UNIVERSITI PUTRA MALAYSIA

CASE STUDY: AN EFFECT OF NOISE IN CHARACTER RECOGNITION SYSTEM USING NEURAL NETWORK

ESMAWATY MOHAMAD

CASE STUDY: AN EFFECT OF NOISE IN CHARACTER RECOGNITION SYSTEM USING NEURAL NETWORK

BY

ESMAWATY MOHAMAD

Thesis Submitted in Fulfilment of the Requirement for the Degree of Master of Science in the Faculty of Computer Science and Information Technology

Universiti Putra Malaysia
May 2003

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations, which have been duly acknowledge. I also declare that it is has not been previously or concurrently submitted for any other degree at UPM or any other institutions.

(Esmawaty Mohamad)

Date : 12 April 2003

Abstract of thesis presented to the Senate of University Putra Malaysia in fulfillment of the requirements for the Bachelor of Computer Science

CASE STUDY : AN EFFECT OF NOISE IN CHARACTER RECOGNITION SYSTEM USING NEURAL NETWORK

By

ESMAWATY MOHAMAD

April 2003

Chairman : Puan The Noranis Mohd Aris

Faculty : Faculty of Computer Science and Information Technology

There has been resurgence of interest in artificial neural networks over the past few years, as a researchers from diverse backgrounds have produced a firms theoretical foundation and demonstrated numerous applications of this rich field of study. Neural networks are useful tools for solving many type of problems. These problems may be characterized as mapping(including pattern association and pattern classification), clustering and constrained optimization.

There has been great deal of work on enhancing neural network performance. Two important parameter are convergence and generalization. Convergence is the amount of time, measured in CPU operations or training epochs, required to find an acceptable

solution for training. Generalization measures the ability to correctly classify new unseen data.

This project studies the generalization ability of trained network to classify noisy data. The aim of this project is to develop a network that is able to recognize various inputs through a series of simulation using Neural Network simulator called MATLAB. The effect of the created network with noise are seen. This projects uses the most popular training method in character recognition problem, namely backpropagation algorithm. The theoretical foundation of this algorithm will be studied and summarized. Simulation experiment results on training and testing data will be recorded and discussed.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk Bacelor Sains Komputer

KAJIAN KES : KESAN BISING KEATAS SISTEM PENEGECAMAN PERKATAAN MENGGUNAKAN KAEDAH NEURAL NETWOK

Oleh

ESMAWATY MOHAMAD

April 2003

Pengerusi : The Noranis Mohd Aris

Fakulti : Fakulti Sains Komputer dan Teknologi Maklumat

Kajian mengenai artificial neural network telah menajadi popular sejak beberapa tahun kebelakangan ini semenjak penemuan beberapa teori dan aplikasi dari pelbagai bidang oleh para penyelidik. Beberapa bidang boleh dikategorikan sebagai pemetaan (termasuk gabungan paten dan klasifikasi paten), clustering dan constrained optimization.

Terdapat banyak penyelidikan dijalankan untuk menambahkan keupayaan neural netwok. Dua parameter penting ialah convergence dan generalization. Convergence adalah berkenaan masa, diukur dari operasi CPU atau bilangan interaksi yang diperlukan untuk melatih data. Generalization pula diukur dari kebolehan neural netwok untuk mengklasifikasikan data yang diuji.

Projek ini mengkaji kebolehan generalization keatas netwok yang telah dilatih. Tujuan projek ini adalah untuk menghasilkan netwok yang boleh mengecam pelbagai input melalui beberapa siri simulasi menggunakan pengsimulasi neural network yang dipanggil MATLAB. Kesan bunyi bising keatas netwok yang dihasilkan dilihat. Ia menggunakan kaedah yang popular digunakan untuk pengecaman perkataan iaitu algoritma backpropagation. Teori algoritma ini dikaji dan disimpulkan. Keputusan keatas latihan dan data yang diuji di rekod dan dibincangkan.

