THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學
Pao Yue-kong Library
包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

Pao Yue-kong Library, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

http://www.lib.polyu.edu.hk

Output Sensitivity of MLPs Derived from Statistical Expectation

Xiaoqin Zeng

PhD

The Hong Kong Polytechnic University

January 2002

Pao Yue-Kong Library
PolyU • Hong Kong

Abstract

The sensitivity of a neural network's output to its parameter perturbation is an important issue in the design and implementation of neural networks. What will be the effects of parameter perturbation on the output of neural networks? How does one measure the degree of the response of neural networks due to parameter perturbation? The objective of this dissertation is to analyse and quantify the sensitivity of the most popular and general feedforward neural network — the Multilayer Perceptron (MLP) to input and weight perturbations.

Based on the structural features of the MLP, a bottom-up approach is followed to study the sensitivity of the MLP. The sensitivity for each neuron is computed in an order from the first layer to the last. Then the results of the neurons in a layer are collected to form the sensitivity for that layer. Finally the sensitivity of the output layer is defined as the sensitivity for the entire neural network.

Sensitivity is defined as the mathematical expectation of output deviations due to input and weight deviations with respect to overall input and weight values in a given continuous interval. An analytical expression that is a function of input and weight deviations is approximately derived for the sensitivity of a single neuron. Two algorithms are then presented to compute the sensitivity for an entire neural network. By analyzing the derived analytical formula and executing one of the given algorithms, some significant observations on the behavior of the MLP under input and weight perturbations are discovered, which can be used as guidelines to aid the design

of an MLP. As intuitively expected, the sensitivity increases with input and weight perturbations, but the increase has an upper bound that is determined by the structural configuration of the MLP, namely the number of neurons per layer and the number of layers. There exists an optimal value for the number of neurons in a layer, which yields the highest sensitivity value. The sensitivity decreases with the number of layers increasing, and the decrease almost levels off when the number becomes large.

Similarly a quantified sensitivity measure to input deviation is developed for a specific MLP with fixed weight and thus fixed network architecture. Based on the derived analytical expressions, two algorithms are given for the computations of the sensitivity of a single neuron and the sensitivity of an entire neural network. The sensitivity measure is a useful means to evaluate the networks' performance such as error-tolerance and generalization capabilities.

The applications of the sensitivity analysis to hardware design, and the sensitivity measure to the selection of weights for a more robust MLP are discussed.

Acknowledgements

There have been many people who have, in various ways, helped me in the development and completion of this dissertation. I would like to take this opportunity to express my thanks to all of them. A few are particularly acknowledged below.

First of all, I am most appreciative of the encouragement, support, and advice that I have received from Prof. Daniel S. Yeung, my chief supervisor, throughout my doctoral studies at Hong Kong Polytechnic University. I have learned a great deal from Prof. Yeung.

My appreciation is extended to Dr. Jiannong Cao, my good friend, for his introducing me to Prof. Yeung and his help during my stay in Hong Kong.

I give my extreme thanks to my wife Yajuan Huang for her understanding and many sacrifices in helping me to complete this research. Her understanding is the strongest support for my study. This dissertation is dedicated to her.

Last, but by no means least, I am greatly indebted to my parents for their endless love and care.

Contents

Abstract ii

Acknowledgments iv

1 Introduction **1**

1.1 Motivation 1
1.2 Problem statement 2
1.3 The Multilayer Perceptron 4
1.3.1 Notation 5
1.3.2 Structure 6
1.3.3 Characteristics of the logistic function 8

2 Related studies on sensitivity **9**

2.1 Sensitivity of Adlines and Madlines 9
2.2 Sensitivity of Neocognitrons 12
2.3 Sensitivity of Multilayer Perceptrons 13
2.4 Application of sensitivity study 18
2.5 Summary 19

3 Research methodology **21**

3.1 Definition of sensitivity for a single neuron 21
3.2 Definition of sensitivity for a layer 23
3.3 Definition of sensitivity for an entire network 24
3.4 Summary 26

4 Sensitivity analysis for an ensemble of MLPs **28**

4.1 Sensitivity for neurons 28
4.2 Sensitivity for networks 32
4.2.1 Recursive algorithm 33
4.2.2 Iterative algorithm 34
4.3 Analysis of the Sensitivity 35
4.3.1 Effects of parameters on the sensitivity of neurons 36
4.3.1.1 Input and weight deviations 36
4.3.1.2 Dimension of input 37

4.3.2 Effects of parameters on the sensitivity of networks 39
4.3.2.1 Input and weight deviations 39
4.3.2.2 Number of neurons 40
4.3.2.3 Number of layers 41
4.4 Simulation verification 42
4.5 Summary 43

5 Sensitivity measure for an individual MLP 46

5.1 Sensitivity for neurons 46
5.2 Sensitivity for networks 49
5.3 Simulation verification 50
5.4 Summary 54

6 Applications of the sensitivity study 56

6.1 Hardware design using sensitivity analysis 56
6.2 Selection of weights using sensitivity measure 57
6.3 Summary 61

7 Conclusion and future work 63

7.1 Conclusion 63
7.2 Suggestions for future research 64
7.2.1 Different assumptions 65
7.2.2 Other kinds of sensitivity 65
7.2.3 Application of the MLP sensitivity results 66

A. Derivation of the sensitivity formula for a neuron 67

B. Three integral formulas 70

C. Approximate integration of the logistic function 73

Bibliography 89

List of Figures

1.1 An unperturbed MLP 4
1.2 A perturbed MLP 4
1.3 The structure of a neuron (i^{th} neuron in layer l) 7
1.4 The structure of the MLP 7

4.1 Sensitivity of a neuron to input and weight deviations 37
4.2 Sensitivity of a neuron to the dimension of input 37
4.3 Sensitivity of MLPs to input and weight deviations 41
4.4 Sensitivity of MLPs to the number of neurons in hidden layer 41
4.5 Sensitivity of MLPs to the dimension of input 42
4.6 Sensitivity of MLPs to the number of neurons in the 1st layer 42
4.7 Sensitivity of MLPs to the number of neurons in the 2nd layer 42
4.8 Sensitivity of MLPs to the number of layers 42

5.1 Sensitivity of the neurons with 2-dimensional weights 52
5.2 Sensitivity of the neurons with 3-dimensional weights 52
5.3 Sensitivity of the neurons with 4-dimensional weights 52
5.4 Sensitivity of the neurons with 5-dimensional weights 52
5.5 Sensitivity of the MLPs to input deviation 54
5.6 Average output deviation of the MLPs to input deviation 54

6.1 Theoretical results for the two MLPs 59
6.2 Simulation results for the two MLPs 59
6.3 Simulation results for the two MLPs at input (0.2,0.8) 60
6.4 Simulation results for the two MLPs at input (0.5,0.5) 60
6.5 Simulation results for the two MLPs at input (0.8,0.2) 61
6.6 Simulation results for the two MLPs at input (0.02,0.08) 61
6.7 Simulation results for the two MLPs at input (0.05,0.05) 61
6.8 Simulation results for the two MLPs at input (0.08,0.02) 61

B.1 The region is the entire Ω 70
B.2 The region is the lower part of Ω 70
B.3 The region is a prism part of Ω 70

C.1 P doesn't cut Ω 78
C.2 P cuts the parallel sides of Ω 78

C.3 P cuts a prism from Ω 80
C.4 P cuts a prism from Ω 80
C.5 The region of Ω_1 81
C.6 The region of Ω_2 81
C.7 The region of Ω_3 81
C.8 The extended region of Ω_3 81
C.9 Ω is divided into two smaller rectangles 87
C.10 Ω is divided into two smaller rectangles 87
C.11 Ω is extended to a bigger rectangle 88

Chapter 1

Introduction

The research on artificial neural networks is aimed at exploring and simulating the mechanism of human brain to achieve human-like function for solving difficult-to-mechanize problems such as recognition, perception, learning, etc. The history of the research underwent a way with ups and downs. Its beginning may be dated back to as early as 1943 when the neurophysiologist McCulloch and the logician Pitts put forward a mathematical model [1] for an artificial neuron. Although a very substantial amount of research work on neural networks has been conducted, it is yet to achieve the ultimate goal, namely, using neural networks to simulate the working of a human brain. Many fundamental questions about the architecture and the behavior of neural networks are still in need of exploring. This dissertation concerns the sensitivity study of the Multilayer Perceptron [2] [3] [4] to its input and weight perturbations.

1.1 Motivation

When considering the design of a neural network, how can network designers determine whether the design would satisfy application requirements under limited precision of digital and analog hardware as well as environmental noise disturbance? How can a more robust neural network be designed and implemented? What will be an appropriate measure to evaluate neural networks' performance such as error-

tolerance and generalization capability? The sensitivity of a neural network's output to its parameters' perturbations is expected to be a key to answer these questions.

After the birth of the Back-Propagation (BP) learning algorithm [5], the Multilayer Perceptron (MLP) became the most popular neural network with a wide range of applications. Theoretically, an MLP is to realize an input-output mapping by selecting a set of weights during the training stage with its learning mechanism. It has been proved that any continuous function can be approximately realized by an MLP with monotonically increasing differentiable activation function [6] [7]. Therefore trained MLPs can be used to solve many problems in the fields of adaptive modeling, pattern recognition, classification, function approximation, inductive learning, etc. It is quite clear that the study on the sensitivity of the MLP has both theoretical and practical values.

An MLP is a supervised learning system. It changes its state by adjusting its tunable parameters to fit the training data and gives the response to the unseen data. The most important tunable parameter is the connection weight. Besides the weight, the training data and the unseen input data are among the other factors that affect the neural network's performance. Thus, the input and weight perturbations are two important factors to be considered when studying the sensitivity of the MLP.

1.2 Problem Statement

The MLP is used to model complex functional relationships between its input and output. This is particularly useful when an analytical model of a process either does

not exist or is not known. Figure 1.1 presents the ideal situation for an MLP. But in real life situations, input and weight may deviate from their ideal values as Figure 1.2 illustrates. Weight deviation in neural networks could be introduced by either hardware imprecision or inadequate training processing, and input perturbation is unavoidable because of the noisy environment. How sensitive is the MLP's output to input and weight deviations? Is there any other factor that may affect the neural networks' output sensitivity due to input and weight deviations? What is an appropriate way to quantify the sensitivity for the performance measurement of the MLP? The objective of this dissertation is to search for answers to these problems.

The first goal of this study is to establish a function that expresses a relationship between the output deviation of an MLP and its input and weight deviations. The mapping is expected to be in the form of

$$\Delta Y = S(\Delta X, \Delta W) \tag{1.1}$$

or

$$\Delta Y = S(\Delta X, W), \tag{1.2}$$

where ΔY is the output deviation vector, ΔX is the input deviation vector, and ΔW is the weight deviation matrix. The mapping $S(\Delta X, \Delta W)$ will then be analyzed to provide information on the sensitivity behavior of the MLP with respect to its input and weight deviations as well as other parameters such as the number of neurons per layer and the number of layers. It would be particularly useful if, for any specific

MLP with given weight, i.e., a fixed network structure, $S(\Delta X, W)$ can provide a quantified measure of the network sensitivity.

In figures 1.1 and 1.2, an MLP is regarded as a black box. But, in order to find solution for the above proposed problems, it is necessary to understand the structure and operation principles of the MLP.

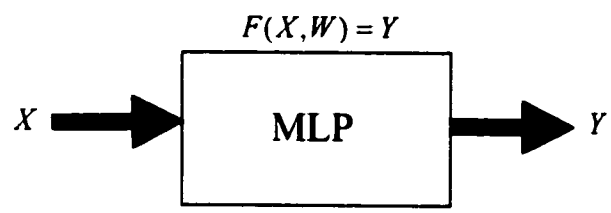

Figure 1.1: An unperturbed MLP.

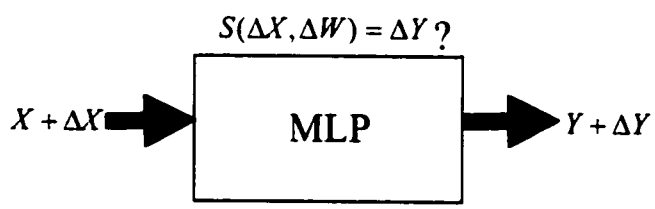

Figure 1.2: A perturbed MLP.

1.3 The Multilayer Perceptron

The MLP with continuous input, weight, and output is the most general feed-forward neural network. The external inputs of the network are fed into the first-layer neurons. The outputs from the first-layer neurons then serve as inputs to the second-layer

neurons, and so on. The outputs of the final layer, also called output layer, are the outputs of the network.

1.3.1 Notation

Generally, an MLP can be assumed to have L layers, and each layer l $(1 \le l \le L)$ has n^l $(n^l \ge 1)$ neurons. The form n^0-n^1-...-n^L is used to represent an MLP with certain structural configuration, in which each n^l $(0 \le l \le L)$ not only indicates the number of neurons (n^0 is an exception which indicates the dimension of the input) but also stands for a layer from left to right including the input layer. n^0 stands for the input layer and n^L stands for the output layer. For a neuron i $(1 \le i \le n^l)$ in layer l, its input vector is $X^l = (x_1^l, \ldots, x_{n^{l-1}}^l)^T$, its weight vector is $W_i^l = (w_{i1}^l, \ldots, w_{in^{l-1}}^l)^T$, and its output is $y_i^l = f(X^l * W_i^l)$, where * is the inner product operator and $f(\cdot)$ is an activation function having the summation of the weighted input elements as its variable. For a layer l, its input vector is X^l, the same as all its neurons' inputs. Its weight set is $W^l = \{W_1^l, \ldots, W_{n^l}^l\}$, and its output vector is $Y^l = (y_1^l, \ldots, y_{n^l}^l)^T$, which is the input of its immediately following layer, namely $Y^l = X^{l+1}$. For an entire network, its input is X^1 or Y^0, its weight is $W = W^1 \cup \ldots \cup W^L$, and its output is Y^L. Let $\Delta X^l = (\Delta x_1^l, \ldots, \Delta x_{n^{l-1}}^l)^T$, $\Delta W_i^l = (\Delta w_{i1}^l, \ldots, \Delta w_{in^{l-1}}^l)^T$, $\Delta W^l = \{\Delta W_1^l, \ldots, \Delta W_{n^l}^l\}$, $\Delta W = \Delta W^1 \cup \ldots \cup \Delta W^L$, and $\Delta Y^l = (\Delta y_1^l, \ldots, \Delta y_{n^l}^l)^T$ be the corresponding deviations for the input, weight and output respectively.

1.3.2 Structure

The structure of a single neuron is shown in Figure 1.3, in which the threshold is ignored because it can be regarded as a special weight with a fixed input element and the extension to include it is trivial. For each neuron, its input and the associated weight and its output from its activation connect it to the external world and to the other neurons in a neural network. The weight is an important, adjustable parameter that determines the mapping between the input and the output of a neuron. The action of a neuron largely depends on its activation function chosen, and in this research the most commonly used logistic function is chosen, that is,

$$f(x) = 1/(1+e^{-x}). \tag{1.3}$$

With neurons as cells, the structure of an MLP is sketched in Figure 1.4. Neurons in the MLP are organized into layers by linking them together. Links only exist between neurons of two adjacent layers, and there is no link between neurons in the same layer and in any two nonadjacent layers. All neurons in a layer are fully connected to all other neurons in its adjacent layers. At each layer except the input layer, the input of each neuron is the output of all the neurons in the previous layer, namely $X^l = Y^{l-1}$, with $X^l \in [0,1]^{n^{l-1}}$ for $l > 1$ due to the characteristics of $f(x)$.

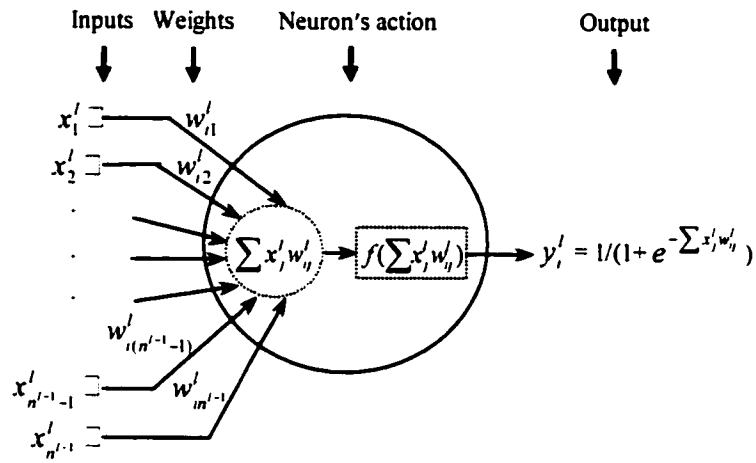

Figure 1.3: The structure of a neuron (i^{th} neuron in layer l).

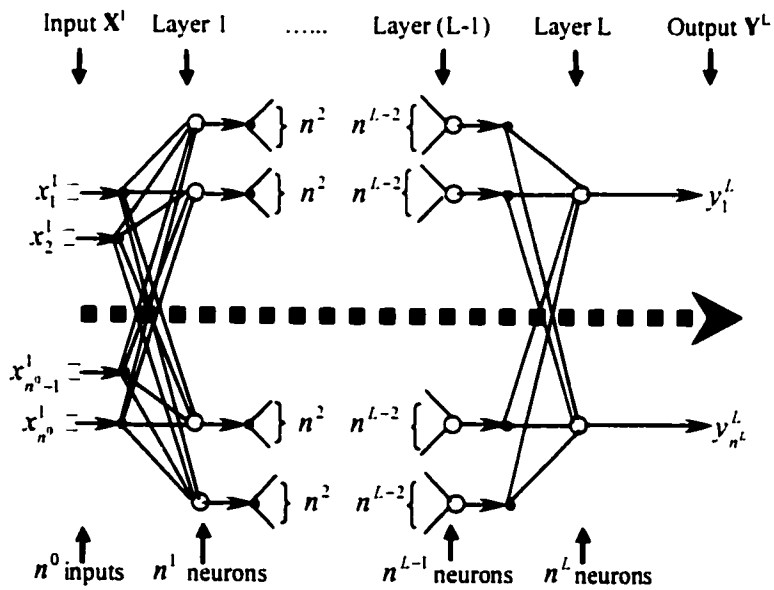

Figure 1.4: The structure of the MLP.

1.3.3 Characteristics of the logistic function

The nonlinear activation function (1.3) plays an important role in the MLP. It has the following characteristics:

- $0 \leq f(x) \leq 1$ (1.4)
- $f(x) + f(-x) = 1$ (1.5)
- $f'(x) = f(x)(1 - f(x))$ (1.6)
- $f(x') < f(x'')$ if $x' < x''$. (1.7)

Chapter 2

Related studies on sensitivity

In the past decade, a variety of studies on the sensitivity of neural networks emerged in the literature [11] — [46]. They varied in their approaches and techniques. A number of neural networks have been investigated which include Hopfields [11] — [14], Madalines [15] — [27], Neocognitrons [28] [29], Multilayer Perceptrons [30] — [42], and some other specific or variant neural networks [43] — [46]. Besides, many applications of the sensitivity of neural networks also appeared [47] — [77]. In this chapter, a literature survey on the sensitivity of neural networks, especially the general feed-forward multilayered neural networks, to input and weight perturbations is presented.

2.1 Sensitivity of Adlines and Madlines

The Madaline proposed by Widrow and Hoff in 1960 [8] [9] is a discrete feed-forward multilayered neural network. A Madaline consists of layers of basic elements called Adalines (Adaptive linear elements) with binary input, weight, and output. The Adalines are analogous to neurons in a neural network. In 1990, Stevenson et al. [15] [16] used n-dimensional geometry as a mathematical tool to theoretically analyze the sensitivity of Adalines and Madalines. The surface of a hypersphere with radius $n^{1/2}$ is used to approximately express the input space of 2^n possible patterns for an

Adaline with n-dimensional input. For a given weight W, the collection of tips of the input vector X that satisfies $W * X = 0$ spans a hyperplane. The hyperplane separates the input X with $W * X > 0$ that yields an output "1" from the input X with $W * X < 0$ that yields an output "-1". Geometrically, the surface of a hypersphere that represents all inputs of an Adaline can be divided by a hyperplane, which is determined by an Adlaline's weights and passes through the origin, into two hemihyperspheres (corresponding to the two outputs of an Adaline). Based on such a geometrical model, Stevenson defined sensitivity as the probability of erroneous output of the Madaline. The sensitivity of an Adaline is derived as a function of the percentage error in inputs and weights under the assumption that the input and weight perturbations are small and the number of Adalines per layer is sufficiently large. In her bottom-up approach, the sensitivity of a single Adaline is computed first, followed by a layer-by-layer accumulative effect, to finally arrive at the total sensitivity of the Madaline network. By analyzing the approximately derived sensitivity expression, she concluded that the sensitivity of the Madaline increases with the number of layers, but it is independent of the number of neurons per layer provided that number is large enough.

