot \eta^-,\ \Delta_{\min}\Big), \tag{4.6}$$

$$\text{if} \quad \Big(g_m(w^{k-1}) \cdot g_m(w^k) = 0\Big) \quad \text{then} \quad \eta_m^k = \eta_m^{k-1}, \tag{4.7}$$

where $0 < \eta^- < 1 < \eta^+$, $\Delta_{\max}$ is the learning rate upper bound, $\Delta_{\min}$ is the learning rate lower bound and

$$\eta_i^k = -\frac{\sum_{\substack{j=1 \\ j\neq i}}^{n} \eta_j^k \, g_j(w^k) + \delta}{g_i(w^k)}, \qquad 0 < \delta \ll \infty, \qquad g_i(w^k) \neq 0, \tag{4.8}$$

it holds that $\lim_{k\to\infty} \| \, g(w^k) \, \| = 0$.

Proof: Evidently, E is bounded below on R^n. The sequence $\{w^k\}_{k=0}^{\infty}$ generated by the iterative scheme (4.4) follows the direction

$$d^k = -\mathrm{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \, \mathrm{sign}\Big(g(w^k)\Big),$$

which is a descent direction if $\eta_m^k,\ m = 1, 2, \ldots, i-1, i+1, \ldots, n$ are positive real numbers derived from Relations (4.5)–(4.7), and η_i^k is given by Relation (4.8), since $g(w^k)^\top d^k < 0$. Following the proof of [141, Theorem 6], since d^k is a descent direction and E is continuously differentiable and bounded below along the radius $\{w^k + \tau d^k \mid \tau > 0\}$, then there always exist τ^k satisfying (4.2)–(4.3) [25, 75]. Moreover, the Wolfe's Theorem [25, 75] suggests that if the cosine of the angle between the descent direction d^k and the $-g(w^k)$ is positive then $\lim_{k\to\infty} \| \, g(w^k) \, \| = 0$. In our case, indeed $\cos\theta_k = \dfrac{-g(w^k)^\top d^k}{\|g(w^k)\| \, \|d^k\|} > 0$. Thus the theorem is proved. □

The modified Rprop, named *GRprop*, is implemented through Relations (4.4)-(4.8). It is worth mentioning that Relation (4.8) can be applied cyclically over the local learning rates, or randomly. In all experiments reported it is replaced each time the smallest learning rate value that yields from the Rprop's schedule with an η_i^k value calculated from relation (4.8).

The role of δ in relation (4.8) is to alleviate problems with limited precision that may occur in simulations, and should take a small value proportional to the square root of the relative machine precision. In our tests we set $\delta = 10^{-6}$ in an attempt to test the convergence accuracy of the proposed strategy. Also $\tau^k = 1$ for all k allows the minimisation step along the resultant search direction

to be explicitly defined by the values of the local learning rates. The length of the minimisation step can be regulated through τ^k tuning to satisfy (4.2)–(4.3). Checking (4.3) at each iteration requires additional gradient evaluations; thus, in practice (4.3) can be enforced simply by placing the lower bound on the acceptable values of the learning rates [66, p.1772], i.e. $\Delta_{\min}$.

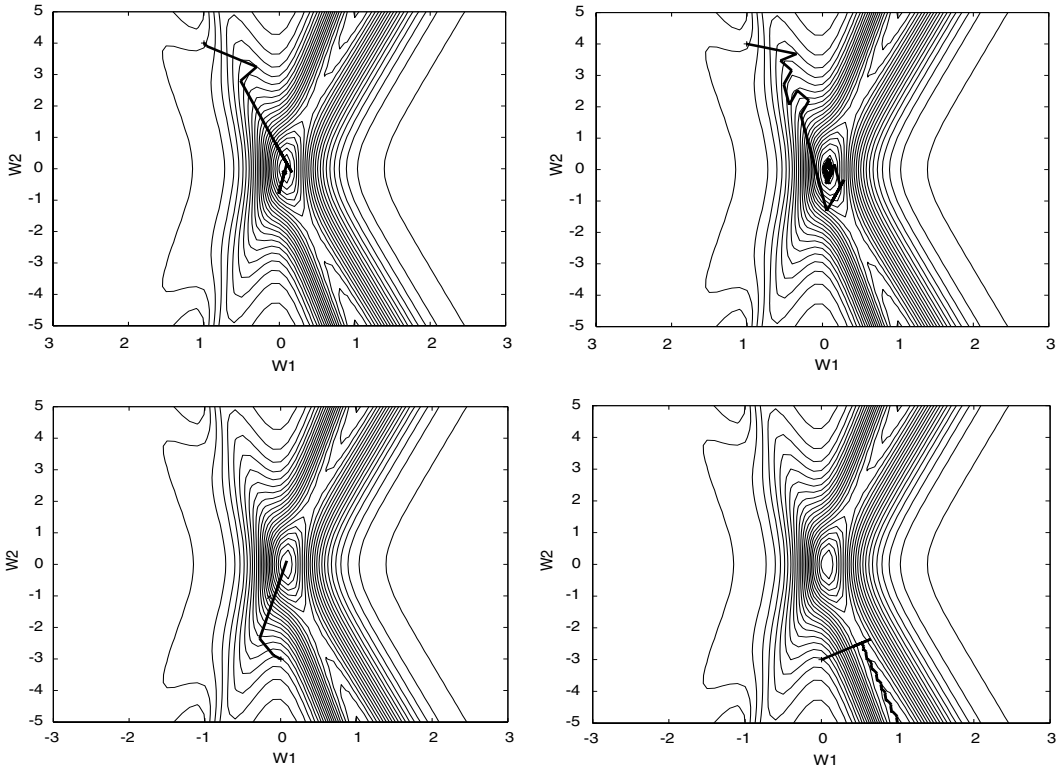


Figure 4.1: Weight trajectories of GRprop (left) and Rprop (right).

A simple problem is used to visualise the behavior of the GRprop and compare it with the original method. It is a single node with two weights and logistic activation function. Figure 4.1 (top row) shows that under the same initial weights and heuristic values, [91], GRprop locates the feasible minimum at the center of the contour plot successfully (Figure 4.1, left), while Rprop oscillates around the neighborhood of the minimiser (Figure 4.1, right). Figure 4.1 (second row) shows how GRprop locates the minimiser successfully, whilst Rprop's trajectory leads to a point with error value higher than the minimiser.

4.3.1 Experimental study using biological data sets

Next, we evaluate the performance of the new method in some benchmarks and compare it with the original Rprop [91]. The recently proposed modification of Rprop, namely IRprop [49], achieves similar results with the Rprop algorithm, so we present only the comparison results with the Rprop. We have used well-studied problems from the UCI Repository of Machine Learning Databases [72], as well as problems studied extensively by other researchers in an attempt to reduce as much as possible biases introduced by the size of the weights space.

Below, we report results from 1000 independent trials for neural network classification problems. These 1000 random weight initialisations are the same for all the learning algorithms. In all cases we have used networks with sigmoid hidden and output nodes, and adopted the notation I-H-O to denote a network architecture with I inputs, H hidden layer nodes and O outputs nodes.

The data sets for the cancer1, diabetes1, thyroid1, and genes2 problems were used as supplied on the PROBEN1 website. PROBEN1 provides explicit instructions for creating training and testing sets and choosing network architectures for many problems [88]. The data sets for the E.coli problem was used as supplied on the UCI repository and the sets for training and testing were created following guidelines published by Horton [47]. There are no standard neural architectures for these two problems so we have done our own preliminary experiments as will be described in the problem subsections. Full description of the tested problems is given in *AppendixA*.

In all experiments the parameters have been set as follows: $\eta^+ = 1.2$; $\eta^- = 0.5$; $\Delta_{ij}^0 = \eta^0 = 0.1$; $\Delta_{\max} = 50$ [91]. Finally we have set $\delta = 10^{-6}$ in an attempt to test the convergence accuracy of the proposed strategy and also $\tau^k = 1$ for all k.

The Cancer problem

In this problem a feed-forward neural network with 9–4–2–2 nodes is used as suggested in the PROBEN1 benchmark collection and in [8]. The error goal in training is $E < 0.02$ [8, 49].

Tables 4.1, 4.2 presents the comparative results and highlight the performance of the GRprop compared to Rprop. There is significant improvement in terms of learning speed and convergence success.

Table 4.1: Comparison of algorithms performance in the Cancer problem for the converged runs

Cancer				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	280 (+)	1.85 ± 1.30 (+)	97.0 (+)	94 (+)
GRprop	246	1.44 ± 0.48	97.3	97

Table 4.2: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Cancer problem with respect to training speed and generalisation.

Cancer	**Times faster algorithm**			**Times better Generalisation**	
Algorithm	Rprop	GRprop	Time	Rprop	GRprop
Rprop	–	440	2.20 ± 2.51	–	372
GRprop	552	–	1.52 ± 0.85	477	–

The Diabetes problem

The applied architecture for this problem is the one suggested by others [49]. The error goal in this case was set at 0.14 to conform to the training error obtained in [49]. The next Tables 4.3, 4.4 correspond to 1000 different runs. The new globally resilient backpropagation algorithm achieves in the most runs to meet

the error target (Convergence success = 95%) as well as reduced the training time. Finally the Table 4.4 shows how many times the GRprop outperforms Rprop in terms of speed and classification success.

Table 4.3: Comparison of algorithms performance in the Diabetes problem for the converged runs

Diabetes				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	455 (+)	2.40 ± 2.1 (+)	75.8 (+)	86 (+)
GRprop	430	2.30 ± 1.9	76.0	95

Table 4.4: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Diabetes problem with respect to training speed and generalisation.

Diabetes	**Times faster algorithm**			**Times better Generalisation**	
Algorithm	Rprop	GRprop	Time	Rprop	GRprop
Rprop	–	298	3.5 ± 3.4	–	90
GRprop	371	–	2.8 ± 2.6	155	–

The Genes2 problem

This benchmark is known as the *genes* problem, a binary problem. The error goal was set at 10^{-5} in an attempt to explore the effectiveness of the algorithms in reaching minimisers with high degree of accuracy.

Table 4.5 shows the performance of each algorithm in terms of: average number of epochs (Epochs), average training time (Time, in secs) to reach the error goal ± the corresponding value of standard deviation, average generalisation (Generalisation, percentage of correctly classified test patterns) and convergence success out of the 1000 runs (Convergence, percentage). For example, GRprop-trained networks always generalise slightly better than other networks: GRprop achieved 100% average generalisation in the runs which converged, while Rprop

achieved an average 99.1% generalisation in the 850 runs that they converged. Figure 4.2 (leftside), shows how GRprop converges to a feasible solution ($E < 10^{-5}$), while Rprop to a minimiser with higher error value. That's highlights the improved convergence rate that GRprop achieves compared to Rprop algorithm.

Table 4.5: Comparison of algorithms performance in the Genes problem for the runs which converged

Genes				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	4860 (+)	73.5 ± 36.4 (+)	99.1 (+)	85 (+)
GRprop	4830	73.1 ± 34.8	100	94

Table 4.6 presents comparative results in terms of training speed (in secs) and generalisation for the 1000 runs. Table 4.6 highlights the number of runs an algorithm is better than the other methods. It also shows the average training time and the corresponding standard deviation for each algorithm calculated over the 1000 runs; it is clear that the Rprop's values show an increase compared to the corresponding values of Table 4.5, since all runs count. The GRprop outperforms Rprop in 536 times, while Rprop is faster than GRprop in 460 runs.

Finally, this table shows how many times an algorithm of the first column generalises better than the other algorithms: the GRprop achieves better generalisation than Rprop 27 times, while Rprop has equal or lower generalisation than GRprop in most runs.

Table 4.6: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Genes problem with respect to training speed and generalisation.

Genes	**Times faster algorithm**			**Times better Generalisation**	
Algorithm	Rprop	GRprop	Time	Rprop	GRprop
Rprop	–	460	84.3 ± 44.6	–	7
GRprop	536	–	79.4 ± 38.1	27	–

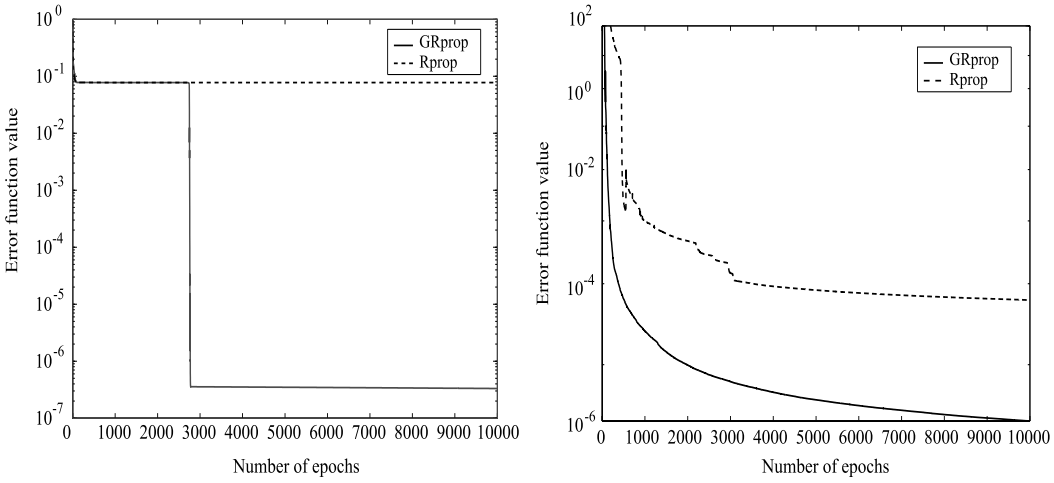


Figure 4.2: GRprop and Rprop learning curves: genes (left) and thyroid (right).

The E.coli problem

This problem concerns the classification of protein localisation patterns into eight classes. A drastically imbalanced data set of 336 patterns is used, where there are classes with 140 patterns and others with only 2 and 5 patterns. More details for this dataset is given in *Appendix A*.

Literature suggests no standard architecture for the E.coli problem. To get an understanding of the requirements of this problem we conducted a set of preliminary experiments to find the most suitable FNN architecture in terms of training speed. In these experiments the neural networks were tested using 4–fold cross validation, as this approach has been used before in the literature for training probabilistic and nearest neighbour classifiers in this problem [47]. We trained several networks with one and two hidden layers using the Rprop algorithm. In particular, we tried various combinations of hidden nodes, i.e. 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes. Each FNN architecture was trained 10 times with different initial weights. The best available architecture found was a 7-16-8 FNN. Rprop–trained FNNs of this architecture achieved better generalisation than the best results reported in the literature [47], when the training error goal was

$E < 0.02$ [6].

Results from 1000 runs for three algorithms using the same architecture are given in Table 4.7. A detailed account of the algorithms' performance is exhibited in Table 4.8. The new learning scheme is faster than Rprop 497 times. It is clear that there are no significant differences in the generalisation performance of the three algorithms.

Table 4.7: Comparison of algorithms performance in the Ecoli problem for the runs which converged

Ecoli				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	157 (+)	1.36 ± 0.35 (+)	90.4 (−)	99 (−)
GRprop	150	1.30 ± 0.29	90.6	100

Table 4.8: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Ecoli problem with respect to training speed and generalisation.

Ecoli	**Times faster algorithm**			**Times better Generalisation**	
Algorithm	Rprop	GRprop	Time	Rprop	GRprop
Rprop	–	478	1.45 ± 2.7	–	370
GRprop	497	–	1.30 ± 0.29	388	–

The Thyroid problem

In this problem, we have applied a network with 21-4-3 nodes, and the error goal was set at 0.0036, as suggested in [124].

Results are given in Table 4.9. GRprop outperforms the other algorithms particularly in training speed. Figure 2 (rightside) illustrates a case where GRprop converges to a minimiser while Rprop gets stuck at a local minimiser with higher error value.

The results in Table 4.10 show that the average training speed and the corresponding standard deviation of the Rprop and IRprop increase when the results of all the runs (1000 cases) are taken into account. This happens because the two algorithms converge only 87 times out of the 1000 runs (see Table 4.9). Nevertheless, the performance of the GRprop appears significantly better: GRprop converges faster in 79 and 78 runs compared to the Rprop and the IRprop, respectively. Moreover, the value of the deviation of the new algorithm is significantly lower than the standard deviation of the other two methods (Table 4.10). In 27 runs, Rprop and IRprop achieve better generalisation than GRprop, which outperforms in 58 runs.

Table 4.9: Comparison of algorithms performance in the Thyroid problem for the runs which converged

Thyroid				
Algorithm	Epochs	Time	Generalisation	Convergence
Rprop	770 (+)	24.20 ± 12.1 (+)	97.9 (+)	80 (+)
GRprop	740	23.10 ± 13.0	98.1	92

Table 4.10: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Thyroid problem with respect to training speed and generalisation.

Thyroid	**Times faster algorithm**			**Times better Generalisation**	
Algorithm	Rprop	JRprop	Time	Rprop	JRprop
Rprop	–	447	28.7 ± 17.9	–	366
GRprop	501	–	25.7 ± 14.5	459	–

4.3.2 Discussion

The new globally convergent batch training algorithm constitutes an efficient improvement of the Rprop algorithm that is built on a theoretical basis. GRprop has exhibited better convergence speed than Rprop, in all cases tested. Table 4.11

gives the summary of the results in terms of GRprop's percentage of improvement in training speed (in secs) over Rprop results.

Table 4.11 shows that the dimensionality of the search space might influence the performance of the tested methods. The GRprop achieves improved results either when the dimensionality of the search space is not high (e.g. Cancer and Diabetes problems) or when the dimensionality ranges from medium to high (as in the Ecoli and Genes problem where the number of the weights is significantly larger than in other problems). Moreover, the results of Table 4.11 acquire additional value when we take into account that the GRprop has also exhibited better convergence success in the majority of the 1000 runs for all problems.

It is worth mentioning the high generalisation performance of the Rprop–family in the E.coli problem. The two members of the family produced, on average, results which are better than the ones reported in the literature [47]. Finally, the standard deviation values in Tables 4.8, 4.10 and 4.5 point out that GRprop produces more consistent behaviour than the other algorithms.

Table 4.11: Summary of GRprop results in terms of learning speed improvement over Rprop.

Problem	Dimensionality	Rprop
Diabetes	30	+5.50 %
Cancer	56	+12.14 %
Genes2	503	+5.40 %
Thyroid	103	+4.80%
E.coli	264	+4.60%

4.4 The Globally Convergent Jacobi-Rprop Method

Based on the previous discussion about the globally convergent algorithms, we proceed with the following convergence result of the JRprop's scheme.

Theorem 2: Suppose that for the error function E conditions (i)-(iii) are fulfilled. Then, for any $w^0 \in R^n$ and any sequence $\{w^k\}_{k=0}^{\infty}$ generated by the JRprop's scheme

$$w^{k+1} = w^k - \tau^k \operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}\left(\nabla E(w^k)\right), \tag{4.9}$$

where $\operatorname{sign}(\nabla E(w^k))$ denotes the column vector of the signs of the components of $\nabla E(w^k) \equiv (\partial_1 E(w^k), \partial_2 E(w^k), \ldots, \partial_n E(w^k))$, $\tau^k > 0$ satisfies the Wolfe's conditions (4.2)–(4.3), η_m^k $(m = 1, 2, \ldots, i-1, i+1, \ldots, n)$ are small positive real numbers generated by the JRprop learning rates' schedule:

if $E(w^k) \leqslant E(w^{k-1})$ {

if $\left(\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) > 0\right)$ **then** $\eta_m^k = \min\left\{\eta_m^{k-1} \cdot \eta^+, \Delta_{\max}\right\}$ (4.10)

if $\left(\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) < 0\right)$ **then** $\eta_m^k = \max\left\{\eta_m^{k-1} \cdot \eta^-, \Delta_{\min}\right\}$ (4.11)

if $\left(\partial_m E(w^{k-1}) \cdot \partial_m E(w^k) = 0\right)$ **then** $\eta_m^k = \eta_m^{k-1}$, (4.12)

}

where $0 < \eta^- < 1 < \eta^+$, $\Delta_{\max}$ is the learning rate upper bound, $\Delta_{\min}$ is the learning rate lower bound, and

$$\eta_i^k = -\frac{\sum_{\substack{j=1 \\ j \neq i}}^{n} \eta_j^k \, \partial_j E(w^k) + \delta}{\partial_i E(w^k)}, \qquad 0 < \delta \ll \infty, \qquad \partial_i E(w^k) \neq 0, \tag{4.13}$$

holds that $\lim\limits_{k \to \infty} \|\nabla E(w^k)\| = 0$.

Proof: Evidently, E is bounded below on R^n. The sequence $\{w^k\}_{k=0}^{\infty}$ generated by the iterative scheme (4.9) follows the direction

$$d^k = -\operatorname{diag}\{\eta_1^k, \ldots, \eta_i^k, \ldots, \eta_n^k\} \operatorname{sign}\left(\nabla f(w^k)\right),$$

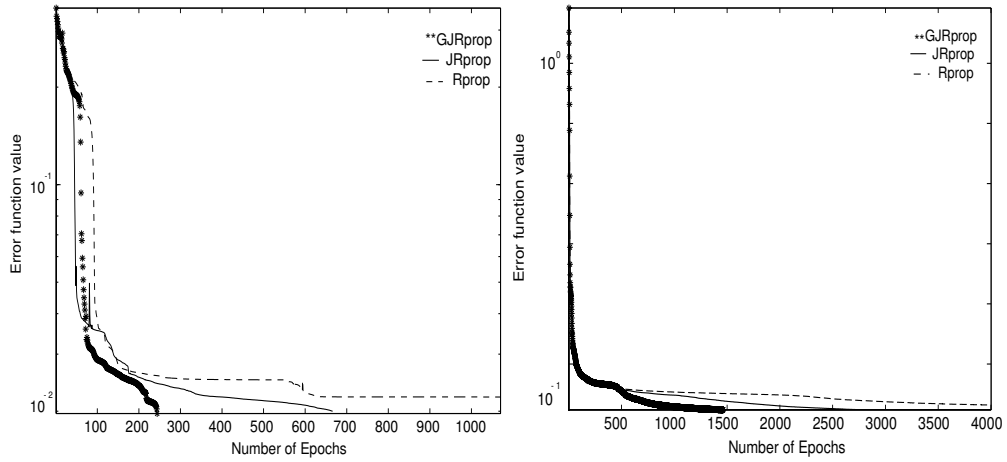
which is a descent direction if η_m^k, where $m = 1, 2, \ldots, i-1, i+1, \ldots, n$, are positive real numbers derived from Relations (4.10)–(4.12), and η_i^k is given by Relation (4.13), since $\nabla f(w^k)^\top d^k < 0$. Following the proof of [141, Theorem 6], since d^k is a descent direction and E is continuously differentiable and bounded below along the radius $\{w^k + \tau d^k \mid \tau > 0\}$, then there always exist τ^k satisfying Relations (4.2)–(4.3) [25, 75]. Moreover, the Wolfe's Theorem [25, 75] suggests that if the cosine of the angle between the descent direction d^k and the $-\nabla f(w^k)$ is positive then $\lim_{k\to\infty} \|\nabla f(w^k)\| = 0$. In our case, indeed $\cos\theta_k = \dfrac{-\nabla f(w^k)^\top d^k}{\|\nabla f(w^k)\| \, \|d^k\|} > 0$. $\square$

The Globally convergent modification of the JRprop, named GJRprop, is implemented through Relations (4.9)–(4.13). It is also important to mention that in case of an error increase then the corresponding weight update procedure of JRprop, descibed in [8], is adopted.

