UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Training and Optimization of Product Unit Neural Networks

by

Adiel Ismail

Submitted in partial fulfillment of the requirements

for the degree

Master of Science

in the Faculty of Natural & Agricultural Science

at

The University of Pretoria

Pretoria

14 November 2001

Declaration

I the undersigned hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Adiel Ismail

14 November 2001

Abstract

Product units in the hidden layer of multilayer neural networks provide a powerful mechanism for neural networks to efficiently learn higher-order combinations of inputs. Training product unit neural networks using local optimization algorithms is difficult due to an increased number of local minima and increased chances of network paralysis. This thesis discusses the problems with using local optimization, especially gradient descent, to train product unit neural networks, and shows that particle swarm optimization, genetic algorithms and leapfrog are efficient alternatives to successfully train product unit neural networks. Architecture selection, i.e. pruning, of product unit neural networks is also studied and applied to determine near optimal neural network architectures that are used in the comparative studies.

Opsomming

Produk-eenhede in die versteekte laag van multi-vlak neurale netwerke verskaf 'n kragtige meganisme aan neurale netwerke om hoë-orde kombinasies van invoer doeltreffend aan te leer. Die leer van neurale netwerke met produk-eenhede word bemoeilik weens die verhoogde aantal lokale minima teenwoordig, asook die verhoogde kans om netwerk paralise te ondervind. Hierdie tesis spreek die probleme aan wanneer lokale optimeringsmetodes gebruik word, veral in die geval van gradientdaling om produk-eenheid neurale netwerke te leer en dui aan dat partikel swerm optimering, genetiese algoritmes en *'leapfrog'* optimering baie doeltreffende alternatiewe is om produk-eenheid neurale netwerke te leer. Argitektuurseleksie, of te wel besnoeiing, van produk-eenheid neurale netwerke word ook bestudeer en toegepas om optimale neurale netwerk argitekture te bepaal, wat gevolglik in die vergelykende studies gebruik word.

Acknowledgements

I wish to thank the following people,

- my supervisor, Prof AP Engelbrecht, for all the positive criticism and useful suggestions.
- my wife, Shirene for her patience and moral support.
- my friend and colleague, Michael Norman, for his advice and encouragement.
- my colleague, Anwar Vahed, for the assistance provided during this project.
- my mother who single-handedly, as a widow from 1980, reared me, my three brothers and sister.

Contents

1 Introduction **1**

1.1 Why Product Unit Neural Networks? 2

1.2 Problems with Training Product Unit Neural Networks using Gradient Descent 3

1.3 Global Optimization Algorithms to Train PUNNs 4

1.4 Objectives 4

1.5 Outline of the Thesis 6

2 Background **8**

2.1 A Brief History of ANNs 8

2.2 What is An Artificial Neural Network? 10

2.3 Advantages of Neural Networks 11

2.4 Limitations of Neural Networks 12

2.5 Why Artificial Neural Networks? 12

2.6 Classes of ANN Applications 13

2.7 A Typical Artificial Neural Network 15

2.8 Learning 16

2.9 Model of An Artificial Neuron 17

2.10 Network Architectures 19

2.11 Activation Functions 24

2.12 Learning Paradigms 26
2.12.1 Error-Correction Learning Rule 26
2.12.2 Hebbian Learning 27
2.12.3 Competitive Learning 29
2.12.4 Stochastic Learning 30
2.13 Learning Paradigms 32
2.13.1 Supervised Learning 32
2.13.2 Unsupervised Learning 32
2.13.3 Reinforcement Learning 33
2.14 Modes of Learning 33
2.15 Performance Measures 35
2.15.1 True Error versus Empirical Error 35
2.16 Approximation Capabilities of Feed-Forward Neural Networks 37
2.16.1 Generalization 38
2.17 Architecture Selection 39
2.18 Back-propagation 45
2.18.1 Overview of Back-propagation 46
2.18.2 The Effect of the Learning Rate 48
2.18.3 The Effect of Momentum on Back-propagation 49
2.18.4 On-line Implementation of Back-propagation 51
2.18.5 Terminating criteria 53
2.18.6 Initialization 53
2.19 Conclusion 55

3 Higher-Order Neural Networks 56
3.1 Sigma-Pi Networks 56
3.2 Pi-Sigma Networks 58

3.3 Functional Link Networks . . . 61
3.4 Second-Order Neural Networks . . . 62
3.5 Product Unit Neural Networks . . . 63
3.6 Product Unit Training Rule . . . 66
3.7 The Bias Unit . . . 68
3.7.1 Case 1 PUNNs . . . 69
3.7.2 Case 2 PUNNs . . . 70
3.7.3 The Distortion Unit . . . 72
3.8 Problems with Training of PUNN using Gradient Descent . . . 74
3.9 Conclusion . . . 79

4 Global Optimization Algorithms 80

4.1 Particle Swarm Optimization . . . 80
4.1.1 PSO Algorithm . . . 83
4.1.2 Inertia Weight . . . 84
4.1.3 Maximum Velocity . . . 84
4.1.4 Acceleration Constants . . . 85
4.1.5 Applications of PSO . . . 85
4.1.6 Advantages of PSO . . . 86
4.2 Genetic Algorithms . . . 88
4.2.1 Applications of GAs . . . 89
4.2.2 Genetic Algorithm . . . 90
4.2.3 Initialization and Size of Population . . . 91
4.2.4 Representation . . . 91
4.2.5 Fitness . . . 92
4.2.6 Crossover . . . 93
4.2.7 Mutation . . . 93