In 1994, Alippi et al. [17] [18] [19] generalized the hypersphere model by considering the sensitivity of the feed-forward multilayered neural networks with multiple-step symmetric activation functions. Each discontinuity point of the function introduces a hyperplane separating the hyperspace into regions, and each of which is associated with an output value. More specifically, the locus of the input and output vectors lies on concentric hyperspheres whose number depends on the number of

steps into which the function is decomposed. Continuous functions can be considered as the limit instance for infinitesimal amplitude of the step-wise interpolation, and the input locus is modified into a hypercube. They also proposed a comprehensive error model related to the neuron's computation and abstracting from any implementation detail. In the model, the following classes of errors are identified:

1. input errors;
2. weight errors;
3. synaptic product errors;
4. summation errors;
5. activation function errors.

For each error, they gave a geometrical interpretation for the error's effects on the neuron's output. The sensitivity they defined is the probability of a network's output error due to one of these error types.

In 1992, Piché [20] — [23] used a statistical rather than a geometrical argument to analyze the effects of the weight error in Madalines. He proposed a stochastic model by assuming that inputs and weights as well as their errors are all independently, identically distributed (iid) with mean zero. Based on such a stochastic model Piché defined sensitivity (he called it noise-to-signal ratio) as the ratio of the variance of the output error to the variance of the output. Under the condition that weight errors are small enough, he derived an analytical expression for the noise-to-signal ratio and extended his derivation to Madalines with 5 different types of activation functions,

including sigmoid, linear, threshold, sinusoid, and saturating. For a given allowance of output error, he concluded, the weights of a Madaline with a threshold function are more accurate than that of a Madaline with sigmoid function. In other words, the Madaline with threshold function is more "sensitive" than one with a sigmoid function.

Piché's results were extended by Yang et al. [24] in 1997. In their results, the constrain that weight errors must be small enough was removed and the correlation between input (weight) and its associative error was taken into consideration.

The above sensitivity studies consider all possible input patterns to the network, without referring to a particular given input. In addition, Orzechowski et al. [25] in 1993 derived sensitivity for a given input pattern, which was expressed as a function of the network parameters, namely, the number of nodes and the weight. The sensitivity was defined as the probability of misclassification error for trained neural networks used in binary-to-binary mappings. Oh and Lee [26] [27] in 1994 derived the sensitivity for a specified input pattern as a function of the trained weight, the input pattern, and the variance of weight perturbation or the bit error probability of the input pattern. Specifically it was defined as the probability of the inversion error for trained Madalines with single hidden-layer.

2.2 Sensitivity of Neocognitrons

The Neocognitron, a special kind of feed-forward multilayered neural networks, was proposed by Fukushima [10] in 1980 with the aim of modeling the function of human

eyes to recognize an object even when the object shifts its position and is slightly deformed. In 1996, Cheng and Yeung [28] [29] also employed n-dimensional geometry as a mathematical tool to investigate the sensitivity analysis of Neocognitrons. Based on the characteristics of Neocognitrons that the vector parameters (input and weight) of the activation function of the S-cell can be approximately transformed to unit vectors, they simplified the activation function of the S-cell and transformed the domain for the input and weight from R^n (n -dimensional space of real numbers) to U^n (n -dimensional space of real unit vectors). So it is possible to use the surface of a unit hypersphere to represent the domain for the input and weight and to analyze the sensitivity of Neocognitrons. With a given threshold angle β, the input X which has an angle with the weight W equal to β ($X * W = \cos\beta$) forms a hypercone, and the hypercone intersects the surface of the hypersphere to form a hyperarc. Only the inputs falling into the hyperarc will fire the S-cell. Based on this model, Cheng and Yeung defined the sensitivity as the mathematical expectation of output errors in terms of percentage error in inputs and weights, and derived a mathematical expression along with an algorithm to calculate the sensitivity of Neocognitrons. They concluded that the sensitivity increases with the number of connections, plans and layers. For a sufficiently large Neocognitron, the degree of change of the sensitivity decreases until it becomes stabilized. When increasing the selectivity parameters, the sensitivity also increases, but its increasing speed is very slow for small perturbations.

2.3 Sensitivity of Multilayer Perceptrons

The Multilayer Perceptron is the most general feed-forward multilayered neural network, in which each neuron's input and weight could be any real number from a continuous interval. The choice for its activation function usually is the sigmoid function that yields a real number output. There are basically two approaches in the sensitivity studies on MLPs. One is an analytical approach in which sensitivity is defined as a partial derivative of output to input. The other is statistically based in which variance is mostly used.

Hashem [30] in 1992 and Fu and Chen [31] in 1993, under the assumption that input perturbations are very small, employed partial differentiation rule to consider the sensitivity of network's output with respect to input perturbations for MLPs with differentiable activation functions. In their method, the sensitivity is defined as the partial derivative of output to input and is computed one layer at a time starting from the output layer and proceeding backwards (backward chaining is used) toward the input layer. Fu and Chen simply expressed the sensitivity as a Jacobian matrix:

$$\frac{dY}{dX^T} = \left[\frac{\partial y_i}{\partial x_j}\right]_{M \times N}, \tag{2.1}$$

where function $Y = F(X)$, $F : D \subset R^N \rightarrow R^M$ is implemented by the MLP. Using this result, they specified some criteria to reduce the sensitivity of an MLP so that the generalization of the network can be improved.

This analytical approach was also used by Chow and Yee [32] to measure the effect of small input perturbation on neural network output. They derived the relationship between the output error and the small input perturbation by the matrix they called sensitivity matrix, which is the same as that proposed by Fu and Chen. Based on this matrix, they defined a *relative input robustness measure*, r, to indicate the relative robustness for a given input subject to input noise. They then defined a *relative network robustness measure*, R, which is the norm measure of r, i.e., $R = \|r\|$, to indicate the overall robustness of the given network. They applied the *relative network robustness measure* to several modified Incipient Fault Detector Artificial Neural Networks for operation in noisy environments and the results showed usefulness and effectiveness.

In 1992 Choi and Choi [33] introduced a statistical sensitivity measure for the MLP with differentiable activation function. They considered firstly the sensitivity of the MLP with a single output neuron for a specific input pattern. The sensitivity is defined as the ratio of the standard deviation of output errors to the standard deviation of input or weight errors under the condition that the latter tends to zero. For example, the sensitivity to weight perturbations in an MLP for a trained weight set W^* and an input pattern p is

$$S^p(W^*) = \lim_{\sigma \to 0} \frac{\sqrt{\mathrm{var}[\Delta X_p(L)]}}{\sigma}, \tag{2.2}$$

where $\Delta X_p(L)$ is the output error, σ is the standard deviation of each element of weight errors, and var[·] is the variance of [·]. By means of first order Taylor approximation, they derived a formula for the computation of the sensitivity. Then based on the result on the MLP with single output neuron for a specific input, they extended the result to the MLP with multiple output neurons for a finite set of input patterns. As for the aggregated sensitivity for more than one input pattern, they suggested

$$S(W^*) = \max\{S^p(W^*), p \in [1, P]\}, \tag{2.3}$$

or

$$S(W^*) = \frac{1}{P}\sum_{p=1}^{P} S^p(W^*), \tag{2.4}$$

where P is the number of input patterns. They also extended the sensitivity for a multiple output MLP, namely,

$$S(W^*) = (S_1(W^*),\ S_2(W^*), \ldots\ldots,\ S_n(W^*))^T, \tag{2.5}$$

where $S_i(W^*)$ $(1 \le i \le n)$ is the sensitivity of the ith neuron in the output layer that has n neurons.

Another one using statistical method is by Xie and Jabri [34] [35] [36] in 1991. They studied the effects of quantization in the neural network, which is exactly a sensitivity problem viewing from the hardware realization perspective. Under a

presumed statistical model to represent the quantized values of weights of neurons and some assumptions on their distributions, they investigated a relationship between bit-resolution, the number of hidden units and layers, and the performance degradation of the network. Based on the work of Xie and Jabri, Düdar and Rose [37] in 1995 extended, with some different approaches, the study of the effects of quantization on the MLP. Besides defining a minimum and maximum for the weight values and then defining quantization width in terms of these, Düdar and Rose changed the activation function from the linearized version presented by Xie and Jabri to the true nonlinear sigmoid function. Two examples were presented to verify their derived relationship, which are representatives of the two typical applications of the MLP. One is a function approximation, namely, a sine generator. The other one represents a pattern classification application and is a four-bit A/D converter.

In 1992, Minai and Williams [38] [39] studied the response of a feed-forward network to the perturbations in internal neuron outputs. The problem was formulated to perturb only one neuron at a time, and the expectation and variance of the deviation of the output was derived. This is one of the ways to study the sensitivity of the neural networks. But it is difficult in getting the sensitivity of the whole network to the perturbation of a particular parameter such as weight.

Recently, Yeung et al. [40] [41] generalized Piché's method by deriving a universal expression of the MLP's sensitivity for antisymmetric squashing activation functions, without any restriction on the magnitude of input and weight perturbations. The effects of neural network design parameters such as the number of layers, the number of neurons per layer and the chosen activation function are analyzed.

Unlike the above statistical approaches which aim at approximating output error statistics, Lovell et al. [42] in 1992 established bounds on the effect of errors in the weights and inputs of an MLP. By assuming that errors in weights and inputs to the network are statistically independent and small, they derived bounds on the expectation and variance of errors at the output of an MLP. This result is useful for the network user to estimate the worse case for respect errors.

2.4 Application of sensitivity study

The applications of the results of the sensitivity study have been explored for a wide range of problems. One concerns the basic research issues on neural networks' architecture and behavior [47] — [72] such as structure pruning, parameter selection, and fault-tolerance improvement, and the other is on concrete applications [73] — [77]. For example, Karnin [47] in 1990 used sensitivity to determine which nodes can be pruned to minimize the size of the network. Engelbrecht and Cloete [54] in 1996 pruned both input and hidden layers based on sensitivity analysis. Zurada et al. [55] in 1997 selected training patterns by using the perturbation-based sensitivity approach to delete redundant inputs for perceptron networks when their training involves a large amount of redundant data. Bernier et al. [72] in 2000 proposed a version of the backpropagation algorithm that used the sensitivity to changes in the weights as a quantitative measure to increase the fault-tolerance capability of feed-forward neural networks. In real life applications, Guo and Uhrig [74] in 1992 constructed a neural network for a nuclear power plant, and its input is the target parameters for change. Each input represented a factor that affects the thermal performance and gross heat

loss of the plant. After completing the sensitivity analysis of the neural network, the most sensitive input factor was found. More control processes were established to prevent the large perturbation of the sensitive factors.

This survey clearly shows that there is an increasing trend of the applications of sensitivity study in recent years, and that sensitivity study is an important consideration when designing neural networks for applications.

2.5 Summary

In the above survey, the results obtained by Stevenson et al. as well as Appli et al. indicate that they are only suitable for networks with discrete neuron inputs, weights and outputs. The method used by Cheng and Yeung is merely applicable to the neural networks whose input and weight vectors are or can be transformed to unit vectors. Unfortunately, the hypersphere model they used fails in the case of the MLP because its input and weight vectors generally do not span a hypersphere surface or volume. The results of Piché's approach are just applicable to an ensemble of Madalines, and not to any individual network because of too strong assumptions on the stochastic model. The same can be said of the extended results made by Yang et al. and Yeung et al. Both the analytical method used by Hashem and the statistical method proposed by Choi and Choi require a constraint on the input and weight deviations that they must be small enough. Furthermore, both results are only useful to measure the output deviation with respect to a given input pattern for a specific MLP network. They are not able to measure the expected output deviation with respect to the overall input

patterns, which is an important factor to the performance measurement of networks, especially for continuous neural networks like MLPs.

The following table summarizes the above review to end this section and chapter.

Neural networks	Researcher(s)	Sensitivity	Constrains
Madalines	M. Stevenson et al. 1990	Probability of erroneous output due to percentage error in weights	•The dimension of input must be large. •$\lvert\Delta W\rvert/\lvert W\rvert$ must be small, where $\lvert W\rvert$ is the magnitude of vector W.
Madalines	S. W. Piché 1995	Ratio of the variance of the output error to the variance of the output.	•Input and weight and their errors are all iid with mean zero. •Input and weight errors must be very small.
Neocognitrons	A. Cheng & D. Yeung 1996	Expectation of output errors due to input and weight errors.	•Input and weight vectors are or can be transformed to unit vectors. •Only S-cells and S-layers are considered.
MLPs	J. Y. Choi & C. H. Choi 1992	Ratio of the standard deviation of output errors to the standard deviation of input or weight errors.	•Input and weight must be given. •Input or weight errors tend to zero.
MLPs	S. Hashem 1992 L. Fu & T. Chen 1993	Partial derivative of output to input.	•Input and weight must be given. •Input error tends to zero.
MLPs	D. Yeung et al. 2000	Ratio of the variance of the output error to the variance of the output.	•The same stochastic model as Piché's •Activation function is the antisymmetric squashing function.

Table 2.1: Some representative studies on the sensitivity of neural networks.

Chapter 3

Research Methodology

This chapter describes the research methodology and the general idea with which the sensitivity is treated in this dissertation. Based on the structural features of the MLP, a bottom-up approach is adopted to consider the sensitivity of different granules in the MLP. First is the sensitivity for a single neuron, then the sensitivity for a layer, and finally the sensitivity for an entire MLP. The sensitivity defined in this dissertation represents or reflects in a statistical sense the output deviation of the MLP due to input and weight deviations.

3.1 Definition of sensitivity for a single neuron

For the case of a single neuron, the most direct and natural way to express its output deviation arising from input and weight deviation is the difference of its deviated and non-deviated output, i.e.,

$$\Delta y_i^l = g(X^l, \Delta X^l, W_i^l, \Delta W_i^l) = f((X^l + \Delta X^l) * (W_i^l + \Delta W_i^l)) - f(X^l * W_i^l). \quad (3.1)$$

According to (3.1), Δy_i^l is a function g of X^l, W_i^l, ΔX^l and ΔW_i^l, and can be readily calculated if the values of the variables are all known. But in real life situations, the values of some of these variables are not always available. For example,

before an MLP is trained, not much information is known about its weights except the interval from which the weights' values may be drawn. Even when those values are known, it may be unnecessary to compute Δy_i^l at some of these values. For example in evaluating a network's performance, it is unnecessary to compute Δy_i^l for an individual input pattern. Rather it would be more desirable to find the relationship between the output deviation Δy_i^l and its cause — input deviation ΔX^l or weight deviation ΔW_i^l or both in a statistical sense by taking overall input patterns into consideration. That is to regard the input variable X^l as a statistical variable. Which variable is chosen as statistical variable is quite application dependent. Our approach, by means of statistical principle, is to consider the mathematical expectation of Δy_i^l by regarding some relevant variables as statistical variables. It is obvious that (3.1) gives a relationship between Δy_i^l and ΔX^l and ΔW_i^l. The problem is to define the sensitivity that keeps this relationship but, in the meanwhile, without reference to X^l and W_i^l. Below is a definition of such sensitivity.

Definition 3.1: For input vector $X^l \in [a_x, b_x]^{n^{l-1}}$ and weight vector $W_i^l \in [a_w, b_w]^{n^{l-1}}$, and given their corresponding deviations ΔX^l and ΔW_i^l, the sensitivity of neuron i in layer l is defined as the mathematical expectation of the absolute value of Δy_i^l with respect to X^l and W_i^l, which is expressed as

$$s_i^l(\Delta X^l, \Delta W_i^l) = E(|g(X^l, W_i^l)|) = E(|f((X^l + \Delta X^l) * (W_i^l + \Delta W_i^l)) - f(X^l * W_i^l)|). \quad (3.2)$$

Obviously, s_i^l is a function of ΔX^l and ΔW_i^l, independent of any specific X^l and W_i^l. In the definition the absolute value is employed so as to avoid the cancellation of the positive and negative output deviations. The formula (3.2) is a general expression, which can have many variations to suit for different applicable situations. For example, to consider input perturbation only, (3.2) can be simplified with $\Delta W_i^l = 0$, and if weights are fixed for a specific MLP after training, (3.2) can be rewritten by only considering the statistical variable X^l. It may also take ΔX^l or ΔW_i^l as statistical variables if there is such a requirement for a particular situation.

3.2 Definition of sensitivity for a layer

According to the MLPs' structure, a layer consists of a set of neurons and each neuron in the same layer has the same input. So, the sensitivity for a layer could be based on the sensitivity of its neurons as follows:

Definition 3.2: For input vector $X^l \in [a_x, b_x]^{n^{l-1}}$ and weight vector set $W^l = \{W_i^l \mid W_i^l \in [a_w, b_w]^{n^{l-1}}, 1 \le i \le n^l\}$, and given their corresponding deviations ΔX^l and $\Delta W^l = \{\Delta W_i^l \mid 1 \le i \le n^l\}$, the sensitivity of layer l is defined as:

$$S^l(\Delta X^l, \Delta W^l) = (s_1^l, \ldots, s_{n^l}^l)^{\mathrm{T}}. \tag{3.3}$$

The above deviation ΔX^l $(l > 1)$ could be in fact regarded as the sensitivity of the previous layer, that is, $\Delta X^l = S^{l-1}$ $(l > 1)$. It is noted that S^l is a vector, which

consists of the sensitivities of all neurons in the layer l. Since sensitivity is generally agreed to be a scalar rather than a vector, one may choose, for instance, $norm(S^l)$, $max(S^l)$, $avg(S^l)$, or $min(S^l)$ as the representation of a layer's sensitivity depending on the nature of the application involved.

3.3 Definition of sensitivity for an entire network

Since the output of the output layer is the network's output, the sensitivity of a neural network is therefore the sensitivity of its output layer.

Definition 3.3: The sensitivity of an MLP is the sensitivity of its output layer. That is

$$S(\Delta X^1, \Delta W) = S^L(\Delta X^L, \Delta W^L) = (s_1^L, \ldots, s_{n^L}^L)^T. \tag{3.4}$$

The definition (3.4) is a recursive definition because S^L depends on ΔX^L which is really S^{L-1}, which in turns depends S^{L-2} and so on until it reaches the initial value S^0, namely ΔX^1. Actually the sensitivity of an MLP is obtained by calculating the sensitivity of each neuron in the MLP with a partial order such that the neurons in the preceding layer should be calculated before those in the succeeding layer and that the neurons in the same layer could be in any order. It is obvious that the sensitivity of a neuron is fundamental to the sensitivity of an MLP.

As the sensitivity definition of a neuron can have many variations, the sensitivity of an MLP would vary depending on the choice of different statistical variables and

different deviation sources and their possible combinations under the assumption that the same choice should be applied to all neurons in the entire network.

Usually the construction of an MLP needs to undergo three stages in its life cycle: design, training, and operation. Different stages may have different requirements and different parameter information on the formation of the sensitivity. In the design stage in which most parameters such as the weight and the architecture of the network are not fixed, the sensitivity analysis to weight deviation is important in considering the effects of weights with limited precision for the hardware and architecture design. In the training and operation stages when the weight and the architecture have been fixed, the sensitivity to input deviation can be an important measure to evaluate the networks' performance such as error tolerance and generalization capability. Table 3.1 lists different sensitivity formations for different applicable situations. The most commonly encountered situations are: neural networks without fixed weights before training or with fixed weights after training, neural networks with input or weight deviations or both, and possible combinations of the two cases. But it should be noticed here that the consideration of weight deviation can not go without the consideration of input deviation for those neurons in the second or later layer because in the MLP the preceding layer's weight deviation will cause input deviation to be injected to its succeeding layer.