Like in the GRprop, we have set the training parameters of GJRprop in the same way ($\delta = 10^{-6}$, $\tau^k = 1$ for all k). The length of the minimisation step can be regulated through τ^k tuning to satisfy Wolfe's Conditions (4.2)–(4.3).

4.4.1 Experimental Study using biological datasets

In this section, we evaluate the performance of the GJRprop (Globally-Jacobi-Rprop), and compare it with the JRprop (Jacobi-Rprop). We have used well–studied problems from the UCI Repository of Machine Learning Databases of the University of California [72]. In all cases we have used networks with classic logistic activations. Below, we report results from 1000 independent trials. These 1000 random weight initialisations are the same for all the learning algorithms. In all cases we have used networks with sigmoid hidden and output nodes, and adopted the same notation as in the GRprop experimental section.



(a) (b)

Figure 4.3: GJRprop, JRprop and Rprop learning curves for (a) the cancer problem and (b) the diabetes problem

For the UCI problems, cancer1, diabetes1, genes, thyroid1, and Ecoli, we have used the data sets as supplied on the PROBEN1 website [88]. PROBEN1 provides explicit instructions for generating training and test sets, and choosing network architectures [88]. The results reported below present the average number of iterations (Epochs), the average training time to reach the error goal $\pm$ the corresponding value of standard deviation, the average generalisation (generalisation is measured as the percentage of correctly classified test patterns), and the percentage of convergence success (this percentage is calculated out of 1000 runs).

In all experiments the parameters have been set as follows: $\eta^+ = 1.2$; $\eta^- = 0.5$; $\Delta_{ij}^0 = \eta^0 = 0.1$; $\Delta_{\max} = 50$ [91]. Finally we have set $\delta = 10^{-6}$ in an attempt to test the convergence accuracy of the proposed strategy and also $\tau^k = 1$ for all k.

The Cancer1 problem

In this problem I have used a feed-forward neural network with 9–4–2–2 nodes as suggested in the PROBEN1 benchmark collection and in [8]. The error goal in training was $E < 0.02$ to harmonise with the training errors obtained in [8, 49]. The results for this pattern classification problem are summarised in Table 4.12. The new algorithm performs significantly better than the other two methods. Table 4.13 presents the number of times each algorithm outperforms the other methods in terms of training speed and generalisation within 1000 independent runs. It yields that the new learning scheme is frequently faster and achieves better generalisation than the other two members of the Rprop family. Figure 4.3(a) presents an example of convergence behaviour starting from the same initial conditions: the Rprop converges to a local minimiser, whilst both JRprop and GJRprop converge to a feasible solution ($E \leq 10^{-2}$) with GJRprop outperforming all other methods.

Table 4.12: Comparison of algorithms performance in the Cancer problem for the converged runs

Cancer				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	280 (+)	1.85 ± 1.30 (+)	97.0 (−)	94 (+)
JRprop	249 (+)	1.45 ± 0.80 (+)	96.9 (+)	97 (−)
GJRprop	240	1.40 ± 0.38	97.1	98

Table 4.13: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Cancer problem with respect to training speed and generalisation.

Cancer	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	JRprop	GJRprop	Time	Rprop	JRprop	GJRprop
Rprop	–	366	270	2.2 ± 2.51	–	231	200
JRprop	526	–	400	1.5 ± 0.90	241	–	172
GJRprop	550	453	–	1.49 ± 0.78	300	209	–

The Diabetes1 problem

The PROBEN1 collection proposes several architectures for this problem, including one with 8–2–2–2 nodes. We decided to use this architecture as it was also suggested by others [8, 49]. The error goal in this case was set at 0.14 to conform to the training error obtained in [8, 49].

Table 4.14 summarises the performance of the tested algorithms. Table 4.15 gives an analytic view of the comparative results in the 1000 trials.

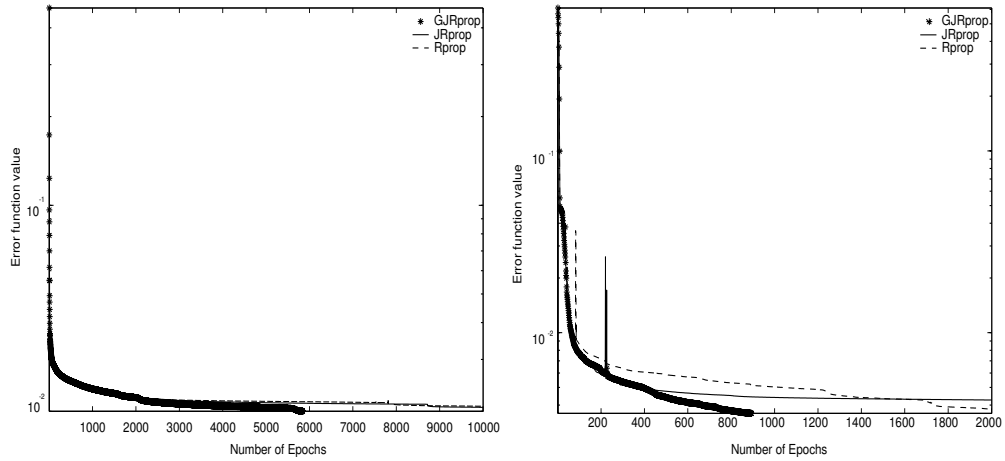
Table 4.14: Comparison of algorithms performance in the Diabetes problem for the converged runs

Diabetes				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	455 (+)	2.40 ± 2.1 (+)	75.8 (+)	86 (+)
JRprop	410 (−)	2.15 ± 1.5 (−)	76.1 (−)	94 (+)
GJRprop	400	2.10 ± 2.0	76.2	97

Table 4.15: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Diabetes problem with respect to training speed and generalisation.

Diabetes	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	JRprop	GJRprop	Time	Rprop	JRprop	GJRprop
Rprop	–	190	198	3.5 ± 3.4	–	108	80
JRprop	644	–	725	2.7 ± 2.1	600	–	590
GJRprop	401	210	–	2.5 ± 2.2	255	180	–

Figure 4.3(b) illustrates a training instance where all the methods start under the same initial conditions: the Rprop converges to a local minimiser, whilst both JRprop and GJRprop converge to a solution with $E \leq 10^{-1}$.



(a) (b)

Figure 4.4: GJRprop, JRprop and Rprop learning curves for (a) the E.coli problem and (b) the thyroid problem.

The Thyroid problem

Comparative results are given in Table 4.16. GJRprop outperforms the other algorithms. Moreover, the value of the deviation of the new algorithm is significantly lower than the standard deviation of the other two methods (see Table 4.16). Finally it is worth mentioning that the GJRprop exhibits significantly improved convergence success compared to the other tested algorithms. This can be attributed to the ability of the new globally convergent algorithm to follow descent directions.

A detailed account of the algorithms' performance is exhibited in Table 4.17. The new learning scheme is faster than Rprop. In terms of generalisation success, GJRprop outperforms Rprop and JRprop.

Figure 4.4(b) illustrates a case where GJRprop converges to a minimiser while Rprop and JRprop get stuck at a local minimiser with higher error value. As

Table 4.16: Comparison of algorithms performance in the Thyroid problem for the converged runs

Thyroid				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	770 (+)	24.20 ± 12.1 (+)	97.9 (+)	80 (+)
JRprop	700 (−)	22.00 ± 9.5 (−)	98.0 (+)	88 (+)
GJRprop	720	22.50 ± 7.5	98.23	92

Table 4.17: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Thyroid problem with respect to training speed and generalisation.

Thyroid	**Times faster algorithm**				**Times better Generalisation**		
Algorithm	Rprop	JRprop	GJRprop	Time	Rprop	JRprop	GRprop
Rprop	–	171	184	28.7 ± 17.9	–	66	57
JRprop	479	–	466	23.7 ± 10.5	169	–	60
GJRprop	386	184	–	23.50 ± 10.5	278	176	–

shown in Figure 4.4(b) Rprop's and JRprop's learning curves exhibit nonmonotone behaviour denoted by two hard peaks: one around time point 100 for the Rprop, and the other around point 200 for the JRprop. The GJRprop decreases monotonically the error function as it always follows a descent direction.

The Ecoli problem

This is a drastically imbalanced data set of 336 patterns, which means that there are classes with 140 patterns and other ones with only 2 and 5 patterns. In these experiments the neural networks were tested using 4–fold cross validation, as this approach has been used before [46, 47].

Results from 1000 runs for three algorithms using the same architecture are given in Table 4.18. A detailed account of the algorithms' performance is exhibited in Table 4.19.

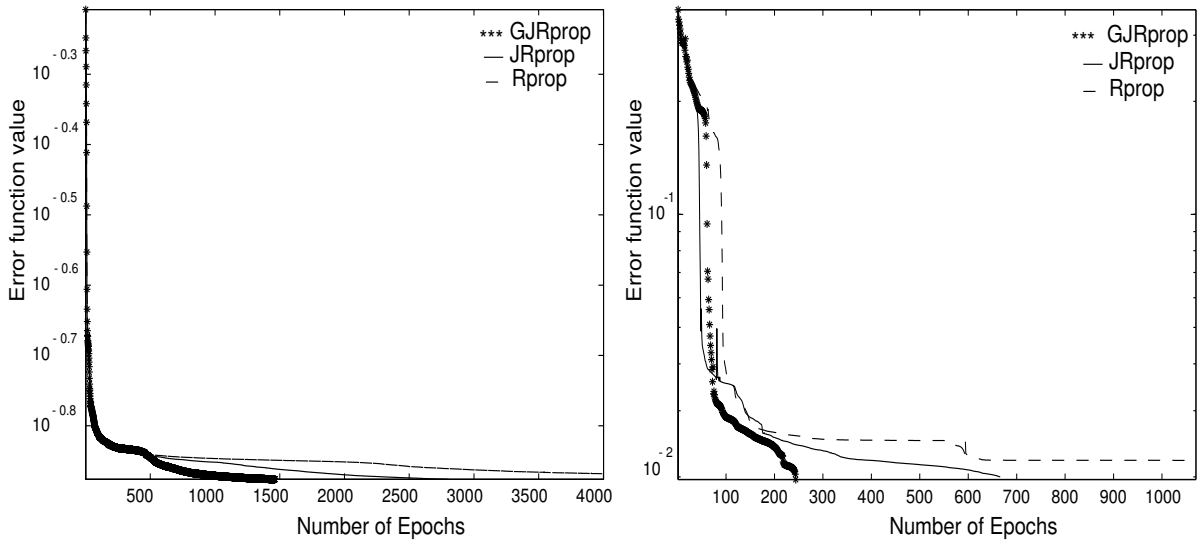


Figure 4.5: GJRprop, JRProp and Rprop learning curves: diabetes (left) and cancer (right).

Table 4.18: Comparison of algorithms' performance in the E.coli problem for the converged runs

E.coli				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	157 (+)	1.36 ± 0.35 (+)	90.4 (−)	99 (−)
JRprop	140 (+)	1.20 ± 0.20 (+)	90.6 (−)	100 (−)
GJRprop	135	1.18 ± 0.18	90.5	100

Table 4.19: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the E.coli problem with respect to training speed and generalisation.

E.coli	**Times faster algorithm**			**Times better Generalisation**		
Algorithm	Rprop	JRprop	GJRprop	Rprop	JRprop	GJRprop
Rprop	–	419	409	–	0	0
JRprop	561	–	460	381	–	87
GJRprop	586	470	–	407	70	–

4.4.2 Boolean function approximation problems

Another set of experiments has been conducted to empirically evaluate the performance of the globally convergent method in a well–studied class of boolean function approximation problems that exhibit strong local minima [17, 38]. These problems include the XOR problem (whose local minina and saddle points have been analysed in detail) and the various parity–N problems, which are considered as classic benchmarks [63, 86, 135]. The error target was set to $E \leq 10^{-7}$ within 2000 iterations in all cases (this is considered low enough to guarantee convergence to a "global" solution), and the adopted architectures were 2–2–1 for the XOR, 3–3–1 for the parity–3, 4–4–1 for the parity–4, 5–5–1 for the partiy–5.

XOR problem

Table 4.20 shows the performance of each algorithm in terms of: average number of iterations, average training time to reach the error goal ± the corresponding value of standard deviation, average generalisation (generalisation is measured as the percentage of correctly classified test patterns), and percentage of convergence success (this percentage is calculated out of 1000 runs). For example, GJRprop exhibits better convergence success than other methods: GJRprop achieved 70% average convergence success, while Rprop and JRprop achieved on average 55% and 60%. The GJRprop outperforms the Rprop significantly in terms of convergence speed and achieves similar learning speed to the JRprop.

Table 4.20: Comparison of algorithms performance in the XOR problem for the converged runs

XOR			
Algorithm	Epochs	Time (secs)	Convergence (%)
Rprop	190 (+)	1.77 ± 1.4 (+)	55 (+)
JRprop	125 (+)	1.42 ± 0.8 (+)	60 (+)
GJRprop	132	1.46 ± 0.7	70

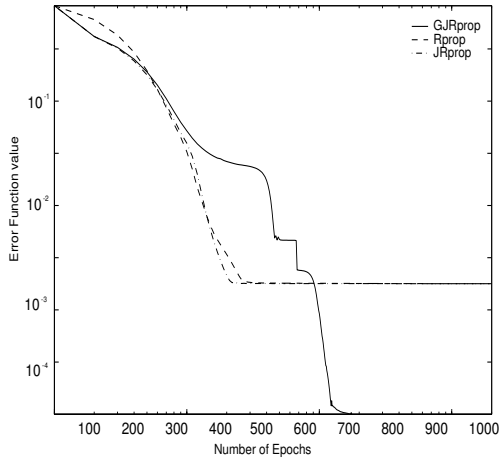


Figure 4.6: Learning error curves for the XOR problem

Figure 4.6 gives an example of the algorithms' convergence. Starting from the same initial conditions, the Rprop and the JRprop converge to a local minimiser, whilst GJRprop reaches a lower value.

Parity–3 problem

Table 4.21 presents comparative results in terms of training speed (in secs) and convergence success for the 1000 runs. It presents the average training time and the corresponding standard deviation for each algorithm calculated over the converged runs. GJRprop shows an increase in the percentage of convergence success. The globally convergent scheme manages to escape from some local minima and finds acceptable solutions with higher possibility than the other two tested methods.

Parity–4 problem

Comparative results for the parity–4 problem are given in Table 4.22. GJRprop outperforms the other algorithms particularly in convergence success. It achieves

Table 4.21: Comparison of algorithms performance in the parity–3 problem for the converged runs

Parity–3			
Algorithm	Epochs	Time (secs)	Convergence (%)
Rprop	1105 (+)	4.8 ± 2.9 (+)	22 (+)
JRprop	1000 (+)	4.5 ± 2.6 (+)	57 (+)
GJRprop	940	4.1 ± 2.3	67

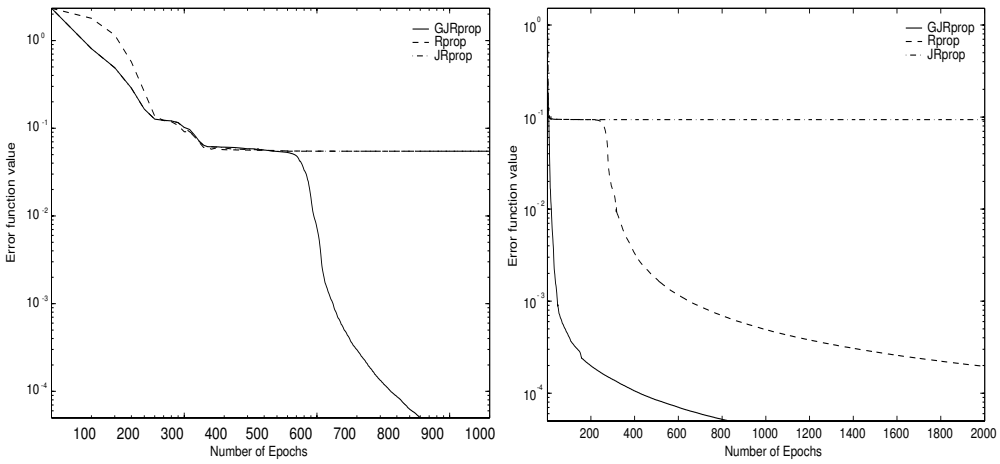
the error goal with 89% success whilst Rprop and JRprop have approximately 70% convergence success. Figure 4.7(a) illustrates a case where GJRprop converges to a minimiser while Rprop and JRprop get stuck at a local minimiser with higher error value.

Table 4.22: Comparison of algorithms performance in the parity–4 problem for the converged runs

Parity–4			
Algorithm	Epochs	Time (secs)	Convergence (%)
Rprop	1010 (+)	6.7 ± 4.4 (+)	69 (+)
JRprop	820 (+)	5.8 ± 3.5 (+)	72 (+)
GJRprop	715	5.2 ± 2.7	89

Parity–5 problem

Comparative results for the parity–5 problem are presented in Table 4.23. The JRprop algorithm achieves the best training speed, while GJRprop exhibits comparable performance. Both of them outperform the Rprop algorithm. Furthermore the GJRprop is more stable and shows an important convergence improvement over the other tested methods. Figure 4.7(b) illustrates a case where GJRprop converges to an acceptable minimiser while the other methods converge to local minima with higher function values.



(a) (b)

Figure 4.7: Typical learning error curves for (a) the parity–4 problem and (b) the parity–5 problem.

Table 4.23: Comparison of algorithms performance in the parity–5 problem for the converged runs

Parity–5			
Algorithm	Epochs	Time (secs)	Convergence (%)
Rprop	1050 (+)	7.5 ± 3.9 (+)	55 (+)
JRprop	830 (+)	5.1 ± 2.2 (+)	60 (+)
GJRprop	855	5.5 ± 2.5	72

4.5 Summary and Contribution of the Chapter

In this chapter globally convergent modifications of the Rprop and JRprop algorithms were introduced, named GRprop and GJRprop respectively. A theoretical justification for their development was provided, and comparative results in well studied benchmark problems were reported. In the tests, both GRprop and GJRprop exhibited better convergence speed and stability than Rprop. It is also important to highlight the fact that GRprop and GJRprop algorithms constitute an efficient improvement of the original Rprop that builds on a theoretical basis. This makes GRprop and GJRprop potentially useful components of global optimisation algorithms. Even though the convergence success rate of the two proposed algorithms with the global convergence property has been improved, there are still cases that converge to saddle points. A possible solution for this occasional problem is to make these algorithms more stochastic or deterministic depending on the tested problem each time. A class of such algorithms is proposed, described and examined in the next chapter.

Chapter 5

Nonextensive Hybrid Learning Schemes

5.1 Introduction

In the previous chapters some new proposed first order training schemes were investigated in depth. An inherent difficulty with these learning schemes is convergence to local minima. Sometimes the local minima can provide acceptable solutions, but they often result in poor network performance. The new algorithms described in the previous chapters, improve the learning speed and the convergence success. However, in some cases the problem with convergence to local minima still exists. In order to overcome this problem, hybrid algorithms which combine the benefits of the gradient descent and global optimisation are addressed in this chapter.

The present chapter introduces the basic notation necessary to study learning problems within the framework of statistical mechanics. A brief introduction to statistical mechanics and neural networks is also given followed by a presentation of the properties of nonextensive statistical mechanics [125]. Based on

this theory, a new hybrid gradient based learning scheme is proposed, namely HLS, that combines the benefits of the gradient descent algorithms and Tsallis Statistics. HLS applies a sign–based weight adjustment, inspired by the previous algorithms, on a perturbed version of the original error function. The HLS algorithm is examined, tested, and its performance is compared in many benchmarks problems with other known algorithms. Experimental results are very promising showing that the new learning scheme has many desirable properties.

5.2 Statistical Mechanics

"A theory is the more impressive the greater the simplicity of its premises, the more different kinds of things it relates, and the more extended its area of applicability. Therefore the deep impression that classical thermodynamics made upon me. It is the only physical theory of universal content which I am convinced will never be overthrown, within the framework of applicability of its basic concepts." A. Einstein

Statistical mechanics was the first foundational physical theory in which probabilistic concepts and probabilistic explanations played a fundamental role. It provides a framework for relating the microscopic properties of individual atoms and molecules to the macroscopic properties of materials that can be observed in every day life. Statistical mechanics is the application of statistics, which includes mathematical tools for dealing with the macroscopic equilibrium properties of large systems of elements that are subject to the microscopic laws of mechanics [60, 28, 130].