ACKNOWLEDGEMENTS

It is with great pleasure for me to take this opportunity to thank Puan The Noranis Binti Mohd Aris for her invaluable guidance and supervision in making this project so beneficial for me.

Besides that, I would like to Zakiah Ayob and Rajina M Raj who being my friends, for their suggestions during our many discussion.

Much appreciation also goes to my husband, Capt Wan Ahmad Rafidi bin Haji wan Daud who have indirectly helped in making this project possible.

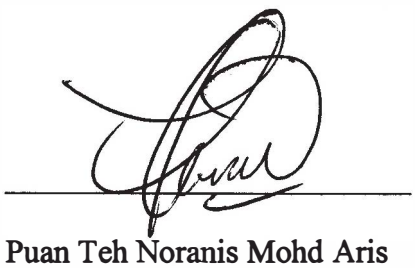
Lastly, I would like to thank my family and others who had given me the necessary supports during the writing of this project.

Esmawaty Mohamad

University Putra Malaysia

APROVAL SHEETS

This thesis submitted to the faculty of computer science and information technology of University Putra Malaysia has been accepted as fulfillment of the requirements for the degree of master of computer science.



Puan Teh Noranis Mohd Aris
Lecturer
Faculty of Computer Science and Information Technology
University Putra Malaysia

TABLE OF CONTENT

CHAPTER III

CHAPTER IV

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background

Neural networks are computing architectures inspired by biological nervous systems. They are useful in applications where formal analysis would be extremely difficult or impossible, such as pattern recognition and nonlinear system identification and control. The behavior of a neural network is defined by its architecture, the way its individual computing elements are connected, and the strength of those connections, or weights. The weights are adjusted by training the network according to a specified learning rule until it performs with the desired error rate. Because neural networks require intensive matrix computations, MATLAB provides a natural framework for rapidly implementing neural networks and for studying their behavior and application. The Neural Network Toolbox provides comprehensive support for the design, implementation, and simulation of many proven network paradigms. Its consistent methodology and modular organization facilitate research, provide a flexible framework for experimentation, and simplify customization.

This project has been dedicated to the studying the network behavior on character recognition problem. A similar project has been done by Kyn (1998) who simulate large information processing network to see the network performance in terms of accuracy. This project has been narrowed down the project scope done by Kyn where it simulate

character recognition problem in the presence of noise. The network performance were obtained in terms of percentage of error recognition by tested network.

Problem Statement

A network is to be designed and trained to recognize the 26 letters of the alphabet. A recognition system developed to digitizes each letter centered in the pattern input file. The result is that each letter is represented as a 20 by 20 grid of boolean values. However, the system is not perfect and the letters may suffer from noise. Perfect classification of ideal input vectors is required, and reasonably accurate classification of noisy vectors. This project explores the effect of noise on the designed character recognition system. The training a network on different sets of noisy vectors forced the network to learn how to deal with noise.

Objectives

The objectives of this project are :-

1. To gain an understanding of the process of neural network design in MATLAB by designing and testing a character recognition system.

2. To gain insight into issues like noise insensitivity (robustness) and generalization ability in neural network design.

Scope

The scope of the project is limited to the input data consisting of twenty-six uppercase pattern from 'A' to 'Z. The system also uses 10 set of noisy data which are created by introducing noise of mean 0 to 0.2 or less to the input data. Both data are used as trained data set. Hundred representation of noisy data then used as test data set. The trained network are tested in different noise level in the range of 0 to 0.5.

Methodology

Simulation experiment is conducted using MATLAB Neural Network Simulator to examine and test the capability of neural network created to solve character recognition problem given.. Backpropagation training method is used to train the network. It is chosen because its ability and popularity in development of Artificial neural network particularly in character recognition area.

