แบบแผนการสุ่มตัวอย่างซ้ำของข้อมูลที่ยากต่อการจัดกลุ่ม

นาย ปริญญา เวียงสมุทร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

DUPLICATE-SAMPLING OF DIFFICULT-TO-CLASSIFY SCHEME

Mr. Parinya Weangsamoot

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computational Science
Department of Mathematics
Faculty of Science
Chulalongkorn University
Academic Year 2008

Thesis Title	DUPLICATE-SAMPLING OF DIFFICULT-TO-CLASSIFY SCHEME
By	Mr. Parinya Weangsamoot
Field of Study	Computational Science
Advisor	Professor Chidchanok Lursinsap, Ph.D.
Co-Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

..... Dean of the Faculty of Science

(Professor Supot Hannongbua, Ph.D.)

THESIS COMMITTEE

..... Chairman

(Siripun Sanguansintukul, Ph.D.)

..... Thesis Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

..... Thesis Co-Advisor

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

..... Examiner

(Khamron Sunat, Ph.D.)

ปริญญา เวียงสมุทร : แบบแผนการสุ่มตัวอย่างซ้ำของชุดข้อมูลที่ยากต่อการจัดกลุ่ม. (DUPLICATE-SAMPLING OF DIFFICULT-TO-CLASSIFY SCHEME)
อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร.ชิดชนก เหลือสินทรัพย์, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม: ผศ. ดร. กรุง สินอภิรมย์สราญ, 49 หน้า.

โครงข่ายประสาทเทียมได้ถูกนำมาใช้ในการแก้ปัญหาการแบ่งกลุ่มข้อมูลอย่างสัมฤทธิ์ผล วิธีการเรียนรู้ที่นิยมใช้ในการสอนโครงข่ายประสาทเทียมคือวิธีการแพร่ย้อนกลับ ปัญหาของวิธีการแพร่ย้อนกลับคือการใช้เวลานานในการคำนวณ ด้วยเหตุนี้วิธีการต่างๆ ได้ถูกพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าว วิธีการส่วนใหญ่ที่พัฒนาขึ้นตั้งอยู่บนแนวคิดในการปรับปรุงการลู่เข้าของค่าความผิดพลาดของโครงข่ายประสาทเทียม งานวิจัยชิ้นนี้ได้เสนอวิธีการเพื่อแก้ปัญหาการใช้เวลานานของวิธีการแพร่ย้อนกลับโดยมีชื่อว่าแบบแผนการสุ่มตัวอย่างซ้ำของชุดข้อมูลที่ยากต่อการจัดกลุ่มซึ่งตั้งอยู่บนพื้นฐานของการกระจายของข้อมูลและค่าอินฟอเมชันเกน วิธีการที่นำเสนอได้ถูกออกแบบมาเพื่อเป็นเครื่องมือที่ทำให้วิธีการแพร่ย้อนกลับทำงานได้ดียิ่งขึ้น วิธีการดังกล่าวใช้ค่าอินฟอเมชันเกนในการแบ่งโดเมนของข้อมูลออกเป็นโดเมนย่อยโดยใช้หลักการแบ่งทวิภาคของคุณลักษณะประจำที่เป็นตัวเลข กลุ่มข้อมูลที่บรรจุอยู่ในโดเมนย่อยที่มีการปนกันของข้อมูลหลายกลุ่มจะถูกเรียกว่ากลุ่มข้อมูลที่ยากต่อการแบ่งกลุ่มซึ่งจะถูกจำลองซ้ำก่อนที่จะเริ่มการเรียนรู้ด้วยวิธีการแพร่ย้อนกลับ ในขณะที่ทำการสอนโครงข่ายประสาทเทียมด้วยวิธีการแพร่ย้อนกลับ การเรียนรู้ของข้อมูลที่ยากต่อการจัดกลุ่มจะถูกเน้นย้ำให้ความสำคัญมากยิ่งขึ้น วิธีการแพร่ย้อนกลับที่เสริมด้วยวิธีการดังกล่าวช่วยให้โครงข่ายประสาทเทียมใช้เวลาในการเรียนรู้น้อยลงในขณะที่ให้ผลการทำนายในระดับเดียวกันหรือสูงกว่า จากผลการทดลองบนแปดชุดข้อมูลที่ได้จากคลังข้อมูลยูซีไอ แสดงให้เห็นถึงการใช้เวลาที่ลดลงของวิธีการแพร่ย้อนกลับ เจ็ดในแปดชุดข้อมูลได้แสดงให้เห็นถึงผลลัพธ์ในการทำนายที่ดีขึ้นของโครงข่ายประสาทเทียม

ภาควิชา คณิตศาสตร์	ลายมือชื่อนิสิต.....
สาขาวิชา วิทยาการคณนา	ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....
ปีการศึกษา 2551	ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม.....