Parameter deviation	Statistical variable	Sensitivity function	Applicable situation
ΔX^1	X^1, W	$S(\Delta X^1)$	Sensitivity to a given input deviation for untrained networks on overall input patterns.
$\Delta X^1, \Delta W$	X^1, W	$S(\Delta X^1, \Delta W)$	Sensitivity to given input and weight deviations for untrained networks on overall input patterns.
$\Delta X^1, \Delta W$	W	$S(X^1, \Delta X^1, \Delta W)$	Sensitivity to given input and weight deviations for untrained networks on a given input pattern.
ΔX^1	X^1	$S(\Delta X^1, W)$	Sensitivity to a given input deviation for trained networks on overall input patterns.
ΔX^1		$S(X^1, \Delta X^1, W)$	Sensitivity to a given input deviation for trained networks on a given input pattern.
ΔX^1	ΔX^1	$S(X^1, W)$	Sensitivity to overall input deviations for trained networks on a given input pattern.
ΔX^1	$\Delta X^1, X^1$	$S(W)$	Sensitivity to overall input deviations for trained networks on overall input patterns.

Table 3.1: List of sensitivity functions.

3.4 Summary

In this chapter, a new approach is proposed to study the sensitivity of MLPs. It is different from the approaches described in chapter 2. Instead of using a hypersphere, a hypercube is employed as the mathematical model to represent the MLP's input or weight space because they usually fall into continuous intervals. Instead of using variance, the mathematical expectation is used as a quantified measure. This approach does offer certain advantages over the other ones. For example, it does not demand that the input perturbation must be very small as the derivative approach requests, and the mathematical expectation used in this approach reflects the network's output error more directly and exactly than the variance does (e.g., some assumption must be made for the expectation prior to considering variance). Moreover, this approach is suitable

for a wide range of situations. It can cover the sensitivity not only due to input deviation or weight deviation individually but also to both of them together, and the sensitivity is not only with respect to a specific input pattern or weight set but also to overall input patterns or weight sets. Its solutions are not only applicable to the situations for neural network designs by regarding weight and architectural parameters as variables, but also to situations for trained neural networks by regarding weight and architecture parameters as being fixed.

In this dissertation, two important cases are investigated. One is to analyze the sensitivity of an ensemble of MLPs to input and weight deviations for untrained neural networks with respect to overall input patterns. The other is to quantify the sensitivity measure of an individual MLP to input deviation for trained networks with respect to its overall input patterns. They are separately discussed in the two chapters that follow.

Chapter 4

Sensitivity analysis for an ensemble of MLPs

The purpose of this chapter is to investigate the behavior of the MLP's sensitivity to input and weight deviations. An analytical expression that is a function of input and weight deviations and the dimension of input (or weight in the same) is derived for the sensitivity of a single neuron, which is treated as a variable neuron without fixing the dimension of weight and the values of weights. Then two algorithms are given to compute the sensitivity of an entire MLP. The algorithms have input and weight deviations, the number of neurons for each layer, and the number of layers as their parameters. By analyzing the derived formula and executing one of the algorithms, some significant observations about the effects of the above mentioned parameters on the outputs of neurons and the MLP networks have been discovered due to input and weight perturbations. The theoretical results have been verified by the simulation results obtained from computer simulations.

4.1 Sensitivity for neurons

In this section, for the sake of simplicity, the superscript that marks the neuron's layer and the subscript that marks the neuron's order in the layer are omitted because only neurons are considered and their positions in the network are irrelevant. Also for

simplicity and without loss of generality, the Definition 3.1 is modified as follows.

Definition 4.1: For input vector $X \in [0,1]^n$ and weight vector $W \in [0,1]^n$, and given their corresponding deviations $|\Delta X| = (|\Delta x_1|,...,|\Delta x_n|)^T$ and $|\Delta W| = (|\Delta w_1|,...,|\Delta w_n|)^T$, the sensitivity of a neuron is defined as the mathematical expectation of the absolute value of Δy with respect to X and W, which is expressed as

$$s = E(|\, g(X,W)\,|) = E(|\, f((X + |\Delta X|) * (W + |\Delta W|)) - f(X * W)\,|)\,. \tag{4.1}$$

Compared with the Definition 3.1, this definition specifies a specific interval for both input and weight and takes the absolute value of both input and weight deviations into consideration. Without loss of generality, narrowing the intervals to $[0,1]^n$ will simplify the derivation of the formula and the computer simulation. Actually, the intervals can be extended to more general continuous intervals. It is noticed that, in the MLP with $f(x) = 1/(1+e^{-x})$ as its activation function, the interval of the input of all layers except the input layer is included in $[0,1]^{n^{l-1}}$ $(l \geq 2)$, and it is possible to limit the input of the first layer to be in $[0,1]^{n^0}$ because any continuous interval $[a,b]$ can be transformed to $[0,1]$ by the one to one mapping $(x^1{}_i - a)/(b-a)$. So the assumption $X^l \in [0,1]^{n^{l-1}}$ $(l \geq 1)$ in the sensitivity definition is practically acceptable. As to the interval for weight, according to the derivation of the sensitivity formula presented in appendix A, it is easy to extend the interval of weight to any positive interval by directly substituting

the new lower and upper limits for that of the integral in the derivation. In extension to a negative interval, the logistic function's characteristic (1.5) can be used to replace the integrand $f(x)$ by $(1-f(-x))$ in the derivation process. For an interval containing both positive and negative values, it can be divided into two intervals (one is the positive part and the other is the negative part) for computation purpose and then add the two results. All in all, the assumption on input and weight intervals may influence the magnitude of the sensitivity only, but it will not change the general trending behaviors of the sensitivity, which is the main objective of this investigation.

The analytical expression for the calculation of the sensitivity can be derived from (4.1). A brief explanation of the derivation is given here, and the details of the derivation are given in appendix A. According to the principle of the mathematical expectation of multivariate continuous random variables, the sensitivity defined in (4.1) can be expressed as

$$
\begin{aligned}
s &= E(|\, g(X,W)\,|) \\
&= E(|\, f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W)\,|) \\
&= \int_{\Omega_X}\int_{\Omega_W} \varphi(X,W)\left|f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W)\right| d\Omega_X\, d\Omega_W\,, \qquad (4.2)
\end{aligned}
$$

where Ω_X and Ω_W are the regions spanned by input and weight vectors, and $\varphi(X,W)$ is the density function. For simplicity, assuming the input X and weight W are

independently and identically distributed in the regions, so the integral region is a hypercube $[0,1]^{2n}$ in $2n$-dimensional hyperspace by taking each element of X and W as a coordinate axis, and the density function satisfies

$$\varphi(X,W)=1. \tag{4.3}$$

In the derivation, the following approximations and integral formula are used to simplify the calculations:

$$1/(1+x)\approx\sum_{k=0}^{p}(-1)^k x^k \text{ for } |x|<1\text{, where } 1\le p<+\infty, \tag{4.4}$$

$$e^x\approx\sum_{h=0}^{q}x^h/h!\text{, where } 1\le q<+\infty\text{, and} \tag{4.5}$$

$$\int x^n e^{ax}dx=\sum_{r=0}^{n}((-1)^r n!x^{n-r}e^{ax}/((n-r)!a^{r+1}))+C. \tag{4.6}$$

The final derivation result is

$$s\approx\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\sum_{h=1}^{q}((k^{h-1}/h)I_1))-(\sum_{h=1}^{q}((k^{h-1}/h)I_2))^n), \tag{4.7}$$

where

$$I_1 = \sum_{r=0}^{h-1} (|\Delta w_j|^{h-1-r} e^{-k(1+|\Delta x_j|)|\Delta w_j|} - (1+|\Delta w_j|)^{h-1-r} e^{-k(1+|\Delta x_j|)(1+|\Delta w_j|)}) / ((h-1-r)!(k(1+|\Delta x_j|))^{r+1}), \tag{4.8}$$

and

$$I_2 = \sum_{r=0}^{h-2} (-e^{-k} / ((h-1-r)!k^{r+1})) + (1-e^{-k}) / k^h. \tag{4.9}$$

In (4.7), if it is let $p = 1$ and $q = 1$, the simplified expression for the sensitivity of a single neuron is then

$$s \approx e^{-n} - e^{-\sum_{j=1}^{n}(1+|\Delta x_j|)(1+|\Delta w_j|)}. \tag{4.10}$$

4.2 Sensitivity for networks

Similar to the Definition 3.2 and Definition 3.3, one could define, based on the Definition 4.1, the sensitivity for a layer and then for an entire network. This section gives two algorithms for the computation of the sensitivity of an entire network, making use of the sensitivity of a single neuron and the structure features of the MLP.

Definition 4.2: For input vector $X^l \in [0,1]^{n^{l-1}}$ and weight vector set $W^l = \{W_i^l \mid W_i^l \in [0,1]^{n^{l-1}}, 1 \le i \le n^l\}$, and given the corresponding deviations $|\Delta X^l| = (|\Delta x_1^l|, \ldots, |\Delta x_{n^{l-1}}^l|)^T$ and $|\Delta W^l| = \{|\Delta W_1^l|, \ldots, |\Delta W_{n^l}^l|\}$, the sensitivity of layer

l is defined as:

$$S^l(|\Delta X^l|,|\Delta W^l|)=(s_1^l,\ldots,s_{n^l}^l)^{\mathrm{T}}. \tag{4.11}$$

Definition 4.3: The sensitivity of an MLP is the sensitivity of its output layer. That is

$$S(|\Delta X^1|,|\Delta W|)=S^L(|\Delta X^L|,|\Delta W^L|)=(s_1^L,\ldots,s_{n^L}^L)^{\mathrm{T}}. \tag{4.12}$$

4.2.1 Recursive algorithm

In Definition 4.3, the sensitivity of an MLP, namely $S^L(|\Delta X^L|,|\Delta W^L|)$, is defined locally on layer L, the output layer. This definition is actually recursive in nature if $|\Delta X^L|$ is regarded as S^{L-1}. It follows that $S^L(S^{L-1},|\Delta W^L|)$. Now denoting (4.7), (4.8), and (4.9) as $s_i^l(S^{l-1},|\Delta W_i^l|)$ and let $S^0=|\Delta X^1|$, the recursive formula for the computation of the sensitivity is

$$S^l=\begin{cases} R(S^0,|\Delta W^1|) & \text{if } l=1 \\ R(S^{l-1},|\Delta W^l|) & \text{if } l>1 \end{cases} \tag{4.13}$$

where $R(S^{l-1},|\Delta W^l|)=(s_1^l(S^{l-1},|\Delta W_1^l|),\ \ldots,s_{n^l}^l(S^{l-1},|\Delta W_{n^l}^l|))$. According to this recursive formula, it is easy to give a recursive algorithm as follows.

•Algorithm MLP_SENSITIVITY_R($|\Delta X^1|$,$|\Delta W|$):

1) if $L=1$ then compute $S^L = R(|\Delta X^1|,|\Delta W^1|)$;

2) if $L>1$ then recursively compute $S^L = R(S^{L-1},|\Delta W^L|)$;

3) return(S^L).

4.2.2 Iterative algorithm

From a global point of view, the sensitivity of an MLP, namely $S(|\Delta X^1|,|\Delta W|)$, is its ouput deviation of the output layer due to the first layer's input deviation $|\Delta X^1|$ and the total weight deviation $|\Delta W|$. As the output of all neurons in a layer is the input to each neuron in the next succeeding layer, the sensitivity of the output layer can be computed neuron by neuron from the first layer to the last layer. That is, S^1 can be computed with $|\Delta X^1|$ and $|\Delta W^1|$ by (4.7), (4.8), and (4.9), then S^2 with $|\Delta X^2|$ (which is S^1) and $|\Delta W^2|$, etc., finally S^L with $|\Delta X^L|$ (which is S^{L-1}) and $|\Delta W^L|$.

The following algorithm MLP_SENSITIVITY_I($|\Delta X^1|$, $|\Delta W|$) is given to compute the sensitivity of an MLP. The input deviation $|\Delta X^1|$ or S^0 and the computed sensitivity S^l ($1 \le l \le L$) can be arranged into a 2-dimensional array. And the total weight deviation $|\Delta W|$ can be arranged into a 3-dimensional array. By controlling the correspondences between the iterative time and the arrays' subscript, it is easy to compute the sensitivity for each neuron, then for each layer, and finally for the entire

MLP.

- Algorithm MLP_SENSITIVITY_I($|\Delta X^1|, |\Delta W|$):

1) Arrange input deviation $|\Delta X^1|$ and weight deviation $|\Delta W|$ into arrays;

2) Compute the sensitivity neuron by neuron from the first layer to the last layer:

   For l = 1 step 1 to L

   For i = 1 step 1 to n^l

   Compute s_i^l by using (4.7), (4.8), and (4.9), and arrange results.

3) S^L is the sensitivity computed.

4.3 Analysis of the sensitivity

In the previous sections, an analytical expression to calculate the sensitivity for a single neuron and two algorithms to compute the sensitivity for an entire MLP have been derived and presented. The analytical expression and the algorithms provide us insights into some significant behaviors of MLPs. In this section, again following a bottom-up approach, the derived analytical expression and the results obtained by executing the algorithm MLP_SENSITIVITY_I($|\Delta X^1|, |\Delta W|$) are analyzed in order to discover some significant observations on the behaviors of both individual neuron and the entire MLP.

4.3.1 Effects of parameters on the sensitivity of neurons

The following analyses are essentially based on (4.7), (4.8), and (4.9). It is so complex that computer aided computations are desirable to make the analytical and computational efforts more reasonable.

4.3.1.1 Input and weight deviations

The simplified formula (4.10) obviously shows that the sensitivity increases with input and weight deviations, and in the extreme cases, the sensitivity is zero when both input and weight deviations are zero or the maximum value is reached, i.e., e^{-n}, when either input or weight deviation is infinite. This result is coincident with what intuitively expected. To be more precise and perceivable, Figure 4.1 is used to illustrate the results computed by (4.7), (4.8), and (4.9). Theoretically the larger the p and q in (4.7) are, the more precise the results will be. But practically they are restricted by the precision limits of the hardware being used. In the computation of the results of Figure 4.1, $p=100000$ and $q=15$ are chosen, which are so selected that if either of them is made any larger it will cause an overflow error. The solid lines in Figure 4.1 show the theoretical relationship between the sensitivity and input and weight deviations, in which the horizontal axis represents both input and weight deviations (for simplicity and without losing generality, it is assumed that the elements of input deviation $|\Delta X^l|$ and weight deviation $|\Delta W_i^l|$ are all identical and thus can be represented by the same point on the horizontal axis). Thus, either from the analysis of (4.10) or from the

computations of (4.7), (4.8), and (4.9), it can be concluded that the sensitivity of a single neuron increases with its input and weight deviations, but the increase has an upper bound that is determined by the dimension of input.

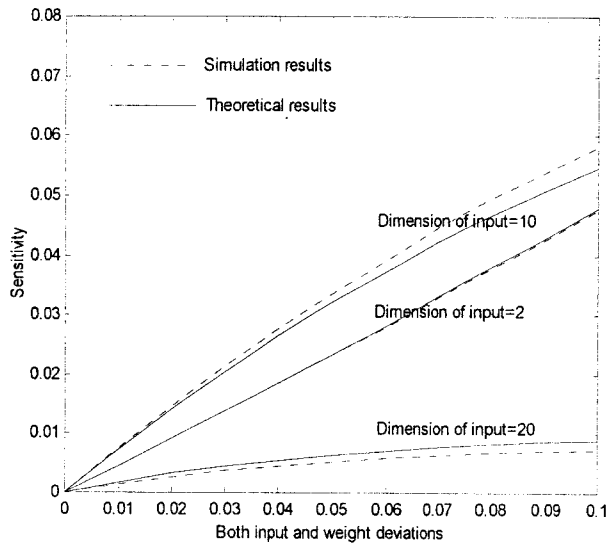

Figure 4.1: Sensitivity of a neuron to input and weight deviations.

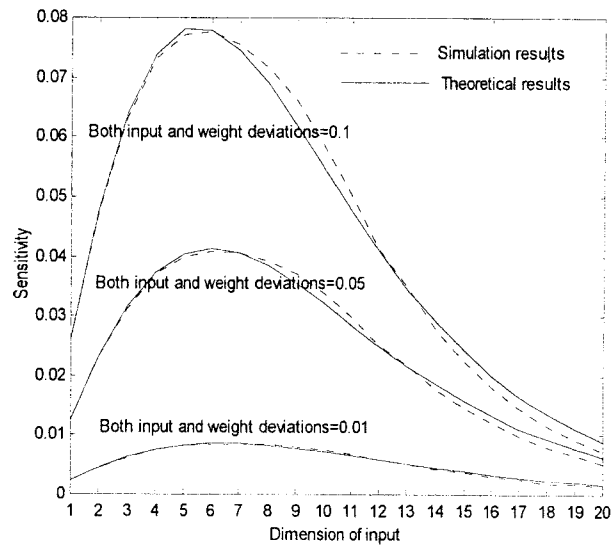

Figure 4.2: Sensitivity of a neuron to the dimension of input.

4.3.1.2 Dimension of input

According to (4.10) with the input and weight deviations held constant, the sensitivity of neuron i in layer l can be represented as the following general form:

$$s = C_1^{\ n} - C_2^{\ n}, \tag{4.12}$$

where n is the dimension of input, C_1 is determined by the ranges of X and W, C_2 is determined by the ranges of X and W, $|\Delta X|$, and $|\Delta W|$, and the inequality

$1 > C_1 \geq C_2 > 0$ is satisfied. By analyzing the feature of the continuous function $s(x) = C_1^{\,x} - C_2^{\,x}$, it is found that $x' = \ln((\ln C_2)/(\ln C_1))/(\ln C_1 - \ln C_2)$ is the function's only maximum point, and x' varies in the interval $(0, +\infty)$ with the variation of C_1 and C_2. So theoretically, it can be concluded that the sensitivity increases with n increasing when $n < x'$ is true and decreases with n increasing when $n \geq x'$. In more detail, assuming that $\lfloor x' \rfloor$ represents the largest integer smaller than x', and m represents the maximum available dimension of input, then the following relationships between the sensitivity and the dimension of input hold:

1) $\lfloor x' \rfloor < 2$: s always decreases with n increasing;
2) $2 \leq \lfloor x' \rfloor < m$: s increases with n until $n \geq \lfloor x' \rfloor$, then decreases with n increasing;
3) $\lfloor x' \rfloor \geq m$: s always increases with n increasing.

But practically, as n should be an integer and small in magnitude, and as the decrease of input and weight deviations force C_2 to increase and this in turn forces x' to increase, it yields the following conclusions about the relationship among the sensitivity, the input and weight deviations, and the dimension of input:

1) The sensitivity may first increase with small n and then decreases with n increasing when the deviations are large,
2) The sensitivity may mostly increase with n when the deviations are small.

The solid lines in Figure 4.2, which are also computed by (4.7), (4.8), and (4.9), again show the theoretical relationship between the sensitivity and the dimension of input.

4.3.2 Effects of parameters on the sensitivity of networks

In order to study the effects of parameters on the sensitivity of an MLP, algorithm MLP_SENSITIVITY_I($|\Delta X^{I}|$, $|\Delta W|$) is implemented and executed for some representative MLPs.

4.3.2.1 Input and weight deviations

Three MLPs are constructed to study the effect of input and weight deviations. They are $2-2-1$, $2-3-1$ and $2-2-2-1$. The sensitivities computed by the algorithm MLP_SENSITIVITY_I($|\Delta X^{I}|$, $|\Delta W|$) for the three MLPs are shown in Figure 4.3, in which the horizontal axis represents both input and weight deviations. The result clearly shows that the sensitivity of an MLP also increases with the input and weight deviations.

From Figure 4.3, it can also be found that the sensitivity of an MLP varies with the structural configuration of an MLP. This phenomenon indicates that the structural configuration of an MLP, such as the number of neurons per layer and the number of layers, has a significant effect on the network sensitivity.

4.3.2.2 Number of neurons

Ten MLPs are constructed to study the effect of the number of neurons in a layer. They are $2-1-1$, $2-2-1$, ..., and $2-10-1$. They can also be represented as a set: $\{2-n-1 \mid 1 \le n \le 10\}$. The difference among them is the number of neurons in layer 1 or the hidden layer. The sensitivities computed by the algorithm MLP_SENSITIVITY_I($|\Delta X^1|$,$|\Delta W|$) for these ten MLPs are shown in Figure 4.4. The results are similar to those presented in figure 4.2 for the dimension of input of a single neuron except a little bit smaller in magnitude.

Since an MLP may have several layers, the number of neurons in different layers may have different effects. To study the effect of the number of neurons in different layers, three sets of MLPs are constructed. They are $\{n-2-2-1 \mid 1 \le n \le 10\}$, $\{2-n-2-1 \mid 1 \le n \le 10\}$, and $\{2-2-n-1 \mid 1 \le n \le 10\}$. The first set is used to study the effect of the dimension of input in the input layer, the second one is on the number of neurons in layer 1, and the third is on the number of neurons in layer 2. The results are shown in figures 4.5, 4.6, and 4.7 respectively. Comparing these three figures, one observes that the effect of the number of neurons in a layer is different for different layers. The nearer a layer to the output layer is, the more effect the number of neurons in the layer has. This, from another angle, shows that the number of layers in an MLP has an effect on the sensitivity.