CHAPTER II

LITERATURE REVIEW

Definition of Neural Network

According to the DARPA **Neural Network** Study (1988, AFCEA International Press, p. 60):

> ... a **neural network** is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

Brief History of Neural Network

The earliest work in neural network goes back to the 1940's when McCulloch and Pitts introduced the first neural network computing model. In the 1950's, Rosenblatt's work resulted in a two-layer network, the perceptron, which was capable of learning certain classifications by adjusting connection weights. Although the perceptron was successful in classifying certain patterns, it had a number of limitations. [2] The perceptron was not able to solve the classic XOR (exclusive or) problem. Such limitations led to the decline of the field of neural networks. However, the perceptron had laid foundations for later work in neural computing.

In the early 1980's, researchers showed renewed interest in neural networks. Recent work includes Boltzmann machines, Hopfield nets, competitive learning models, multilayer networks, and adaptive resonance theory models. [1]

Typical Architecture

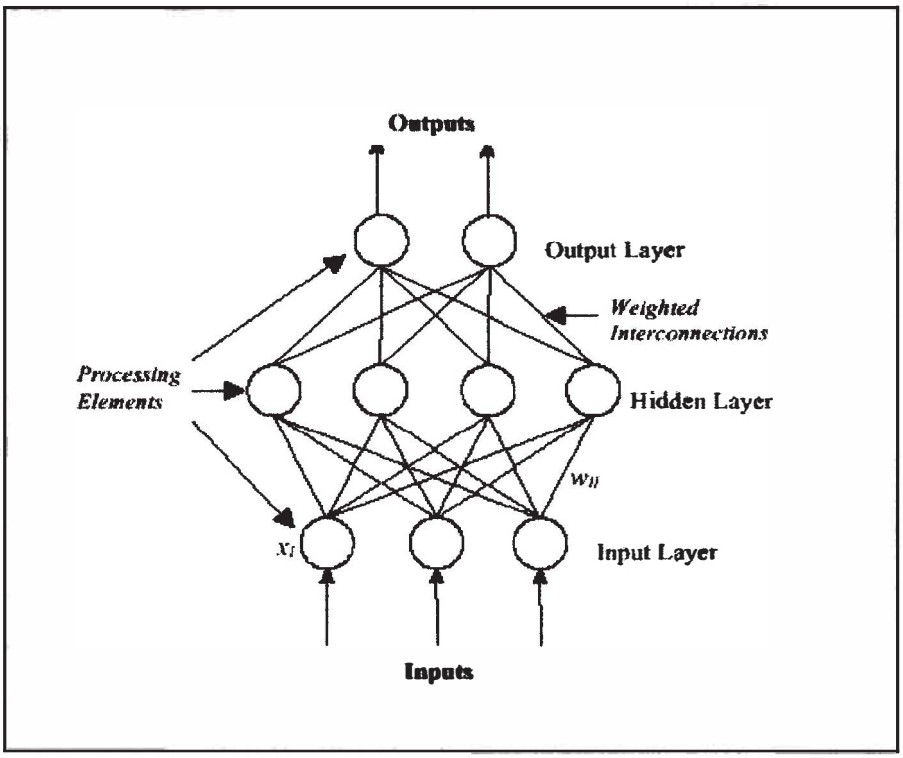


Figure 1 : A classic neural net architecture representation

The Input layer contains the independent variable values. This layer is connected to the Hidden Layer, which can have a variable number of *processing elements* . All processing elements in the Hidden Layer are connected to the elements in the Output layer, which contain the dependent variables for the given modeling problem. The *interconnections*

that connect all the processing elements in this structure are weighted, and it is the weights that are adjusted during the training process.

Training Method For ANN

In this project, the Backpropagation Training Method is selected from among the many training methods (or algorithms) to solve a given pattern recognition problem. Hence, before proceed to actually solving the problem itself, it will b e good to take a look at the Backpropagation algorithm. A close examination of this algorithm will help to understand or at least, grasp a rough idea on how an ANN trained by this method can learn a pattern recognition problem

Backpropagation Training Method

Introduction

The Back propagation algorithm is actually a generalization of the Least-Mean-Squared (LMS) algorithm (or sometimes also known as the delta rule). The LMS algorithm was first developed by Bernard Widrow and his student, Marcian (Ted) Hoff in the 1960s. [4] This algorithm or rule is closely related to the perceptron learning rule in the sense that both of them adjust the connection weights in an ANN whenever there is an error between the net output and the target value. The slight difference or improvement found in Back propagation algorithm, however, gives this algorithm the added advantage of

added advantage of increased ability in generalization (i.e., better ability to response to inputs that are similar, but not identical, to that of training ones.)