4872364123 : MAJOR COMPUTATIONAL SCIENCE
KEYWORDS: MULTILAYER PERCEPTRON NETWORK / BACKPROPAGATION ALGORITHM / INFORMATION GAIN / BINARY-SPLITTING / DIFFICULT-TO-CLASSIFY DATA / IMPURITY SUBSPACE

PARINYA WEANGSAMOOT : DUPLICATE-SAMPLING OF DIFFICULT-TO-CLASSIFY SCHEME. ADVISOR : PROF. CHIDCHANOK LURSINSAP, Ph.D., THESIS CO-ADVISOR: ASST. PROF. KRUNG SINAPIROMSRAN, Ph.D., 49 pp.

The multilayer perceptron (MLP) network is applied successfully in solving many pattern classification problems. The famous learning algorithms used to train the MLP network is the backpropagation (BP) algorithm. One disadvantage regarding to the BP algorithm is its lengthy computational time. Therefore, a number of techniques were developed to handle this situation. Most of those techniques still based on a concept of improving the convergence of the network's error. In this thesis, we present another view, based on distribution of data and information gain, to solve the time-consuming problem called duplicate-sampling of difficult-to-classify scheme. The proposed technique is designed as an enhancement tool to apply with the standard BP algorithm. The technique utilizes the information gain to split data space into subspaces based on binary splitting of numeric attribute. Data located in the impure subspaces are identified as difficult-to-classify data which are duplicated before starting the BP algorithm. During the learning process, the difficult-to-classify data will be emphasized by the BP learning. The BP learning with this technique requires less computational time to achieve the same or higher accuracy. The experiments performed on eight data sets taken from the UCI repository indicate a computational time improvement. Seven out of eight data sets showed a better result.

Department: Mathematics Student's Signature.....
Field of Study: Computational Science Advisor's Signature.....
Academic Year 2008 Co-Advisor's Signature.....

ACKNOWLEDGEMENT

I am especially deeply grateful to many people who encourage me and help me the course of this study, especially Professor Dr. Chidhanok Lursinsup and Assistant Professor Dr. Krung Sinapiromsran. Their valuable suggestions and comments make this work feasible and keep it in the right direction. I would also like to thank my committee members for their corrections of my thesis as well as providing valuable comments at the committee meeting.

I would like to give a lot of thanks to people in the Advance Virtual and Intelligence Computing (AVIC) Research Center who show me how to be a good researcher. They always give me many useful suggestions and, of course, being my good friends. Their experiences are very useful to me. Without them, I probably have a harder time in doing my first thesis.

My special thanks go to my beloved parents. They always support me everything. Their love and encouragement are the most important things in my life. Furthermore, I also wish to express my special thanks to my younger brother and sister. They always told me "*keep going ahead*, *don't give up*", these words are very encourage me.

Finally my deep appreciation goes to my girlfriend, *Dtuk*. She always stands beside me, and always believes in my potential. Her love, care, encouragement and some complaining keep me on the way to success. Without her, I don't know what kind of person I will be.

CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I
INTRODUCTION

1.1 Problem and Motivation

Among all classification techniques, one outstanding classifier is a neural network. The advantages of using the neural network dealing with pattern classification problems are its robustness and its high predictive capability. In the past decades, various types of neural network techniques have been employed to handle the real-world pattern classification problems. Especially, the one which involves in many fields of research is the multilayer perceptrons (MLP) network trained with the backpropagation (BP) learning algorithm. The BP algorithm, which is performed under a supervised manner, is a concept of searching an optimal group of vectors in weight space that yields the lowest network's error [1]. The MLP network with BP algorithm showed many successes in solving a large number of complicated problems. Nevertheless, to get a valid MLP network, it requires a long learning time.