It is useful, at this stage, to briefly discuss some concepts that are used in statistical mechanics. A system is called microscopic if it is roughly of atomic dimensions, or smaller. On the other hand, a system is macroscopic when it is large enough to be visible in the ordinary sense. This is a rather inaccurate

definition. The exact definition depends on the number of particles in the system, which we shall call N. Typically, a system is macroscopic if $\frac{1}{\sqrt{N}} \ll 1$, which means that statistical arguments can be applied with reasonable accuracy. For instance, if we wish to keep the statistical error below one percent, then a macroscopic system would have to contain more than about ten thousand particles. Any system containing sensibly less than this number of particles would be regarded as essentially microscopic. Hence, statistical arguments could not be applied to such a system without unacceptable error.

In many cases of interest, the physical systems being simulated are believed to be ergodic. A system is ergodic if, for almost all initial conditions, the long-time average is equal to the state-space average [118]. For the physicist, ergodicity is a property that is usually postulated for a system in order to make many analytical and computational tasks tractable. The postulate is seldom justified from first principles. Instead, calculations for a system are performed assuming ergodicity. Then if theoretical predictions agree with experiments (actual or numerical), this is taken as evidence that the assumption is valid. Over the years, physicists have accumulated extensive experience about which systems at which energies are well modeled by the assumption of ergodicity [122]. However, it is nowadays clear that many interesting systems are in fact nonergodic! Such complex systems can, in many cases, be handled within the so called Nonextensive Statistical Mechanics [125, 36], a current generalisation of the standard statistical mechanics.

Statistical mechanics deals with the macroscopic equilibrium properties of large systems of elements that are subject to the microscopic laws of mechanics [56]. A standard assumption of statistical mechanics is that quantities like energy are "extensive" variables, meaning that the total energy of the system is proportional to the system size; similarly the entropy also supposed to be extensive. Generally, at least for the energy, this is justified by appealing to the short-range nature of the interactions which hold matter together, form chemical bonds, etc.

On the other hand, suppose one deals with long-range interactions, most prominently gravity; one can then find that energy is not extensive. This might makes necessary a generalisation of the standard theory.

Another important key concept in statistical mechanics is the ensemble. An ensemble is a collection of microstates of system of molecules, all having one or more extensive properties in common. Additionally, an ensemble is associated with a probability distribution according to a weight for each element (microstate) of the ensemble. The more important ensembles are "Microcanonical ensemble","Canonical ensemble", "Isothermal-isobaric ensemble", "Grand-canonical ensemble" [34].

Statistical mechanics is clearly mechanics plus theory of probabilities. Applications of the techniques of statistical mechanics are widespread, and include applications to physical, chemical, biological systems, and other interdisciplinary applications such as optimisation techniques [32, 69, 136, 146].

The statistical mechanical framework allows the description of complex systems with relatively simple models. A class of complex systems is neural networks. As they may have thousands of degrees of freedom (e.g. weights) it is possible to get inspiration from the theory of statistical mechanics.

5.2.1 Boltzmann's Statistical Mechanics

Statistical mechanics began as an effort to explain the macroscopic laws of thermodynamics by considering the microscopic application of Newton's laws to the particles that a material is made of. Newtonian mechanics is considered by many physicists as *eternal*, but by no means consider it as universal [128].

A diffuse belief exists that Boltzmann-Gibbs (BG) statistical mechanics and standard thermodynamics are eternal and universal. Indeed, for more than one century highly successful applications of the magnificent Boltzmann's connection of Clausius macroscopic entropy to the theory of probabilities are applied to the microscopic world. BG thermal statistics can easily be considered as one of the pillars of modern science. However, it is unavoidable to think that, like all other products of human mind, this formalism must have physical restrictions, i.e., domains of applicability, out of which it can at best be an approximation [128]. We will come back onto this point later.

Entropies play a crucial role in statistical mechanics [48]. In the last decades much attention has been devoted to this subject [27]. The concept of entropy is fundamental in the foundation of statistical physics. It first appeared in thermodynamics through the second law of thermodynamics. Both Boltzmann and Gibbs entropies are, in fact, the pillars of statistical mechanics and are the basis of all the entropy concepts in modern physics. Mathematical analysis and practical applications of both Boltzmann and Gibbs entropies have been used [142].

Boltzmann's Statistical Mechanics is particularly famous for Boltzmann's statistical interpolation of the second law of thermodynamics. The celebrated Boltzmann principle is $S = k\ln(W)$, where k is a thermodynamic unit of measurement of entropy and is known as Boltzmann constant ($k = 1.33 \cdot 10^{-23}\frac{J}{K}$), and W, called thermodynamic probability or statistical weight, or "degree of disorder', which is the total number of microscopic complexions compatible with the macroscopic state of the system. We avoid the name thermodynamic probability for the term W as it leads to many confusions [23].

Boltzmann's Statistical Mechanics is based on the Boltzmann–Gibbs entropy, which has also been defined fruitfully by Shannon as follows: $S_{BG} = -k\sum_{i=1}^{W} p_i ln(p_i)$, with $\sum_{i=1}^{W} p_i = 1$, that provides exponential laws for describing stationary states

and basic time–dependent phenomena, where $\{p_i\}$ are the probabilities of the microscopic configurations, and $k > 0$. This form is known to be the correct entropic form for ergodic systems [36].

Boltzmann entropy plays a basic role in the connection of the nonmechanical science of thermodynamics with mechanics through statistics. In thermodynamics, two fundamental properties of Boltzmann entropy are (i) its nondecrease: if no heat enters or leaves a system, its entropy cannot decrease; i.e the entropy S of the system is a monotonic increasing function of "degree of disorder' W, that is, $S(W) \leqslant S(W+1)$. (ii) its additivity: the entropy of two independent systems, taken together, is the sum of their separate entropies, i.e the entropy S is assumed to be an additive function for two statistically independent systems with degrees of disorder W1 and W2, respectively. The entropy of the composite system is given by $S(W1 \cdot W2) = S(W1) + S(W2)$.

Boltzmann entropy plays an important role not only in the foundation of statistical mechanics, but also in the other branches of science, which will be described briefly in next sections.

5.3 Statistical Mechanics and Neural Networks

Statistical mechanics sets out to explain the behaviour of macroscopic systems by studying the statistical properties of their microscopic constituents. A macroscopic phenomenon of the image of a face of a person can be a smile. The microscopic phenomenon is reflected in the correlation between the pixels in the image. Macroscopic phenomena are related to form. Clouds are another example of how microscopic or molecular forces have a macroscopic effect. The cloud is a large scale structure and the water molecules in it are on a small scale. Much of the research in neural networks is about explaining these properties.

One of major concepts in neural networks is the interaction between microscopic and macroscopic phenomena. Neural Networks are widely used in many classification applications such as pattern classification, speech recognition etc. Neural Networks are often used to classify or categorise. Once a representation of a group of persons is stored in the network, then the neural network can infer who looks sad or happy. The macroscopic properties in this case include the number of faces that the network can recognise, the speed with which this is done or the number of classes into which it can split up the faces (smile).

Statistical mechanical methods have successfully been applied to the study of neural network models of associative memory [3, 40, 54]. These models are biologically plausible and can be trained very quickly in some cases, compared with the popular neural networks proposed earlier (such as multi-layered perceptron), which have been shown to work satisfactorily. One of the models that constructs associative memories is the learning matrix proposed by Steinbuch in 1961 [116]. Some interesting properties of this model is that it can be able to store a number of input/output associations in one matrix, and also to generalise by recognising similar, but not identical, input patterns and producing the same output pattern. It has therefore the ability to produce the correct output from incomplete or corrupted input patterns. Some of learning matrices are described in [117]. Similar solutions are feedback networks that use a simple Hebbian learning and the training is achieved with a single calculation.

One of the most widely used models of associative memory is the Hopfield network [44]. In the Hopfield network, the weights are calculated using a matrix method procedure. This neural network model overcome some of the drawbacks that learning matrix has. However, this model of associative memory has still the drawback to stuck at local minimum. These local minima correspond many times to undesired stable states and therefore the outputs are incorrect. A possible cause of this fact is that these models use hard-limiters [100].

5.3.1 Annealing schedules in neural networks learning

Despite the fact that noise plays a influential role in the operation of real neurons, e.g. neural cells' responses to identical stimuli have been found to be stochastic in nature, the effect of noise on the operation of artificial neural networks has not been investigated in depth. One of the most famous neural networks model operating with noise is the Boltzmann machine, [1, 10]. It is inspired by the Boltzman–Gibbs entropy $S_{BG} = -k \sum_i p_i ln p_i$ that provides exponential laws for describing stationary states and basic time–dependent phenomena, where $\{p_i\}$ are the probabilities of the microscopic configurations, and $k > 0$. In addition, attempts to explore the benefits of introducing noise during learning, such as in [1, 21, 95], have been based on the use of Gaussian distributions.

In particular the use of Simulated Annealing has been explored for learning of the Boltzmann machine [1, 10]. Simulated Annealing (SA) refers to the process in which random noise in a system is systematically decreased so as to enhance the response of the system [52]. In the numerical optimisation framework, SA is a procedure that has the capability to move out of regions near local minima [24, 119]. SA is based on random evaluations of the objective function, in such a way that transitions out of a local minimum are possible. First, it reaches an area in the function domain space where a global minimiser should be present, following the gross behavior irrespectively of small local minima found on the way. It then develops finer details, finding a good, near–optimal local minimiser, if not the global minimum itself [57].

In the context of neural networks learning the performance of the classical SA is not the appropriate one: the method needs a greater number of function evaluations than usually required for a single run of first–order learning algorithms and does not exploit derivative related information. Notice that the problem of minimising a neural network's error function is not the well defined local minima but the broad regions that are nearly flat. In this case, the so–called Metropolis

move is not strong enough to move the algorithm out of these regions [143]. To alleviate this situation, [21] has suggested to incorporate an annealing schedule in the steepest descent algorithm:

$$w^{k+1} = w^k - \eta \nabla E(w^k) + \rho \cdot c \cdot 2^{-d \cdot k}, \tag{5.1}$$

where k is the iteration number, η is a common fixed stepsize for all weights, ρ is a constant controlling the initial intensity of the noise, $c \in (-0.5, +0.5)$ is a random number and d is the noise decay constant. This approach does not use the notion of the acceptance probability, such as the Metropolis algorithm in the classic SA [52], or the generalised acceptance probability in the generalised SA [127]. Instead, it implements a form of Langevin noise that has been proved quite effective in neural systems learning, [45, 95], and has motivated the development of other methods, such as the *Simulated Annealing Rprop*–SARprop and the SARprop with Restarts–ReSARprop [124].

5.3.2 Boltzmann's Statistical Mechanics and Neural Networks

Neural networks is one important domain that Boltzmann's Statistical Mechanics is applied. The Boltzmann machine seems to be the first multilayered learning machine inspired by statistical mechanics. It learns the statistical distribution of a data set, and can use this for tasks like pattern completion. The Boltzmann machine is a kind of neural network which behaves as a discrete time Hopfield network or as a stochastic network in which the neuron state is binary and decisions are taken probabilistically [1]. It can rigorously be established that the Boltzmann machine will probabilistically converge to a global optimum point. This neural model tries to overcome the Hopfield network's problem to settle at local energy minima rather than at the global minimum.

The Boltzmann machine [1, 10] is a probabilistic feedback network in which the

neurons are designed to mimic the particles in a thermodynamic system. In these systems the particles or neurons can be in either of two states. The probability of being in these states also obeys the Boltzmann distribution, hence the name of the network. The Boltzmann distribution for two states S_a and S_b leads to the next equation:

$$\frac{p_a}{p_b} = \exp^{-(E_a - E_b)/T} \tag{5.2}$$

where T is the temperature and E is the energy in each state. Generally, the aim of the Boltzmann machine is to move in direction of decreasing energy. Occasionally, it accepts a move that increases energy. This will be done with high probability at first steps, but lower probability as annealing progresses is occurred. This settles to a thermal equilibrium, and the global mimimum energy is reached.

The Boltzmann machine has many problems. Difficulties in the weights adjustments i.e how many weights are changed at time, calculation of the probabilities and the way to adjust the temperature during the simulated annealing procedure [99, 100]. However the main drawback of the Boltzmann machine is that the training is extremely slow [43].

An effort to overcome the problem with the training time has been done by introducing the Cauchy machine [119]. This model is based on the same idea with the Boltzmann machine but it uses different distribution function. It is claimed that this change in the probability function allows jumps to higher energy states more often, so that faster annealing can take place. The Cauchy distribution is defined as:

$$\frac{p_a}{p_b} = \frac{T}{T^2 + (E_a - E_b)^2} \tag{5.3}$$

5.4 Nonextensive Statistical Mechanics

Boltzmann's Statistical Mechanics have some limitations. Some kind of extension appears to become necessary [128]. Indeed, an everyday increasing list of physical anomalies are, here and there, being pointed out which defy (not to say that plainly violates) the standard Boltzmann-Gibbs (BG) prescriptions.

At this point, it is worthy to mention some important efforts to generalise the BG entropy. Renyi entropy is one of the important generalisations of the BG entropy [90, 142]. It is an extensive entropy for independent systems and defined as follows:

$$S_q^R \equiv (\ln \sum_i^W p_i^q)/(1-q) \tag{5.4}$$

where q is a continuous parameter.

Boltzmann's Statistical Mechanics is widely used for systems that are in stationary states characterised by thermal equilibrium consistent with ergodicity. Nonextensive Statistical Mechanics is an alternative which is proposed as a way of dealing with anomalous systems through mathematical methods [125, 36]. A Nonextensive thermostatistics, which recovers the extensive BG mechanics as a particular case was proposed in 1988, by Tsallis, which might correctly cover at least some of the known anomalies [125]. Some anomalous systems are considered to be nonergodic systems with stationary states that are metastable and long-lived. Nonextensive Statistical Mechanics exhibits apparent success for certain closed systems as well as in many open systems in biology, economics and other fields.

The Nonextensive Statistical Mechanics are based on the Tsallis entropy. In particular, Tsallis has defined the generalised entropy [125]:

$$S_q \equiv k \, \frac{1 - \sum_{i=1}^W p_i^q}{q-1} \quad (q \in \mathbb{R}), \tag{5.5}$$

where W is the total number of microscopic configurations, whose probabilities

are $\{p_i\}$, and k is a conventional positive constant. When $q = 1$ it reproduces the S_{BG} entropic form.

The nonextensive entropy S_q achieves its extreme value at the equiprobability $p_i = 1/W, \forall i$ and this value is equal to $S_q = \dfrac{W^{1-q} - 1}{1 - q}$ [125, 36]. Another important property is the nonextensivity of this entropic form. For independent systems, it is subextensive for $q > 1$, superextensive for $q < 1$ and while $q = 1$ recovers to BG entropy, which is extensive [36].

Recently another generalisation of the BG form is the normalised nonextensive entropy independently introduced in [55, 89]. The form of this entropy is given below:

$$S_q^N \equiv (1 - [\sum_i^W p_i^q]^{-1})/(1 - q) \tag{5.6}$$

In this stage it is necessary to briefly describe three remarkable properties. The first one is the concavity, which is related to thermodynamic stability or robustness concerning the fluctuations of energy and other quantities. The second property is the stability or continuity that is the experimental robustness, i.e similar experiments should provide quantitatively similar results. Finally the finiteness of the entropy that characterises the gradual exploration of the available phase space is the third property. It is important to remark that for $q > 0$ both of S_q and S_{BG} entropy satisfy the three above important properties, while the Renyi and S_q^N entropy violates all three [36].

Next we give the relationship between the previously defined entropies and the Tsallis entropy, which will be used in the rest of this chapter and in our applications. Renyi entropy is related through a monotonic function with the nonextensive entropy S_q:

$$S_q^R \equiv (\ln \sum_i p_i^q)/(1 - q) = \ln[1 + (1 - q)S_q]/(1 - q) \tag{5.7}$$

For $q = 1$ the S_{BG} is represented by S_1, or S_1^R, or finally by S_1^N. Finally the Normalised entropy has a strong relationship with S_q by:

$$S_q^N \equiv (S_q)/(\sum_i p_i^q) = S_q/[1 + (1-q)S_q] \tag{5.8}$$

The S_q, S_q^R, S_q^N entropies have in common the optimising distribution, under the same conditions. All three entropies depend on $\sum_i p_i^q$, hence any of them could be expressed as a function of the other two. All of them lead to the same q exponential optimising distribution [36].

Nowadays the idea of nonextensivity has been used in many applications. Nonextensive statistical mechanics has successfully been applied in physics (astrophysics, astronomy, cosmology, nonlinear dynamics etc) [107, 120, 108], chemistry [103], biology [131], economics [129], computer sciences and other important sciences [36]. Further discussion about the nonextensivity in Artificial Intelligence domain is given in next section.

5.4.1 Nonextensive Statistical Mechanics and Neural Networks

The problem of finding the global minimum of a complex cost function, which has a large number of local minima, is very difficult [43, 97]. A variety of global optimisation algorithms have been introduced over the years to overcome this problem. As already said, one of the most popular methods is the Simulated annealing. It uses BG statistics at two different steps, namely at the *visitation step*, which uses a Gaussian distribution, and at the *acceptance step*, that uses the Boltzmann factor. Nonextensive Statistical Mechanics is applied in the Simulated annealing by generalising both the Gaussian distribution and the Boltzmann factor. Empirical results show an improvement in the speed, the precision and

in the success rate [127]. In what follows we discuss about nonextensive entropy that have successful results in training feedforward neural networks.

The next section will introduce an adaptive search strategy that aims to alleviate the problem of occasional convergence to local minima in supervised training. Our approach adapts the weights using only information from the sign of a gradient vector, which is calculated on a perturbed error function, and uses adaptive steps along each weight directions. The perturbations are generated from a noise sources that replaces the usual Boltzmann–Gibbs factor used in annealing schedules by the *q–exponential function* of the generalised entropy of nonextensive statistical mechanics [125, 126].

5.4.2 Nonextensive Entropy and the Perturbed Error Function

In general, additive noise can be introduced in neural network learning by formulating the *perturbed* energy function:

$$\tilde{E}(w,x) = E(w^k) + c^k \cdot \sum_{i=1}^{n} w_i^k x_i^k, \tag{5.9}$$

where k indicates iterations, $E(w)$ is given by (2.2), $x = (x_1, \ldots, x_n)^\top$ defines a vector of independent noise sources, and c is a parameter that regulates the influence of the noise.

In this approach noise is generated by a noise source that is characterised by the nonextensive entropic index q. The optimisation of the entropic form (5.5) under appropriate constraints, [125], yields for the canonical ensemble

$$p_i \propto [1-(1-q)\beta E_i]^{\frac{1}{(1-q)}} \equiv e_q^{-\beta E_i}, \tag{5.10}$$

where β is a Lagrange parameter, $\{E_i\}$ is the energy spectrum, and the *q-exponential function* is defined as:

$$e_q^x \equiv [1 + (1-q)x]^{\frac{1}{(1-q)}} = \frac{1}{[1-(q-1)x]^{\frac{1}{(q-1)}}} \tag{5.11}$$

Following the above discussion and inspired by [21, 127], in this method, noise is generated according to a schedule that can be expressed as

$$e_q^{-T(\ln 2)\cdot k} = [1-(1-q)T(\ln 2)\cdot k]^{\frac{1}{1-q}}, \tag{5.12}$$

where T is the temperature; k indicates iterations. In this approach, noise is not applied proportionally to the size of each weight; instead a form of weight decay is used, which is considered beneficial for achieving a robust neural network that generalises well [39, 124]. Thus, noise is introduced in neural network learning by formulating the *perturbed* energy function:

$$\tilde{E}(w^k) = E(w^k) + \mu \cdot \sum_{i=1}^{n} \frac{\left(w_i^k\right)^2}{\left[1+\left(w_i^k\right)^2\right]} \cdot [1-(1-q)T(\ln 2)\cdot k]^{\frac{1}{1-q}}, \tag{5.13}$$

where k indicates iterations, $E(w)$ is given by (2.2), $\sum_i w_i^2/(1+w_i^2)$ is the weight decay bias term which can decay small weights more rapidly than large weights, and μ is a parameter that regulates the influence of the combined weight decay/noise effect. This form of weight decay modifies the energy landscape so that smaller weights are favored at the beginning of the training but as learning progresses the magnitude of the weight decay is reduced to favor the growth of large weights. Thus, as the energy landscape is modified during training the search method is allowed to explore regions of the energy surface that were previously unavailable. Minimisation of (5.13) requires calculating the gradient of the energy term with respect to each weight

$$\tilde{g}_i(w^k) = g_i(w^k) + \mu' \cdot \frac{w_i^k}{\left[1 + (w_i^k)^2\right]^2} \cdot [1 - (1-q)T(\ln 2) \cdot k]^{\frac{1}{1-q}}, \qquad (5.14)$$

where $\mu' > 0$ (in our experiments, reported in the next section, a fixed value of $\mu' = 0.01$ was used).

The Formulation of the New Method

The proposed hybrid strategy applies the sign–based weight adjustment of Rprop, defined by Relation (4.4), on the perturbed energy function (5.13) using the gradient term of Equation (5.14). Also the learning rates are adapted by means of Conditions (4.10)–(4.12), where $\tilde{g}_i(w^k)$ is used instead of $g_i(w^k)$. Lastly, an additional condition is introduced in order to avoid using relatively small weight adjustments

$$\begin{aligned} &if \quad \left(\eta_i^{k-1} < \rho \cdot [1 - (1-q)T(\ln 2) \cdot k]^{\frac{2}{1-q}}\right) \\ &then \quad \eta_i^k = max\left(\eta_i^{k-1} \cdot \eta^- + 2c\rho \cdot [1 - (1-q)T(\ln 2) \cdot k]^{\frac{2}{1-q}}, \Delta_{min}\right) \end{aligned} \qquad (5.15)$$

where $0 < \rho < 1$ and $c \in (0, 1)$ is a random number.

Below, a simple problem is used to visualise the behavior of the *Hybrid Learning Scheme*–(HLS) for different values of T keeping q fixed. It is a single node with two weights and a logistic activation function. The error landscape of Figures 5.2, 5.1 and Figure 5.3 has a global minimum and two local minima.