4.3.2.3 Number of layers

To study the effect of the number of layers, ten MLPs are constructed. They are $2-1$, $2-2-1$, $2-2-2-1$, ..., and $2-2-2-2-2-2-2-2-2-2-1$. From the second to the last, each time a layer with 2 neurons is added into the previous MLP and keep the output layer with only one neuron. The sensitivities computed by the algorithm MLP_SENSITIVITY_I($|\Delta X^1|$, $|\Delta W|$) for these MLPs are shown in Figure 4.8. It is found that the sensitivity decreases with the number increasing when the number is small and the decrease almost levels off when the number becomes large. Hence, when deviation propagates in an MLP, the neurons in the preceding layers may reduce it more than the neurons in the succeeding layers, and with the number of layers increasing the output deviation of the MLP becomes stable.

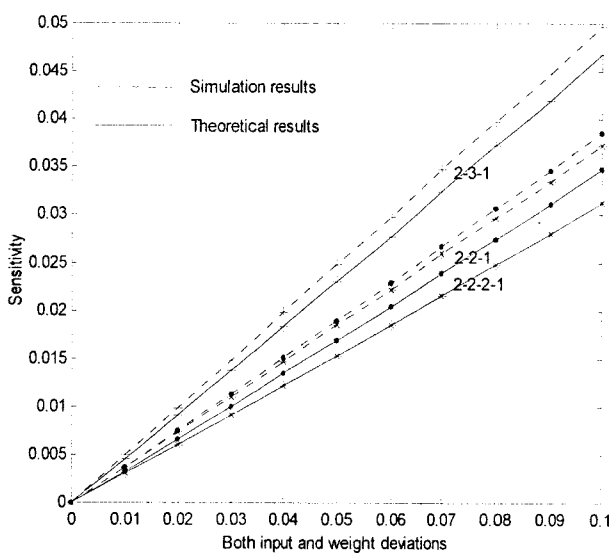

Figure 4.3: Sensitivity of MLPs to input and weight deviations.

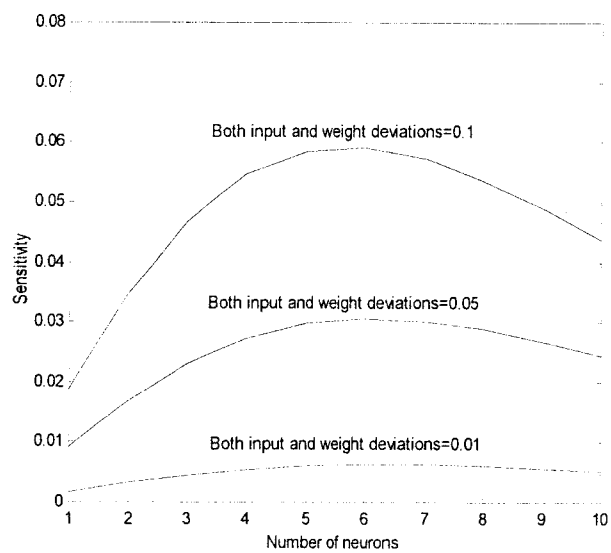

Figure 4.4: Sensitivity of MLPs to the number of neurons in hidden layer.

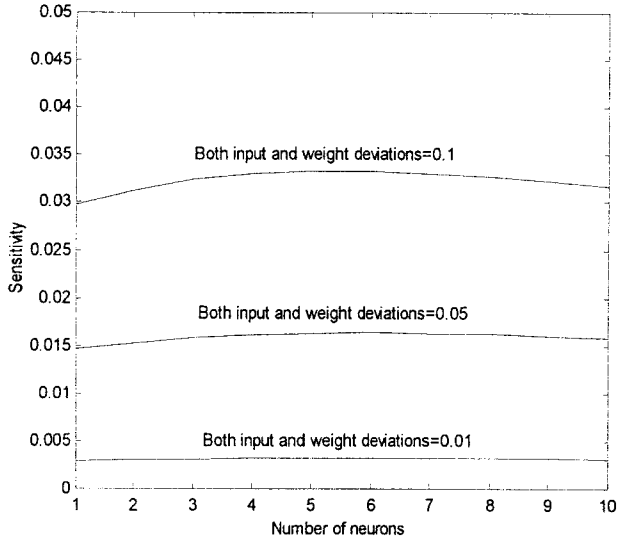

Figure 4.5: Sensitivity of MLPs to the dimension of input.

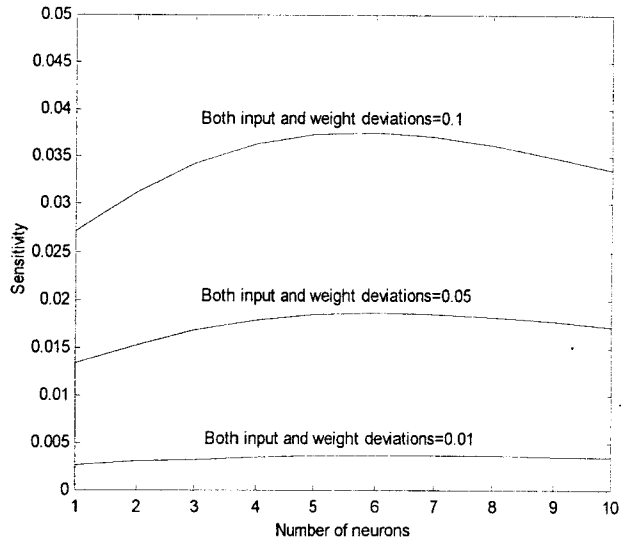

Figure 4.6: Sensitivity of MLPs to the number of neurons in the 1st layer.

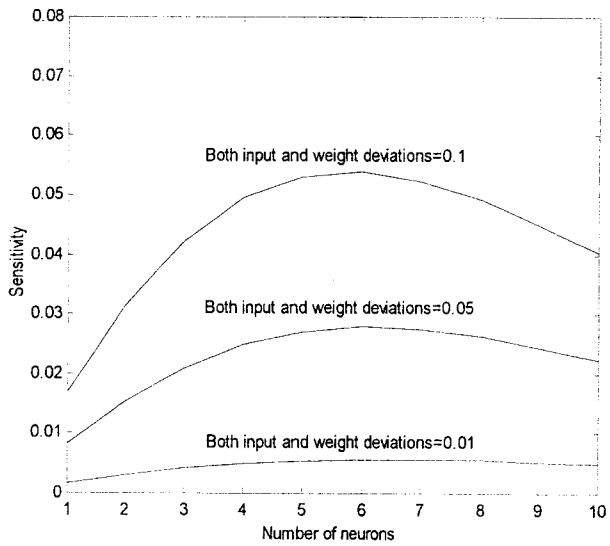

Figure 4.7: Sensitivity of MLPs to the number of neurons in the 2nd layer.

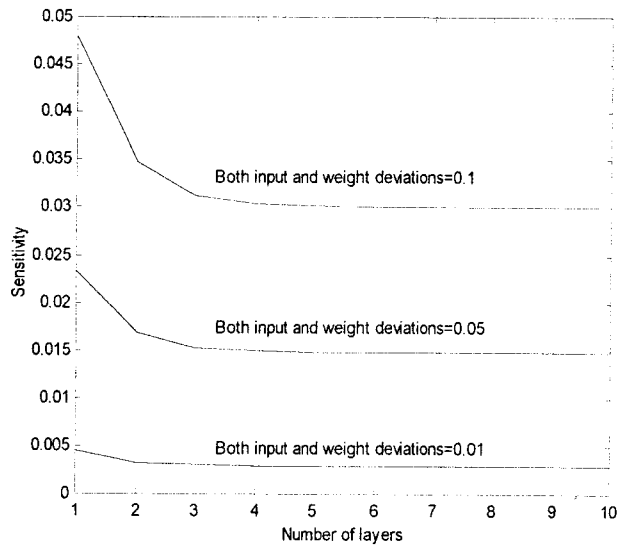

Figure 4.8: Sensitivity of MLPs to the number of layers.

4.4 Simulation verification

In order to verify the above theoretical results, a computer simulation was run to obtain simulation results. As the sensitivity of a single neuron is fundamental to the sensitivity

of an entire MLP, the sensitivity for a single neuron is considered first. In the simulation with a given n^{l-1} (the dimension of input), more than 100000000 samples are randomly selected with different inputs and weights distributed uniformly in the hypercube $[0,1]^{2n^{l-1}}$. For a given input deviation and weight deviation (to be consistent with the above theoretical results, they are assigned identical values), the average output deviation for a neuron over all the selected samples is computed. The dotted lines in figures 4.1 and 4.2 show the simulation results. The simulation for an entire MLP was done with organizing such 3 MLPs: $2-2-1$, $2-3-1$, and $2-2-2-1$. Similar to a neuron, the average output deviation for each network is computed over all selected samples under a given deviation for both input and weight. The dotted lines in Figure 4.3 show the simulation results. The agreement between the theoretical and the simulation results is quite good. The differences between the theoretical and the simulation results in the three figures are caused more or less by insufficient samples in the simulation. For Figure 4.3, the main cause is the interval $[0,1]^{n^{l-1}}$ used in the theoretical formulae (4.7), (4.8) and (4.9) for the inputs of the layers that are not the first layer. Since the summation of weighed inputs is always positive and so the value of the logistic function is always greater than or equal to $\frac{1}{2}$, the match will be better if the interval $[0,1]^{n^{l-1}}$ is substituted by $[\frac{1}{2},1]^{n^{l-1}}$ for the inputs of each layer l, where $l \geq 2$.

4.5 Summary

In this chapter, the derived formula for the sensitivity of a single neuron, especially its

simplified form (4.10), clearly expresses the relationship between neuron's output deviation and its causes — input and weight deviations. The given algorithms for the computation of the sensitivity of an entire MLP make it possible to analyze the behaviors of the MLP's sensitivity. As intuitively expected, the sensitivity increases with input and weight deviations, but the increase has an upper bound that is determined by the structural configuration of the MLP, namely the number of neurons per layer and the number of layers. There exists an optimal value for the number of neurons in a layer, which yields the highest sensitivity value. The effect caused by the number of layers is quite unexpected. The sensitivity of a neural network may decrease at first and then almost keeps constant while the number increases. The theoretical results obtained are in reasonable agreement with simulation results.

The analysis result thus obtained, although being too general to be applicable to a specific MLP, does disclose some significant behaviors of MLPs. For example, there is a well-known claim that with a sufficient number of neurons the MLP will tend to memorize the training patterns and thus has poor generalization ability. This phenomenon can be explained and justified by the above analysis results. It is known that a trained MLP will memorize each trained input pattern, say X'. A new input X'' (not in the training data set) can be regarded as a deviated trained pattern, denoted as $X'' = X' + \Delta X$. According to the above analysis that more number of neurons (over a maximum value) in a layer and more layers of an MLP will decrease the sensitivity

more, sufficient number of neurons, which implies more neurons per layer or more layers, will decrease the sensitivity. Less sensitivity means the output for the new pattern X^n tends to that of the trained pattern X^t.

Generally, the sensitivity analysis could provide some a priori information and guidelines to aid the networks' design. For instance, it can be helpful to the feasibility study of hardware design. With the analysis results designers could know prior to hardware implementation how much deviation the limited hardware will cause and whether the expected output deviation will exceed the design specification. The architecture design of the neural networks can also benefit from the sensitivity analysis. Under certain sensitivity requirements, the network designers could be guided with the analysis results to find the appropriate number of neurons in a layer and the appropriate number of layers in a network.

Chapter 5

Sensitivity measure for an individual MLP

The results obtained in the last chapter provide useful information to aid the network design, but they are too rough to measure the sensitivity for any specific individual MLP that has fixed weight and thus fixed architectural parameters. Being different from the last chapter, the purpose of this chapter is to investigate a quantified sensitivity measure for an individual MLP. Since weight is assumed fixed, there is no meaning to consider weight deviation any more, and the sensitivity here is only considered arising from input deviation. In order to quantify the sensitivity, some analytical expressions that are functions of input deviation are derived. Based on these expressions two algorithms are given to compute the sensitivity for a single neuron and an entire network.

5.1 Sensitivity for neurons

In this section, the superscript and the subscript are omitted in the expressions of neuron's input and weight. And the sensitivity of a neuron is modified as follows.

Definition 5.1: For input vector $X \in [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n] \subseteq [0,1]^n$ and its corresponding deviation ΔX, and a given weight vector $W \in R^n$, the sensitivity of a neuron is defined as the mathematical expectation of the absolute value of Δy with

respect to X, which is expressed as

$$s = E(|g(X)|) = E(|f((X+\Delta X)*W) - f(X*W)|). \tag{5.1}$$

It is noticed that the fixed weight of a neuron may cause its output, which is the input of the neurons in the succeeding layer, to fall into a narrower interval than $[0,1]$. To be suitable for the situation of the networks trained with fixed weights, this definition specifies that each component of input could vary in a small continuous interval contained by $[0,1]$, and that each component of weight to be a real number rather than to vary in a continuous interval.

According to the principle of the mathematical expectation of multivariate continuous random variables, (5.1) can be expressed as

$$s = \int_{\Omega} \cdots \int \varphi(X) \, | f((X+\Delta X)*W) - f(X*W) | \, d\Omega, \tag{5.2}$$

where $\varphi(X)$ is the density function and Ω is the integral region spanned by stochastic variable X. By assuming that the elements of X are independent of each other, (5.2) can further be rewritten as

$$s = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_n}^{b_n} \varphi(X) \, | f((X+\Delta X)*W) - f(X*W) | \, dx_1 dx_2 \ldots dx_n . \tag{5.3}$$

Because of (1.7), namely, $f(x)$ being a monotone increasing function, and the fact that $\varphi(X) \geq 0$, the symbol of the absolute value can be moved out of the integral. Thus

$$s = \begin{cases} \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f((X+\Delta X)*W)dx_1dx_2..dx_n - \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f(X*W)dx_1dx_2..dx_n, \text{ if } \Delta X*W \geq 0 \\ \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f(X*W)dx_1dx_2..dx_n - \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f((X+\Delta X)*W)dx_1dx_2..dx_n, \text{ if } \Delta X*W < 0 \end{cases}$$

$$= \Big| \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f((X+\Delta X)*W)dx_1dx_2...dx_n - \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f(X*W)dx_1dx_2...dx_n \Big|. \tag{5.4}$$

By substituting the inner product with summation and $f(\cdot)$ with logistic function and by denoting $\sigma = \sum_{j=1}^{n}\Delta x_j w_j$, (5.4) becomes

$$s = \Big| \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f(\sum_{j=1}^{n}(x_j+\Delta x_j)w_j)dx_1dx_2..dx_n - \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)f(\sum_{j=1}^{n}x_jw_j)dx_1dx_2..dx_n \Big|$$

$$= \Big| \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)/(1+e^{-\sum_{j=1}^{n}x_jw_j-\sigma})dx_1dx_2...dx_n - \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)/(1+e^{-\sum_{j=1}^{n}x_jw_j})dx_1dx_2...dx_n \Big|. \tag{5.5}$$

Therefore, the computation of the sensitivity is attributed to the calculation of the integral

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)/(1+e^{-\sum_{i=1}^{n}x_i w_i-\sigma})dx_1dx_2\ldots dx_n. \tag{5.6}$$

A detailed approximate solution for the calculation of (5.6) under the assumption of $\varphi(X)=1$, namely X being uniformly distributed in the hyperrectangle $[a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_n,b_n]$, is given in appendix C. With this solution, an algorithm to compute the sensitivity for a single neuron is given as follows, in which $A=\{a_1,a_2,\ldots,a_n\}$ and $B=\{b_1,b_2,\ldots,b_n\}$.

•Algorithm: NEURON_SENSITIVITY(A, B, $\varphi(X)$, ΔX, W)

1) $\sigma=\Delta X * W$;

2) $I(\sigma)\approx\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)/(1+e^{-\sum_{i=1}^{n}x_i w_i-\sigma})dx_1dx_2\ldots dx_n$;

3) $I(0)\approx\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}\varphi(X)/(1+e^{-\sum_{i=1}^{n}x_i w_i})dx_1dx_2\ldots dx_n$;

4) $s=|I(\sigma)-I(0)|$ is the sensitivity for a neuron.

5.2 Sensitivity for networks

With the same rationale, the sensitivity of the entire network can be computed neuron by neuron from the first layer to the last layer. But, since the weight is fixed, the input space and distribution of a neuron in layer l $(2\le l\le L)$ may be changed by the neurons in its previous layers. For the sake of a better accuracy, the limits A and B, and the

density function $\varphi(X)$ for each layer in the above algorithm may need to be adjusted.

•Algorithm: MLP_SENSITIVITY($A, B, \varphi(X^1), \Delta X^1, W$)

1) Arrange given input deviation ΔX^1, i.e. S^0, and weight set W into arrays;

2) Compute the sensitivity neuron by neuron and layer by layer:

   For $l = 1$ step 1 to L

   If $l > 1$ then adjusting A, B, and $\varphi(X^l)$;

   For i = 1 step 1 to n^l

   Call NEURON_SENSITIVITY($A, B, \varphi(X^l), S^{l-1}, W_i^l$) and save result s_i^l;

3) S^L is the computed sensitivity for the MLP.

5.3 Simulation verification

To verify the derived theoretical results, experiments are conducted. Again a single neuron is considered first. In the experiments, for the sake of simplicity and without losing generality of spatial dimension and direction, eight neurons with 4 different input dimensions from $n = 2$ to $n = 5$ and different weight values are used. Each neuron holds either all positive weight values or separating positive and negative weight values. As to the case of the neurons with all negative weight values, it could be reduced to the case of all positive elements by using the characteristic (1.6) of the logistic function. Some abstract values of the weights for these neurons are given in table 5.1. Both theoretical computation and computer simulation are based on the

contents of this table. The theoretical results, i.e., the sensitivities, for each neuron is computed by the algorithm NEURON_SENSITIVITY($A, B, \varphi(X), \Delta X, W$) with the parameters $A = \{0\}$, $B = \{1\}$, $\varphi(X) = 1$ (i.e., X is uniformly distributed in $[0,1]^n$, $2 \le n \le 5$), and the elements of ΔX being all identical and ranging from 0 to 0.1 with step 0.01. The simulation results, i.e., the average output deviations, for each neuron is obtained with respect to more than 100000000 inputs that are selected randomly and uniformly from the interval $[0,1]^n$ ($2 \le n \le 5$). Both theoretical results and simulation results are shown in figures 5.1, 5.2, 5.3, and 5.4. It is clear that the agreement between the two results is very good. However, these figures also suggest that the differences between the two results for a single neuron seem to increase when the input dimension n increases. This could be explained by the fact that an increase in n would result in using more approximations in computing the integral (5.6) in a higher dimensional space. It would also result in an exponential increase of the number of sample points needed in the simulations. Both increases in computational efforts are severely affected by the precision of the computer hardware used in our experiments.

Parameter	n	$w(1)$	$w(2)$	$w(3)$	$w(4)$	$w(5)$
Neuron 1	2	1.0	2.0			
Neuron 2	2	1.0	-2.0			
Neuron 3	3	1.0	2.0	3.0		
Neuron 4	3	1.0	-2.0	3.0		
Neuron 5	4	1.0	2.0	3.0	4.0	
Neuron 6	4	1.0	-2.0	3.0	-4.0	
Neuron 7	5	1.0	2.0	3.0	4.0	5.0
Neuron 8	5	1.0	-2.0	3.0	-4.0	5.0

Table 5.1: The weight dimension and weight values for the 8 neurons.

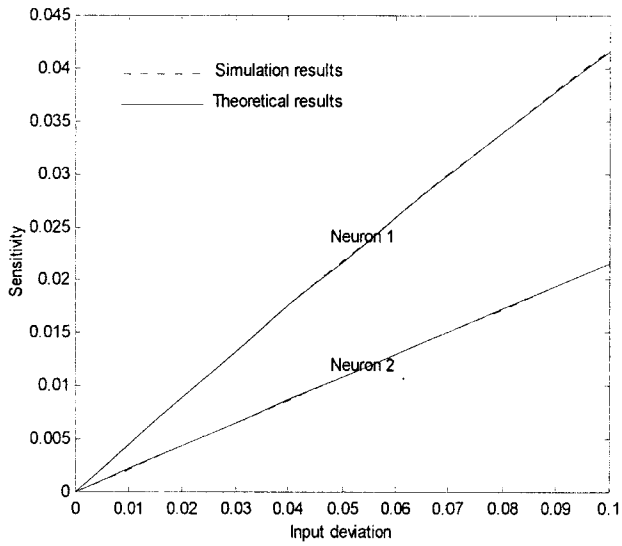

Figure 5.1: Sensitivity of the neurons with 2-dimensional weights to input deviation.