In this research, we present a new technique to reduce computational time of the BP algorithm called *duplicate-sampling of difficult-to-classify scheme*. The fundamental idea of the proposed technique based on a concept that each datum does not contain the same useful information. Thus, the difficulty in identifying the class of each datum should depend on the information it carried. We extend this concept to the BP learning and make an assumption that the data with higher useful information require small amount of learning time (small number of epochs) to be correctly classified by the network. On the other hand, the ones with less useful information require large amount of learning time (large number of epochs) to be correctly classified by the network. According to this assumption, we categorize data into two types which are *difficult-to-classify data* which requires a large number of epochs and *easy-to-classify data* which requires a small number of epochs. We try to reduce the BP learning time by intensifying the presentation of the difficult-to-classify data to the network; while, the presentation of the easy-to-classify data to the network is not changed. Consequently, we utilize the information gain measurement as a tool to identify the difficult-to-classify and easy-to-classify data. Then, we duplicate the difficult-to-classify data. Hence, in each epoch of the BP learning process, the presentation of the difficult-to-classify data is emphasized.

1.2 Objective

The objective of this study is to develop a learning scheme that requires less computational time than the standard backpropagation algorithm while still maintains the high predictive ability.

1.3 Scope of Work and Constraints

This study is constrained by the following conditions:

1. The study focuses on solving pattern classification problems.

2. The study concentrates on numeric data sets only.

1.4 Literatures Related to Multilayer perceptron network and Backpropagation Algorithm.

Jacobs, R.A. [2] presented techniques to modify the backpropagation algorithm called delta-bar-delta-learning rules. These rules intend to accelerate the convergence of backpropagation algorithm. This issue contains four important schemes:

1. Every adjustable network parameter of the cost function should have its own individual learning-rate parameter.

2. Every learning-rate parameter should be allowed to vary from one iteration to the next.

3. When the derivative of the cost function with respect to a synaptic weight has the same algebraic sign for several consecutive iterations of the algorithm, the learning rate parameter for that particular weight should be increased.

4. When the algebraic sign of the derivative of the cost function with respect to a particular synaptic weight alternated for several consecutive iterations of the algorithm, the learning-rate parameter for that weight should be decreased.

Salomon and van Hemmen [3] presented a method to accelerate the backpropagation algorithm called dynamic self-adaptation procedure. The underlying idea is to take the learning rate of the previous time step, increasing and decreasing it slightly, evaluating the cost function for both new values of the learning-rate parameter, and then choose the particular one that gives the lower value of the cost function.

LeCun Y. [4] described general rules for training any example with backpropagation algorithm called maximizing information content. The details of these rules are:

1. The use of an example should result in the largest, as possible, scale of training error.

2. The use of an example should be radically different from all those previously used.

In practice, the second rule can be performed, easily, by shuffling the order of examples presented to the multilayer perceptrons from one epoch to the next.

LeCun [4] described a technique to make the BP algorithm perform faster called *emphasizing scheme.* This scheme is performed by presenting the difficult data to the MLP network more often than the easy ones. The difficult data are determined by examining the error of a particular datum in each epoch and comparing it with the error in the previous epoch.

Stone, M. [5] described a statistical technique to create any classification model with generalization as a goal. This technique is called cross validation. First, the data set is randomly split into training and test sets. The training set is further split into two disjoint subsets:

1. Estimation subset.
2. Validation subset.

The motivation here is to validate the model on a data set different from the one used for the parameter estimation. In this way, we may use the training set to assess the performance of various candidate models, and, thereby, choose the best one.

Morgan and Bourlard [6] presented a method to train multilayer perceptrons with backpropagation called early-stopping method. This technique is designed for encountering overfitting problem of neural network. The method suggested the stopping criterion for backpropagation algorithm. The learning process should be stopped when the mean squared error on training set decreases while the mean squared error of validating set increases.

Suresh, Omkar and Mani [7] presented the MLP network with BP algorithm on network of workstation. This research tries to reduce the BP learning time under a concept of parallel computing.