Figure 5.1 shows that under the same initial conditions, HLS escapes a saddle point and a valley that leads to a local minimum, and converges to the global minimiser located at the center of the contour plot (Figure 5.1, left), while Rprop converges to the local minimiser (Figure 5.1, right). The top rows of Figures 5.2 and 5.3 show that Rprop converges to the local minimiser from two different

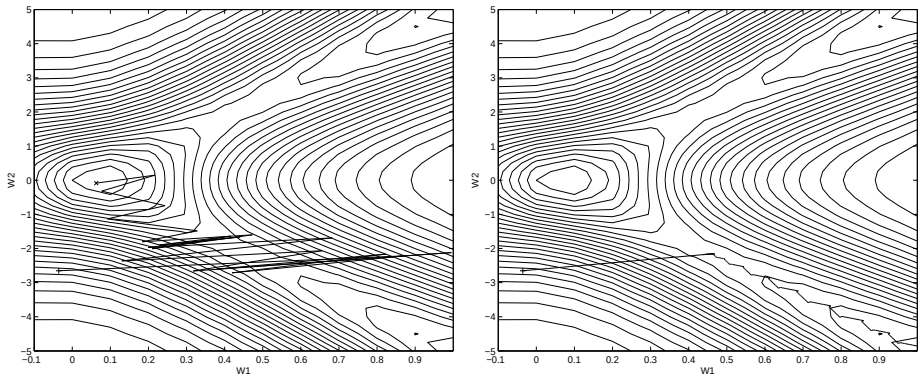


Figure 5.1: The weights trajectory of the Hybrid Learning Scheme converges to the global minimum (left), whilst the trajectory of Rprop to a local minimiser (right).

initial weights. We have applied the adaptive learning scheme with $q = 1.2$ for the following temperatures: $T = 0.01$ (Figures 5.2 and 5.3, second row), $T = 0.001$ (Figures 5.2 and 5.3, third row), $T = 0.0001$ (Figures 5.2 and 5.3, bottom). The HLS escapes the region around the local minimum, and converges to the global minimiser located at the center of the contour plot in all cases. The value of T influences the shape of the HLS trajectory. Small values generate more stochastic paths, while larger values lead to more deterministic behavior. Best results for this problem are achieved by setting $T = 0.01$.

5.5 Experimental Study

In this section, we evaluate the performance of the HLS and compare it with the Rprop and the SARprop algorithms. SARprop introduced by Treadgold and Gedeon[124], tries to solve the problem of poor local minima. It attempts to address this problem by using the method of Simulated Annealing (SA) [57],[119].

We have used well–studied problems from the UCI Repository of Machine Learning Databases of the University of California [72], as well as problems studied

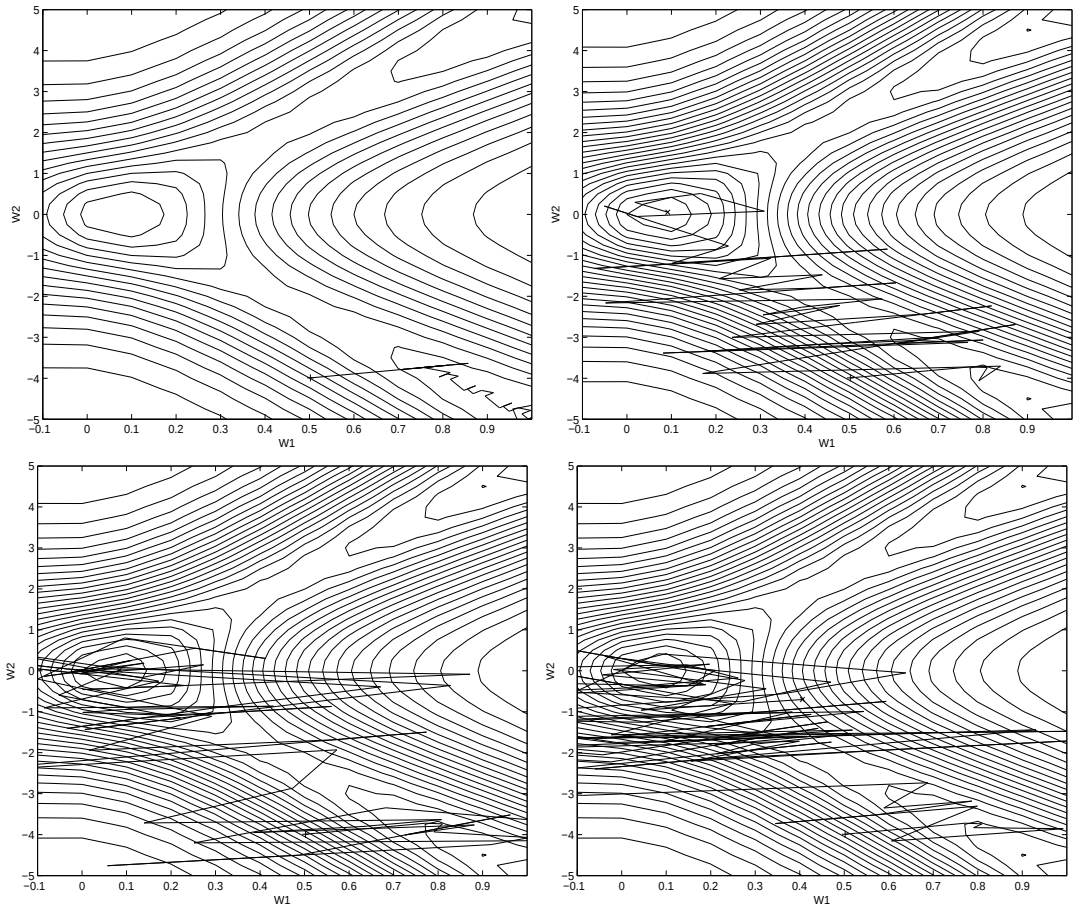


Figure 5.2: Starting from the same initial weights, the trajectory of the Rprop converges to a local minimiser (top), whilst the trajectory of HLS converges to the global minimum (3 different values for the Temperature are shown – see text for details).

extensively by other researchers in an attempt to reduce as much as possible biases introduced by the size of the weights space. In all cases we have used networks with classic logistic activations. The guidelines of [91] and [124] were adopted for setting the learning parameters of Rprop and SARprop respectively. The parameters of the HLS were set to the same values for all experiments in an attempt to test the robustness of the method in different types of problems: the entropic index $q = 1.2$ and the temperature $T = 0.1$.

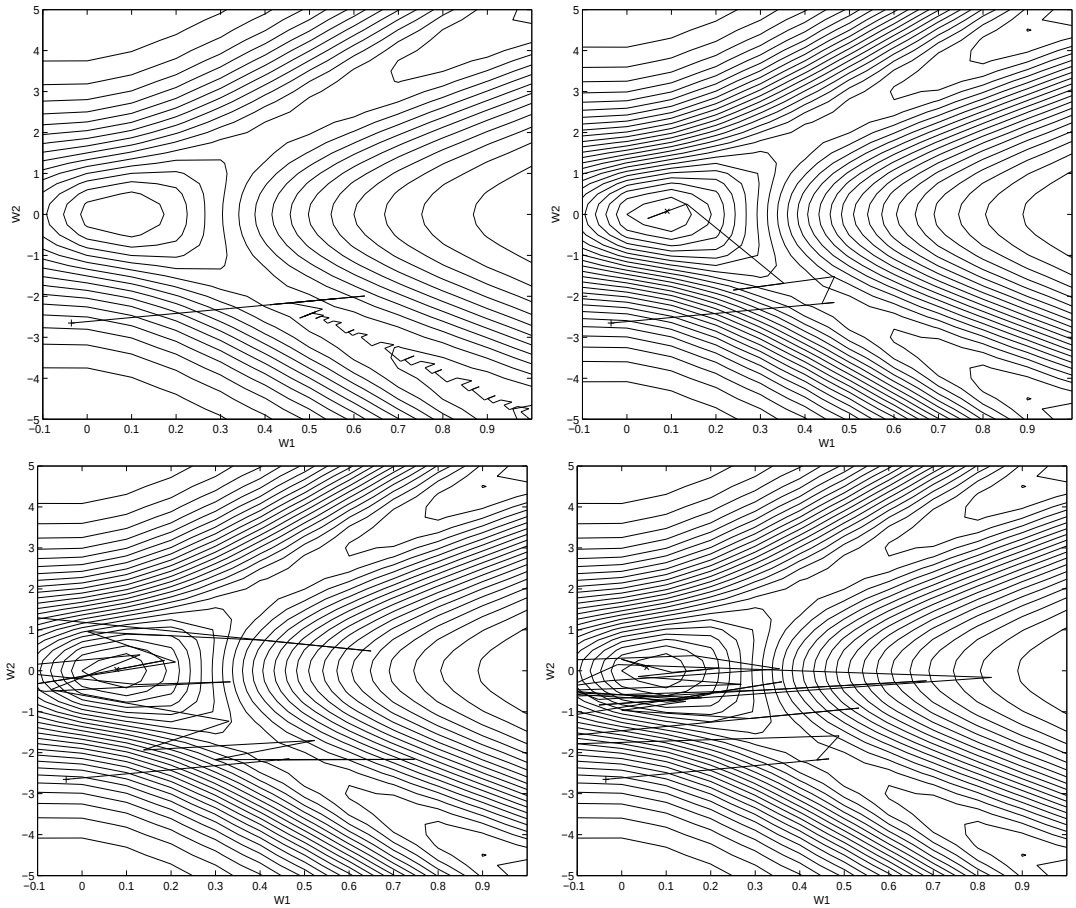


Figure 5.3: Starting from the same initial weights, the trajectory of the Rprop converges to a local minimiser (top) , whilst the trajectory of HLS converges to the global minimum (3 different values for the Temperature are shown – see text for details).

5.5.1 Fisher's Iris data, a benchmark problem

The first benchmark is known as the *Fisher's Iris* problem [72, 88]. The data set consists of 120 examples and the test set of 30 examples. Following [124], an 4–2–3 FNN (4 input–2 hidden–3 output nodes; 19 weights overall) was used, and the maximum number of iterations to find a "near–optimal" weight configuration (defined as a weight set w^* that results to an error function value $E(w^*) \leq 0.01$) was set to 2000. Table 5.1 shows the average performance in terms of: iterations to converge to the error target (*Epochs*), success of convergence to the

target, within 2000 epochs (*Convergence*, out of the 1000 runs), and generalisation (*Generalisation*, percentage of correctly classified test examples); a "+" indicates statistical significance of the HLS results over another method).

Table 5.1: Comparison of algorithms performance in the Iris problem for the converged runs

Iris				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	2556 (+)	7.8 ± 5.0 (+)	97.2 (+)	74 (+)
SARprop	1591 (+)	4.9 ± 4.0 (+)	99.0 (+)	96 (+)
HLS	1175	3.9 ± 1.8	99.7	99

Table 5.2 presents the number of 1000 runs in which each algorithm performs better than the other methods with respect to training speed and generalisation. Both SARprop and HLS outperform comparing to Rprop algorithm in terms of speed and success. However the new learning stochastic scheme achieves the best performance.

Table 5.2: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Iris problem with respect to training speed and generalisation.

Iris	**Times faster algorithm**			**Times better Generalisation**		
Algorithm	Rprop	SARprop	HLS	Rprop	SARprop	HLS
Rprop	–	192	106	–	95	107
SARprop	800	–	437	161	–	97
HLS	894	562	–	166	120	–

5.5.2 Application to biological data

Below, we report results from 1000 independent trials for four UCI problems. These 1000 random weight initialisations are the same for the three learning algorithms, and the training and testing sets were created according to *Proben1* [88].

It is also investigated the performance of the tested algorithms in some well known Boolean function approximation problems, which characterised of the existence of many local minima. In this class of problems 1000 independent runs are applied. Next comparative results are given.

The statistical significance of the results has been analysed using the Wilcoxon test [114]. This is a nonparametric method that is considered an alternative to the paired t–test. It assumes there is information in the magnitudes of the differences between paired observations, as well as the signs. All statements in the tables reported below refer to a significance level of 0.05.

Cancer problem

The second benchmark is the *breast cancer diagnosis* problem which classifies a tumor as benign or malignant based on 9 features [72, 88]. We have used an FNN with 9–4–2–2 nodes (a total of 56 weights) as suggested in [88]. The comparative results are shown in Table 5.3. The new proposed scheme affects positive to meet fast the error goal. The SARprop algorithm improves the convergence success compared to Rprop but the application of the nonextensive term in the learning procedure increase significantly the convergence success.

Table 5.3: Comparison of algorithms performance in the Cancer problem for the converged runs

Cancer				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	280 (+)	1.85 ± 1.30 (+)	97.0 (+)	94 (+)
SARprop	250 (+)	1.60 ± 1.70 (+)	97.6 (−)	98 (−)
HLS	141	0.95 ±0.26	97.5	100

Table 5.4 shows more analytically the performance of the tested algorithms within 1000 runs. The 790 out of 1000 runs that HLS is faster than Rprop and 740 times better than SARprop highlights the increased learning speed of the new training

algorithm.

Table 5.4: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Cancer problem with respect to training speed and generalisation.

Cancer	**Times faster algorithm**			**Times better Generalisation**		
Algorithm	Rprop	SARprop	HLS	Rprop	SARprop	HLS
Rprop	–	394	208	–	411	500
SARprop	598	–	249	594	–	584
HLS	790	740	–	511	431	–

Diabetes problem

The *diabetes1* benchmark is a real-world classification task which concerns deciding when a Pima Indian individual is diabetes positive or not [72, 88]. There are 8 features representing personal data and results from a medical examination. The Proben1 collection suggests a 8–2–2–2 FNN (34 weights overall). The termination criterion is $E \leq 0.1$ within 2000 iterations.

Judging from the table 5.5 is obvious that the Rprop algorithm converges many times in local minima. The new stochastic learning algorithm overcomes this problem in the most cases. Its convergence success is 97% while SARprop has 92% and Rprop 86%. Furthermore the HLS is the fastest algorithm and improves significantly the Generalisation success compared to Rprop. Table 5.6 shows analytically view of the algorithms' performance.

Table 5.5: Comparison of algorithms performance in the Diabetes problem for the converged runs

Diabetes				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	455 (+)	2.40 ± 2.1 (+)	75.8 (+)	86 (+)
SARprop	410 (+)	2.25 ± 1.8 (+)	76.2 (−)	92 (+)
HLS	276	1.50 ± 1.1	76.4	97

Table 5.6: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Diabetes problem with respect to training speed and generalisation.

Diabetes	**Times faster algorithm**			**Times better Generalisation**		
Algorithm	Rprop	SARprop	HLS	Rprop	SARprop	HLS
Rprop	–	301	301	–	508	530
SARprop	696	–	478	600	–	590
HLS	692	518	–	580	520	–

Thyroid problem

Lastly, the *thyroid1* problem, [72, 88], uses a 21–4–3 nodes FNN, suggested by [124], to decide whether the patient's thyroid has over function, normal function, or under function. A data set with 3600 examples is used and the target is to find within a maximum of 2000 iterations a weight set that produces $E \leq 0.0036$. Table 5.7 gives the average performance of the three algorithms in the two problems. The new method outperforms the other methods in the number of iterations required to reach a suitable solution, and converges in all cases.

It is important to highlight in table 5.8 the number of times that HLS is faster than Rprop and SARprop (660 and 1000 times respectively). However the HLS is still better in terms of Generalisation and Convergence success than Rprop and SARprop.

Table 5.7: Comparison of algorithms performance in the Thyroid problem for the converged runs

Thyroid				
Algorithm	Epochs	Time (secs)	Generalisation (%)	Convergence (%)
Rprop	770 (+)	24.20 ± 12.1 (+)	97.9 (+)	80 (+)
SARprop	810 (+)	29.75 ± 13.1 (+)	98.1 (−)	90 (+)
HLS	276	18.10 ± 4.1	98.2	97

Table 5.8: Number of times, out of 1000 runs, each algorithm performs better than the other methods in the Thyroid problem with respect to training speed and generalisation.

Thyroid	**Times faster algorithm**			**Times better Generalisation**		
Algorithm	Rprop	SARprop	HLS	Rprop	SARprop	HLS
Rprop	–	770	340	–	85	75
SARprop	230	–	0	101	–	97
HLS	660	1000	–	121	110	–

5.5.3 Boolean function approximation problems

Another set of experiments has been conducted to empirically evaluate the performance of the new method in a well–studied class of boolean function approximation problems that exhibit strong local minima [17, 38]. This class includes the XOR problem (whose local minima and saddle points have been analysed in detail) and the various parity–N problems, which are considered as classic benchmarks [63, 86, 124, 135]. The error target was set to $E \leq 10^{-7}$ within 2000 iterations in all cases (this is considered low enough to guarantee convergence to a "global" solution, especially for the XOR problem), and the adopted architectures were 2–4–1 for the XOR, 3–3–1 for the parity–3, 4–6–1 for the parity–4, 5–7–1 for the parity–5, following the recommendations of [124]. The results are presented in Table 5.9 and Table 5.10. Figure 5.5 gives a typical example of algorithms' convergence. Starting from the same initial conditions, the Rprop

converges to a local minimiser, whilst both SARprop and HLS escape from the local minimum. However, HLS converges to a feasible solution much faster than SARprop.

Table 5.9: Average performance in the XOR and Parity–4 problems

	XOR		**Parity4**	
Algorithm	Epochs	Convergence	Epochs	Convergence
Rprop	1110 (+)	23 (+)	1360 (+)	42 (+)
SARprop	150 (+)	75 (+)	1378 (+)	48 (+)
HLS	69	88	1270	80

Table 5.10: Average algorithm performance in the Parity–3 and Parity–5 problems

	Parity3		**Parity5**	
Algorithm	Epochs	Convergence	Epochs	Convergence
Rprop	1105 (+)	22 (+)	416 (+)	67 (+)
SARprop	882 (+)	58 (+)	394 (+)	80 (+)
HLS	640	78	20	90

Additional experiments have been performed to explore the influence of the entropic index q on the convergence speed of the HLS. According to our experiments, large values of q cause an increase to the average number of iterations required to achieve the error target but seems not to affect the convergence success. In the XOR problem, for example, the HLS requires on average 378 iterations to converge when using $q = 1.7$ and $T = 0.1$ (cf. with Table 5.9, where 69 iterations are required with $q = 1.2$ and $T = 0.1$). The HLS exhibits similar behavior in the parity–5 problem, where an average of 440 iterations is required when $q = 1.7$ and $T = 0.1$ (cf. with Table 5.10 where $q = 1.2$ and $T = 0.1$). In all tables the results are based on $q = 1.2$ and the temperature $T = 0.1$.

A typical run for the XOR problem, for the Rprop method, SARprop and the HLS is shown in Figure 5.4. As we can notice from Figure 5.4, starting with the same initial weights and learning parameters in the XOR problem, Rprop got stuck

in a local minimum with higher error function value while the Hybrid Learning Scheme (HLS) successfully converges to a feasible solution ($E(w) \leqslant 10^{-16}$) after 200 iterations and the SARprop approximately in 400 iterations.

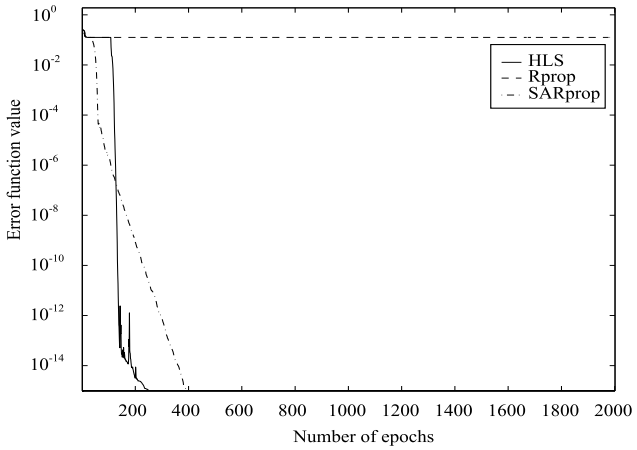


Figure 5.4: Typical learning error curve for the XOR function

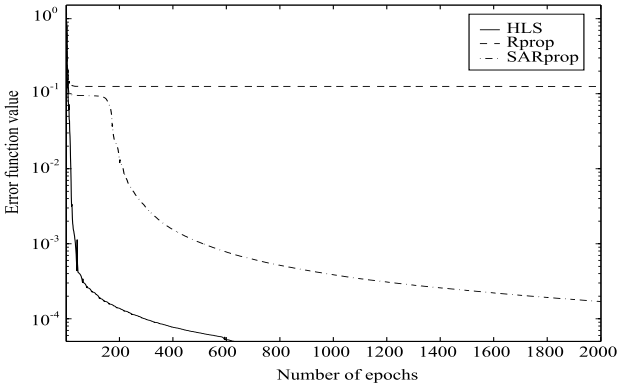


Figure 5.5: Typical learning error curve for the Parity–3 problem

Finally, Figure 5.5 highlights the performance of the HLS, which converges fast to the target point while the other tested algorithms never meet the optimal solution.

5.6 Summary and Contribution of the Chapter

The proposed hybrid learning scheme builds on ideas from global search methods. In general, global search methods are expected to lead to "optimal" or "near-optimal" weight configurations by allowing the network to escape local minima during training. It is worth noting that global search algorithms possess strong convergence properties, and, at least in principle, are straightforward to implement and apply. Issues related to their numerical efficiency are considered by equipping global optimisation algorithms with a "traditional" local minimisation phase.

Global convergence, however, needs to be guaranteed by the global–scope algorithm component which, theoretically, should be used in a complete, "exhaustive" fashion. These remarks indicate the inherent computational demands of the global optimisation algorithms, which increases non–polynomially, as a function of problem–size, even in the simplest cases [102]. To alleviate this situation hybrid schemes for neural networks learning have been developed in an attempt to achieve improved convergence rates compared to the standard global optimisation, and in some cases even maintains the guarantee of convergence to a global minimiser [12, 21, 115].