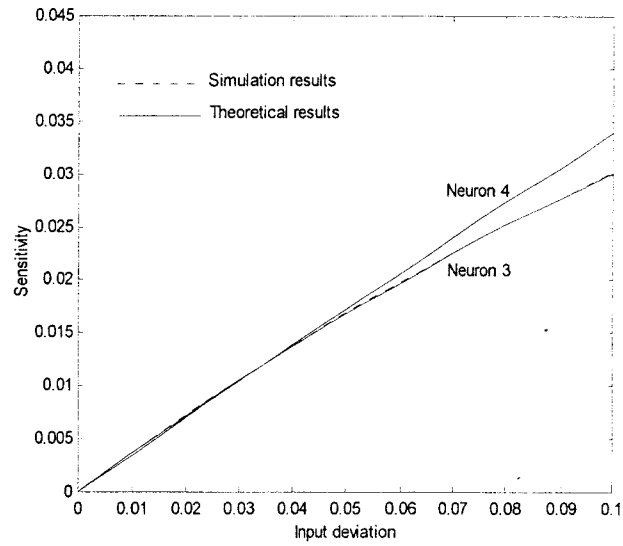

Figure 5.2: Sensitivity of the neurons with 3-dimensional weights to input deviation.

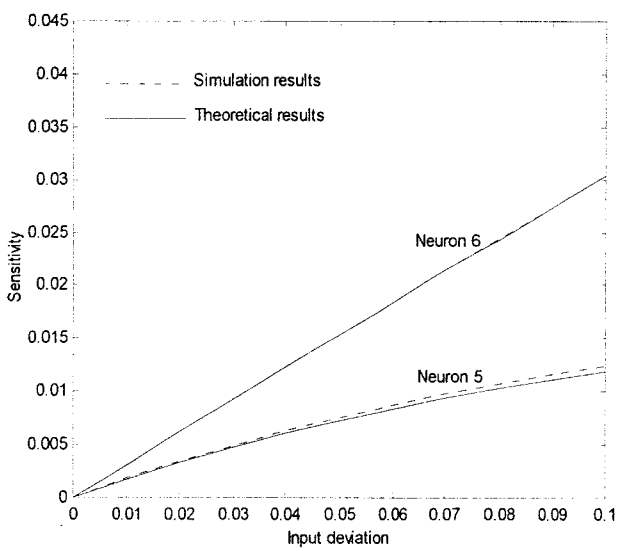

Figure 5.3: Sensitivity of the neurons with 4-dimensional weights to input deviation.

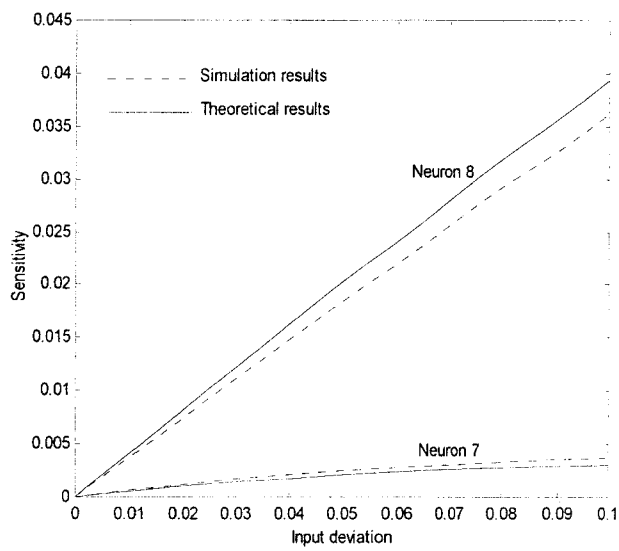

Figure 5.4: Sensitivity of the neurons with 5-dimensional weights to input deviation.

The experiments on the entire networks were made for 3 MLPs: $2-2-1$, $2-3-1$, and $2-2-2-1$. Their abstract weight values are given in Table 5.2. Similar to the verification of individual neurons, the theoretical computation and computer simulation

are made on the given weight sets and the input deviations are ranging from 0 to 0.1 with step 0.01. The sensitivity for each network is computed by the algorithm MLP_SENSITIVITY(A , B , $\varphi(X)$, ΔX^1 , W) with $A = \{0\}$, $B = \{1\}$, $\varphi(X) = 1$, and they are not adjusted throughout the computations. The average output deviations for each network is with respect to more than 100000000 inputs that are selected uniformly from interval $[0,1]^2$. As for the case of the entire network, we would only expect the both the theoretical results and the simulation results show similar trends (as indicated in figures 5.5 and 5.6), but not close approximations to each other. This is especially true when the number of layers increases since the integration limits and density function for the integral (5.6) in each successive layer are not modified in order to ease the computational efforts involved. Both the theoretical and the simulation results are shown in figures 5.5 and 5.6 separately.

W	$w(1;1,1)$	$w(1;1,2)$	$w(1;2,1)$	$w(1;2,2)$	$w(1;3,1)$	$w(1;3,2)$
	$w(2;1,1)$	$w(2;1,2)$	$w(2;1,3)$	$w(2;2,1)$	$w(2;2,2)$	
	$w(3;1,1)$	$w(3;1,2)$				
2-2-1	-5.0	6.0	-3.0	4.0		
	-1.0	2.0				
2-3-1	1.0	-2.0	3.0	-4.0	5.0	-6.0
	7.0	-8.0	9.0			
2-2-2-1	10.0	-9.0	8.0	-7.0		
	-5.0	6.0	-3.0	4.0		
	-1.0	2.0				

Table 5.2: Three sets of weight values.

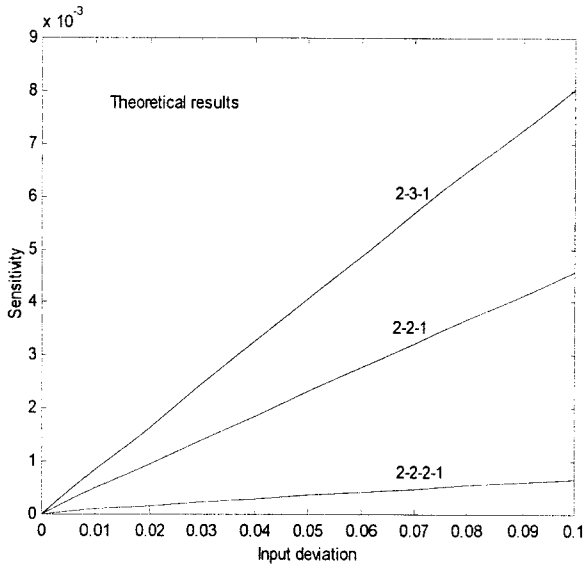

Figure 5.5: Sensitivity of the MLPs to input deviation.

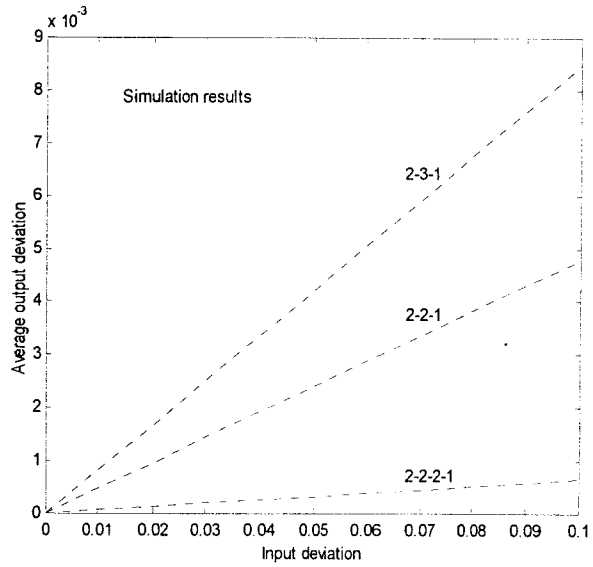

Figure 5.6: Average output deviation of the MLPs to input deviation.

5.4 Summary

The sensitivity measure is something like an analogue rule, which network users can use as a tool to evaluate their networks' performance. In fact, different applications may have different sensitivity requirements. Without loss of generality, an MLP can be regarded as a function approximator. From this point of view, it can be imaged that there exits a relationship between the function to be approximated and the sensitivity of the network to realize the function. For example, the two ends of the sensitivity rule are separately required by two extreme types of functions. One is the constant function that requires a corresponding network with the sensitivity being zero. The other is a function with the highest slope curve that requires a network with the sensitivity being highest. Knowing the sensitivity requirement of an application, one can use the

sensitivity rule to measure and compare a set of candidate networks, and then select the best one. The selection of weights for a more robust MLP during the training stage is one of the applications that will be discussed in the next chapter.

Chapter 6

Applications of the sensitivity study

In the last two chapters, two kinds of sensitivity are given for different situations with different purposes. One is for networks under design in which weight has not been determined and architectural parameters could be adjusted. It has been shown that sensitivity information could be helpful to give some guidance for network design. The other is for networks with fixed weight and architectural parameters. The network users could use the sensitivity as a rule to measure the networks' performance. This chapter presents one application for each type.

6.1 Hardware design using sensitivity analysis

One of the most important factors for hardware design is to find out how much the limited hardware precision affects its output accuracy so that one could predict whether the expected output deviation will exceed the design specification.

Let's consider a simple example that shows the use of the sensitivity in hardware design. Suppose the network weight connections can be perfectly represented by a 32-bit hardware. However, due to the limitation of the hardware representation, the network can represent the weight with 16 bits only. That means the least insignificant 16 bits are truncated. Alternatively it can be regarded as adding weight deviations to the original weight such that the last 16 bits of the original weight become zeros. In

this case, the weight interval is $[0,2^{32}-1]$ and the maximum weight deviation is $2^{16}-1$. With this information, the algorithm given in chapter 4 can be used to find the sensitivity for the MLP to be designed with given architectural configurations. The sensitivity can aid the network designers to predict whether the suggested truncation from 32 to 16 bits is acceptable.

Furthermore, the network designers can make use of the knowledge obtained from sensitivity analysis in chapter 4 to see if it is possible to reduce the sensitivity by adjusting the architectural parameters such as the number of layers in the network and the number of neurons in each layer under the condition that the adjustments made will not violate the design specification.

6.2 Selection of weights using sensitivity measure

The sensitivity measure given in chapter 5 can be used as a relative criterion to evaluate networks' performance. An application of the sensitivity measure to the selection of weights for a more robust MLP is demonstrated in this section.

It is well known that with different initial weight sets the BP algorithm will finally obtain different sets of weights in training, and even with the same initial weight set the BP algorithm may also find different sets of weights that satisfy the application requirements. Now, a practical question is which one among those obtained weight sets should be chosen so that the trained MLP has more error-tolerance capability? Obviously, with the sensitivity measure this question can be answered. Since the sensitivity measure actually reflects the average output deviation for the overall inputs

under perturbations, so from the error-tolerance point of view, the sensitivity of the network in training should be the smaller the better.

In order to illustrate the selection of weights for a more robust MLP, a continuous mapping example given in [33] is employed here. In the example, a real-valued continuous mapping

$$F(x_1, x_2) = 0.5x_1x_2 + x_1e^{-x_1x_2}, \tag{6.1}$$

where $(x_1, x_2) \in [0,1] \times [0,1]$ is realized by a three-layer MLP with an architecture $2-5-1$ and the logistic function as the activation function. [33] gives four sets of trained weights for the MLPs and states that they have the same performance in realizing the mapping (6.1). Actually MLPs with the same structure 2-5-1 but different weight sets are considered to be different. In the following illustration, two of the four weight sets are used and they are listed in Table 6.1.

W	$w(1;1,1)$	$w(1;1,2)$	$w(1;2,1)$	$w(1;2,2)$	$w(1;3,1)$
	$w(1;3,2)$	$w(1;4,1)$	$w(1;4,2)$	$w(1;5,1)$	$w(1;5,2)$
	$w(2;1,1)$	$w(2;1,2)$	$w(2;1,3)$	$w(2;1,4)$	$w(2;1,5)$
Set 1	6.145	-5.106	2.701	0.443	-8.550
	-0.230	-1.927	-1.490	-1.165	0.640
	5.588	3.769	-6.820	-1.370	-1.413
Set 2	8.879	0.180	-1.659	0.134	-1.958
	-0.265	-1.426	-0.595	6.172	-5.013
	6.133	-2.393	-3.569	-1.842	5.397

Table 6.1: Two sets of weight values.

With the two weight sets and given the input deviations ranging from 0 to 0.1 with step 0.01, sensitivities can be computed by the algorithm MLP-SENSITIVITY($\Delta X^1, W$) for the two resulting MLPs. The results are shown in Figure 6.1. According to the sensitivity results, it can be concluded that the MLP with set 2 is more robust than the MLP with set 1. This conclusion can be verified by computer simulation. In the simulation, the two MLPs are implemented and the averages of their output deviations are obtained by feeding them with more than 100000000 inputs that are uniformly distributed in the interval $[0,1]^2$ with deviations ranging from 0 to 0.1 with step 0.01. The simulation results are shown in Figure 6.2. It is clear that the simulation results support the theoretical results. Hence, under the assumption that the MLP with set 1 and the MLP with set 2 have the same performance in realizing the mapping (6.1), and aiming at obtaining a more robust MLP, it is set 2 rather than set 1 that should be selected as the MLP's weight set.

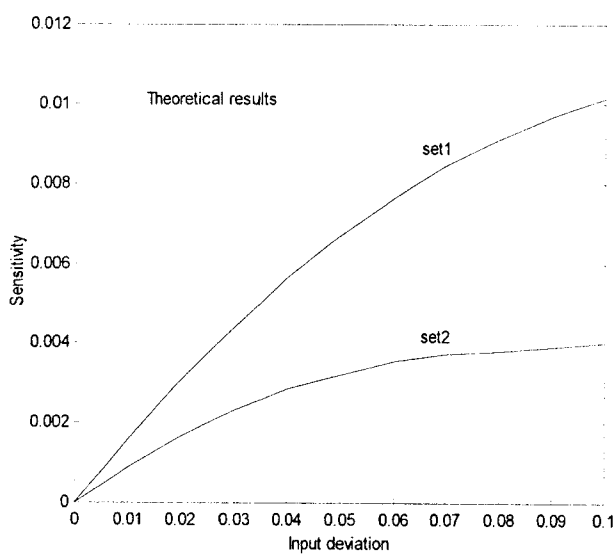

Figure 6.1: Theoretical results for the two MLPs.

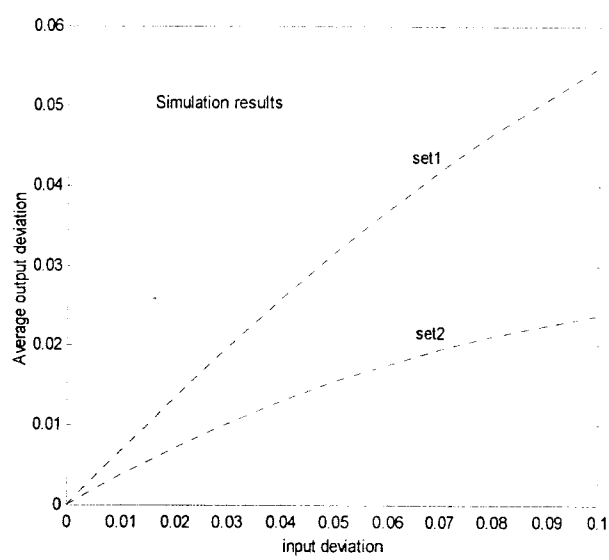

Figure 6.2: Simulation results for the two MLPs.

Usually the required expectation with respect to overall inputs may not be true for every specific input. But it must be true for most of the inputs of the MLP with a smooth activation function. Below are some simulation results on the two given MLPs which show the output deviations obtained by feeding them with specific input (x_1, x_2) which has deviations ranging from 0 to 0.1 with step 0.01. The simulation results at inputs (0.02, 0.08), (0.05, 0.05), and (0.08, 0.02) are shown in figures 6.3 to 6.5 for input data on scale 10^{-2}, and (0.2, 0.8), (0.5, 0.5), and (0.8, 0.2) in figures 6.5 to 6.7 for input data on scale 10^{-1}. These results all support the above theoretical conclusion of the weight selection.

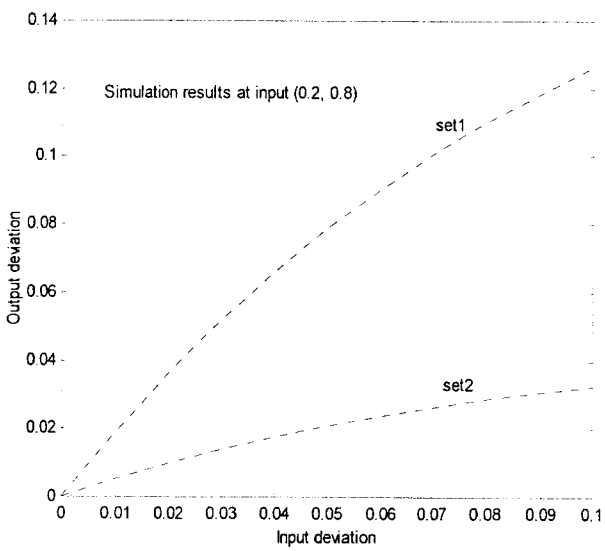

Figure 6.3: Simulation results for the two MLPs at input (0.2 ,0.8).

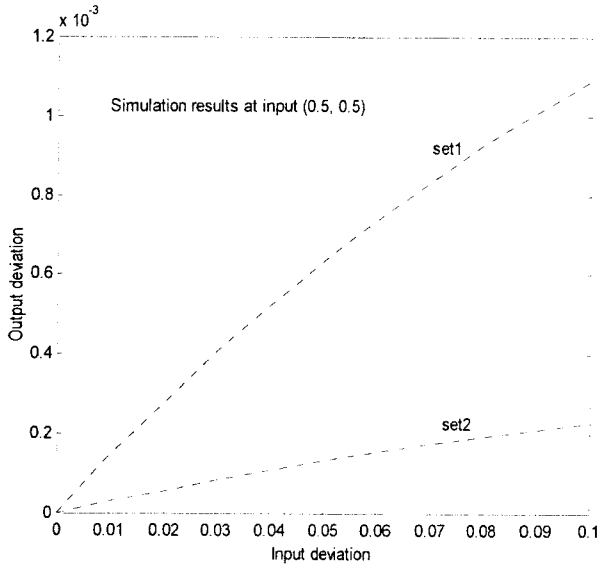

Figure 6.4: Simulation results for the two MLPs at input (0.5 ,0.5).

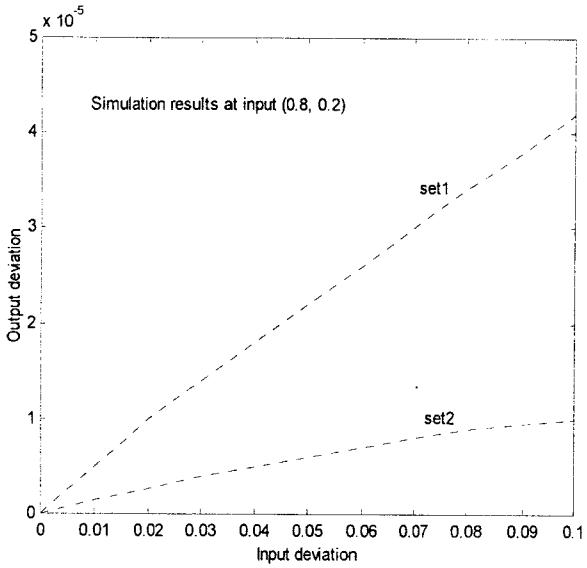

Figure 6.5: Simulation results for the two MLPs at the input (0.8 ,0.2).