Arit Thammano and Asavin Meengen [8] presented a new classifier technique called *new evolutionary neural network classifier*. The technique applied the concepts of

fuzzy c-means algorithm and the evolutionary algorithm to the neural network. During training, the fuzzy c-means algorithm is used to form the clusters in the cluster layer (hidden layer) and the evolutionary algorithm plays role in optimizing those clusters and their parameters. Applying it to a test data, the class of any particular pattern is determined by examining which cluster node (each cluster node representing a different class) returns the maximum output value.

1.5 Literatures Related to information gain.

Ross Quinlan [9] presented a technique to construct a decision tree based on nominal data called *Iterative Dichotomiser 3* (ID3) algorithm. This algorithm employs the information gain as a tool to select an appropriate attribute for branching nodes of a tree under a concept of getting the smallest tree as possible.

Ross Quinlan [10] presented a C4.5 decision tree algorithm, the successor of the ID3 algorithm. The C4.5 still based on the principals used in ID3. The improvements of C4.5 over the ID3 are: an extending to discrete and continuous attributes, techniques for handling the missing values and rules for pruning trees.

John T. Kent [11] investigated in using the information gain to measure correlation between two random quantities, X and Y, in a parametric model of dependence. The investigation aimed to measure the correlation for both the usual product-moment correlation coefficient for the bivariate normal model and the multiple correlation coefficients in the standard linear regression model.

David J. C. Mackay [12] presented a technique to make a Bayesien learning framework more efficient called *Information-Based Objective Functions for Active Data Selection*. This technique described three different criterions to select salient data points from data space to gain more information during learning.

Lisa Borland, Angel R. Plastino and Constantino Tsallis [13] discussed an issue of information gain within nonextensive thermostatics. The discussion is about the general properties and a consistent test for measuring the degree of correlation between random variables is proposed. Moreover, minimum entropy distributions are also discussed and the H-theorem is proved within the generalized context.

1.6 Organization

The thesis is organized as follows. Chapter 2 discusses the underlying idea of the pattern classification problems, the multilayer perceptron network, the backpropagation algorithm and the information gain in a decision tree algorithm. Chapter 3 explains the mathematical details of the backpropagation algorithm and the duplicate-sampling of difficult-to-classify scheme. Chapter 4 shows the experimental results. Finally, Chapter 5 concludes the proposed scheme and suggests some possible future works.

CHAPTER II
THEORETICAL BACKGROUND

The purpose of this chapter is to describe the fundamental theory of the pattern classification problem, the multilayer perceptron network, the backpropagation algorithm and the information gain in a decision tree algorithm.

2.1 Pattern Classification Problem

Pattern classification or data classification is a process of constructing model from a data set with predefined class (training set or training sample). The constructed model will be used to predict a class of new data [14]. The training set is collected in a format as shown in Table 2.1. Each datum or pattern is described by attributes and one of the attributes is used to define a class of data called class label attribute.

Table 2.1 Some parts of Iris data set.

Instance	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	7.0	3.2	4.7	1.4	Iris-versicolor
6	6.4	3.2	4.5	1.5	Iris-versicolor
7	6.9	3.1	4.9	1.5	Iris-versicolor
8	5.5	2.3	4.0	1.3	Iris-versicolor
9	6.5	2.8	4.6	1.5	Iris-versicolor
10	6.3	3.3	6.0	2.5	Iris-virginica
11	5.8	2.7	5.1	1.9	Iris-virginica
12	7.1	3.0	5.9	2.1	Iris-virginica
13	6.3	2.9	5.6	1.8	Iris-virginica
14	6.5	3.0	5.8	2.2	Iris-virginica

The Iris data set [15] contains three different classes. Each class refers to a type of iris plant, and each attribute refers to a physical detail of the iris plant. Each datum can be considered as a mathematical pattern such as the first instance which can be viewed as a pattern or a vector with a value of < 5.1, 3.5, 1.4, 0.2, 1 > where the last member (1) refers to its corresponding class (Iris-setosa). Pattern classification is a challenging problem. A number of mathematical models have been developed to solve it, and the famous one involved in this research is a neural network.

2.2 An Overview of Artificial Neural Network

Human brain has a high ability in learning and solving many complicated problems. It is a highly complex organ of human which consists of billions of specially built cell called neurons. It is estimated that there are ten billion neurons in the human brain. The neuron consists of four major components which are soma, axon, dendrite and synapse as shown in Figure 2.1.