The proposed approach belongs to the special class of adaptive training algorithms that employ a different adaptive stepsize for each weight. Algorithms of this class avoid slow convergence in the flat directions and oscillations in the steep directions, and exploit the parallelism inherent in the evaluation of learning error $E(w)$ and gradient $\nabla E(w)$ by the Back-Propagation (BP) algorithm [97]. Various algorithms of this class have been suggested in the literature, such as [66, 81, 82, 91]. Among them the *Resilient Propagation* (Rprop) algorithm is one of the most popular methods [91].

In this chapter a new hybrid learning scheme that combines deterministic and

stochastic search by employing a different adaptive stepsize for each weight, and a form of noise that is characterised by the nonextensive entropic index q that is regulated by a weight decay term were discussed. This allows to modify the error surface during training so that exploration of new regions of the error landscape is achieved. It is important to mention that the behavior of the learning scheme can be more stochastic or deterministic depending on the trade off between T and q.

Experiments with the hybrid scheme, and comparisons with two other popular learning methods, namely the Rprop and the SARprop, were very encouraging: accelerated and reliable neural learning was achieved in all cases tested. In next chapter, the performance of the HLS in two bioinformatics problems is presented. Finally, an investigation of the FNNs and Ensemble FNNs performance trained with the HLS algorithm has been done.

Chapter 6

Training Neural Network Ensembles in Bioinformatics problems

6.1 Introduction

Scientists, involved in the area of proteomics, are currently seeking integrated, customised and validated research solutions to better expedite their work in proteomics analyses and drug discoveries. Some drugs and most of their cell targets are proteins, because proteins dictate biological phenotype. In this context, the automated analysis of protein localisation is more complex than the automated analysis of DNA sequences; nevertheless the benefits to be derived are of same or greater importance. In order to accomplish this target, the right choice of the kind of the methods for these applications, especially when the data set is drastically imbalanced, is very important and crucial.

In this chapter, the performance of some commonly used classifiers was investigated, such as the K nearest neighbours, and a feed-forward neural network

with and without applying cross validation, in a class of imbalanced problems from the bioinformatics domain. Furthermore, ensemble-based schemes were constructed using the notion of diversity, and empirically their performance was tested on the same problems. The experimental results favour the generation of neural network ensembles as these are able to produce good generalisation ability and significant improvement compared to other existing single classifier methods. Finally, a recently proposed training algorithm is applied in Feed-forward Neural Networks and in ensemble diversity neural networks. The new hybrid learning scheme HLS described in the previous chapter shows an increased generalisation and convergence success compared to the other recently proposed algorithms such as GRprop,JRprop and GJRprop. In this specific class of the problems tested in this chapter, I am interested in improving the generalisation of the FNNs and ensembles neural networks. The stochastic nature of the HLS is important and helpful to create efficient ensembles. Therefore, a section based on this idea is further investigated in this chapter and comparative results are given. In this chapter investigation of several supervised learning schemes is also done [98, 91, 92], in order to improve the classification success of neural networks. Furthermore the methods that were used to predict the protein localisation sites are described. Lastly, the experimental results and comparisons are presented.

6.2 Description of the Problem and Related Works

The ability to identify known proteins with similar sequence and similar localisation is becoming increasingly important, as structural, functional and localisation information is needed to accompany the raw sequences. In particular, the study of protein localisation (in order to function properly, proteins must be transported to various locations within a particular cell) is considered very useful in the post-genomics and proteomics era, as it provides information about each protein that is complementary to the protein sequence and structure data [15].

Two of the most thoroughly studied single-cell organisms are the bacterium Escherichia coli and the eukaryote Saccharomyces cerevisiae, also called Yeast. Both organisms use mainstream metabolic pathways that are recognisably similar to the corresponding metabolic functions in all life forms including higher eukaryotes. The entire genome sequence has been determined for both organisms. The relations between genetics and biochemistry that constitute the fundamental processes of life in these single-cell organisms, serve as a foundation for ongoing investigations on the processes that operate in the more complex, higher forms of life [58]. E.coli and Yeast are the last characterised organisms as they can be very easily manipulated. They have rapid growth rate and very simple nutritional requirements. Many applications have been done to analyse the gene expression of these proteins. Recently a neuro-fuzzy approach for functional genomics has been proposed. More precicely the objective of this approach was to learn and predict the fuctional classes of the E.coli genes [76]. In this study, I am interested in a different problem, which is the prediction of the localisation sites of proteins, such as the E.coli and the Yeast (S. cerevisiae).

The first approach for predicting the localisation sites of proteins from their amino acid sequences was an expert system developed by Nakai and Kanehisa [73, 74]. Later, expert identified features were combined with a probabilistic model, which could learn its parameters from a set of training data [46]. Better prediction accuracy has been achieved by using standard classification algorithms such as K nearest neighbours (KNN), binary decision trees, and nave Bayesian classifiers. The KNN achieved the best classification accuracy compared to these methods [47]. E.coli proteins were classified into 8 classes with an average accuracy of 86%, while Yeast proteins were classified into 10 classes with an average accuracy of 60% by applying cross validation. More recently, attempts to improve the classification success have been made using back-propagation neural networks, genetic algorithms, growing cell structures, and expanding range rules, but no significant improvements over the KNN algorithm were reported

[22]. A data selection method for Probabilistic Neural Networks (PNN) was also applied to E.coli dataset achieving better performance than the KNN (90% accuracy) [20]. Lastly, an empirical study showed that combined methods can achieve better performance than individual ones [2].

6.2.1 The Horton-Nakai Model

The first method, which has been specifically designed for the protein localisation problem [46], is a probabilistic model, referred to as the Horton-Nakai (HN) model. The HN model is consisted of a rooted binary tree of classification variables. Each non-leaf node of the binary tree is a feature variable. The leaves of the tree are the possible classes that a new pattern is going to be classified. A non-leaf node n represents all the classes that belong to leaves that are descendants of n. Each node has a probability associated with it. The probability of n being true represents the probability that an object belongs to n class. In these experiments, it used the version of the HN model that employs sigmoid conditional probability functions, as those exhibited gave the best results in previously published studies [46].

6.2.2 The K Nearest Neighbours Algorithm

The K Nearest Neighbours is a simple and effective classification algorithm. It is widely used in machine learning and has numerous variants. Given a test sample of unknown labels, it finds the K nearest neighbors in the training set and assigns the label of the test sample according to the labels of these neighbors. The vote from each neighbor is weighted by its rank in terms of the distance to the test sample.

In a more formal way, the function of the KNN algorithm can be expressed as

follows: let $X = x^i = (x_1^i, \ldots, x_d^i), i = 1, \ldots, N$ be a collection of d-dimensional training samples and $C = C_{1,\ldots,C_M}$is a set of M classes. Each sample x^i will first be assumed to possess a class label $L_i \in 1, \ldots, M$ indicating with certainty its membership to a class in C. Assume also that x^s is the incoming sample to be classified. Classifying x^s corresponds to assigning it to one of the classes in C, i.e. deciding among a set of M hypotheses: $x^s \in C_q$, $q = 1, \ldots, M$. Let Φ^s be the set of the K nearest neighbours of x^s in X. For any $x^i \in \Phi^s$, the knowledge that $L_i = q$ can be regarded as evidence that increases our belief that x^s also belongs to one of the classes of C. However, this piece of evidence does not provide certainty by itself .

The KNN method requires selecting a distance metric and choosing a value for parameter K. The KNN, as suggested by Duda and Hart [26], stores the training data, denoted as a pair (X, L), and classifies new sample to the majority class among the K closest examples in the training data. This is usually done by calculating the Euclidean distance measure. The Euclidean distance D, between the point x and the prototype p^i of the d-dimensional training samples, is given by the following equation:

$$D^i = \| x - p^i \|, \tag{6.1}$$

where $\|\|$ denotes the euclidean norm.

One of the drawbacks of KNN algorithm is that it needs to compare a test sample with all samples in the training set. In addition, the performance of this algorithm greatly depends on the appropriate choice for the parameter K. The K Nearest Neighbor (KNN) based classification techniques are very popular in the biological domain because of their simplicity and their ability to capture sequential constraints present in the sequences. In order to classify a test sequence, the KNN first locates K training sequences, which are most similar to the test sequence. It then assigns the class label that most frequently occurs among those K sequences (majority function) to the test sequence. The key component of the KNN classifier is the method used for computing the similarity between the two

sequences.

6.2.3 The PSORT System

The PSORT system (http://psort.ims.u-tokyo.ac.jp/) is a tool for the prediction of protein subcellular localisation in a sense that it can deal with proteins localised at almost all the subcellular compartments. The last version of PSORT is a widely used computational method to predict the subcellular localisation sites of proteins from their amino acid sequences. The reasoning algorithm is the K Nearest Neighbors classifier. It is used to assess the probability of localising at each candidate sites (Horton and Nakai, 1997) [47]. For each query protein, such as the Gram-positive or Gram-negative or eukaryotic proteins, the output values of the subprograms for these proteins are normalised and simple Euclidean distances to all of the data points contained in the training data are calculated. Then, the prediction is performed using the K Nearest neighbor, where K is a predefined integer parameter. For example if these K Nearest neighbor contain 50% nuclear proteins then the query is predicted to be localised to the nucleus class with a probability of 50%.

6.3 Ensemble-based Methods

Ensemble based methods enable an increase in generalisation performance by combining several individual neural networks trained on the same task. The ensemble approach has been justified both theoretically [42, 53] and empirically [78]. The creation of an ensemble is often divided into two steps [104]; the first is to generate individual ensemble members and the second to appropriately combine individual members outputs to produce the output of the ensemble. The simplest method for creating ensemble members is to train each member network using

randomly initialised weights. A more advanced approach is to train the different networks on different subsets of the training set as Bagging [18] does, where each training set is created by resampling and replacement of the original one with uniform probability. Boosting [33] also uses resampling of the training set, but the data points previously poorly classified, receive a higher probability.

Finally, there is a class of methods for creating ensembles that focuses on creating classifiers that disagree partially on their decisions. In general terms, these methods alter the training process in an attempt to produce classifiers that will generate different classifications. In the neural networks context, these methods include techniques for training with different network topologies, different initial weights, different learning parameters, and/or learning different portions of the training set (see [105] for a review and comparisons).

6.3.1 The Notion of Diversity and its Levels

Networks belonging to an ensemble are thought to be diverse with respect to a test set if they make different generalisation errors on that test set. Different patterns of generalisations can be produced when networks are trained either on different training sets, or from different initial conditions, or with different numbers or hidden nodes, or using different algorithms [105].

In 1997, Sharkey and Sharkey, [105], introduced the term levels of the diversity. They proposed four levels of diversity ranging from the best cases of Level 1 and Level 2 Diversity to the minimum diversity of Level 4. Level 1 Diversity requires more than two members in an ensemble and considers that for every test input there is always a member that produces the correct output. Level 2 Diversity corresponds to at least five members in an ensemble. It is possible diversity of Level 2 to lead to Level 1 Diversity by removing some of the members. An ensemble of Level 2 Diversity is also known as upwardly mobile and eliminates

coincident failures. However, in ensembles that belong to this level, the majority is always correct. The Level 3 Diversity may contain subsets of classifiers with either Level 1 or 2 Diversity and can be upwardly mobile. In this case, the correct output for each input pattern from a test set is always produced by at least one member. Finally, the Level 4 Diversity is equivalent to the minimal diversity that can be used to improve generalisation. This level can never be reliable since the ensemble members exhbit similar failures (see [105] for details).

6.3.2 Measuring Ensemble Diversity.

As it has already mentioned the measure of the diversity is an important factor to create ensembles that can achieve good generalisation performance. There are many ways to quantify the ensemble diversity, which usually associated with the particular error measure.

In the context of regression problems, Krogh and Vedelsby suggest to calculate the diversity as [53]:

$$d_i(p_k) = [A_i(p_k) - A^*(p_k)]^2, \tag{6.2}$$

where d_i is the diversity of the i^{th} classifier on pattern p_k, $A_i(p)$ is the i^{th} classifier prediction and $A^*(p)$ is the ensemble prediction. Then in this case, we obtain for the mean squared ensemble error the following equation:

$$E_{ens} = \bar{E} - \bar{D}, \tag{6.3}$$

where $\bar{D}$ is the average ensemble's diversity and $\bar{E}$ is the mean single classifier's errors.

Another measure of the diversity is related to the conditional entropy error measure. In the experiments, emphasis is given on determining the contribution of the individual ensemble member to diversity. In this case, an entropy based measure would not have been useful because it does not allow determining individual

contributions [148].

Thus, for the classification problems the most widely used diversity measure is a simple 0/1 loss function. More precisely, if $A_i(p)$ is the prediction accuracy of the i-th classifier for the label of p and assuming that the ensemble's accuracy is $A^*(p)$ then the diversity of the i-th classifier on our tested example p is given by the following procedure:

$$d_i(p) = \begin{cases} 0, & \text{if } A_i(p) = A^*(p) \\ \\ 1, & \text{if } otherwise \end{cases} \tag{6.4}$$

The equation for the ensemble error is the same as in the regression problems, provided that the loss function used is the squared error function, and that the ensemble prediction is still given as the weighted average of the single classifier predictions.

To compute the diversity of an ensemble of size n, on a training set of size m, the average of the above term is:

$$D_{ij} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} d_i(p_j), \tag{6.5}$$

This expression estimates the probability that disagree with the prediction of the ensemble as a whole. The proposed approach is to build ensembles that are consistent with the training data and that attempt to maximise this diversity term. The average diversity is an interesting factor for the operation of the learning scheme. Taking into account small values for the mean squared ensemble error ($E_{ens} = \bar{E} - \bar{D}$, $E_{ens} < 0.1$), corresponds to simple ensemble. For this work, it is used the disagreement of an ensemble member from the ensemble's prediction as a measure of diversity. Thus, the mean squared ensemble error is equal to the average squared error of the individual networks minus the average diversity. Generally, in order to obtain small ensemble error, we want the diversity to be large and the individual errors to be small [148].

Below an example is given to identify the diversity measure that will help to determine the contribution of an individual ensemble member to diversity. An ensemble of three different individual members (FNNs in these experiments) try to improve the accuracy for the Yeast data and specifically for the class me2 using a 10-fold cross validation. Each neural net of our ensemble has an output vector with 8 nodes, which are combined giving the ensemble's output. In Table 6.1, the indication of the winner node is taken into account, using five patterns and three members in the ensemble.

Table 6.1: Example with 3 classifiers and 5 data patterns of Yeast for me2 class

Target Output Values	$p1$ 1	$p2$ 1	$p3$ 1	$p4$ 1	$p5$ 1	**Classification Errors**
$FNN1 : A_1(x)$	0	1	0	1	0	$E1 = 0.6$
$FNN1 : A_2(x)$	1	0	1	1	0	$E2 = 0.4$
$FNN1 : A_3(x)$	1	1	1	0	1	$E3 = 0.2$
$Ensemble : A_n(x)$	1	1	1	1	0	$E1 = 0.2$

By applying the values in the equation for the Diversity Term we take $\bar{D} = 0.33$. The mean squared ensemble error ($E_{ens} = \bar{E} - \bar{D}$) is shown in 6.2.

Table 6.2: The mean squared ensemble error for our example

Average Error ($\bar{E}$)	**Mean Diversity** $\bar{D}$	**Mean squared ensemble error** $E_{ens} = \bar{E} - \bar{D}$
0.4	0.33	0.07

Figure 6.1 illustrates how FNNs are combined to produce the ensemble. The major aim is to create an ensemble of networks with good predictive performance. Therefore, we consider a population of neural networks (100 FNNs). First, we consider the properties of each individual FNN and then we combine FNNs that achieve better performance for the desired application.

The implementation of the ensemble-based method consists of the following steps:

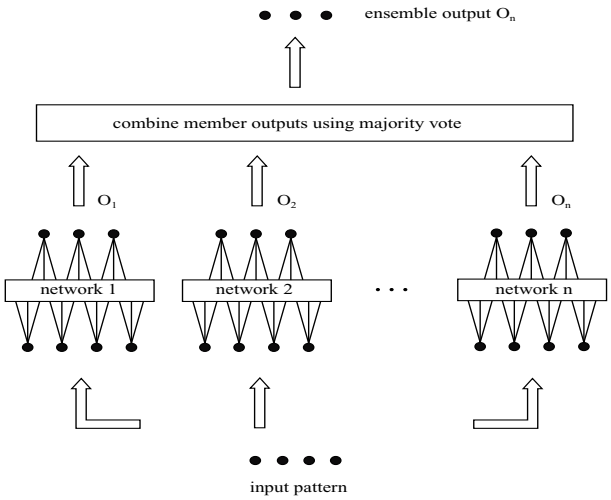


Figure 6.1: Ensemble Scheme

- Step1: Create n FNNs where each one uses the same training set and differs only in its random initial weights.
- Step2: Select the k Neural Networks (k¡n), which when they do fail to classify the data, they fail on different inputs patterns so that failures on one FNN can be compensated by successes of others.
- Step3: Combine the ensemble members' outputs using majority voting to get ensemble's output.

In the present work, two different approaches were investigated for creating ensembles. The first approach consisted of creating a simple neural network ensemble by combining FNNs that use the full training set but differs in their random weight initialisations. The second approach was based on the notion of diversity that was mentioned above and used three different FNNs. Training and testing sets were generated by applying cross validation. Similar approaches often produce results as good as bagging [78]. In these experiments, the behaviour of neural ensembles was also investigated, which belong to Level 4 Diversity, [105], when the cross validation method is applied and Level 3 Diversity [105] when the test set contains all proteins.

6.4 Experimental Study

In the experiments with the KNN, K=7 was set for the E.coli proteins and K=21 for the Yeast dataset; these values have been found empirically to give the best performance [47]. It has been investigated by Horton and Nakai, that the KNN algorithm achieves better performance than the Horton-Nakai model [46, 47]. Thus, in this experimental study only the KNN results are presented, and compared with the FNN and the neural networks ensemble's performance.

6.4.1 Description of Datasets

The datasets that used in this study have been submitted to the UCI Machine Learning Data Repository by Murphy and Aha [72] and are described in [74, 46, 47]. They include the Escherichia dataset with 336 different proteins labeled according to 8 localisation sites and the Yeast data set with 1484 proteins labeled according to 10 sites. More details and fully description of these datasets is given in *Appendix A*.

6.4.2 Classifying E.coli Proteins Using a Feed-forward Neural Network

A set of preliminary experiments conducted to find the most suitable FNN architecture in terms of training speed. The Rprop algorithm was used to train several networks with one hidden layer with various combinations of hidden nodes, i.e. 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes. Each FNN architecture was trained 10 times with different initial weights. The best available architecture found was a 7-16-8 FNN and this architecture was used throughout the experiments. In order to explore the effect of the network error on the accuracy of the classifications one hundred (100) independent trials were performed with two different

termination criteria, which are based on the Mean Squared Error of the Error function, $E_{MSE} < 0.05$ and $E_{MSE} < 0.015$. Following previous work in this area [73, 74, 46, 47, 22], I conducted experiments using all the data for testing, as well as data sets produced by 4-fold cross validation. Also, additional experiments were done by applying leave-one-out cross validation to further investigate the performance of the methods.

Classification of E.coli proteins on the entire dataset

The particular nature of the problem makes important for biologists to know exactly the performance of a method on each individual protein that is included in the dataset. Thus, this experiment trained and tested FNNs using the entire dataset. Table 6.3 presents the classification success for each class.

Table 6.3: The accuracy of classification of E.coli proteins for each class

patterns	**Class**	**KNN (%)**	**FNN**(%) $E_{MSE} < 0.05$	**FNN**(%) $E_{MSE} < 0.015$
77	im	75.3	82.5	89.0
143	cp	98.6	98.1	99.0
2	imL	0.0	85.0	91.8
5	omL	80.0	100.0	100
35	imU	65.7	68.0	88.3
2	imS	0.0	6.5	49.0
20	om	90.0	86.6	93.2
52	pp	90.3	88.7	93.6
Mean		**62.5**	**76.9**	**88.0**
Stdv		**39.8**	**30.1**	**16.3**

The results of the FNN for each class exhibit average performance over 100 trials. It is evident that the FNNs outperforms KNN in almost every class. It is very important to highlight the neural network classification success in inner membrane with cleavable signal sequence (imS) and in inner membrane with lipoprotein (imL) classes. In the first case, FNNs trained with $E_{MSE} < 0.05$ exhibit a 6.5% success and the FNNs with $E_{MSE} < 0.015$ have 49% success,

Table 6.4: Confusion matrix for E.coli proteins with KNN.

No of Patterns	Class	cp	imL	imS	imU	im	omL	om	pp
143	cp	141	0	0	0	0	0	0	2
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	0
35	imU	0	0	0	23	11	0	0	0
77	im	3	0	0	14	58	0	0	2
5	omL	0	0	0	0	0	4	1	0
20	om	0	0	0	0	0	0	18	2
52	pp	4	0	0	0	1	0	0	47

while the KNN method has 0.0% in imS class. In the second case, the FNNs with $E_{MSE} < 0.05$ and the FNNs with $E_{MSE} < 0.015$ exhibits 85% and 91.8% success respectively. The KNN method fails in both cases to produce correct classifications.

It is worth noticing that results for FNNs become even better when an $E_{MSE} < 0.015$ was used for training. This obviously caused an increase to the average number of epochs required to converge (approximately 2000 epochs were needed on average), but at the same time led to considerable improvements: the average classification accuracy over 100 runs was 93% with a standard deviation of 0.33. This behaviour provides evidence that a small variation in the value of the error goal might affect the classification success. This might provide explanation for the unsatisfactory FNNs performance reported in [22], where no details on the error goals used in the training phase were provided.

In order to identify common misclassifications, the confusion matrix was calculated for the KNN and FNN that exhibited the best training speed for an $E_{MSE} < 0.015$. These results are shown in Tables 6.4 and 6.5. The afore mentioned neural network achieved high percentage of classification compared to other methods [47]. These results also show that fast convergence achieved by the FNN by no means affect its classification success.

Table 6.5: Confusion matrix for E.coli proteins with FNN.

