Chapter 1

Introduction

The recent resurgence of interest in neural networks can be ascribed to the recognition that the brain does not perform calculations in the same way as conventional (Von Neumann) computers. Despite the fact that computers execute instructions at extremely fast speeds, human beings whose brains operate at much slower speeds still outperform computers at tasks such as speech recognition, face recognition, etc. The human brain consists of an extremely large number of interconnected nerve cells, or neurons, which operate in parallel to process information. An artificial neural network (ANN) is an information processing system that mimics the structure and operating principles found in the information processing systems of human beings. The study of neural networks is one of the most rapidly expanding fields attracting researchers from a wide variety of disciplines such as biology, engineering, linguistics, mathematics, medicine, neuroscience, physics, psychology and statistics. ANNs have been applied successfully in many applications such as speech recognition [Cohen *et al* 1993], handwritten character recognition [Guyon 1990], steering of an autonomous vehicle [Pomerlau 1989], medical diagnosis of heart attacks [Harrison *et al* 1991], radar target detection and classification [Haykin *et al* 1992], and many more.

1.1 Why Product Unit Neural Networks?

Standard neural networks use summation units (SUs), where the net input signal to a unit is the weighted sum of the inputs connected to that unit. Research has shown that these summation unit neural networks (SUNNs) can approximate any continuous function to an arbitrary degree of accuracy, provided that the hidden layers contain a sufficient number of hidden units [Funahashi 1989, Hornik *et al* 1989a]. However, these networks require a large number of summation units (SUs) when approximating complex functions that involve higher order combinations of its inputs [Leerink *et al* 1995]. When approximating polynomials, higher-order combinations of inputs, such as x^3y^7, are often required. Networks that utilize higher-order combinations of its inputs will greatly reduce the number of processing units required to represent these complex functions [Janson *et al* 1993].

Several neural network models have been developed to gain an advantage in using higher-order terms [Gurney 1992, Leerink *et al* 1995, Redding *et al* 1993]. Examples of these higher-order neural networks are: pi-sigma network (PSN) [Ghosh *et al* 1992], sigma-pi networks [Lee Giles 1987], second-order neural networks [Milenković *et al* 1996] and functional link neural networks [Ghosh *et al* 1992, Hussain *et al* 1997, Zurada 1992]. An alternative network that also employs higher-order terms, is a product unit neural network (PUNN), where the net input is now a product of terms; each term consisting of an input raised to a weight [Durbin *et al* 1989]. Advantages of PUNNs are increased information capacity and the ability to form higher-order combination of inputs. Durbin and Rumelhart determined empirically that the information capacity of product units PUs (as measured by their capacity for learning random boolean patterns) is approximately $3N$, compared to $2N$ of a SU network for a single threshold logic function, where N denotes the number of inputs to

the network [Durbin *et al* 1989].

The next section briefly describes the problems associated with the training of PUNNs using back-propagation by gradient descent.

1.2 Problems with Training Product Unit Neural Networks using Gradient Descent

The back-propagation algorithm, independently developed by Werbos [Werbos 1974] and Bryson [Bryson *et al* 1969], provides a computationally efficient method for the training of multilayer neural networks. Its greatest strength is in finding non-linear solutions to ill-defined problems [Haykin 1994]. Unfortunately, the search space for PUNNs can be extremely convoluted, with numerous local minima that trap gradient descent [Durbin *et al* 1989, Leerink *et al* 1995]. While it is possible for local minima to occur in SU networks, they are particularly prevalent in networks containing PUs, due to the effect of the exponential terms in the learning equations. Thus, while PUs increase a neural network's capability, they also add complications in the training process. Although gradient descent has shown to be successful in training SUNNs, gradient descent fails to train PUNNs in general. The reason for its failure is discussed in more detail in section 3.8 on page 74. Gradient descent requires auxiliary information such as function derivatives, in order to calculate the minimum. The search space of PUNNs have an increased number of local minima, deep ravines and valleys, often surrounded by steep gradients that lead to huge adjustments of the weights when gradient descent is used, and consequent saturation.