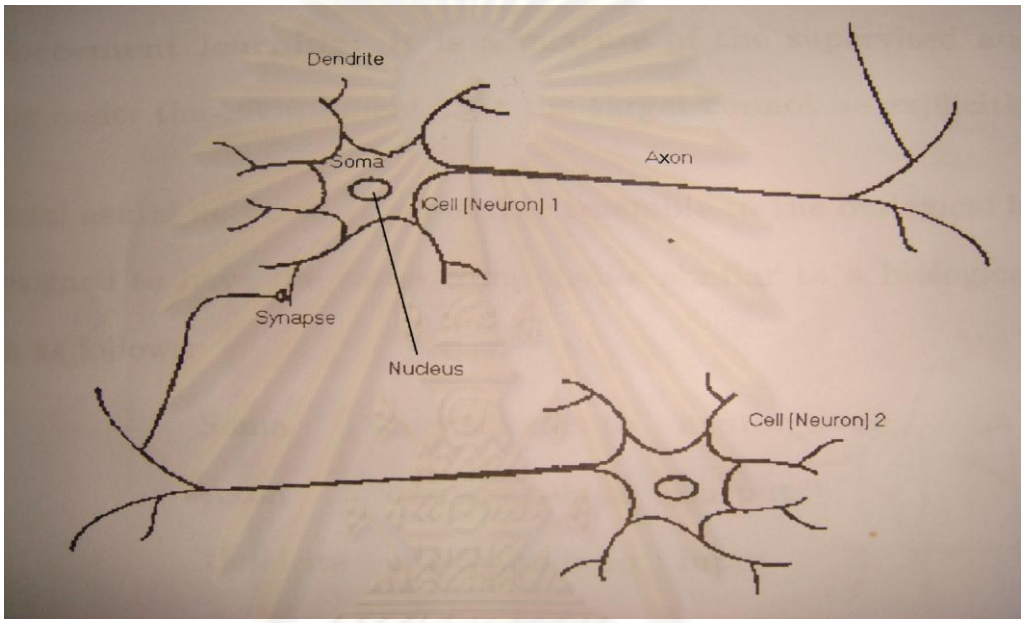


Figure 2.1 The human neurons.

Each single neuron is connected to others forming an enormous network called neural network. Any single neuron acts as an information-processing unit. When a particular neuron is stimulated by an input signal, the signal will be transmitted via the dendrite to the soma. Then, the input signal will be transmitted, again, via the axon to the other connected neurons. Before the signal will be transmitted from one neuron to others, the strength of the transmitted signal will be controlled (be improved or worsened) by the chemical and biological reaction called synaptic resistance. This reaction allows the neuron to control the strength of the transmitted signal. Therefore, any neuron can create a particular output signal (a signal with specific strength) corresponding to its input signal. By the processing of each neuron, the neural network is able to create a distinctive output signal to respond to a particular input signal. To get the distinctive output signal, the network has to learn to adjust the strength of the input signal during an internal signal transmission (transmitting signal from one neural in the network to others). As mentioned previously, the process that plays role in

adjusting strength of the transmitted signal is the synaptic resistance. This process can be considered, in a mathematical sense, as a process of weight adjustment where the weights refer to the chemical and biological factors. When the weights are changed, it is compared as an occurring of chemical and biological reactions that yields the altering of signal's strength.

The learning process of the neural network is a process of adjusting the weights of all neurons in the network. The purpose of the adjustment is to make the network able to create a particular output signal corresponding to its received input signal. The learning process of the neural network can be categorized into three categories as follows.

1. Supervised learning. The neural network is forced to generate a specific target signal corresponding to each particular input signal (the weight adjustment process is performed with respect to a value of the target signal).

2. Unsupervised learning. There is no a specific target signal for each input signal. The neural network learns by adjusting its weights equal to a value of the input signal (the weight adjustment process is performed with respect to a value of the input signal).

3. Reinforcement learning. It is a combination of both previous learning schemes under the environment that the target output cannot be clarified.

An artificial neuron is made up to mimic the human's neuron in both sense of structure and function. Figure 2.2 shows structure of the artificial neuron.