No of Patterns	Class	cp	imL	imS	imU	im	omL	om	pp
143	cp	142	0	0	0	0	0	0	1
2	imL	0	2	0	0	0	0	0	0
2	imS	0	0	1	1	0	0	0	0
35	imU	0	0	0	31	4	0	0	0
77	im	2	0	0	6	69	0	0	0
5	omL	0	0	0	0	0	5	0	0
20	om	0	0	0	0	0	0	19	0
52	pp	3	0	0	0	0	0	0	49

Classification of E.coli proteins using 4-fold cross validation

In the second experiment, a 4-fold cross validation test performed by randomly partitioning the dataset into 4 equally sized subsets, as suggested in [46, 47]. Three subsets are used for training, while the remaining one is used for testing. Table 6.6 presents the best KNN and FNN classification accuracy for each class in the second partition. The overall classification success is also given in Table 6.7, where results are in terms of percentage of sucess for each partition. Lastly, it is important to mention that the performance of KNN algorithm is considerably improved when 4-fold cross validation is used but still lacks in performance compared to the best FNN.

Classification of E.coli proteins using leave one out cross validation

In this experiment, E. coli proteins were classified using a FNN by applying a leave one out cross validation. Table 6.8 gives the best results for each method using leave one out cross validation. As shown in the table, the KNN exhibited better performance than the FNN in this case. Nevertheless, it still lacks compared to KNN with 4-fold cross validation.

The overall pattern classification success for all classes using FNNs with $E_{MSE} <$

Table 6.6: Best classification success for each method with 4 fold cross-validation for E.coli proteins(2nd partition).

patterns	**Class**	**KNN (%)**	**FNN**(%) $E_{MSE} < 0.015$
77	im	84.0	79.5
143	cp	100	97.2
2	imL	0.0	0.0
5	omL	100.0	100
35	imU	62.2	87.5
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	92.2	100
Mean		**64.8**	**68.1**
Stdv		**41.8**	**42.7**

Table 6.7: Best overall performance for each method with 4-fold cross validation for each partition.

Cross Validation	Partition	KNN (%)	FNN $E_{MSE} < 0.015$ (%)
	0	89.3	91.7
	1	95.2	88.1
	2	84.5	84.5
	3	76.2	88.1
Mean		**86.3**	**88.1**
Mean		**8.04**	**2.92**

0.015 was 85.42%, which can be considered slightly worst compared to the 86% of the KNN. Table 8 exhibits the average classification success for each class. The differences in mean performance of the two methods in Table 6.8 reflect the fact that the KNN classified correctly the five patterns of the class omL while the FNN misclassified one of these patterns. In order to identify the misclassifications in the E. coli dataset we created the confusion matrix for the KNN and FNN, which are shown in 6.9, 6.10 respectively.

Table 6.8: The accuracy of classification of E.coli proteins for each class using leave one out cross validation.

patterns	**Class**	**KNN**	**FNN**(%)
patterns	**Class**	**(%)**	$E_{MSE} < 0.015$
77	im	76.6	79.3
143	cp	97.9	97.2
2	imL	0.0	0.0
5	omL	100.0	80.0
35	imU	57.1	65.7
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	88.5	84.6
Mean		**62.5**	**60.8**
Stdv		**40.8**	**38.5**

Table 6.9: Confusion matrix for E.coli proteins with KNN with leave one out cross validation.

No of Patterns	Class	cp	imL	imS	imU	im	omL	om	pp
143	cp	140	0	0	0	0	0	0	3
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	1
35	imU	1	0	0	20	14	0	0	0
77	im	4	0	0	10	59	0	0	4
5	omL	0	0	0	0	0	5	0	0
20	om	0	0	0	0	0	1	16	3
52	pp	5	0	0	0	1	0	0	46

6.4.3 Classifying Yeast Patterns Using a Feed-forward Neural Network

A set of preliminary experiments was conducted, as with the E.coli dataset, in order to find the most suitable architecture. An 8-16-10 FNN architecture exhibited the best performance. 100 FNNs were trained with the Rprop algorithm using different initial weights. As previously we conducted two experiments following the guidelines of [46, 47].

Table 6.10: Confusion matrix for E.coli proteins with FNN with leave one out cross validation.

No of Patterns	Class	cp	imL	imS	imU	im	omL	om	pp
143	cp	139	0	0	0	0	0	0	4
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	1
35	imU	0	0	0	23	11	0	0	1
77	im	3	0	0	11	61	0	1	1
5	omL	0	0	0	0	0	4	1	0
20	om	0	0	0	0	0	2	16	2
52	pp	4	0	0	0	3	0	1	44

Classification of Yeast proteins on the entire dataset

The first experiment concerns testing using the whole dataset. The FNN outperforms significantly the other methods. The average classification accuracy of the FNN's is 67%; the worst-case performance was 64% (which is still an improvement over other methods) and the best one 69%. The result of the KNN is 59.5%. We have trained the FNNs for 10000 epochs or until achieving an $E_{MSE} < 0.005$ .On the average, approximately 3500 epochs are needed in order to reach convergence.

Table 6.11 shows the classification success achieved for each class. The results of the FNN represent the average of 100 trials. The neural network outperforms the other methods in almost every class. It is very important to highlight the neural network classification success in the POX and ERL classes.

To identify the misclassifications in the Yeast dataset we have created the confusion matrix for the FNN that exhibited the fastest convergence to an $E_{MSE} < 0.005$. The results are shown in Table 6.12. It is important to highlight the significant improvement to classify the localisation sites in each class compared with the other previous attempts as shown in Table 13 [47].

Table 6.11: The accuracy of classification of Yeast proteins for each class.

No of Patterns	Class	KNN (%)	FNN E_{MSE} <0.05 (%)
463	cyt	70.7	66.7
5	erl	0.0	99.6
35	exc	62.9	62.7
44	me1	75.0	82.9
51	me2	21.6	47.8
163	me3	74.9	85.6
244	mit	57.8	61.3
429	nuc	50.7	57.7
20	pox	55.0	54.6
30	vac	0.0	4.1
Mean		**46.8**	**62.3**
Stdv		**29.1**	**25.9**

Classification of Yeast proteins using 10-fold cross validation

The second experiment involves the use of 10 fold cross validation method with 10 equally sized partitions. The results are shown in Table 6.14. The KNN algorithm improves its generalisation success achieving an overall success of 59.5%. The performance of the FNNs is also improved, achieving average classification success of 64.9%. In these experiments, an $E_{MSE} < 0.05$ used to train the neural networks. The results of the Wilcoxon test for the Table 14, gives $T = 7$, which satisfies the condition $T < 14$. This proves that the improved mean performance achieved by the FNNs is statistically significant when compared against the results of the KNN algorithm.

Classification of Yeast proteins using leave one out cross validation

In the third experiment a leave one cross validation is implemented to classify the Yeast Patterns. The leave one out validation helps to explore further the performance of the tested methods. The best overall classification success for all the patterns in the Yeast dataset using a FNN with leave one out cross

Table 6.12: The confusion matrix of Yeast proteins for each class using a neural network.

No of Patterns	Class	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
463	cyt	309	0	0	3	0	12	30	109	0	0
5	erl	0	5	0	0	1	1	0	0	0	0
35	exc	4	0	23	2	2	0	2	2	0	0
44	me1	3	0	1	37	0	2	1	0	0	0
51	me2	6	0	2	3	25	5	8	2	0	0
163	me3	9	0	0	0	0	140	2	10	0	2
244	mit	50	0	0	2	4	10	150	28	0	0
429	nuc	110	0	0	0	42	10	20	247	0	0
20	pox	6	0	0	0	0	0	2	1	11	0
30	vac	11	0	2	0	1	6	2	7	0	1

validation is 59.9%, while the KNN algorithm exhibits approximately 58%. Both of their performance still lacks comparing to 10 fold cross validation performance. Table 6.15 shows the comparative results for the tested methods using leave one out cross validation and presents the classification success for each class. The statistics reveal that FNN outperforms significantly to KNN algorithm exhibiting a more balanced performance.

6.4.4 Classifying Protein Patterns Using Ensemble-based Techniques

In this section, ensembles were created focusing on classifiers that disagree on their decisions so when they do fail to classify the data, fail on different inputs so that failures on one FNN can be compensated by successes on others. These Diversity Network Ensembles (DNE) allow to weight the outputs of the network in such a way that either the correct answer is obtained or at least that the correct output is obtained often enough so that the generalisation is improved. In order to get a small ensemble error we try the diversity to be large and the

Table 6.13: The confusion matrix of Yeast proteins for each class using the KNN algorithm.

No of Patterns	Class	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
463	cyt	314	0	1	0	2	3	32	91	1	0
5	erl	0	0	3	1	1	0	0	0	0	0
35	exc	4	0	22	4	2	0	2	1	0	0
44	me1	0	0	8	33	0	1	2	0	0	0
51	me2	9	0	7	10	11	3	7	4	0	0
163	me3	18	0	0	0	1	122	6	16	0	2
244	mit	62	0	4	2	5	8	141	19	3	0
429	nuc	171	0	0	0	2	10	27	216	0	0
20	pox	4	0	1	1	0	0	1	2	11	0
30	vac	13	0	3	1	1	6	1	5	0	0

number of individual errors to be small.

Experiments using the full datasets

In this study, the reported results based on using a Simple Network Ensemble (SNE) and a Diverse Neural Networks (DNN). I performed two different experiments. The implementation of the ensemble, in the first experiment consisted of five networks and belongs to the so called level 3 diversity. Table 6.16 shows the results of the two ensemble-based methods on the E.coli dataset. The overall classification success of the SNE method was 93.5%, while the overall classification success of the DNN method was 96.8% using all the data as suggested by Horton and Nakai [46, 47]. I decided to concentrate on the DNN method and created an ensemble for the Yeast dataset. The results are shown in Table 6.16. The DNE significantly outperforms all other methods used so far in the literature (cf. with the results reported in [46, 47, 22]).

Table 6.14: Best performance for each method using 10 fold cross-validation for Yeast proteins.

Cross Validation	Partition	KNN (%)	FNN $E_{MSE} < 0.05$ (%)
	0	55.8	65.1
	1	59.2	66.4
	2	61.0	63.3
	3	65.8	65.8
	4	48.6	66.4
	5	62.3	68.5
	6	68.5	61.8
	7	58.9	59.8
	8	56.9	66.4
	9	58.2	65.6
Mean		**59.5**	**64.9**
Std. Dev		**5.49**	**2.57**

Experiments using leave one out cross validation.

In this case it is difficult to build an ensemble with feedforward neural nets to do different errors in different patterns. By applying a leave one out cross validation most of the neural nets fail in the same patterns so the classification ability of the ensemble is not significant improving comparing to the neural networks. Based on these experimental results the strategy of building neural network ensembles has to do with the tested problem. Therefore for the E.coli and Yeast datasets a 4-fold and 10-fold cross validation method respectively is suggested to be applied [47]. The performance of the FNNs and the diverse neural networks ensemble, using the leave one out cross validation, are shown in the tables below (6.18, 6.19).

Classifying E.coli Patterns Using Ensemble methods with 4-fold cross validation.

In these experiments an ensemble of three Neural Networks is investigated. The FNNs failures for predicting the imS class, are shared by all the networks. This

Table 6.15: Best performance for each method using one fold cross-validation for Yeast proteins for each class.

No of Patterns	Class	KNN (%)	FNN E_{MSE} <0.05 (%)
463	cyt	55.4	66.7
5	erl	0.0	99.6
35	exc	61.7	62.7
44	me1	65.1	82.9
51	me2	26.0	47.8
163	me3	75.8	85.6
244	mit	58.4	61.3
429	nuc	50.3	57.7
20	pox	57.8	54.6
30	vac	0.0	4.1
Mean		**46.0**	**62.3**
Stdv		**26.9**	**25.9**

type of ensemble corresponds to level 4 diversity. It can never be reliable but it can still be used to improve the generalisation. By applying the cross validation, the training and testing data are changed. So in order to train properly the FNN we set smaller error target $E_{MSE} < 0.01$. In this case we focus to find Neural Networks that can classify the class with the few patterns such the imL class. The divesrity measure in these experiments is $D = 0.33$. The ensemble is based on predicting correctly the patterns from the imL class.

Table 6.20 presents the number of mistakes for each members of the ensemble and for different combinations. Ensemble members FNN1, FNN2, and FNN3 fail to classify 15, 16, and 11 mistakes respectively. Combining different FNNs produces various numbers of common mistakes (failures) depending on the diversity of the members which are combined. For example, as shown in Table 6.20, although FNN1 and FNN2 make 15 and 16 mistakes, a combination of the two produces significantly smaller number of common failures, i.e. only two common mistakes. In our experiments we used a combination of the three FNNs using majority voting. This ensemble also produces two common failures but the overall success is imporoved. Detailed results are shown in Table 6.21.

Table 6.16: Accuracy of classification for E.coli proteins using ensemble-based techniques.

No of Patterns	Class	SNE E_{MSE} <0.015 (%)	DNN E_{MSE} < 0.015 (%)
77	im	87.0	92.22
143	cp	98.0	100
2	imL	100	100
5	omL	100	100
35	imU	77.2	94.3
2	imS	50.0	50
20	om	80.0	100
52	pp	91.4	96.15
Mean		**85.45**	**91.7**
Stdv		**16.8**	**17.1**

Table 6.17: Accuracy of classification for Yeast proteins using diverse neural networks

No of Patterns	Class	DNN E_{MSE} <0.05 (%)
463	cyt	69.2
5	erl	100
35	exc	64.3
44	me1	84.9
51	me2	54.9
163	me3	88.4
244	mit	61.7
429	nuc	57.7
20	pox	55.0
30	vac	10.0
Mean		**64.6**
Stdv		**24.6**

Table 6.18: Best performance for each method using one fold cross-validation for E.coli proteins by ensemble-based techniques.

No of Patterns	Class	FNN E_{MSE} <0.015 (%)	DNN $E_{MSE} < 0.015$ (%)
77	im	79.3	80.6
143	cp	97.2	97.2
2	imL	0.0	0.0
5	omL	80.0	80.0
35	imU	65.7	65.7
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	84.6	86.6
Mean		**60.8**	**61.3**
Stdv		**38.5**	**38.8**

It is important to mention that using a DNE allows to classify correctly the imL class, which has only two patterns. This can increase the average class accuracy for E. coli proteins, as shown in Table 6.21.

Various combinations of diverse FNNs can be produced focusing on improving the classification for particular class labels. For example, if predicting correctly classes with a few patterns (e.g. imS) is not a priority then it is possible to improve the overall classification success by focusing on classifying correctly classes with many patterns (e.g. cp). In this case the ensemble would combine diverse Neural Networks that achieve better performance in classes with many patterns, such as the cp class.

Classifying Yeast Patterns Using Ensemble methods with 10-fold cross validation.

In this case, it created an ensemble of three Neural Networks, which corresponds to Level 3 diversity [105]. However, a simple majority vote will not always produce the correct answer, but at least one member in the ensemble will produce the correct output for each input pattern in the test set. Table 6.22 shows

Table 6.19: Best performance for each method using one fold cross-validation for Yeast proteins for each class.

No of Patterns	Class	FNN E_{MSE} <0.05 (%)	FNN E_{MSE} <0.05 (%)
463	cyt	66.7	66.7
5	erl	80.0	100
35	exc	62.7	62.7
44	me1	82.9	80.0
51	me2	47.8	47.8
163	me3	85.6	91.7
244	mit	61.3	61.3
429	nuc	57.7	57.7
20	pox	54.6	54.6
30	vac	4.1	4.1
Mean		**60.3**	**62.6**
Stdv		**23.4**	**26.4**

classification success for an FNN and two ensembles (DNE1 and DNE2) using 10-fold cross validation. The FNN performance is the best available from a set of 100 trials, while the diversity value for DNN1, which focuses on predicting cyt patterns, is D=0.36 giving $E_{ens} = 0.023$, and for DNN2, which is built based on vac patterns, is D=0.33 and $E_{ens} = 0.022$.

It is important to mention that DNEs, achieved better performance compared to other classification methods. The KNN overall success was 59.3% and the FNN success was 63.4% when $E_{MSE} < 0.05$ was used in training. DNN1 shows significant improvement in overall pattern classification success, reaching 67.6% but it cannot classify correctly any vac pattern; it focuses on cyt patterns instead. The overall classification success for DNN2 was 62.75% with predictions for all classes of the S. cerevisiae proteins, and an average success per class of 66.2%, as shown in Table 6.22.

Table 6.20: Number of patterns each neural network fails to classify correctly ($FNN1, FNN2, FNN3$), and common number of patterns neural networks fail to classify correctly ($FNN12$, $FNN13$,$FNN23$,$FNN123$) for the Ecoli protein problem using 4 fold cross validation and diverse neural networks.

Neural Networks	**Failures**
$FNN1$	15
$FNN2$	16
$FNN3$	11
$FNN12$	2
$FNN13$	5
$FNN23$	3
$FNN123$	2

Table 6.21: Accuracy of classification for Ecoli proteins using 4 fold cross validation and diverse neural networks.

No of Patterns	Class	FNN E_{MSE} <0.01 (%)	DNN $E_{MSE} < 0.01$ (%)
77	im	89.0	88.8
143	cp	97.2	97.2
2	imL	0.0	100.0
5	omL	100.0	100.0
35	imU	75.0	87.5
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	100.0	92.3
Mean		**67.6**	**80.7**
Stdv		**42.7**	**33.3**

Table 6.22: The accuracy of classification of Yeast proteins for each class using 10 fold cross validation.

patterns	**Class**	**KNN** (%)	**FNN1**(%) $E_{MSE} < 0.05$	**DNN1**(%) $E_{MSE} < 0.05$	**DNN2**(%) $E_{MSE} < 0.05$
463	cyt	70.7	71.7	76.1	59.0
5	erl	0.0	100.0	100.0	100.0
35	exc	62.9	25.0	75.0	50.0
44	me1	75.0	80.0	60.0	80.0
51	me2	21.6	60.0	40.0	60.0
163	me3	74.9	87.5	100.0	100.0
244	mit	57.8	75.0	91.7	58.3
429	nuc	50.7	42.9	52.5	54.8
20	pox	55.0	66.7	33.3	66.7
30	vac	0.0	0.0	0.0	33.3
Mean		**46.8**	**60.9**	**62.9**	**66.2**
Stdv		**29.1**	**30.3**	**32.2**	**21.3**

6.4.5 Classifying Ecoli and Yeast Patterns Using the Hybrid Learning Scheme to Train Neural Networks.

The complexity of Ecoli and Yeast classification problems makes Neural Network training important and crucial factor. Well studied learning algorithms, such as the Backpropagation Algorithm and the Resilient Propagation algorithm, failed to classify correctly classes with few patterns. The recently proposed algorithms JRprop, GJRprop and GRprop showed classification improvements but still misclassified protein classes with few patterns.

The new adaptive gradient–based learning scheme (HLS) inspired from the Rprop algorithm, and based on the theory of nonextensive statistical mechanics [125] gives better results. The specific characteristics of this algorithm make it attractive to be used in wide range of bioinformatics applications, such as is the classification of the proteins into localisation sites and protein folding. Next experimental results are presented and comparisons are made. The training of the FNNs was done by using the HLS algorithm in all runs.

Table 6.23 shows the accuracy of classification of E.coli proteins for each class using the HLS to train the neural networks, which is slightly better than these neural networks trained with the Rprop algorithm. It is important to highlight the increased classification success that the FNN with the HLS algorithm achieve in classes with few patterns such as imL and imS.

Table 6.23: The accuracy of classification of E.coli proteins for each class Using the HLS algorithm to train the Neural Networks

patterns	**Class**	**FNN Rprop**(%) $E_{MSE} < 0.015$	**FNN HLS**(%) $E_{MSE} < 0.015$
77	im	89.0	89.3
143	cp	99.0	99.3
2	imL	91.8	95.0
5	omL	100.0	100
35	imU	88.3	85.3
2	imS	49.0	50.0
20	om	93.2	93.3
52	pp	93.6	93.5
Mean		**88.0**	**88.2**
Stdv		**16.3**	**16.1**

Table 6.24 presents the best overall performance for the FNN's trained with different algorithms.

Table 6.24: Best overall performance for each method with 4-fold cross validation for each partition.

Cross Validation	Partition	FNN Rprop $E_{MSE} < 0.015$ (%)	FNN HLS $E_{MSE} < 0.015$ (%)
	0	91.7	92.0
	1	88.1	88.3
	2	84.5	84.3
	3	88.1	88.5
Mean		**88.1**	**88.3**
Stdv		**2.92**	**3.1**

The next table 6.25 gives the average performance for Neural networks trained with the Rprop and HLS algorithms. As shown in Table 6.25 the FNN's trained

with the HLS outperforms the FNN's with Rprop in the number of epochs required to reach a suitable solution, classification success and converges in all partitions.

Table 6.25: Mean behaviour in terms of speed, convergence and testing classification success for each method with 4 fold cross-validation for E.coli proteins.

Cross Validation	**FNN Rprop**			**FNN HLS**		
Partition	$E_{MSE} < 0.015$ *Success*	*Epochs*	*Convergence*	$E_{MSE} < 0.015$ *Success*	*Epochs*	*Convergence*
0	82.9	6837	58.0	84.8	4998	80.0
1	83.4	4634	97.0	84.9	2832	100.0
2	79.0	3784	98.0	80.6	2100	100.0
3	80.2	5395	74.0	82.7	3679	90.0
Mean	**81.3**	**5162**	**81.7**	**83.3**	**3402**	**92.5**

Judging from the previous tables, it can be drawn that the nonextensive training algorithm (HLS) improves the performance of the neural networks and generates different classifications. Therefore it is very possible the Ensemble Neural Nets trained with the HLS ($DFNN\ HLS$) achieve better classification success. Table 6.26 confirms this improvement performance. Our tested ensemble is belong to 'Level 4' diversity. This ensemble ($DFNN\ HLS$) is consisted of 3 FNN trained with the HLS algorithm, which two of these are from the first partition and the third from the second partition. In this stage, it is important to clarify that the classification success is the testing classification success for all the patterns that consist the testing data, and it is not the class classification success.

Table 6.26: The Ensemble performance using 4 fold cross-validation for E.coli proteins for 50 runs using the new training algorithm.