Least-Mean-Squared Or Delta Rule (Widrow-Hoff Learning Rule

As mentioned earlier, the Least-Mean-Squared (LMS) rule developed by Widrow and Hoff [4] is the precursor of the Backpropagation algorithm. It was initially developed for the Single-Layer-Perceptron (SLP) network. Backpropagation, on the other hand was developed to meet the need of a much-advanced Multi-Layer-Perceptron (MLP).

In order to have a better picture of what is the main idea these algorithms, it will be good for us to investigate the more 'primitive' or 'original' algorithm, that is the LMS rule. Through the examination of this rule, we hope gain a better understanding of how this rule contributes to the learning of an ANN.

The LMS rule change the weights of the neural connections so as to minimize the difference between the net input to the output unit, y_in, and the target value, t. Its aim is to minimize the overall error through reducing the error contributed by each training pattern, one at a time. Weight corrections can also be accumulated over a number of training patterns and this known as batch updating of the weights.

The LMS rule for adjusting the weight from the i_th. input unit ti the j_th. Output unit for each pattern is given as :

$$\Delta w_{ij} = \alpha (t_j - y\_in_j)x_i$$

Where x_i is the activation of input unit – i,

Y_in_j is the net input to output unit Y_j, and

$$Y\_in_j = \sum_{i=1}^{n} x_i w_{ii},$$

α is the learning rate,

t_j is the target output of pouput unit-j

The squared error for a particular training pattern is defined as :

$$E = \sum_{i=1}^{m} (t_j - y\_in_j)^2$$

Vector E is a function of all of weights. The gradient of E will consist of the partial derivatives of E with respect to each of the weight. This gradient which is also a vector, giver the direction of most rapid increase in E. Therefore, by reversing the sign (or direction) of gradient E, we obtain the direction of the most rapid decrease in the error. In other words, by adjusting the weight w_{ij} in the direction of $-\square E / \square w_{ij}$ will enable us to reduce the error most rapidly.

An explicit formula for $\partial E / \partial w_{ij}$ for an arbitrary weighr w_{ij} is given below.

$$\frac{\partial}{\partial w} = \frac{\partial}{\partial w} = \sum_{J=1}^{m} (t_j - y\_in_J)^2$$

For a particular weight, say w_{ij},

$$\frac{\partial}{\partial w} = \frac{\partial}{\partial w} = [(t_J - y\_in_J)^2]$$

Together with

$$Y\_in_J = \sum_{i=1}^{n} x_i w_{iJ},$$

We obtain

$$\frac{\partial}{\partial w} = -2(t_J - y\_in_J) \frac{\partial}{\partial w}$$

$$= -2(t_J - y\_in_J) x_I$$

Hence, the local error will be reduced most rapidly by adjusting the weights according to the LMS rule as mentioned before :

$$\Delta w_{ij} = \alpha (t_I - y\_in_J) x_I$$

Backpropagation

In general, training a network by standard Backpropagation involves three stages :

(a) The Feed – forward Phase,

(b) The Backpropagation Phase, and

(c) The Adjustment or Learning Phase.

(a) The Feed – forward Phase

During this phase, the input units receive input signals and broadcast them to each of the hidden units. Each hidden unit then computers its activation and sends its signal to all units in the next layer. This relay of signals continue until the output units are reached. The output units will computer their activation to form the response of the net for the given input pattern.