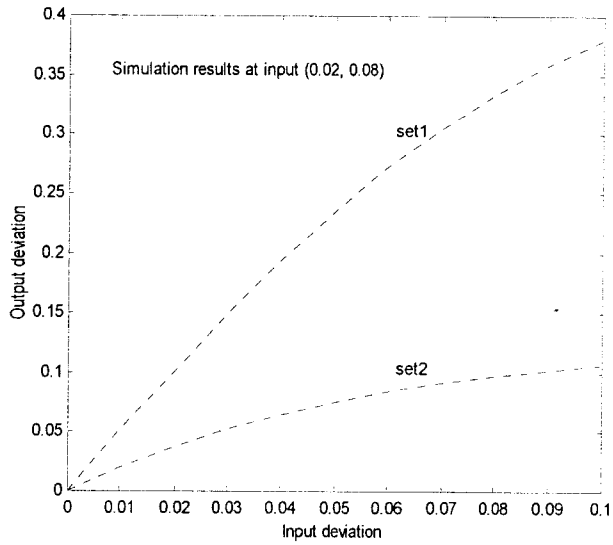

Figure 6.6: Simulation results for the two MLPs at the input (0.02 ,0.08).

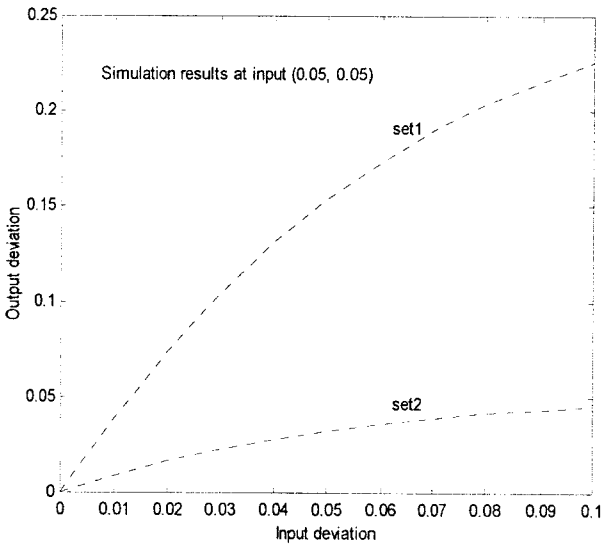

Figure 6.7: Simulation results for the two MLPs at the input (0.05 ,0.05).

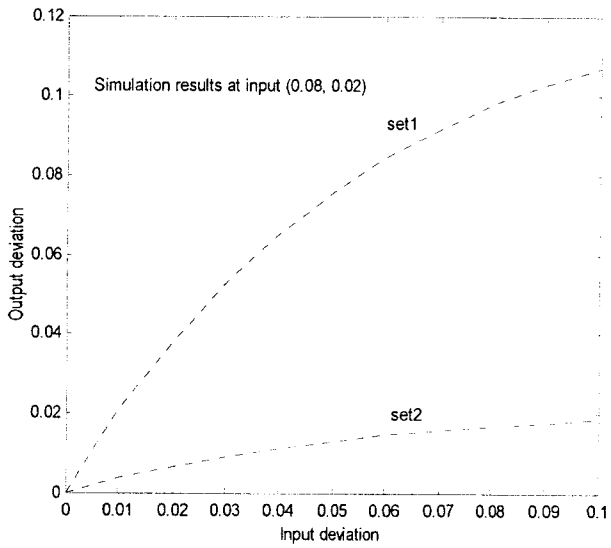

Figure 6.8: Simulation results for the two MLPs at the input (0.08 ,0.02).

6.3 Summary

This chapter discusses two applications treating sensitivity as output error to predict and improve the networks' error-tolerance capability. Form another point of view, if

the sensitivity is regarded as an output increment, it may be possible to use sensitivity to assess a network's generalization ability. It seems that there exits a relationship among error tolerance, sensitivity, and generalization ability of a network under certain application requirements. By clearly defining this relationship and using sensitivity as a tool, it may be possible to find an appropriate balance between the error tolerance and the generalization ability in designing MLPs to achieve higher performance.

Chapter 7

Conclusion and future research

7.1 Conclusion

In this dissertation the sensitivity of the MLP to input and weight perturbations has been systematically and theoretically studied. To match the structural feature of the MLP, a bottom up approach is followed from the sensitivity of a neuron to a layer and finally to the entire network. To reflect neural networks' statistical nature, mathematical expectation is used to define the sensitivity as the average of output deviations due to input and weight perturbations with respect to overall inputs and weights in given intervals. Two typical situations in the MLP's life cycle are mainly addressed. One is the MLP without fixed weight and architecture during the design phase. The other is the trained MLP with fixed weight and thus fixed architecture. Since most network parameters are variables in the former case, the sensitivity derived under this circumstance can only be expected to give some rough information on the behaviors of an ensemble of MLPs in the presence of input and weight perturbations. With more fixed parameters in the latter case, the sensitivity can be more accurate as a measure to evaluate the performance of an individual MLP.

The analysis of the sensitivity derived for an ensemble of MLPs given in chapter 4, although it is too rough to be applied to an individual MLP, does reveal some significant trends of behaviors of the MLP's sensitivity. It has been found that the

sensitivity increases with input and weight deviations. But the increase has an upper bound that is related to the number of neurons per layer and the number of layers. There exists an optimal value for the number of neurons in a layer, which yields the highest sensitivity value. As to the number of layers, the sensitivity may decrease at first and then almost keeps constant while the number increases.

The sensitivity derived as a measure for an individual MLP given in chapter 5 is more accurate and suitable for the performance evaluation of individual MLPs.

The results of the sensitivity analysis are useful guidelines for networks' designers to predict the feasibility of hardware designs and to find the appropriate network architecture. The sensitivity measure is a useful tool for networks' users to estimate the performance of implemented networks. In chapter 6, the applications of the sensitivity analysis to hardware design and the sensitivity measure to the selection of weights for a more robust MLP are discussed.

7.2 Suggestion for future research

Three possible research areas are suggested in this section. The first one is to relax or strengthen the assumptions on which the MLP sensitivity results are based to obtain more applicable or accurate results. The second suggested area is to consider the sensitivity with respect to other statistical variables such as input or weight deviations, and derive formulas for its computations. And the third suggested area is to use the sensitivity results to study other theoretical issues of MLPs. These three suggested research areas are discussed briefly in the following subsections.

7.2.1 Different assumptions

In the computations of the sensitivity presented in the previous chapters, it is assumed that the input interval for each layer is [0,1] and the inputs are uniformly distributed in the interval. Actually these assumptions can be relaxed or tightened to gain more precise, but more complicated, results and to be more suitable for real applications. Because the input interval of a neuron in layer l $(2 \le l \le L)$ depends on the output of the neurons in layer $l-1$, the adjustment of the interval bounds for each layer l may be needed. Taking (1.7) into consideration, it is not difficult to find the new input bounds for layer l $(2 \le l \le L)$ by the bounds and the given weight information of layer $l-1$.

As to the selection of distribution density, it is quite application dependent. But the sensitivity formulas for a neuron need to be re-derived for every new density function. Furthermore, in chapter 5, the input distribution for layer l $(2 \le l \le L)$ may be different from layer $l-1$ because of the given weights in layer $l-1$. The re-derivation of the formulas and especially the determination of the varying density functions are difficult problems in need of more research work.

7.2.2 Other kinds of sensitivity

Table 3.1 lists several cases to which the sensitivity study is applicable. Some of them, although less general, do have practical values. For example, an application that has very limited number of inputs but the input deviations are scattered in a given range may often encountered in real life situations. In this case the input is no longer

appropriate to be taken as a statistical variable and instead it should be the input deviation. To accommodate these changes, new sensitivity definition and its computations are worthy of exploring.

7.2.3 Application of the MLP sensitivity results

The Applications of the results presented in this dissertation are very interesting and promising research topics. In chapter 6, two direct applications of the results have shown the merits of the sensitivity study. It is expected that some other applications such as the measurement of the network's generalization ability, the pruning of the network architecture, and neural network driven control problems will benefit from the sensitivity study.

Appendix A

Derivation of the sensitivity formula for a neuron

This appendix gives the detailed derivation of the analytical expression for the computation of the sensitivity of a neuron without fixed weight. According to the principle of the mathematical expectation of multivariate continuous random variables, the formula for the sensitivity defined in (4.1) can be derived as follows, in which Ω_X and Ω_W are the hypercubes $[0,1]^n$ spanned by input and weight vectors.

$$s = E(|g(X,W)|)$$

$$= E(|f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W)|)$$

$$= \int_{\Omega_X}\int_{\Omega_W} \varphi(X,W)\,|f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W)|\,d\Omega_X d\Omega_W$$

$$= \int_{\Omega_X}\int_{\Omega_W} |f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W)|\,d\Omega_X d\Omega_W \qquad \text{by (4.3)}$$

$$= \int_{\Omega_X}\int_{\Omega_W} (f((X+|\Delta X|)*(W+|\Delta W|)) - f(X*W))\,d\Omega_X d\Omega_W \qquad \text{by (1.7)}$$

$$= \int_{\Omega_X}\int_{\Omega_W} (1/(1+e^{-\sum_{j=1}^{n}(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)}) - 1/(1+e^{-\sum_{j=1}^{n}x_j w_j}))\,d\Omega_X d\Omega_W \qquad \text{by (1.3)}$$

$$\approx \int_{\Omega_X}\int_{\Omega_W} (\sum_{k=0}^{p}(-1)^k e^{-k\sum_{j=1}^{n}(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)} - \sum_{k=0}^{p}(-1)^k e^{-k\sum_{j=1}^{n}x_j w_j})\,d\Omega_X d\Omega_W \qquad \text{by (4.4)}$$

$$= \int_{\Omega_X}\int_{\Omega_W} (\sum_{k=1}^{p}(-1)^k (e^{-k\sum_{j=1}^{n}(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)} - e^{-k\sum_{j=1}^{n}x_j w_j}))\,d\Omega_X d\Omega_W$$

$$= \int_{\Omega_X}\int_{\Omega_W} (\sum_{k=1}^{p}(-1)^k (\prod_{j=1}^{n} e^{-k(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)} - \prod_{j=1}^{n} e^{-kx_j w_j}))\,d\Omega_X d\Omega_W$$

$$=\sum_{k=1}^{p}(-1)^k(\int_{\Omega_X}\int_{\Omega_W}(\prod_{j=1}^{n}e^{-k(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)}-\prod_{j=1}^{n}e^{-kx_jw_j})d\Omega_X d\Omega_W)$$

$$=\sum_{k=1}^{p}(-1)^k(\int_0^1\ldots\int_0^1(\prod_{j=1}^{n}e^{-k(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)}-\prod_{j=1}^{n}e^{-kx_jw_j})dx_1\ldots dx_n dw_1\ldots dw_n)$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\int_0^1\int_0^1 e^{-k(x_j+|\Delta x_j|)(w_j+|\Delta w_j|)}dx_j dw_j)-\prod_{j=1}^{n}(\int_0^1\int_0^1 e^{-kx_jw_j}dx_j dw_j))$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\int_0^1[-e^{-k(x_j+\Delta x_j)(w_j+\Delta w_j)}/(k(w_j+|\Delta w_j|))]_0^1 dw_j)-\prod_{j=1}^{n}(\int_0^1[-e^{-kx_jw_j}/(kw_j)]_0^1 dw_j))$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\int_0^1(e^{-k(\Delta x_j)(w_j+\Delta w_j)}-e^{-k(1+\Delta x_j)(w_j+\Delta w_j)})/(k(w_j+|\Delta w_j|))dw_j)-\prod_{j=1}^{n}(\int_0^1(1-e^{-kw_j})/(kw_j)dw_j))$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\int_0^1 e^{-k(1+\Delta x_j)(w_j+\Delta w_j)}(e^{k(w_j+\Delta w_j)}-1)/(k(w_j+|\Delta w_j|))dw_j)-\prod_{j=1}^{n}(\int_0^1 e^{-kw_j}(e^{kw_j}-1)/(kw_j)dw_j))$$

$$\approx\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\int_0^1 e^{-k(1+\Delta x_j)(w_j+\Delta w_j)}(\sum_{h=1}^{q}(k(w_j+|\Delta w_j|))^{h-1}/h!)dw_j)-\prod_{j=1}^{n}(\int_0^1 e^{-kw_j}(\sum_{h=1}^{q}(kw_j)^{h-1}/h!)dw_j))$$

by (4.5)

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\sum_{h=1}^{q}(\int_0^1 e^{-k(1+\Delta x_j)(w_j+\Delta w_j)}(k(w_j+|\Delta w_j|))^{h-1}/h!dw_j))-\prod_{j=1}^{n}(\sum_{h=1}^{q}(\int_0^1 e^{-kw_j}(kw_j)^{h-1}/h!dw_j)))$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\sum_{h=1}^{q}((k^{h-1}/h)I_1))-\prod_{j=1}^{n}(\sum_{h=1}^{q}((k^{h-1}/h)I_2)))$$

$$=\sum_{k=1}^{p}(-1)^k(\prod_{j=1}^{n}(\sum_{h=1}^{q}((k^{h-1}/h)I_1))-(\sum_{h=1}^{q}((k^{h-1}/h)I_2))^n),$$

where

$$I_1=\int_0^1(w_j+|\Delta w_j|)^{h-1}e^{-k(1+\Delta x_j)(w_j+\Delta w_j)}/(h-1)!dw_j$$

$$=[\sum_{r=0}^{h-1}(-1)^r(w_j+|\Delta w_j|)^{h-1-r}e^{-k(1+\Delta x_j)(w_j+\Delta w_j)}/((h-1-r)!(-k(1+|\Delta x_j|))^{r+1})]_0^1 \quad \text{by (4.6)}$$

$$=\sum_{r=0}^{h-1}(|\Delta w_j|^{h-1-r}\, e^{-k(1+|\Delta x_j|)|\Delta w_j|}-(1+|\Delta w_j|)^{h-1-r}\, e^{-k(1+|\Delta x_j|)(1+|\Delta w_j|)})/((h-1-r)!(k(1+|\Delta x_j|))^{r+1}),$$

and

$$I_2=\int_0^1 (w_j)^{h-1} e^{-kw_j}/(h-1)!\,dw_j$$

$$=[\sum_{r=0}^{h-1}(-1)^r (w_j)^{h-1-r} e^{-kw_j}/((h-1-r)!(-k)^{r+1})]_0^1 \qquad \text{by (4.6)}$$

$$=\sum_{r=0}^{h-2}(-e^{-k}/((h-1-r)!k^{r+1}))+(1-e^{-k})/k^h.$$

Appendix B

Three integral formulas

Below are three integral formulas that will be used in the derivations of appendix C. These formulas can be proven by the mathematical induction on n. In the formulas, it is assumed that k is an integer number, σ is a real number, and $A=\{a_1, a_2, \ldots, a_n\}$, $B=\{b_1, b_2, \ldots, b_n\}$, and $W=\{w_1, w_2, \ldots, w_n\}$ are three sets having n real numbers as elements with $b_j > a_j$ and $w_j \neq 0$ $(1 \leq j \leq n)$. These formulas all have the same integrand, i.e. $e^{k\sum_{j=1}^{n} x_j w_j - k\sigma}$, but different integral regions. Each region represents a typical closed volume from a hyperrectangle Ω, $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$, to part of it. In obvious reason, only 3-dimensional figures are used for illustration.

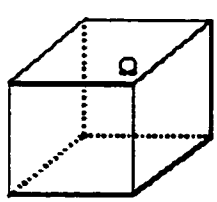

Figure B.1: The region is the entire Ω.

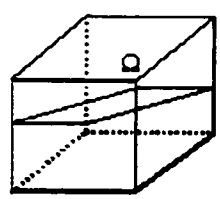

Figure B.2: The region is the lower part of Ω.

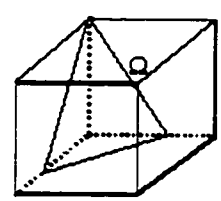

Figure B.3: The region is a prism part of Ω.

1) $$I_1(k,\sigma,A,B,W)=\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} e^{k\sum_{j=1}^{n}x_j w_j+k\sigma}\,dx_1dx_2\ldots dx_n$$

$$=\begin{cases}\prod_{j=1}^{n}(b_j-a_j), & \text{if k=0}\\[2ex] e^{k\sigma}\prod_{j=1}^{n}(e^{kb_jw_j}-e^{ka_jw_j})/(kw_j), & \text{if k}\neq 0.\end{cases} \tag{B.1}$$

The integral region of this formula is whole Ω, which is enclosed by $2n$ planes perpendicular each other, i.e. $x_j=a_j$ and $x_j=b_j$ $(1\le j\le n)$, as Figure B.1 illustrates.

2) $$I_2(k,\sigma,A,B-\{b_n\},W)=\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_{n-1}}^{b_{n-1}}\int_{a_n}^{-(\sum_{j=1}^{n-1}x_jw_j+\sigma)/w_n} e^{k\sum_{j=1}^{n}x_jw_j+k\sigma}\,dx_1dx_2\ldots dx_n$$

$$=\begin{cases}-((\sum_{j=1}^{n-1}w_j(b_j+a_j)/2)+a_nw_n+\sigma)(\prod_{j=1}^{n-1}(b_j-a_j))/w_n, & \text{if k=0}\\[2ex] \prod_{j=1}^{n-1}(b_j-a_j)/(kw_n)-e^{ka_nw_n+k\sigma}\prod_{j=1}^{n-1}(e^{kb_jw_j}-e^{ka_jw_j})/(k^n\prod_{j=1}^{n}w_j), & \text{if k}\neq 0.\end{cases} \tag{B.2}$$

The integral region of this formula is the lower part of Ω cut by hyperplane P, $\sum_{j=1}^{n}x_jw_j+\sigma=0$, through all edges parallel to an axis, say x_n-axis, as Figure B.2 illustrates. Actually this formula is also suitable for the upper part of Ω by adjusting the integral limits. Furthermore, if one is only interested in the scalar of the integral,

the condition $b_j > a_j$ can be removed because the lower limit and the upper limit for each variable in the integral can be arbitrarily exchanged without affecting the absolute value of the integral. The previous formula (B.1) and the next formula (B.3) also possess this characteristic.

$$3)\quad I_3(k,\sigma,A,W)=\int_{a_1}^{-(\sum_{j=2}^{n}a_jw_j+\sigma)/w_1}\int_{a_2}^{-(x_1w_1+\sum_{j=3}^{n}a_jw_j+\sigma)/w_2}\cdots\int_{a_{n-1}}^{-(\sum_{j=1}^{n-2}x_jw_j+a_nw_n+\sigma)/w_{n-1}}\int_{a_n}^{-(\sum_{j=1}^{n-1}x_jw_j+\sigma)/w_n}e^{k\sum_{j=1}^{n}x_jw_j+k\sigma}dx_1dx_2\ldots dx_n$$

$$=\begin{cases}(-1)^n(\sum_{j=1}^{n}a_jw_j+\sigma)^n/(n!\prod_{j=1}^{n}w_j), & \text{if } k=0\\[2ex] \sum_{l=1}^{n-1}(-1)^{n+1}(\sum_{j=1}^{n}a_jw_j+\sigma)^l/(k^{n-l}l!\prod_{j=1}^{n}w_j)+(-1)^{n+1}(1-e^{k\sum_{j=1}^{n}a_jw_j+k\sigma})/(k^n\prod_{j=1}^{n}w_j), & \text{if } k\neq 0.\end{cases}\qquad\text{(B.3)}$$

The integral region of this formula is a prism part of Ω as Figure B.3 illustrates which is enclosed by P and n planes perpendicular to each other, i.e. $x_j = a_j\ (1 \le j \le n)$. This formula is suitable for any prism part of a hyperrectangle cut by a hyperplane. The hyperplane part in the prism is its base and the coordinates $(a_1, a_2, \ldots, a_n)$ indicate the vertex not on the base.