4.2.8 Reproduction or Selection 94
4.2.9 Advantages of GAs 95
4.2.10 Disadvantages of GAs 96
4.3 Leapfrog 97
4.3.1 Leapfrog Algorithm 97
4.4 Experimental Results 98
4.4.1 Test Functions 99
4.4.2 Performance Criteria 101
4.5 Experimental procedure 102
4.5.1 Parameters for the Optimization Methods 103
4.6 Optimizing the Parameters 105
4.6.1 Optimal Parameters for PSO 105
4.6.2 Optimal Parameters for BP 107
4.6.3 Optimal parameters for GA 110
4.6.4 Optimal parameters for LFOP 110
4.7 Initial Neural Network Architectures 112
4.8 Best Configuration for SUNNs and PUNNs 114
4.9 Comparison Between PUNNs Containing Bias and Distortion Units . . 117
4.10 Comparison of Global Optimization Algorithms 119
4.11 Analysis of Results 120
4.12 Conclusion 128

5 Architecture selection 140

5.1 Overview of Sensitivity Analysis 144
5.2 The Variance Nullity Pruning Approach 145
5.3 Sensitivity equations 150
5.3.1 Output-Hidden Layer Analysis 151

5.3.2 Output-Input Layer Analysis 151

5.4 Application of the Variance Nullity Pruning Algorithm to PUNNs . . . 153

5.5 Conclusion 155

6 Conclusions **167**

6.1 Possible Improvements and Future Research 170

Bibliography **172**

A Derivation of learning rules for PUNNs **192**

A.1 Learning rules for a PUNN using a bias unit 194

A.2 Learning rules for PUNN using a distortion unit 199

B Publications from this thesis **202**

C Symbols and notation **203**

List of Tables

4.1 Range of values for inertia weight for PSO . . . 105
4.2 Range of values for maximum velocity for PSO . . . 106
4.3 Range of values for acceleration constant for PSO . . . 106
4.4 Best parameters for PSO using PUs . . . 106
4.5 Best parameters for PSO using SUs . . . 107
4.6 Intervals for initial weights for BP . . . 108
4.7 Range of values for learning rate for BP . . . 108
4.8 Range of values for momentum for BP . . . 108
4.9 Best parameters for BP using PUs . . . 109
4.10 Best parameters for BP using SUs . . . 109
4.11 Range of values for crossover . . . 110
4.12 Range of values for mutation . . . 110
4.13 Best parameters for GA using PUs . . . 111
4.14 Best parameters for GA using SUs . . . 111
4.15 Range of values for δ . . . 112
4.16 Range of values for δ_1 . . . 112
4.17 Range of values for Δt . . . 112
4.18 Best parameters for LFOP using PUs . . . 113
4.19 Best parameters for LFOP using SUs . . . 113
4.20 Initial configuration for PUNNs . . . 114

4.21 Initial configuration for SUNNs 115

4.22 Configuration for PUNNs 116

4.23 Configuration for SUNNs 116

4.24 Comparison of MSEs on PUNN containing a bias and a PUNN containing a distortion unit 118

4.25 Mean squared error results for PUs 121

4.26 Mean squared error results for SUs 122

4.27 Epochs needed to reach MSE levels 133

4.28 Epochs needed to reach MSE levels 134

4.29 Epochs needed to reach MSE levels 135

4.30 Percentage simulations that converged to MSE levels 136

4.31 Percentage simulations that converged to MSE levels 137

4.32 Percentage simulations that converged to MSE levels 138

4.33 Percentage simulations that converged to MSE levels 139

5.1 Pruning of hidden units - function F1 156

5.2 Pruning of hidden units - function F2 157

5.3 Pruning of hidden units - function F3 158

5.4 Pruning of hidden units - function F4 159

5.5 Pruning of hidden units - function F5 160

5.6 Pruning of hidden units - function F6 161

5.7 Pruning of hidden units - function F7 162

5.8 Pruning of hidden units - function F8 163

5.9 Pruning of input units - function F1 164

5.10 Pruning of input units - function F2 165

5.11 Pruning of input units - function F4 166

List of Figures

2.1 Model neuron using a Summation Unit 18
2.2 Typical Recurrent Neural Networks 21
2.3 Multilayer feed-forward Neural Network 23

3.1 Sigma-pi network 57
3.2 Pi-Sigma Network 60
3.3 Functional Link Network 61
3.4 Two types of PUNNs 64
3.5 Case 1 PUNN 69
3.6 Case 2 PUNN 71
3.7 PUNNs with a distortion unit 72
3.8 Effect of the distortion unit in approximating $f(z) = z^2$ 73
3.9 MSE values with weight w fixed at 1 for $f(z) = z^2$ 77
3.10 Error surface for the straight line between 3 minima, $f(z_1, z_2) = z_1^2 + z_2^2$ 78
3.11 PUNN to approximate $f(z_1, z_2) = z_1^2 + z_2^2$ 79

4.1 Functions to be approximated 100
4.2 Functions to be approximated 101
4.3 Learning profiles for functions F1, F2 and F3 130
4.4 Learning profiles for functions F4, F5 and F6 131
4.5 Learning profiles for functions F7 and F8 132