(b) The Backpropagation Phase

In this phase, an information of the errors corresponding to each input pattern is propagation backward from the output layer to the preview layers (i.e., the hidden layers). This information which is normally represented by a factor, δ_k (the subscript k indicates the corresponding unit from which the error information is obtained) is computed based on the error obtained through comparison between the activation value of the output and

its target value. (The mathematics involved in determining δ_k are given in the coming sub-section where the algorithm is presented)

(c) The Adjustment or Learning Phase

From the δ_k computed in the Backpropagation Phase, the required adjustment of weights is carried out in this phase. This phase is also called the Learning Phase because it is through the adjustment of weights, the ANN under training is seemed to be adapting itself to a particular state that it to solve the given problem.

Activation Function

An activation function for a Backpropagation-trained ANN should have several important characteristics. It should be :

i. Continuous

ii. Differentiable, and its derivatives must be easy to compute, and

iii. Monotonically non – decreasing

Two commonly used activation for Backpropagation-trained ANN are the binary sigmoid function and the hyperbolic tangent function (Figure 2).

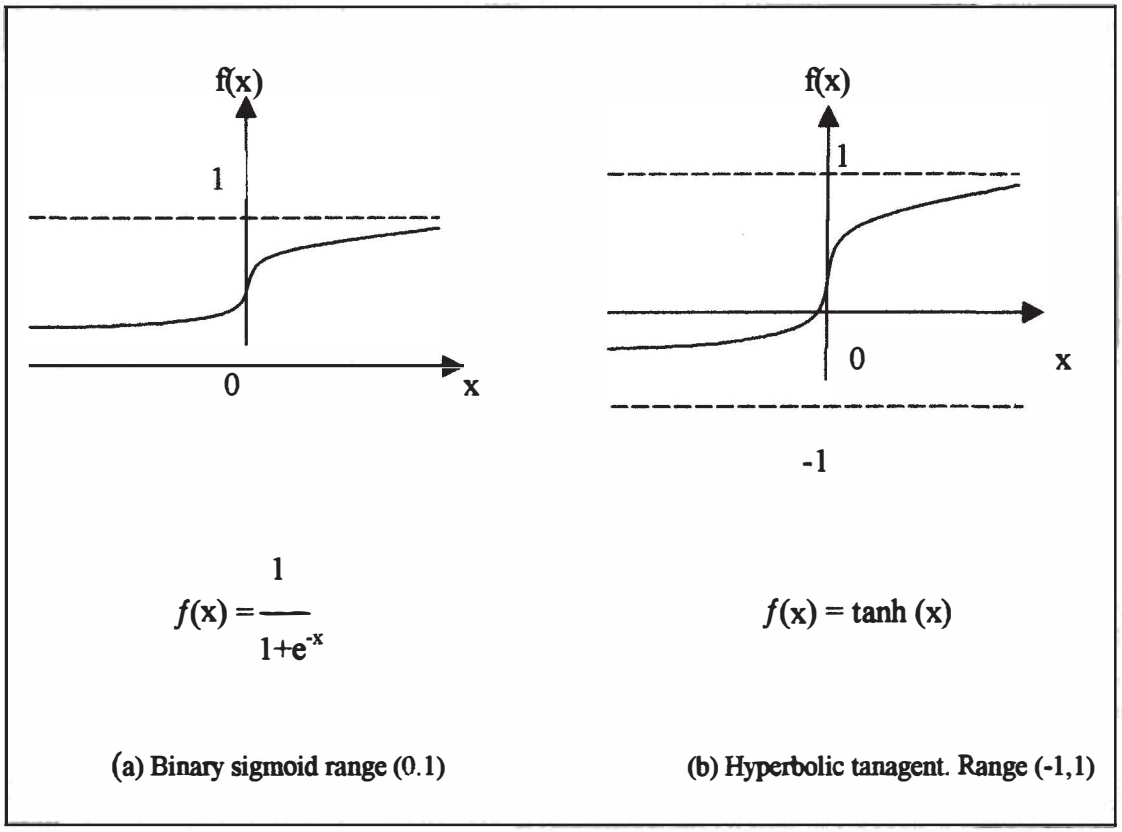


Figure 2 : Two types of activation functions used in Backpropagation-trained ANN