Appendix C

Approximate integration of the logistic function

The computation of the sensitivity in chapter 5 is finally attributed to the calculation of the following multiple integral with the logistic function $f(X) = 1/(1+e^{-\sum_{i=1}^{n} x_i w_i - \sigma})$ as the integrand and the hyperrectangle Ω, $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$, as the integral region,

$$I = \int_{\Omega} \ldots \int f(X) d\Omega. \tag{C.1}$$

Since no primitive function can be found for the integrand, the Taylor series

$$1/(1+x) \approx \sum_{k=0}^{p} (-1)^k x^k \tag{C.2}$$

under the convergent condition $|x| < 1$ is employed to approximate the integrand. By substituting $e^{-\sum_{i=1}^{n} x_i w_i - \sigma}$ for x in (C.2), the logistic function could be partially expressed as

$$f(X) \approx f^p(X) = \sum_{k=0}^{p} (-1)^k e^{-k\sum_{i=1}^{n} x_i w_i - k\sigma}, \tag{C.3}$$

only if $|e^{-\sum_{j=1}^{n} x_j w_j - \sigma}|<1$, namely $\sum_{j=1}^{n} x_j w_j + \sigma > 0$, is satisfied. But how to deal with $f(X)$ under the condition $\sum_{j=1}^{n} x_j w_j + \sigma \le 0$? It is quite clear that, when X satisfying $\sum_{j=1}^{n} x_j w_j + \sigma < 0$, $f^p(X)$ will certainly be divergent. Fortunately the logistic function characteristic (1.6), i.e., $f(-x) = 1 - f(x)$, can be used to solve this problem. Under this circumstance, the logistic function could be further expressed as

$$f(X) \approx f^p(X) = \begin{cases} \sum_{k=0}^{p} (-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma}, & \text{if } \sum_{j=1}^{n} x_j w_j + \sigma > 0 \\ 1 - \sum_{k=0}^{p} (-1)^k e^{k\sum_{j=1}^{n} x_j w_j - k\sigma}, & \text{if } \sum_{j=1}^{n} x_j w_j + \sigma < 0. \end{cases} \tag{C.4}$$

As to X that satisfies $\sum_{j=1}^{n} x_j w_j + \sigma = 0$, it is noticed that $f^p(X)$ will oscillate between 0 and 1, that is $f^p(X) = 0$ when p is an odd number, say p_1, and $f^p(X) = 1$ when p is an even number, say p_2. But actually it has $f(X) = 1/2$ when X satisfies $\sum_{j=1}^{n} x_j w_j + \sigma = 0$. Thus it is necessary to amend the imprecision arising from X satisfying $\sum_{j=1}^{n} x_j w_j + \sigma = 0$ in Ω. The above analysis indicates that

the amendment could be achieved by calculating (C.4) twice with p_1 and p_2 each time, and then to average the two results. That is

$$f(X) \approx f^p(X) = (f^{p_1}(X) + f^{p_2}(X))/2, \qquad \text{if } \sum_{j=1}^{n} x_j w_j + \sigma = 0. \tag{C.5}$$

By combining (C.4) and (C.5), the logistic function could be totally expressed as

$$f(X) \approx f^p(X) = \begin{cases} \left(\sum_{k=0}^{p_1}(-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} + \sum_{k=0}^{p_2}(-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma}\right)/2, \text{ if } \sum_{j=1}^{n} x_j w_j + \sigma \geq 0 \\ \\ 1 - \left(\sum_{k=0}^{p_1}(-1)^k e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} + \sum_{k=0}^{p_2}(-1)^k e^{k\sum_{j=1}^{n} x_j w_j + k\sigma}\right)/2, \text{ if } \sum_{j=1}^{n} x_j w_j + \sigma \leq 0. \end{cases} \tag{C.6}$$

So (C.1) could be approximately transformed to the following form

$$I \approx \int_{\Omega} \cdots \int f^p(X) d\Omega. \tag{C.7}$$

Because of (C.5), the condition under $\sum_{j=1}^{n} x_j w_j + \sigma = 0$ is no longer a problem. Below only the satisfaction of the two convergent conditions of (C.4) in Ω is considered. From a geometric point of view, the hyperplane P, $\sum_{j=1}^{n} x_j w_j + \sigma = 0$, can be regarded as a dividing plane, which divides the n-dimensional space into three

parts, namely the points on P, the points below P, and the points above P. There is no doubt that the points on P satisfying $\sum_{j=1}^{n} x_j w_j + \sigma = 0$, and each of the other two parts holds either $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ or $\sum_{j=1}^{n} x_j w_j + \sigma < 0$. They are two opposite cases. If one holds $\sum_{j=1}^{n} x_j w_j + \sigma > 0$, the other must satisfy $\sum_{j=1}^{n} x_j w_j + \sigma < 0$, and vice versa. Now, the problem is how to find out whether P cuts Ω or not. If it does, how can the two parts for the calculation of (C.4) be separated?

A solution to this problem is proposed below, which is based on analytical geometry. The idea is to locate the intersection points of P and those parallel lines that are the extensions of the edges of Ω. Each line must have one and only one intersection point with P under the assumption $w_j \neq 0$ for $1 \leq j \leq n$. Then by comparing the coordinates on a given axis of those intersection points with the lower bound and the upper bound of Ω on that given axis, it is easy to identify the intersection situations between P and Ω. It is known that Ω has 2^{n-1} edges parallel to a given coordinate axis. Taking x_n-axis as an example, each edge parallel to x_n-axis can be determined by assigning $n-1$ coordinates (x_1, ..., x_{n-1}) with either $x_j = a_j$ or $x_j = b_j$. Thus the x_n-coordinate for a given line's intersection point can be calculated by

$$x_n = -(\sum_{j=1}^{n-1} x_j w_j + \sigma)/w_n \tag{C.8}$$

with its $n-1$ fixed coordinates. This will yield a set of 2^{n-1} coordinates, denoted as $E_n = \{e_{n,1}, e_{n,2}, \ldots, e_{n,2^{n-1}}\}$, for 2^{n-1} edges. It is worth to notice here that during the calculation of E_n an order is actually set to select the 2^{n-1} edges, which is represented by the second subscript of the elements of E_n. This means that the second subscript uniquely corresponds to an edge and thus its $n-1$ fixed coordinates, denoting as $(x_1, \ldots, x_{n-1})$, or $\{x_1, \ldots, x_{n-1}\}$, for $e_{n,i}$ $(1 \le i \le 2^{n-1})$. This property will be used later. Then through the selection $e_{n,l} = \max\{e_{n,1}, \ldots, e_{n,2^{n-1}}\}$ and $e_{n,s} = \min\{e_{n,1}, \ldots, e_{n,2^{n-1}}\}$, and the comparison among $e_{n,s}$, $e_{n,l}$, a_n, and b_n, it has the following possible cases and their corresponding formulas for the calculation of (C.7).

•**Case 1.** $e_{n,s} \ge b_n$ or $e_{n,l} \le a_n$:

In this case as illustrated by Figure C.1, P doesn't cut Ω. Hence in Ω only one convergent condition of (C.4) is satisfied. If Ω holds $\sum_{j=1}^{n} x_j w_j + \sigma > 0$, it has

$$I \approx \int_{\Omega} \ldots \int (\sum_{k=0}^{p} (-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma}) d\Omega$$

$$= \int_{a_1}^{b_1} \int_{a_2}^{b_2} \ldots \int_{a_n}^{b_n} \sum_{k=0}^{p} (-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n$$

$$= \sum_{k=0}^{p} (-1)^k \int_{a_1}^{b_1} \int_{a_2}^{b_2} \ldots \int_{a_n}^{b_n} e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n$$

$$= \sum_{k=0}^{p} (-1)^k I_1(-k, \sigma, A, B, W). \tag{C.9}$$

Otherwise, the formula for Ω satisfying $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ is

$$I \approx \sum_{k=1}^{p} (-1)^{k+1} I_1(k, \sigma, A, B, W). \tag{C.10}$$

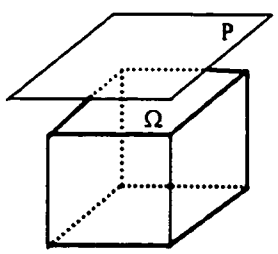

Figure C.1: P doesn't cut Ω.

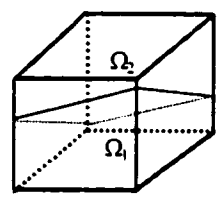

Figure C.2: P cuts the parallel sides of Ω.

•Case 2. $e_{n,x} \geq a_n$ and $e_{n,l} \leq b_n$:

In this case as illustrated by Figure C.2, P cuts Ω by all the edges that parallel to x_n-axis, and divides Ω into two parts, Ω_1 and Ω_2. Supposing $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ in Ω_1 and $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ in Ω_2, it follows that

$$I \approx \int_{\Omega_1} \cdots \int (1 - \sum_{k=0}^{p} (-1)^k e^{k\sum_{j=1}^{n} x_j w_j + k\sigma}) dx_1 dx_2 \ldots dx_n + \int_{\Omega_2} \cdots \int \sum_{k=0}^{p} (-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n$$

$$=\sum_{k=1}^{p}(-1)^{k+1}\int_{\Omega_1}\cdots\int e^{k\sum_{j=1}^{n}x_j w_j+k\sigma}dx_1dx_2\ldots dx_n+\sum_{k=0}^{p}(-1)^{k}\int_{\Omega_2}\cdots\int e^{-k\sum_{j=1}^{n}x_j w_j-k\sigma}dx_1dx_2\ldots dx_n$$

$$=\sum_{k=1}^{p}(-1)^{k+1}\,|\,I_2(k,\sigma,A,B-\{b_n\},W)\,|+\sum_{k=0}^{p}(-1)^{k}\,|\,I_2(-k,\sigma,B,A-\{a_n\},W)\,|, \qquad \text{(C.11)}$$

where

$$\int_{\Omega_1}\cdots\int e^{k\sum_{j=1}^{n}x_j w_j+k\sigma}dx_1dx_2\ldots dx_n=\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_{n-1}}^{b_{n-1}}\int_{a_n}^{-(\sum_{j=1}^{n-1}x_j w_j+\sigma)/w_n}e^{k\sum_{j=1}^{n}x_j w_j+k\sigma}dx_1dx_2\ldots dx_n\ , \qquad \text{(C.12)}$$

and

$$\int_{\Omega_2}\cdots\int\sum_{k=0}^{p}(-1)^{k}e^{-k\sum_{j=1}^{n}x_j w_j-k\sigma}dx_1dx_2\ldots dx_n=\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_{n-1}}^{b_{n-1}}\int_{-(\sum_{j=1}^{n-1}x_j w_j+\sigma)/w_n}^{b_n}e^{-k\sum_{j=1}^{n}x_j w_j-k\sigma}dx_1dx_2\ldots dx_n\ . \qquad \text{(C.13)}$$

Similarly, the formula for $\sum_{j=1}^{n}x_j w_j+\sigma>0$ in Ω_1 and $\sum_{j=1}^{n}x_j w_j+\sigma<0$ in Ω_2 is

$$I\approx\sum_{k=0}^{p}(-1)^{k}\,|\,I_2(-k,\sigma,A,B-\{b_n\},W)\,|+\sum_{k=1}^{p}(-1)^{k+1}\,|\,I_2(k,\sigma,B,A-\{a_n\},W)\,|. \qquad \text{(C.14)}$$

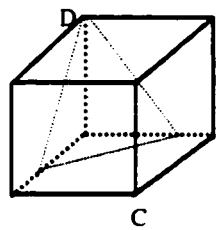

Figure C.3: P cuts a prism from Ω.

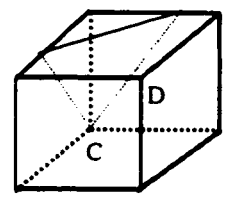

Figure C.4: P cuts a prism from Ω.

•**Case 3.** $e_{n,s} = a_n$ and $e_{n,l} > b_n$ or $e_{n,s} < a_n$ and $e_{n,l} = b_n$:

In this case as illustrated by Figure C.3 or Figure C.4, P cuts a prism from Ω and they have a special intersection point that is a vertex of Ω. For general consideration, it is supposed a point C($c_1, c_2, \ldots, c_n$) that is the lower vertex of Ω on the edge corresponding to $e_{n,s}$ and a point D($d_1, d_2, \ldots, d_n$) that is the upper vertex of Ω on the edge corresponding to $e_{n,l}$, then either C or D is the special intersection point, and it must be true that $c_n = a_n$ and $d_n = b_n$. The other $n-1$ coordinates of point C and point D can be determined by $e_{n,s}$, namely $(x_1, \ldots, x_{n-1})_s$, and $e_{n,l}$, namely $(x_1, \ldots, x_{n-1})_l$, as mentioned above. The coordinates of these two points are very important in setting the boundary for the integral in the calculation of (C.7). For the simplicity of expression, they are also regarded as two sets, i.e. C=$\{c_1, c_2, \ldots, c_n\}$ and D=$\{d_1, d_2, \ldots, d_n\}$. Referring to the discussion in appendix B, three regions are used, Ω_1, Ω_2 and Ω_3, to replace Ω as Figure C.5, Figure C.6, and Figure C.7 illustrate. It is obvious that Ω_1 is suitable for (B.3) and Ω_2 for (B.2). But Ω_3, as Figure C.8

illustrates, needs to be extended with $2^{n-1}-2$ small prisms to be suitable for (B.3). In fact, this situation may appear to Ω_1 and not to Ω_3 too. In order to give a general solution, (B.3) is first used to calculate the integral by assuming Ω_1 and Ω_3 to be prisms, and then the results are adjusted by subtracting the integral in each small prism under the condition that the integrand expansion in that small prism is convergent. The integrand expansion in each of the small prisms will diverge if Ω_1 or Ω_3 actually does not need such a prism for the extension. Thus according to whether $c_n = e_{n,s}$ or $d_n = e_{n,J}$, and the satisfaction of the convergent condition of (C.4) in Ω_1, Ω_2 and Ω_3, it has the following four groups of formulas for different conditions. Below only the derivation for one group is given and the results for the other three groups are listed.

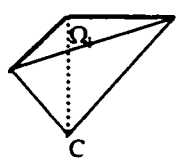

Figure C.5: The region of Ω_1.

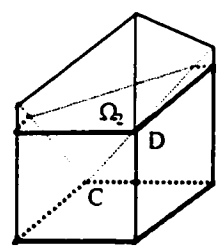

Figure C.6: The region of Ω_2.

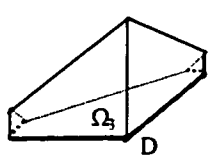

Figure C.7: The region of Ω_3.

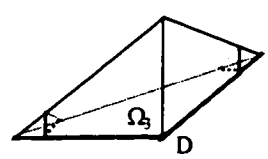

Figure C.8: The extended region of Ω_3.

1) $c_n = e_{n,s}$, $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ in Ω_1, and $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ in Ω_2 and Ω_3 :

$$
\begin{aligned}
I \approx & \int_{\Omega_1} \cdots \int \sum_{k=0}^{p} (-1)^k e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n \\
& + \int_{\Omega_2} \cdots \int (1 - \sum_{k=0}^{p} (-1)^k e^{k\sum_{j=1}^{n} x_j w_j + k\sigma}) dx_1 dx_2 \ldots dx_n \\
& - \int_{\Omega_3} \cdots \int (1 - \sum_{k=0}^{p} (-1)^k e^{k\sum_{j=1}^{n} x_j w_j + k\sigma}) dx_1 dx_2 \ldots dx_n \\
= & \sum_{k=0}^{p} (-1)^k \int_{\Omega_1} \cdots \int e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n \\
& + \sum_{k=1}^{p} (-1)^{k+1} \int_{\Omega_2} \cdots \int e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 \ldots dx_n \\
& - \sum_{k=1}^{p} (-1)^{k+1} \int_{\Omega_3} \cdots \int e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 \ldots dx_n \\
= & \sum_{k=0}^{p} (-1)^k (|I_3(-k, \sigma, (C - \{c_n\}) \cup \{d_n\}, W)| - \sum_{h=1 \& h \neq s \& h \neq l}^{2^{n-1}} \delta_h) \\
& + \sum_{k=1}^{p} (-1)^{k+1} |I_2(k, \sigma, C, D - \{d_n\}, W)| \\
& - \sum_{k=1}^{p} (-1)^{k+1} (|I_3(k, \sigma, D, W)| - \sum_{h=1 \& h \neq s \& h \neq l}^{2^{n-1}} \varepsilon_h),
\end{aligned}
\tag{C.15}
$$

where

$$\int_{\Omega_1} \cdots \int e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 \ldots dx_n$$

$$=\left|\int_{c_1}^{-(\sum_{j=2}^{n-1} c_j w_j + d_n w_n + \sigma)/w_1} \int_{c_2}^{-(x_1 w_1 + \sum_{j=1}^{n-1} c_j w_j + d_n w_n + \sigma)/w_2} \cdots \int_{c_{n-1}}^{-(\sum_{j=1}^{n-2} x_j w_j + d_n w_n + \sigma)/w_{n-1}} \int_{-(\sum_{j=1}^{n-1} x_j w_j + \sigma)/w_n}^{d_n} e^{-k\sum_{j=1}^{n} x_j w_j - k\sigma} dx_1 dx_2 .. dx_n \right|$$

$$- \sum_{h=1 \& h \neq s \& h \neq l}^{2^{n-1}} \delta_h \quad , \tag{C.16}$$

$$\int_{\Omega_2} \cdots \int e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 \ldots dx_n$$

$$=\left|\sum_{k=1}^{p} (-1)^{k+1} \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_{n-1}}^{b_{n-1}} \int_{a_n}^{-(\sum_{j=1}^{n-1} x_j w_j + \sigma)/w_n} e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 \ldots dx_n \right|, \tag{C.17}$$

and

$$\int_{\Omega_1} \cdots \int e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 \ldots dx_n$$

$$=\left|\int_{-(\sum_{j=2}^{n} d_j w_j + \sigma)/w_1}^{d_1} \int_{-(x_1 w_1 + \sum_{j=1}^{n} d_j w_j + \sigma)/w_2}^{d_2} \cdots \int_{-(\sum_{j=1}^{n-2} x_j w_j + d_n w_n + \sigma)/w_{n-1}}^{d_{n-1}} \int_{d_n}^{-(\sum_{j=1}^{n-1} x_j w_j + \sigma)/w_n} e^{k\sum_{j=1}^{n} x_j w_j + k\sigma} dx_1 dx_2 .. dx_n \right|$$

$$- \sum_{h=1 \& h \neq s \& h \neq l}^{2^{n-1}} \varepsilon_h \quad . \tag{C.18}$$

Further,

$$\delta_h = \begin{cases} \left| I_3(-k, \sigma, \{x_1, \ldots, x_{n-1}\}_h \cup \{d_n\}, W) \right|, \\ \qquad \text{if } \sum_{k=0}^{p} (-1)^k \left| I_3(-k, \sigma, \{x_1, \ldots, x_{n-1}\}_h \cup \{d_n\}, W) \right| < \sum_{k=0}^{p} (-1)^k \left| I_3(-k, \sigma, (C - \{c_n\}) \cup \{d_n\}, W) \right| \\ 0, \quad \text{otherwise} \end{cases} \tag{C.19}$$

and

$$\varepsilon_h = \begin{cases} \left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right|, \\ \qquad \text{if } \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right| < \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,D,W)\right| \\ 0, \quad \text{otherwise.} \end{cases} \tag{C.20}$$

In the similar way, the other three groups could be derived as:

2) $c_n = e_{n,s}$, $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ in Ω_1, $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ in Ω_2 and Ω_3:

$$\begin{aligned} I \approx & \sum_{k=1}^{p}(-1)^{k+1}\Big(\left|I_3(k,\sigma,(C-\{c_n\}) \cup \{d_n\},W)\right| - \sum_{h=1\&h\neq s\&h\neq l}^{2^{n-1}} \delta_h \Big) \\ & + \sum_{k=0}^{p}(-1)^{k} I_2(-k,\sigma,C,D-\{d_n\},W) \\ & - \sum_{k=0}^{p}(-1)^{k}\Big(\left|I_3(-k,\sigma,D,W)\right| - \sum_{h=1\&h\neq s\&h\neq l}^{2^{n-1}} \varepsilon_h \Big), \end{aligned} \tag{C.21}$$

where

$$\delta_h = \begin{cases} \left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right|, \\ \quad \text{if } \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right| < \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,(C-\{c_n\})\cup\{d_n\},W)\right| \\ 0, \text{ otherwise} \end{cases} \tag{C.22}$$

and

$$\varepsilon_h = \begin{cases} \left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right|, \\ \quad \text{if } \sum_{k=0}^{p}(-1)^{k}\left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{d_n\},W)\right| < \sum_{k=0}^{p}(-1)^{k}\left|I_3(-k,\sigma,D,W)\right| \\ 0, \quad \text{otherwise.} \end{cases} \tag{C.23}$$