Cross Validation	FNN Rprop(%)	FNN HLS (%)	DFNN HLS (%)
Ecoli Proteins	$E_{MSE} < 0.015$	$E_{MSE} < 0.015$	$E_{MSE} < 0.015$
$Classification\ Success$	81.3	83.3	88.8

The problem of the Yeast proteins is more complicated. The effect of HLS algorithm in training of the FNN's is presented in Table 6.27. The mean performance of the Neural nets trained with the HLS is significantly better compared with the Rprop-FNN's in terms of speed and testing classification success.

Table 6.27: Mean behaviour in terms of speed, convergence and testing classification success for each method with 10 fold cross-validation for Yeast proteins.

Cross Validation	**FNN Rprop**			**FNN HLS**		
Partition	$E_{MSE} < 0.05$ *Success*	*Epochs*	*Convergence*	$E_{MSE} < 0.05$ *Success*	*Epochs*	*Convergence*
0	61.6	983	100.0	63.2	486	100.0
1	61.0	1052	100.0	64.0	632	100.0
2	58.6	934	100.0	61.0	513	100.0
3	61.9	970	100.0	64.2	582	100.0
4	60.4	973	100.0	62.1	580	100.0
5	62.2	1370	100.0	65.5	749	100.0
6	56.3	705	100.0	58.9	446	100.0
7	52.8	534	100.0	54.9	339	100.0
8	60.9	920	100.0	63.1	569	100.0
9	60.7	667	100.0	63.3	447	100.0
Mean	**59.6**	**911**	**100.0**	**62.1**	**544**	**100.0**

Finally an ensemble consisted of 3 FNN's trained with the HLS (Level4 diversity)is tested. The ensemble has one FNN from the first partition, one from the second and the last one from the 6 th partition. The Ensemble performance using 10 fold cross-validation for 50 runs is given in table 6.28. As shown in this table, there is significant improvement for the Ensemble as the new stochastic learning scheme creates different classifications that can be intelligent combined to generate better classification performance.

Table 6.28: The Ensemble performance using 10 fold cross-validation for Yeast proteins for 50 runs using the new training algorithm.

Cross Validation Ecoli Proteins	FNN Rprop(%) $E_{MSE} < 0.015$	FNN HLS (%) $E_{MSE} < 0.015$)	DFNN HLS (%) $E_{MSE} < 0.015$)
$Classification$ *Success*	59.6	62.1	70.5

6.5 Summary and Contribution of the Chapter

In this chapter, it was explored the use of a neural network approach in predicting the localisation sites of proteins in the organisms. The performance of Neural Networks in a single mode and in an ensemble formulation was investigated. The use of diverse network ensembles exhibiting Level 3 and 4 diversity with and without cross validation was also explored. Numerical results using supervised learning schemes with and without the use of cross validation, are better than the best previous attempts.

Ensemble for neural networks is a subject of active research. It enables an increase in generalisation performance, by combining several individual neural networks trained on the same task. Using the notion of the diversity in building our ensembles, helps to improve the classification success. Furthermore, the generation of ensemble with diversity FNNs showed a significant improvement compared to all other approaches in the literature for the two organisms. Training with the recently proposed hybrid algorithm appears to have a positive effect on their performance. The nonextensive concept in this algorithm generates different classifications, which is important characteristic to create efficient ensembles of neural nets.

To sum up, the proposed algorithms are applied for training feed-forward neural networks and diverse neural ensembles in biological and bioinformatics datasets. Finally building a neural ensemble with neural networks which are trained by different training algorithms and in different training sets are on going research.

Chapter 7

Conclusions and Future work

'I am neither Athenian nor Greek but a citizen of the world.' Socrates 500bc.

7.1 Discussion

Neural Networks are widely used in many classification applications such as pattern classification, speech recognition etc. Neural Networks are often used to classify or categorise. The goal of Feedforward Neural Network (FNN) learning is to iteratively adjust the weights, in order to globally minimise a measure of the difference between the actual output of the network and the desired output, as specified by a teacher, for all examples in a training set [43].

Finding this global minimum of a complex cost function, which has a large number of local minima, is a very intricate task [43, 97]. Many existing supervised learning algorithms based on the technique of gradient descent try to find the optimal solution. A variety of approaches, inspired from unconstrained optimisation theory, have also been applied, in order to use second derivative related information to accelerate the learning process. A problem with these algorithms

is that they occasionally converge to undesired local minima. While some local minima can provide acceptable solutions, they often result in poor network performance. This problem can be overcome through the use of global optimisation. The drawback of these algorithms is that they are computationally expensive.

The applications and problems studied in my PhD work are devoted to complex phenomena, drawing input from a wide variety of fields, including biology, genomics, bioinformatics and pattern recognition. Among the various applications developed in my PhD, is the classification of protein localisation patterns into known categories, which is considered particularly useful in the post-genomics era. The E.coli and Yeast proteins are characterised by the drastically imbalanced nature of their dataset. The new proposed schemes applied in these problems, show a significant improvement in terms of learning speed and classification success, when compared to well known existing training neural networks algorithms.

In my PhD work, also emphasis is given on the development of well-performing supervised learning schemes to apply to classification problems and especially using biological data. It proposed algorithms to improve the learning speed and the convergence success compared to well known existing training algorithms. More specific this PhD proposes a new class of sign–based schemes with adaptive learning rates that are based on the composite nonlinear Jacobi process. It develops adaptation strategy that ensures the search direction is a descent one and guarantees the decrease of the batch error. It also equips the new algorithms with the global convergence property; i.e. it proves convergence to a local minimiser from any remote starting point. However, in some cases the problem with the convergence to local minima continues to exist.

To improve Neural Networks' ability to correctly predict complex phenomena, ideas based on the statistical physics have also been applied to accomplish this target. Nonextensive statistical mechanics, proposed by Tsallis, exhibits apparent success for certain systems in biology, economics, linguistics, the physics of

turbulence, and other fields. Therefore in my PhD, the difficult problem of occasional convergence to local minima is dealt by proposing a hybrid adaptive learning scheme that combines deterministic and stochastic search. Stochastic search is explored in the context of Nonextensive Statistical Mechanics, by modifying the error surface during training using the q-nonextensive entropic index.

Finally, the Neural Networks' performance was investigated in an ensemble formulation. I have explored the use of network ensembles in Bioinformatics problems, and the proposed algorithms were applied for training feed-forward neural networks and diverse neural ensembles in biological and bioinformatics datasets. This phenomenon effects their performance positively.

7.2 Future work

My research work contributes to the theory of learning algorithms. The new algorithms provided improved learning speed, and better convergence behaviour compared to well known training algorithms. However, these algorithms have some limitations. It is still not guaranteed that the new schemes converge to global minima. Further research into the performance of JRprop, GJRprop and GRprop, has to be done. Experimental results provide evidence that the synergy of techniques from nonextensive statistics provides neural learning schemes significant benefits. It is likely this idea, if it is suitably adopted in the new proposed schemes, can improve the convergence and classification success. Thus, one direction for future work is to conduct experiments exploring the influence of the entropic index q in the new algorithms's performance.

Moreover, the Hybrid Learning Scheme HLS that was built on the Tsallis theory, had the temperature T and the entropic index q a priori constantly fixed during the training. On-line adaptation of these two crucial parameters can help to improve its performance. Furthermore, a pre-training of these two algorithms,

or an adaptive scheme between the T and q could possibly give better results. Also, the behavior of the learning scheme can be more stochastic or deterministic, by applying a trade-off formula between T and q. Thus, through this formula, we can achieve better control of the training, as we can make the algorithm more deterministic or stochastic, depending on the application. Finally, when the HLS cannot find the optimal solution or for many iterations cannot reduce the error, then we can investigate the performance of HLS in a restarting mode, i.e to reset the Temperature. All these ideas are likely to improve the convergence speed and stability of the method.

An additional challenge is to properly train large complex dynamics networks. Random neural networks and spiking neural networks are both stochastic dynamic neural models. Training of these models is a difficult task. Ideas based on the statistical physics could accomplish this goal. In particular, implementation of methods based on nonextensive statistics and global optimisation theory should be very promising.

The development and use of computational methods for the acquisition, analysis and interpretation of biological information is also an interesting open problem. Particularly, I am interested in clustering biomolecular data, complex biological structures, integration of biological sequence, structure and function data.

Moreover, much of the on-going statistical analysis of DNA sequences is focused on the estimation of characteristics of coding and non-coding regions that would possibly allow discrimination of these regions. To estimate the level and type of correlation in these regions we can apply various statistical methods. Deterministic and stochastic models are promising tools. Methods based on statistical mechanics and Tsallis statistics can help the statistical methods to find whether the correlations do exist in the DNA sequence. It is possible to find a range for the q entropic index that can describe these correlations in detail.

Furthermore, characterising the complexity of networks is a nontrivial task, and is

still an unresolved problem. Some known complexity measures themselves have a high computational complexity, therefore an alternative complexity measure can be useful in some cases. It is difficult to determine whether we can find a measure that reliably qualifies a system's *structure* as complex when we know that the *dynamics used to generate it* are complex. It is apparent that the term complexity is not well defined. Future work will involve the development and analysis of certain measures that will give an insight as to what complexity is. An entropic measure, inspired from nonextensive statistics as proposed by Tsallis, could provide evidence for the complexity of dynamical networks.

Finally, we are going to explore further the properties of Tsallis entropy into Optimisation methods in Artificial Intelligence applications. Probabilistic models for global exploration of a space of potential solutions are usually based on the Boltzmann Gibbs entropy and employ Gaussian distributions. They have been used in various computational intelligence applications such as training of Boltzmann machines and multilayer neural networks with noise, model reduction of control systems using evolutionary computing, and designing of gaussian mutations in swarm intelligence. An alternative model that will be based on the nonextensive entropy, can be beneficial by incorporating this model with computational intelligence methods. Investigation of the Tsallis entropy or Statistical mechanics in the context of neural networks learning and swarm intelligence is another future target.

These ideas certainly deserve further investigation. Hopefully, the implementation of my PhD will prove a powerful tool to accomplish these future targets.

Appendix A

Problems Description–Datasets–Evaluation Methods

A.1 Problems description

Classification problems from the UCI repository of machine learning database have been used. In all cases,except of the bioinformatics problems which are Ecoli and Yeast problems, we follow the guidelines of PROBEN1. For the protein classification problems, I apply the guidelines which suggested by other researchers, Horton and Nakai [46], [47] who studied extensively these problems. Description of the datasets and the basic features of the tested problems follow.

A.1.1 The Cancer1 problem.

This is the *breast cancer diagnosis* problem which classifies a tumor as benign or malignant based on 9 features [72, 88]. A FNN with 9–4–2–2 nodes (a total of 56 weights) is used, as suggested in [88]. The link for this problem is: http://www.ics.uci.edu/ mlearn/databases/breast-cancer-wisconsin/

A.1.2 The Diabetes1 problem.

The aim of this real-world classification task is to decide when a Pima Indian individual is diabetes positive or not. There are 8 inputs, which represents personal data and results from a medical examination. The data set consists of 384 patterns[72, 88], and the link is: http://www.ics.uci.edu/ mlearn/databases/pima-indians-diabetes/

A.1.3 The Genes2 problem.

It is a binary problem. The goal of this classification task is to decide, from a window of 60 DNA sequence elements (nucleotides), whether the middle is either an intron/exon boundary (a donor), or an exon/intron boundary (an acceptor), or none of these. The data set consists of 1588 patterns. This data set was created based on the 'splice junction' problem dataset from the UCI repository of machine learning database [72, 88].

A.1.4 The Thyroid problem.

This problem is based on patient query data and patient examination data. The task is to decide whether the patient's thyroid has over function, normal function, or under function. The data set consists of 3600 patterns. I

used the thyroid1 dataset, which is not a permutation of the original data, but retains the original order instead [72, 88]. The url to download the data is: http://www.ics.uci.edu/ mlearn/databases/thyroid-disease/

A.1.5 Fisher's Iris problem

This benchmark is known as the *Fisher's Iris* problem [72, 88]. The data set consists of 120 examples and the test set of 30 examples. The url of this problem is http://www.ics.uci.edu/ mlearn/databases/iris/

A.1.6 The Ecoli problem.

The dataset used has been submitted to the UCI Machine Learning Data Repository by Murphy and Aha [72] and is described in [74, 46, 47]. Escherichia dataset with 336 different proteins labelled according to 8 localisation sites

Ecoli, being a prokaryotic gram-negative bacterium, is an important component of the biosphere. It colonises the lower gut of animals and survives, as it is a facultive anaerobe, when realise to the natural environment, allowing widespread to new hosts [16, 61]. Three major and distinctive types of proteins are characterised in E.Coli: enzymes, transporters and regulators. The largest number of genes encodes enzymes (34%) (this should include all the cytoplasm proteins) followed by the genes for transport functions and the genes for regulatory possess (11.5%) [58].

For this problem, seven different attributes were used as in [47, 73, 74]. The first attribute is generated by applying McGeoch's method for signal sequence recognition. The second one is a result of the von Heijne's method for signal sequence recognition, and the third one is the von Heijne's Signal Peptidase II consensus sequence score. The fourth attribute represents the presence of charge

on N-terminus of predicted lipoproteins. The fifth attribute is the score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins, and the sixth one is the score of the ALOM membrane spanning region prediction program. The last attribute gives the score of ALOM program after excluding putative cleavable signal regions from the sequence.

In particular, protein patterns in the E.coli data set are organised as follows: 143 patterns of cytoplasm (cp), 77 of inner membrane without signal sequence (im), 52 of periplasm (pp), 35 of inner membrane with uncleavable signal sequence (imU), 20 of outer membrane without lipoprotein (om), 5 of outer membrane with lipoprotein (omL), 2 of inner membrane with lipoprotein (imL) and 2 patterns of inner membrane with cleavable signal sequence (imS). The corresponding url is: http://www.imcb.osaka-u.ac.jp/nakai/psort.html

A.1.7 The Yeast problem.

Saccharomyces cerevisiae (Yeast) is the simplest Eukaryotic organism. Yeast as more complicated form of life than E.coli possesses different types of proteins related to the cytoskeletal structure of the cell, the nucleus organisation, membrane transporters and metabolic related proteins (as mitochondrial proteins). Of major importance are the yeast membrane transporter proteins as they are responsible for nutrient uptake, drug resistance, salt tolerance, control of cell volume, efflux of undesirable metabolites and sensing of extracellular nutrients [16, 61].

The Yeast data set with 1484 proteins labelled according to 10 sites. This is another drastically imbalanced pattern classification problem. Yeast proteins are organised as follows: there are 463 patterns of cytoplasm (cyt), 429 of nucleus (nuc), 244 of mitochondria (mit), 163 of membrane protein without N-terminal signal (me3), 51 of membrane protein with uncleavable signal (me2), 44 of membrane protein with cleavable signal (me1), 35 of extracellular (exc), 30 of vacuole

(vac), 20 of peroxisome (pox) and 5 patterns of endoplasmic reticulum (erl)[72]. The corresponding link is http://www.imcb.osaka-u.ac.jp/nakai/psort.html

A.2 Evaluation Methods

A.2.1 Cross Validation

In k-fold cross validation a dataset D is randomly split into k mutually exclusive subsets $D_{i,\ldots,D_k}$ of approximately equal size. The classifier is trained and tested k times; each time $t \in 1, 2, \ldots, k$ it is trained on all $D_i, i = 1, \ldots, k$, with $i \neq t$, and tested on D_t. The cross validation estimate of accuracy is the overall number of correct classifications divided by the number of instances in the dataset. The proportion of the number of the patterns for all the classes, is equal in each partition as this procedure provides more accurate results than a plain cross validation does [51].

A.2.2 The Wilcoxon Test of Statistical Significance

The Wilcoxon signed rank test is a nonparametric method [114]. It is an alternative to the paired t-test. It has been introduced by Wilcoxon in 1945, and it is designed to test whether a particular sample comes from a population with a specific median. It can also be used in paired difference experiments. This test assumes that there is information in the magnitudes of the differences between paired observations, as well as the signs. It is a very popular statistical test used by researchers to prove the significance of the reported results [77, 49]. Next, I briefly describe the implementation of this method.

Firstly, the paired observations are taken, I calculate the differences and then we rank them from smallest to largest by their absolute value. Adding all the ranks

associated with positive and negative differences gives the so called T_+ and T_- statistics respectively. Finally, the probability value associated with this statistic is found from the appropriate table.

More precisely the data consists of n^* observations on the respective bivariate random variables. Assume that the sample of differences, DF_i, is randomly selected from the population of differences. The DF_i's are mutually independent and the probability distribution for the sampled paired differences is continuous. Let $|DF_i| = |X_i - Y_i|$ be the absolute differences for $i = 1, 2, \ldots, n^*$, where $X_i = (x_1, \ldots, x_n^*)$ is the population A, and $Y_i = (y_1, \ldots, y_n^*)$ is the population B. Let n be the number of non zero differences. Assume that T_+ is the sum of signed rank of positive DF_i, T_- is the sum of signed rank of negative DF_i and $T = min(T_+, T_-)$. I assigned ranks to these n absolute differences according to the relative size of the absolute differences. I implemented the Right-tailed Test. In this application the population A is shifted to the right of B. The Hypothesis in this case is: $H_0 : DF_i = 0$, and $H_a : DF_i > 0$, and the rejection region is $T \leq T_0$ for small sample sizes, where T_0 is given by a standard table and it is the critical value of T [114]. When the sample sizes are greater than 25, the large sample approximation procedure is used, which is: $Z_c > Z_a$, where Z_c is equal to:

$$Z_c = \frac{T_+ - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \tag{A.1}$$

and Z_a takes a standard value $Z_a = 1.96$, and n is the number of the paired differences, which are not zero.

In the experiments, i analysed the statistical significance by implementing the Wilcoxon rank sum test as proposed in [114]. The implementation of the Wilcoxon Signed Rank algorithm consists of the following steps:

- Step 1: Take the absolute difference $|DF_i| = |X_i - Y_i|$ for each pair;
- Step 2: Omit from consideration those cases where $|X_i - Y_i| = 0$;
- Step 3: Rank the remaining absolute differences, from smallest to largest, employing tied ranks where appropriate;
- Step 4: Assign to each such rank a '+' sign when the difference of $X_i - Y_i > 0$ and a '-' sign when the difference of $X_i - Y_i < 0$. T_+ is the sum of signed rank of positive and T_- is the sum of signed rank of negative;
- Step 5: Calculate the value of Z_c for the Wilcoxon test if the sample sizes are greater than 25, or calculate the value of T for small samples, which is equal to the minimum of the sum of the signed ranks $T = min(T_+, T_-)$.

All statements refer to a significance level of 5% [49].

References

[1] Ackley, D., Hinton, G., and Sejnowski, T., "A learning algorithm for Boltzmann machines", *Cognitive Science*, 9, 147–169, 1985.

[2] Aik Choon Tan and David Gilbert., "An empirical comparison of supervised machine learning techniques in bioinformatics", In the Proceedings of the *First Asia Pacific Bioinformatics Conference (APBC 2003)*, Adelaide, Australia. Sydney: Australian Computer Society. P. Chen (editor) Conferences in Research and Practice in Information Technology, 19: 219-222, 2003.

[3] Amit, D.J., Geutfreund, H., and Sompolinsky, H.,"Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks", *Phys. Rev. Lett.*, 55, 1530–1533, 1985.

[4] Anastasiadis A.D., Magoulas G.D., "Nonextensive statistical mechanics for hybrid learning of neural networks", *Physica A: Statistical Mechanics and its Applications*, vol.344, 372-382, 2004.

[5] Anastasiadis A.D., Magoulas G.D., "Nonextensive Entropy and Regularization for Adaptive Learning", in Proceedings of the *International Joint Conference on Neural Networks (IJCNN-04)*, Budapest, Hungary, vol. 2, 1067-1072, 2004.

[6] Anastasiadis A.D., Magoulas G.D., "Analysing the Localisation Sites of Proteins through Neural Networks Ensembles", *Neural Computing and Applications*, in press, 2006.

[7] Anastasiadis A.D., Magoulas G.D., Vrahatis M.N., "An efficient improvement of the Rprop algorithm." in Proceedings of the *1st International Workshop on Artificial Neural Networks in Pattern Recognition*, Florence, Italy, *IAPR2003*, 197-201, 2003.

[8] Anastasiadis A.D., Magoulas G.D., Vrahatis M.N., "Sign-based Learning Schemes for Pattern Classification", *Pattern recognition Letters*, vol. 26, 1926-1936, 2005.

[9] Anastasiadis A.D., Magoulas G.D., "Neural Network-based Prediction of Proteins Localisation Sites", in Proceedings of the *European Symposium on Intelligent Technologies, Hybrid and Adaptive Systems (EUNITE 2003)*, Oulu, Finland, July 2003.

[10] Arts E. H. L., and Korst, J., *Simulated Annealing and Boltzmann Machines*. New York: Wiley, 1989.

[11] Axelsson, O., Iterative Solution Methods, Cambridge Univ. Press, New York, 1996.

[12] Baba, N., Mogami, Y., Kohzaki, M., Shiraishi, Y., and Yoshida, Y., "A hybrid algorithm for finding the global minimum of error function of neural networks and its applications", *Neural Networks*, 7, 1253-1265, 1994.

[13] Battiti R., "First– and second–order methods for learning: between steepest descent and Newton's method", *Neural Computation*, 4, 141–166, 1992.

[14] Bishop, C.M. , Neural Networks for Pattern Recognition, Oxford: Oxford University Press, 1995.

[15] Boland M.V. and Murphy R.F., "After sequencing: quantitative analysis of protein localization", *IEEE Engineering in Medicine and Biology*, Sept/Oct., 115-119, 1999.

[16] Blattner F.R, Plunkett G, Bloch C.A, Perna N.T, Burland V, Riley M, Collado-Vides J, Glasner J.D, Rode C.K, Mayhew G.F, Gregor J, Davis N.W, Kirkpatrick H.A, Goeden M.A, Rose D.J, Mau B, Shao Y., (1997), "The complete genome sequence of Escherichia coli K-12", *Science*, 277(5331): 1453-1474, 1997.