1.3 Global Optimization Algorithms to Train PUNNs

What is needed, is a global optimization method instead of gradient descent, which is a local optimizer, to allow searching for larger parts of the search space, and which has the ability to get out of local minima. Genetic algorithms (GA) and particle swarm optimization (PSO) are global optimization methods that do not require auxiliary information, such as function derivatives, about the function being approximated in order to calculate the minimum. Leapfrog optimization (LFOP), a derivative based global optimizer, will also be used in training PUNNs. Each optimization algorithm has its own set of parameters. Optimal parameters are determined for each of these optimization algorithms. This thesis is dedicated to the training of PUNNs using PSO, GA and LFOP. Architecture selection, i.e. pruning, of PUNNs is also studied and applied to determine near optimal neural network architectures. The variance nullity pruning algorithm of Engelbrecht is applied to PUNN to determine near optimal network architectures [Engelbrecht *et al* 1999c, Engelbrecht 2001].

1.4 Objectives

The main objective of this thesis is to illustrate that gradient descent fails to train PUNNs and to show that global optimization algorithms, such as particle swarm optimization, genetic algorithms and leapfrog optimization, are more successful at training PUNNs.

The second objective is to determine which global optimization algorithm is more efficient and robust in training PUNNs. This thesis assumes a PUNN architecture with a

bias (for an explanation of a bias, refer to section 2.7 on page 15) to the output units and no bias to the hidden units. Instead, an extra unit, referred to as a 'distortion unit', is included in the hidden layer (refer to section 3.7.3 on page 72 for an explanation of the distortion unit). In this case, product units compute the net input signal as:

$$net_{y_j} = \prod_{i=1}^{I+1} z_i^{v_{ji}} \tag{1.1}$$

instead of

$$net_{y_j} = \prod_{i=1}^{I} z_i^{v_{ji}} + z_{I+1} \cdot v_{j,I+1} \tag{1.2}$$

where net_{y_j} is the net input to hidden unit Y_j, I is the total number of input units, Z_i is an input unit, z_i is an input signal to unit Z_i, v_{ji} is the weight between input unit Z_i and hidden unit Y_j. In equation (1.2), the threshold (or bias) is denoted by $v_{j,I+1}$, z_{I+1} is the input to the bias unit and has a value of -1. In equation (1.1), z_{I+1} cannot be viewed as the input to the bias, since it does not perform the function of a bias, i.e. it does not act as an offset to the other hidden units, but rather distorts the activation function to more accurately fit the data. In this case, z_{I+1} is now referred to as the input to the 'distortion' unit, Z_{I+1} with value -1, clearly distinguishing it from a bias unit. Note, equation (1.1) does not contain a bias to the hidden units. The linear activation function is assumed for the hidden and output units of the PUNNs, while the sigmoidal activation function is assumed for the hidden and output units of the SUNNs.

The third objective is to find the smallest architecture in training PUNNs for a particular function using the variance nullity pruning algorithm of Engelbrecht [Engelbrecht *et al* 1999c, Engelbrecht 2001]. The thesis also compares the pruned architectures of PUNNs and SUNNs to determine whether there is any gain in architecture complexity and performance using PUs.

1.5 Outline of the Thesis

The thesis is organized as follows. Chapter 2 presents an overview of ANNs. Topics covered include: description of a typical ANN, advantages and limitations of ANNs, training of ANNs, types of network architectures, activation functions, classification of learning rules, different learning paradigms, performance measures for ANNs, approximation capabilities of feed-forward neural networks, architecture selection and back-propagation by gradient descent.

An overview of the different higher-order neural networks such as pi-sigma, sigma-pi, functional link and second-order neural networks is given in chapter 3. A product unit neural network is described and the training rule for PUNNs presented. The problems associated with training of PUNNs using gradient descent are also investigated.

Chapter 4 is dedicated to a detailed discussion of genetic algorithms, particle swarm optimization and leapfrog optimization. Optimal parameters for each optimization algorithm are determined for each of the test functions. Results of these global optimization algorithms applied to 8 test functions are also discussed.

Chapter 5 discusses pruning of artificial neural networks. The variance nullity pruning algorithm of Engelbrecht is then adapted to prune PUNNs [Engelbrecht *et al* 1999c, Engelbrecht 2001]. Results of pruning are tabulated for PUNNs.

Chapter 6 summarizes the main conclusions and the goals, shortcomings and possible

improvements are suggested.