3) $d_n = e_{n,l}$, $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ in Ω_1, $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ in Ω_2 and Ω_3:

$$\begin{aligned} I \approx & \sum_{k=0}^{p}(-1)^{k}\Big(\left|I_3(-k,\sigma,(D-\{d_n\})\cup\{c_n\},W)\right| - \sum_{h=1\&h\neq s\&h\neq l}^{2^{n-1}}\delta_h\Big) \\ & + \sum_{k=1}^{p}(-1)^{k+1}\left|I_2(k,\sigma,D,C-\{c_n\},W)\right| \\ & - \sum_{k=1}^{p}(-1)^{k+1}\Big(\left|I_3(k,\sigma,C,W)\right| - \sum_{h=1\&h\neq s\&h\neq l}^{2^{n-1}}\varepsilon_h\Big), \end{aligned} \tag{C.24}$$

where

$$\delta_h = \begin{cases} \left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right|, \\ \quad \text{if } \sum_{k=0}^{p}(-1)^k\left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right| < \sum_{k=0}^{p}(-1)^k\left|I_3(-k,\sigma,(D-\{d_n\})\cup\{c_n\},W)\right| \\ 0, \text{ otherwise} \end{cases} \tag{C.25}$$

and

$$\varepsilon_h = \begin{cases} \left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right|, \\ \quad \text{if } \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right| < \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,C,W)\right| \\ 0, \quad \text{otherwise.} \end{cases} \tag{C.26}$$

4) $d_n = e_{n,l}$, $\sum_{j=1}^{n} x_j w_j + \sigma < 0$ in Ω_1, and $\sum_{j=1}^{n} x_j w_j + \sigma > 0$ in Ω_2 and Ω_3:

$$\begin{aligned} I \approx & \sum_{k=1}^{p}(-1)^{k+1}\Big(\left|I_3(k,\sigma,(D-\{d_n\})\cup\{c_n\},W)\right| - \sum_{h=1\&h\neq r\&h\neq l}^{2^{n-1}}\delta_h\Big) \\ & + \sum_{k=0}^{p}(-1)^k\left|I_2(-k,\sigma,D,C-\{c_n\},W)\right| \\ & - \sum_{k=0}^{p}(-1)^k\Big(\left|I_3(-k,\sigma,C,W)\right| - \sum_{h=1\&h\neq r\&h\neq l}^{2^{n-1}}\varepsilon_h\Big), \end{aligned} \tag{C.27}$$

where

$$\delta_h = \begin{cases} \left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right|, \\ \qquad \text{if } \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right| < \sum_{k=1}^{p}(-1)^{k+1}\left|I_3(k,\sigma,(D-\{d_n\})\cup\{c_n\},W)\right| \\ 0, \quad \text{otherwise} \end{cases} \tag{C.28}$$

and

$$\varepsilon_h = \begin{cases} \left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right|, \\ \qquad \text{if } \sum_{k=0}^{p}(-1)^{k}\left|I_3(-k,\sigma,\{x_1,\ldots,x_{n-1}\}_h \cup \{c_n\},W)\right| < \sum_{k=0}^{p}(-1)^{k}\left|I_3(-k,\sigma,C,W)\right| \\ 0, \quad \text{otherwise.} \end{cases} \tag{C.29}$$

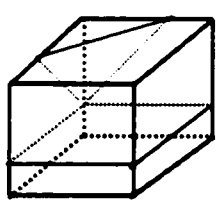

Figure C.9: Ω is divided into two smaller rectangles.

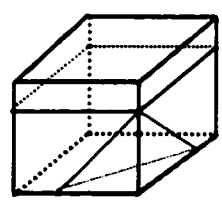

Figure C.10: Ω is divided into two smaller rectangles.

•**Case 4.** $e_{n,s} > a_n$ and $e_{n,l} > b_n$ or $e_{n,s} < a_n$ and $e_{n,l} < b_n$:

In this case P cuts Ω as Figure C.9 or Figure C.10 illustrate. It can be attributed to case 1 and case 3 by cutting Ω with hyperplane $x_n = e_{n,s}$ or $x_n = e_{n,l}$ into two smaller hyperrectangles.

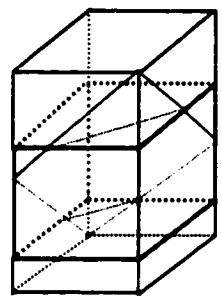

Figure C.11: Ω is extended to a bigger rectangle with two extra rectangles.

•**Case 5.** $e_{n,s} < a_n$ and $e_{n,l} > b_n$:

This case, as Figure C.11 illustrates, can be attributed to case 2 and case 3 by considering a bigger hyperrectangle first as case 2 does and then dealing with two extra hyperrectangles on the top and bottom as case 3 does.

Bibliography

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics 5, pp. 115-133, 1943.

[2] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain", Psychological Review, Vol. 65, pp. 386-408, 1958.

[3] M. Minsky amd S. Papert, *Perceptron: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.

[4] S. Haykin, *Neural Networks — A Comprehensive Foundation*, Prentice Hall International, 1994.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1, Chapter 8, MIT Press, Cambridge, MA, 1986.

[6] K. M. Hornik, M. Stinchcombea and H. white, "Miltilayer feedforward networks are universal approximators", Neural Networks vol. 2, pp. 359-366, 1989.

[7] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks", Neural Networks, vol. 2, pp. 183-192, 1989.

[8] B. Widrow and M. E. Hoff, "Adaptive switching circuits", IRE Western Electric Show and Convention Record, pt. 4, pp. 96-104, Aug. 1960.

[9] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline and backpropagation," Proc. of IEEE, vol. 78, no. 9, pp. 1415-1442, Sept. 1990.

[10] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", Biological Cybernetics, vol. 36, no. 4, pp. 193-202, Apr. 1980.

[11] G. W. Davis, "Sensitivity analysis of hopfield neural net", Proc. of IEEE Int. First Conf. on Neural Network, vol. 3, pp. 325-328, San Diego, CA, 1987.

[12] G. W. Davis, "Sensitivity analysis in neural net solutions", IEEE Trans. on System, Man, and Cybernetics, vol. 19, no. 5, pp. 1078-1082, Sept./Oct. 1989.

[13] L. C. Jiao and Z. Bao, "Neural networks via nonlinear sensitivity approach", Proc. of IEEE Int. Symp. on Circuits and Systems, vol. 5, pp. 2542-2545, 1991.

[14] A. N. Michel and K. Wang, "Sensitivity and perturbation analysis of artificial feedback neural networks", Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics, vol. 2, pp. 1332, 1992.

[15] M. Stevenson, Sensitivity of Madalines to Weight Errors, PhD dissertation, Stanford University, Stanford, CA, Dec. 1990.

[16] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weight errors", IEEE Trans. on Neural Networks, vol. 1, no. 1, pp. 71-80, Mar. 1990.

[17] C. Alippi, V. Piuri, and M. G. Sami, "The issues of error sensitivity in neural networks", Proc. of Int. Conf. on Massively Parallel Computing Systems, pp. 177-189, Ischia , Italy, May 1994.

[18] C. Alippi, V. Piuri, and M. G. Sami, "Sensitivity to errors in artificial neural networks: a behavioral approach", Proc. of IEEE Int. Symp. on Circuits and Systems, pp. 459-462, London, June 1994.

[19] C. Alippi, V. Piuri, and M. G. Sami, "Sensitivity to errors in artificial neural networks: a behavioral approach", IEEE Trans. on Circuits and Systems-I: Fundamental Theory and Applications, vol. 42, no. 6, pp. 358-361, June 1995.

[20] S. W. Piché, "Robustness of feedforward neural networks", Proc. of Int. Joint Conf. on Neural Networks, vol. 2, pp. 346-351, Baltimore, MD, June 1992.

[21] S. W. Piché, Selection of Weight Accuracies for Neural Networks, PhD dissertation, Stanford University, Stanford, CA, May 1992.

[22] S. W. Piché, "The effects of weight errors in neural networks", Proc. of WCNN93, vol. 4, pp. 559-565, Portland, OR, July 1993.

[23] S. W. Piché, "The selection of weight accuracies for Madalines", IEEE Trans. on Neural Networks, vol. 6, no. 2, pp. 432-445, Mar. 1995.

[24] L. Yang, D. Hu, Y. Luo, and X. Zhang, "Robustness analysis of feedforward neural networks composed of threshold neurons", Proc. of IEEE Int. Conf. on Intelligent Processing Systems, vol.1, pp.502-506, Oct. 1997.

[25] N. S. Orzechowski, S. R. T. Kumara, and C. R. Das, Performance of multilayer neural networks in binary-to-binary mappings under weight errors", Proc. of IEEE Int. Conf. on Neural Networks, pp. 1684-1689, San Francisco, CA, May 1993.

[26] S. H. Oh and Y. Lee, "Sensitivity of trained neural networks with threshold function", Proc. of IEEE Int. Conf. on Neural Networks, vol. 2, pp. 986-991, June 1994.

[27] S. H. Oh and Y. Lee, "Sensitivity analysis of a single hidden-layer neural networks with threshold function", IEEE Trans. on Neural Networks, vol. 6, no. 4, pp. 1005-1007, July 1995.

[28] A. Y. Cheng, Sensitivity Analysis and Performance Tuning of Neocognitron, M.Phil. thesis, The Hong Kong Polytechnic University, Hong Kong, China, Oct. 1996.

[29] A. Y. Cheng and D. S. Yeung, "Sensitivity analysis of Neocognitron", IEEE Trans. on System, Man, and Cybernetics — Part C: Applications and Reviews, vol.29, no. 2, pp. 238-249, May 1999.

[30] S. Hashem, "Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions", Proc. of Int. Joint Conf. on Neural Networks, vol. 1, pp. 419-424, Baltimore, 1992.

[31] L. Fu and T. Chen, "Sensitivity analysis for input vector in multilayer feedforward neural networks", Proc. of IEEE Int. Conf. on Neural Networks, vol. 1, pp. 215-218, San Francisco, CA, 1993.

[32] M. Y. Chow and S. O. Yee, "A measure of relative robustness for feedforward neural networks subject to small input perturbations", International Journal of Neural Systems, vol. 3, no. 3, pp. 291-299, 1992.

[33] J. Y. Choi and C. H. Choi, "Sensitivity analysis of multilayer perceptron with differentiable activation functions", IEEE Trans. on Neural Networks, vol. 3, no. 1, pp. 101-107, Jan. 1992.

[34] Y. Xie and M. A. Jabri, "Analysis of the effects of quantization in multi-layer neural networks using a statistical model", Proc. of Int. Joint Conf. on Neural Networks, vol. 2, no. 2, pp. 503-508, 1991.

[35] Y. Xie and M. A. Jabri, "Analysis of effects of quantization in multilayer neural networks using statistical model", Electronics Letters, vol. 27, pp. 1196-1198, 1991.

[36] Y. Xie and M. A. Jabri, "Analysis of the effects of quantization in multilayer neural networks using a statistical model", IEEE Trans. on Neural Networks, vol. 3, no. 2, pp. 334-338, Mar. 1992.

[37] G. Düdar and K. Rose, "The effects of quantization on multilayer neural networks", IEEE Trans. on Neural Networks, vol. 6, no. 6, pp. 1446-1451, Nov. 1995.

[38] A. A. Minai and R. D. Williams, "Perturbation response in feed-forward neural networks" Proc. of Int. Joint Conf. on Neural Networks, vol. 3, pp. 857-862, 1992.

[39] A. A. Minai and R. D. Williams, "Perturbation response in feedforward networks", Neural Networks, vol. 7, no. 5, pp. 783-796, 1994,

[40] D. S. Yeung, X. Sun, and **X. Zeng**, "Sensitivity analysis of Multilayer Perceptron", Proc. of IJCAI-01, vol. 2, pp. 809-814, Seattle, WA, Aug. 2001.

[41] D. S. Yeung and X. Sun, "Using function approximation to analyze the sensitivity of the MLP with antisymmetric squashing activation function", IEEE Trans. on Neural Networks, vol. 13, no. 1, pp. 34-44, Jan. 2002.

[42] D. Lovell, P. Bartlett, and T. Downs, "Error and variance bounds on sigmoidal neurons with weigh and input errors", Electronics Letters, vol. 28, no 8, pp. 760-762, Apr. 1992.

[43] S. Oh, R. J. MarksII, "Noise sensitivity of projection neural networks", Proc. of IEEE Int. Symp. on Circuits and systems, vol. 3, pp. 2201-2204, 1989.

[44] D. Anguita, S. Ridella, S. Rovetta, and R. Zunino, "Limiting the effects of weight errors in feed forward networks using interval arithmetic", Proc. of IEEE Int. Conf. on Neural Networks, vol. 1, pp. 414-417, 1996.

[45] D. Anguita, S. Ridella, and S. Rovetta, "Worst case analysis of weight inaccuracy effects in Multilayer Perceptrons", IEEE Trans. on Neural Networks, vol. 10, no. 2, pp. 415-418, Mar. 1999.

[46] L. Ozyilmaz and T. Yildirim, "Sensitivity analysis for conic section function neural networks", Proc. of Int. Joint Conf. on Neural Networks, vol. 4, pp. 244-249, 2000.

[47] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks", IEEE Trans. on Neural Networks, vol. 1, no. 2, pp. 239-242, June 1990.

[48] J. M. Zurada, A. Malinowski and I. Cloete, "Sensitivity analysis for pruning of training data in feedforward neural networks", Proc. of First Australian and New Zeland Conf. on Intelligent Information Systems, pp. 288-292, Perth, Dec. 1993.

[49] T. H. Goh, "Semantic extraction using neural network modeling and sensitivity analysis", Proc. of IEEE Int. Joint Conf. on Neural Networks, vol. 1, pp. 1031-1034, Nagoya, 1993.

[50] J. M. Zurada, A. Malinowski and I. Cloete, "Sensitivity analysis for minimization of input data dimension for feedforward neural network", Proc. of IEEE Int. Symp. on Circuits and Systems, pp. 447-450, London, 1994.

[51] A. P. Engelbrecht, I. Cloete and J. M. Zurada, "Determining the significance of input parameters using sensitivity analysis", Lecture Notes in Computer Science, vol. 930, pp. 382-388, 1995.

[52] H. L. Viktor, A. P. Engelbrecht and I. Cloete, "Reduction of symbolic rules from artificial neural networks using sensitivity analysis", Proc. of IEEE Int. Conf. on Neural Networks, vol. 4, pp. 1788-1793, 1995.

[53] T. Czernichow, "Architecture selection through statistical sensitivity analysis", Proc. of Int. Conf. on Artificial Neural Networks, pp. 179-184, Germany, 1996.

[54] A. P. Engelbrecht, I. Cloete, "A sensitivity analysis algorithm for pruning feedforward neural networks", Proc. of IEEE Int. Conf. on Neural Networks, vol. 2, pp. 1274-1278, Washington, 1996.

[55] J. M. Zurada, A. Malinowski, S. Usui, "Perturbation method for deleting redundant inputs of Perceptron networks", Neurocomputing, vol. 14, pp. 177-193, 1997.

[56] A. P. Engelbrecht, I. Cloete, "Selective learning using sensitivity analysis", Proc. of IEEE Int. Joint Conf. on Neural Networks, vol. 2, pp. 1150-1155, Alaska, 1998.

[57] A. P. Engelbrecht, I. Cloete, "Feature extraction from feedforward neural networks using sensitivity analysis", Proc. of Int. Conf. on Systems, Signals, Control, Computers, vol. 2, pp. 221-225, Durban, South Africa, Sept. 1998.

[58] A. P. Engelbrecht, L. Fletcher and I. Cloete, "Variance analysis of sensitivity information for pruning multilayer feedforward neural networks", Proc. of Int. Joint Conf. on Neural Networks, vol. 3, pp. 1829-1833, Washington DC, 1999.

[59] A. P. Engelbrecht, "Sensitivity analysis for decision boundaries", Neural Processing Letters, pp. 253-266, 1999.

[60] A. P. Engelbrecht and H. L. Viktor, "Rule improvement through decision boundary detection using sensitivity analysis", Proc. of Int. Workshop on Artificial Neural Networks, Alicante, Spain, June 1999.

[61] A. P. Engelbrecht and I. Cloete, "Incremental learning using sensitivity analysis", Proc. of Int. Joint Conf. on Neural Networks, vol. 2, pp. 1350-1355, Washington DC, 1999.

[62] A. P. Engelbrecht and R. Brits, "A clustering approach to incremental learning for feedforward neural networks", Proc. of Int. Joint Conf. on Neural Networks, vol. 3, pp. 2019-2024, Washington DC, 2001.

[63] A. P. Engelbrecht, "A new pruning heuristic based on variance analysis of sensitivity information", IEEE Trans. on Neural Networks, vol. 12, no. 6, pp. 1386-1399, Nov. 2001.

[64] F. Frost and V. Karri, "Determining the influence of input parameters on BP neural network output error using sensitivity analysis", Proc. of Third Int. Conf. on Computational Intelligence and Multimedia Applications, pp. 45-49, 1999.

[65] C. T. Chiu, K. Mehrotra,C. K. Mohan, and S. Ranka, "Robustness of feedforward neural networks", Proc. of IEEE Int. Conf. on Neural Networks, vol. 2, pp. 783-788, 1993.

[66] Y. Lee and S. H. Oh, "Input noise immunity of Multiplayer Perceptrons", ETRI Journal, vol. 16, no. 1, pp. 35-43, 1994.

[67] C. Alippi, "Feedforward neural networks with improved insensitivity abilities", Proc. of IEEE Int. Symp. on Circuits and Systems, vol. 5, pp. 359 –362, 1999.

[68] J.L. Bernier, J. Ortega, and A. Prieto, "A modified backpropagation algorithm to tolerate weight errors", Lecture Notes in Computer Science, vol. 1240, pp. 763-771, Springer, Berlin, 1997.

[69] J.L. Bernier, J. Ortega, E. Ros, I. Rojas and A. Prieto, "A new measurement of noise immunity and generalization ability for MLPs", International Journal of Neural Systems, vol. 9, no. 6, pp. 511-522, 1999.

[70] J.L. Bernier, J. Ortega, I. Rojas, E. Ros, and A. Prieto, "An accuracy measure for Multilayer Perceptron tolerance to weight deviations", Neural Processing Letters, vol. 10, no. 2, pp. 121-130, 1999.

[71] J.L. Bernier, J. Ortega, E. Ros, I. Rojas and A. Prieto, "A quantitive study of fault tolerance, noise immunity and generalization ability of MLPs", Neural Computation, vol. 12, pp. 2941-2964, 2000.

[72] J.L. Bernier, J. Ortega, I. Rojas and A. Prieto, "Improving the tolerance of Multilayer Perceptrons by minimizing the statistical sensitivity to weight deviations", Neurocomputing, vol. 31, no.1-4, pp. 87-103, 2000.

[73] J. Castellanos, A. Pazos, J. Riso and L. J. Zafra, "Sensitivity analysis on neural networks for meteorological variable forecasting", Proc. of IEEE Workshop on Neural Networks for Signal Processing, pp. 587-595, 1994.

[74] Z. Guo and R. E. Uhrig, "Sensitivity analysis and applications to nuclear power plant", Proc. of Int. Joint Conf. on Neural Networks, vol. 2, pp. 453-458, 1992.

[75] D. F. Cook and R. E. Shannon, "A sensitivity analysis of a Back-Propagation neural network for manufacturing process parameters", Journal of Intelligent Manufacturing, pp. 155-164, 1991.

[76] A. H. Rambhia, R. Glenny and J. N. Hwang, "Critical input data channels selection for progressive work exercise test by neural network sensitivity analysis", Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, vol. 2, pp. 1097-1100, 1999.

[77] A. G. Bahbah and A. A. Girgis, "Input feature selection for real-time transient stability assessment for artificial neural network (ANN) using ANN sensitivity analysis", Proc. of IEEE Int. Conf. on Power Industry Computer Applications, pp. 295-300, 1999.

[78] **X. Zeng**, D. S. Yeung and X. Sun, "Sensitivity analysis of Multilayer Perceptron to input perturbation", Proc. of IEEE Int. Conf. on System, Man, and Cybernetics, vol. 4, pp. 2509-2514, Nashville, TN, Oct. 2000.

[79] **X. Zeng** and D. S. Yeung, "Sensitivity analysis of Multilayer Perceptron to input and weight perturbations", IEEE Trans. on Neural Networks, vol. 12, no. 6, pp. 1358-1366, Nov. 2001.

[80] **X. Zeng** and D. S. Yeung, "A quantified sensitivity measure for Multilayer Perceptron to input perturbation", submitted to Neural Computation (under minor revision).

[81] **X. Zeng** and D. S. Yeung, "Selection of weights for a robust Multilayer Perceptron using sensitivity measure", to be submitted.