[17] Blum, E.K., "Approximation of Boolean functions by sigmoidal networks: Part I: XOR and other two variable functions", *Neural Computation*, 1, 532–540, 1989.

[18] Breiman L., "Bagging predictors", *Machine Learning*, vol. 24: 123-140, 1996.

[19] Brewster, M. E. and Kannan, R., "Nonlinear successive over-relaxation", *Numer. Math.*, 44, 309–315, 1984.

[20] Blent Bolat and Tlay Yildirim., "A data selection method for Probabilistic Neural Networks" , International *XII. Turkish Symposium on Artificial Intelligence and Neural Networks, TAINN*, 2003.

[21] Burton R.M., and Mpitsos G.J., "Event dependent control of noise enhances learning in neural networks", *Neural Networks*, 5, 627-637, 1992.

[22] Cairns, P. Huyck, C. Mitchell, I. Wu, W., A "Comparison of Categorisation Algorithms for Predicting the Cellular Localization Sites of Proteins", Proceedings *IEEE International Workshop on Database and Expert Systems Applications*, 296-300, 2001.

[23] Chakrabarti C.G., and Kajal De, "Boltzmann-Gibbs Entropy: Axiomatic Characterization and Application", *International Journal of Mathematics and Mathematical Sciences*, Vol. 23, No. 4, 243251, 2000.

[24] Corana A., Marchesi, M., Martini, C., and Ridella, S., "Minimizing multimodal functions of continuous variables with the Simulated Annealing algorithm", *ACM Trans. Math. Soft.*, 13, 262–280, 1987.

[25] Dennis J.E., and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and nonlinear equations*, SIAM, Philadelphia, 1996. Originally published: Prentice Hall, Inc., New Jersey, 1983.

[26] Duda, R.O. and Hart.: Pattern Classification and Scene Analysis. John Wiley and Sons, 1973.

[27] Dugdale, J.S., Entropy and its Physical Meaning, Taylor and Francis Inc., Philadelphia, 1996.

[28] Engel, E., and C. Van den Broeck, "Statistical Mechanics of Learning", Cambridge University Press, 2001.

[29] Fahlman. S. E., "Faster-learning variations on backpropagation: an empirical study", In D.S. Touretzky, G.E. Hinton, and T.J. Sejnowski, editors, Proceedings of the 1988 Connectionist Models Summer School. Morgan Kaufmann, San Mateo, CA, 38–51, 1988.

[30] Fang, L and Li, T, "A globally optimal annealing learning algorithm for multilayer perceptrons with applications", in proc. *AI'90: Austrtalian Joint Conf. Artificial Intell.*, Perth Australia: World Scientific, 201-206, 1990.

[31] Fletcher, R., *Practical Methods of Optimization*, John Wiley & Sons, 1975.

[32] Fondecave R., and Brochard-Wyart, F., "Application of statistical mechanics to the wetting of complex liquids", *Physica A: Statistical Mechanics and its Applications*, Vol. 274, Issues 1-2, 19-29, 1999.

[33] Freund Y. and Schapire R. E.,(1996), "Experiments with a new boosting algorithm", in *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156, 1996.

[34] Gibbs, J. Willard., "Elementary Principles in Statistical Mechanics", 1902.

[35] Gilbert, J.C., and Nocedal,J., "Global convergence properties of conjugate gradient methods for optimization", *SIAM J. Optimization*, Vol. 2, 2142, 1992.

[36] Gell-Mann, M., and Tsallis, C., eds., *Nonextensive Entropy–Interdisciplinary Applications*, Oxford University Press, New York, 2004.

[37] Gill, P. E., Murray, W., and Wright, M. H, Practical Optimization, Academic Press, NY, 1981.

[38] Gori M. and Tesi A., "On the problem of local minima in backpropagation", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14, 76–85, 1992.

[39] Gupta A., and Lam, S.M., "Weight decay backpropagation for noisy data", *Neural Networks*, 11, 1127–1137, 1998.

[40] Gyorgyi, G., "Techniques of replica symmetry breaking and the storage problem of a McCulloch-Pitts neuron", *Physics Reports*, Vol. 342, issue 4-5, 263-392, 2001.

[41] Hagan M.T., and Menhaj, M.B., "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, 5, 989-993, 1994.

[42] Hansen L. K and Salamon P., "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12: 993-1001, 1990.

[43] Haykin, S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, 1994.

[44] Hopfield, J.J., "Neural Networks and physical systems with emergent collective computational abilities", *Proc. of the National Academy of Science, Biophysics*, 81, 3088-3092, 1982.

[45] Hoptroff R., and Hall, T., "Learning by diffusion for multilayer perceptron", *Electronic Letters*, 25, 531–533, 1989.

[46] Horton, P., and Nakai, K., "A probabilistic classification system for predicting the cellular localization sites of proteins", Proceedings of the *Fourth International Conference on Intelligent Systems for Molecular Biology*, 109-115, 1996.

[47] Horton, P., and Nakai, K., "Better Prediction of Protein Cellular Localization Sites with the k Nearest Neighbors Classifier", Proceedings of *Intelligent Systems in Molecular Biology*, 368-383, 1997.

[48] Huang, K., Statistical Mechanics, Wiley, New York, 1987.

[49] Igel, C. and Husken, M., "Empirical evaluation of the improved Rprop learning algorithms", *Neurocomputing*, 50, 105-123, 2003.

[50] Jacobs, R., "Increased rates of convergence through learning rate adaptation", *Neural Networks*, 1 (4), 295–307, 1988.

[51] Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection", *International Joint Conference on Artificial Intelligence*, 223-228, AAAI Press and MIT Press, 1995.

[52] Kirkpatrick, S., C.D. Gelatt Jr., and Vecchi, M.P., "Optimization by simulated annealing", *Science*, 220, 671–680, 1983.

[53] Krogh A. and Vedelsby J., "Neural network ensembles, cross validation, and active learning", in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 2: 650-659, 1995.

[54] Kuhn, R., Bos, J., "Statistical mechanics for neural networks with continuous-time dynamics", *J. Phys. Math.Gen. A*, 26, 831-857, 1993.

[55] Landsberg, Peter T., and Vlatko Vedral,"Distributions and channel capacities in generalized statistical mechanics", *Physics Letters A* 247, 211-217, 1998.

[56] Lavis, D.A., "Is equilibrium a useful concept in statistical mechanics?" *Conference on Philosophical and Foundational Issues in Statistical Physics*, Utrecht, November 2003.

[57] Laarhoven P.J.M.Vanand., Arts, E.H.L., *Simulated Annealing: Theory and Applications.* Dordrecht, The Netherlands: D. Reidel, 1988.

[58] Liang, P., B. Labedan, and M. Riley, "Physiological genomics of Escherichia coli protein families. *Physiol Genomics*, 9(1): 15-26, 2002.

[59] Liu R., Dong G., and Ling X., A convergence analysis for neural networks with constant learning ra tes and non-stationary inputs., In *Proceedings of the 34th Conference on Decision and Control,* New Orleans, 1278–1283, 1995

[60] Lebowitz, Joel L., "Statistical mechanics:A selective review of two central issues", *Reviews of Modern Physics*, Vol. 71, No. 2, 346-357, 1999.

[61] Lodish, H., Berk, A., Zipursky, S. L., Matsudaira, P., Baltimore, D. and James Darnell, J., Molecular Cell Biology , Freeman, 5th edn, 2003.

[62] Looney, C. G., "Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists", Oxford University Press, 171-172, 1997.

[63] Magoulas G.D., Vrahatis M.N., and Androulakis G.S., "On the alleviation of the problem of local minima in back–propagation", *Nonlinear Analysis: Theory, Methods and Applications*, 30, 4545–4550, 1997.

[64] Magoulas G.D., Vrahatis M.N., and Androulakis G.S., "Effective backptopagation with variables stepsize", *Neural Networks*, vol 10, 69–82, 1997.

[65] Magoulas G.D., Vrahatis M.N., Grapsa, T.N., and Androulakis, G.S., "Neural network supervised training based on a dimension reducing method", In: S.W. Ellacot, J.C. Mason, and I.J. Anderson (eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, 245–249, Kluwer, 1997.

[66] Magoulas G.D., Vrahatis M.N., and Androulakis G.S., "Improving the Convergence of the Backpropagation Algorithm Using Learning Rate Adaptation Methods", *Neural Computation*, 11, 1769–1796, 1999.

[67] Magoulas G.D. and Vrahatis M.N., "A Class of Adaptive Learning Rate Algorithms Derived by One-Dimensional Subminimization Methods", *Neural, Parallel and Scientific Computations*, 8, 147-168, 2000.

[68] Magoulas G.D., Plagianakos V.P., and Vrahatis M.N., "Globally convergent algorithms with local learning rates", *IEEE Tr. Neural Networks*, vol. 13, no. 3, 774-779, 2002.

[69] Mantegna, Rosario N., Palgyi Zoltn, and Stanley, H.Eugene., "Applications of statistical mechanics to finance", *Physica A: Statistical Mechanics and its Applications*, Vol. 274, Issues 1-2, 216-221, 1999.

[70] McCulloch, W.S. and Pitts, W. (1943). "A logical calculus of the ideas imminent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-133, 1943.

[71] Møller, M.F., "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, 6, 525–533, 1993.

[72] Murphy, P. M. and Aha, D. W., UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science, 1994, http://www.ics.uci.edu/ mlearn/MLRepository.html.

[73] Nakai, K. and Kanehisa, M., "Expert system for predicting protein localization sites in gram-negative bacteria", *PROTEINS: Structure, Function,*

and Genetics, 11: 95-110, 1991.

[74] Nakai, K. and Kanehisa, M., "A knowledge base for predicting protein localization sites in eukaryotic cells", *Genomics*, 14: 897-911, 1992.

[75] Nocedal J., "Theory of algorithms for unconstrained optimization", *Acta Numerica*, 1, 199–242, 1992.

[76] Neagu, D. and Palade, V., "A neuro-fuzzy approach for fuctional genomics data interpretation and analysis", *Neural Computing and Applications*, 12: 153-159, 2003.

[77] Nugent C D., Lopez J A., Smith A E., and Norman D. Black, "Prediction models in the design of neural network based ECG classifiers: A neural network and genetic programming approach", *BMC Medical Informatics and Decision Making* , 2(1), 2002.

[78] Opitz D. and Maclin R., "Popular Ensemble Methods: An Empirical Study", *Journal of Articial Intelligence Research*, 11: 169-198, 1999.

[79] Ortega, J. M. and Rheinboldt, W. C., "Iterative Solution of Nonlinear Equations in Several Variables", Academic Press, NY, 1970.

[80] Patnaik L.M., and Rajan, K., "Target detection through image processing and resilient propagation algorithms", *Neurocomputing*, 35, No. 1-4, 123–135, 2000.

[81] Pfister M., and Rojas, R., "Speeding–up backpropagation– A comparison of orthogonal techniques", *Proc. of the Joint Conference on Neural Networks*, Nagoya, Japan, 517–523, 1993.

[82] Pfister M., and Rojas, R., "Qrprop–a hybrid learning algorithm which adaptively includes second order information", *Proc. of the 4th Dortmund Fuzzy Days*, 55–62, 1994.

[83] Pfister M., and Rojas R., "Hybrid Learning Algorithms for Neural Networks-The adaptive Inclusion of Second Order Information", *Thesis*, 1995.

[84] Penrose, O., Foundations of Statistical Mechanics. A Deductive Treatment, International Series of Monographs in Natural Philosophy, vol. 22, Pergamon Press, Oxford, New York, Toronto, 1970.

[85] Picton, P., "Neural Networks", Second Edition, Grassroots Series, 2000.

[86] Plagianakos, V.P., Magoulas G.D., and Vrahatis, M.N., "Learning in multilayer perceptrons using global optimization strategies", *Nonlinear Analysis: Theory, Methods and Applications*, 47, 3431–3436, 2001.

[87] Plagianakos, V.P., Magoulas G.D., and Vrahatis, M.N., "Supervised training using global search methods", N. Hadjisavvas and P. Pardalos (eds.), *Advances in Convex Analysis and Global Optimization, vol. 54, Noncovex Optimization and its Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 421-432, 2001.

[88] Prechelt, L, , PROBEN1-A set of benchmarks and benchmarking rules for neural network training algorithms, Technical report 21/94, Fakultt fr Informatik, Universitt Karlsruhe, 1994.

[89] Rajagopal, A.K., Abe, S., *Phys. Rev. Lett.*, 83, 1711, 1999.

[90] Renyi, A., Wahrscheinlichkeitsrechmung, Deutscher Verlag der Wissenschften, Berlin, 1966.

[91] Riedmiller M., and Braun, H., "A direct adaptive method for faster backpropagation learning: The Rprop algorithm", *Proc. International Conference on Neural Networks*, San Francisco, CA, 586-591, 1993.

[92] Riedmiller, M., "Rprop - Description and Implementation Details", Technical Report, University of Karlsruhe, January, 1994.

[93] Riedmiller, M., "Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques", *Int. Journal of Computer Standards and Interfaces*, Special Issue on Neural Networks, 16(3), 265275, 1994.

[94] Ripley, B., "Statistical aspects of neural networks", in: J. Bornndor -Nielsen, J. Jensen, W. Kendal (Eds.), *Networks on Chaos: Statistical and Probabilistic Aspects*, Chapman and Hall, 1993.

[95] Rögnvaldsson, T., "On Langevin updating in multilayer perceptrons", *Neural Computation*, 6, 916–926, 1994.

[96] Rosenblatt, F. (1958). "The perceptron:a probabilistic model of information storage and organisation in the brain", *Psychological Review*, 65, 386-408.

[97] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning internal representations by error propagation", D.E. Rumelhart, J.L. McClelland (eds.), *Parallel Distributed Processing:Explorations in the Microstructure of Cognition 1*, MIT Press, 318–362, 1986.

[98] Rumelhart, D.E., Hinton, G.E, and Williams, R. J., "Learning internal representations by error propagation", In D. E Rumelhart and J. L. McClellend (Eda.), Parallel distributed processing: Explorations in the microstructure of cognition. 318-362, Cambridge, MIT Press, 1986.

[99] Rumelhart, D.E., and McClelland, J.L., "Parallel Distributed Processing", Cambridge, MIT press, 1986.

[100] Rumelhart, D.E., and McClelland, J.L., "Explorations in Parallel Distributed Processing", Cambridge, MIT press, 1988.

[101] Scales L.E. Introduction to non-linear optimization, MacMillan Publishers LTD, 34-35, 1985.

[102] Shang Y., and Wah, B., "Global optimization for neural network training", *IEEE Computer*, 45–54, 1996.

[103] Serra, P., Stanton, A.F., Kais, S., and Bleil, R. E., "Comparison study of pivot methods for global optimization", *The Journal of Chemical Physics*, Vol. 106, Issue 17, 7170-7177, 1997.

[104] Sharkey A.J.C., "On combining artificial neural nets", *Connection Science*, vol. 8: 299-314, 1996.

[105] Sharkey A.J.C., and Sharkey N.E., "Combining diverse neural nets", *The Knowledge Engineering Review*, 12: 231-247, 1997.

[106] Shepherd, J. Andrian, "Second-Order Methods for Neural Networks", Springer, 1997.

[107] Shibata, Hiroshi., "Statistics of phase turbulence II", *Physica A: Statistical Mechanics and its Applications*, Vol. 317, Issues 3-4, 391-400, 2003.

[108] Siekman, Will H., "The entropic index of the planets of the solar-system", *Chaos, Solitons and Fractals*, Vol. 16, Issue 1, 119-124, 2003.

[109] Sima, J., "Back Propagation is Not Efficient", *Neural Networks*, 6: 1017-1023, 1996.

[110] Sikorski, K., "Bisection is optimal", Numer. Math., 40, 111-117, 1982.

[111] Sikorski, K., Optimal Solution of Nonlinear Equations, Oxford University Press, New York, 2001.

[112] Stewart G.W., Introduction to Matrix Computations, Academic Press, New York, 1973.

[113] Silva, M.F., and Almeida, Luis B., "Speeding up backpropagation", In R. Eckmiller editor, *Advanced Neural Computers*, 151-158, 1990.

[114] Snedecor, G., and Cochran, W., Statistical Methods, Iowa State University Press, 8th edition, 1989.

[115] Styblinski M.A., and Tang, T.S., "Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing", *Neural Networks*, 3, 467-483, 1990.

[116] Steinbuch, K., "Die lernmatrix", *Kybernetic*, 1, pp 36-45, 1961.

[117] Steinbuch, K., and Piske, U., "Learning matrices and their application", *IEEE Trans. on Neural Computing*, EC-12, 846-862, 1963.

[118] Szasz, D., *Boltzmann's Ergodic Hypothesis, a Conjecture for Centuries?*, in D. Szasz, ed., Hard Ball Systems and the Lorentz Gas, Springer, 421-446, 2000.

[119] Szu, H., "Nonconvex optimization by fast simulated annealing", *Proceedings of IEEE*, 75, 1538–1540, 1987.

[120] Taruya Atsushi., and Masa-aki Sakagami., "Gravothermal catastrophe and Tsallis generalized entropy of self-gravitating systems", *Physica A: Statistical Mechanics and its Applications*, Vol. 307, Issues 1-2, 185-206, 2002.

[121] Thomas Udelhoven. and Brigitta Schutt., Capability of feed-forward neural networks for a chemical evaluation of sediments with diffuse reflectance spectroscopy, *Chemometrics and Intelligent Laboratory Systems*, 51: 9-22, 2000.

[122] Tolman, R.C., The principles of statistical mechanics. Clarendon Press, Oxford, 1938.

[123] Tollenaere T., "Supersab: Fast adaptive backpropagation with good scaling properties", *Neural Networks*, 3(5), 561 - 573, 1990.

[124] Treadgold N.K., and Gedeon T.D., "Simulated Annealing and Weight Decay in Adaptive Learning: The SARPROP Algorithm", *IEEE Tr. Neural Networks*, 9, 4, 662–668, 1998.

[125] Tsallis, C., "Possible Generalization of Boltzmann-Gibbs Statistics", *J. Statistical Physics*, 52(1–2),479–487, 1988.

[126] Tsallis, C., Mendes R.S., and Plastino, A.R., *Physica A: Statistical Mechanics and its Applications*, 261, 534-554, 1998.

[127] Tsallis, C., and Stariolo, D.A., "Generalized Simulated Annealing", *Physica A: Statistical Mechanics and its Applications*, 233, 395–406, 1996.

[128] Tsallis, C., "Nonextensive Statistics: Theoretical, Experimental and Computational Evidences and Connections", *Brazilian Journal of Physics*, vol. 29, no. 1, 1999.

[129] Tsallis, C., Anteneodo, C., Borland, L., and Osorio, R., "Nonextensive statistical mechanics and economics', *Physica A: Statistical Mechanics and its Applications*, vol. 324, Issues 1-2, 89-100, 2003.

[130] Tsallis, C., "What should a statistical mechanics satisfy to reflect nature", *Physica D*, 193, 3–34, 2004.

[131] Upadhyaya, A., Rieu, J., Glazier, J.A., and Sawada, Y., "Anomalous diffusion and non-Gaussian velocity distribution of Hydra cells in cellular aggregates", *Physica A: Statistical Mechanics and its Applications*, Vol. 293, Issues 3-4, 549-558, 2001.

[132] Varga, R., Matrix Iterative Analysis, Second Edition, Springer–Verlag, Berlin, 2000.

[133] Van Belle D. and Andre B., "A genomic view of Yeast membrane transporters", *Current Opinion in Cell Biology*, 13, No. 4, 389-98, 2001.

[134] Vogl T. P., Mangis J. K., Rigler A. K., Zink W. T. and Alkon D. L., "Accelerating the convergence of the back-propagation method", *Biological Cybernetics, 59, 257-263, 1988.*

[135] Van der Smagt P.P., "Minimization Methods for training feedforward neural networks", *Neural Networks*, 7, 1–11, 1994.

[136] Vicsek, T., Andrs Czirk, Ills J. Farkas and Dirk Helbing, "Application of statistical mechanics to collective motion in biology", *Physica A: Statistical Mechanics and its Applications*, Vol. 274, Issues 1-2, 182-189, 1999.

[137] Vrahatis M.N., "Solving systems of nonlinear equations using the nonzero value of the topological degree", *ACM Transactions Mathematical Software*, 14, 312–329, 1988.

[138] Vrahatis M.N., "CHABIS : A mathematical software package for locating and evaluating roots of systems of nonlinear equations", *ACM Transactions Mathematical Software*, 14, 330–336, 1988.

[139] Vrahatis M.N., Magoulas G.D. and Plagianakos V.P., "Globally convergent modification of the Qprop method", *Neural Processing Letters*, 12, 2, 159-170, 2000.

[140] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos and G.D. Magoulas, "A class of gradient unconstrained minimization algorithms with adaptive stepsize", *Journal of Computational and Applied Mathematics*, 114, No. 2, 367–386, 2000.

[141] Vrahatis M.N., Magoulas G.D. and Plagianakos V.P., "From linear to nonlinear iterative methods", *Applied Numerical Mathematics*, 45, 1, 59-77, 2003.

[142] Wehrl, A., "General properties of entropy", Rev. Modern Phys. 50, no. 2, 221–260, 1978.

[143] Weslstead, S.T., *Neural network and fuzzy logic applications in C/C++*, Wiley, 1994.

[144] Wolfe P., "Convergence conditions for ascent methods", *SIAM Review*, 11, 226–235, 1969.

[145] Wolfe P., "Convergence conditions for ascent methods. II: Some corrections", *SIAM Review*, 13, 185–188, 1971.

[146] Wolf, M., "Applications of statistical mechanics in number theory", *Physica A: Statistical Mechanics and its Applications*, Vol. 274, Issues 1-2, pp 149-157, 1999.

[147] Young D., "Iterative methods for solving partial difference equations of elliptic type", *Trans. Amer. Math. Soc.*, 92-111, 1954.

[148] Zenobi, G., Cunningham, P., "Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error", In Proceedings of the *European Conference on Machine Learning*, pp 576-587, 2001.