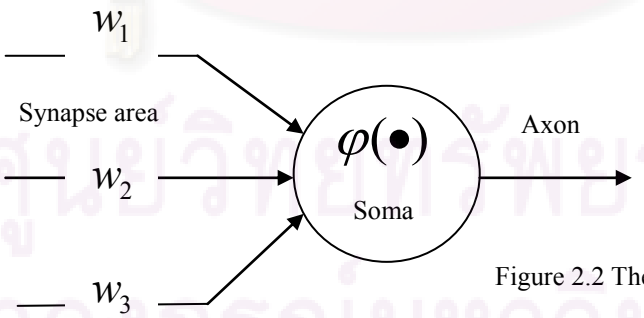


Figure 2.2 The artificial neuron.

w_i is a weight of a neuron, and $\varphi(\bullet)$ is an activation function of the neuron. Similarly to the real neuron, a single artificial neuron connects to others to form a network called artificial neural network (ANN). In addition, the learning process of the ANN can also be performed under the three learning schemes as previously described. In the

next section, we will describe the detail of its activation function, the vital part of the artificial neuron.

2.3 Activation Function

An activation function plays an important role in the learning process. The use of the artificial neuron with different activation function yields as the different value of the output signal. Figure 2.3 shows structure and some mathematical notations of the artificial neuron.

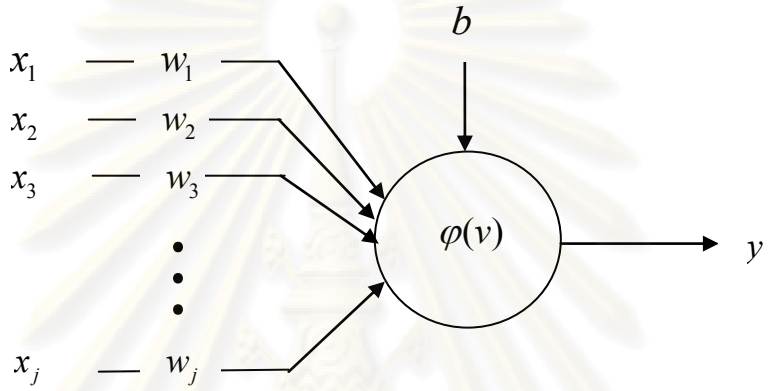


Figure 2.3 The structure of the artificial neurons with some mathematical notations. $<x_1,x_2,...,x_j>$ refers to an input signal, $<w_1,w_2,...,w_j>$ refers the adjustable weights, b is a bias, *v* is a linear combination of an input signal and weights, $\varphi(v)$ is an activation function and *y* is an output of neuron. When the artificial neuron is stimulated by the input signal (i.e., the input signal is presented to the neuron), each element of the input signal will be multiplied by the associated weight. Then, the results of each multiplication are summed together yielding an induced local field *v* (a linear combination of an input signal). Consequently, the induced local field can be defined by

$$v = (\sum_{j} w_j x_j) + b \qquad 2.1$$

The index *j* is the order of the members of the input signal. The bias *b* is presence to make the activation function set off from zero [16]. After getting the induced local field *v*, the computation of an output signal is performed as in equation 2.2.

$$y = \varphi(v) \qquad 2.2$$

y is the output signal, and $\varphi(\bullet)$ is an activation function. Equations described above are the common computation to create an output signal of the artificial neuron.

Many studies in the neural network have investigated on finding a better activation function. Consequently, a number of activation functions were created. In this thesis, we present just the three well-known activation functions which are Threshold function, Piecewise-linear function and Sigmoid function.

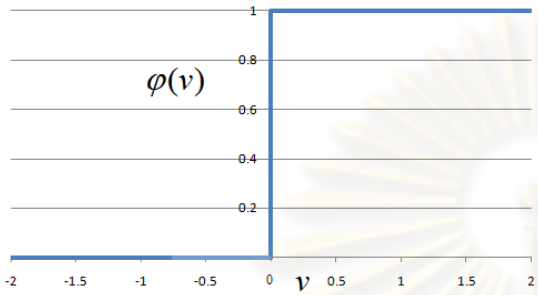


Figure 2.4 (a) The threshold function.

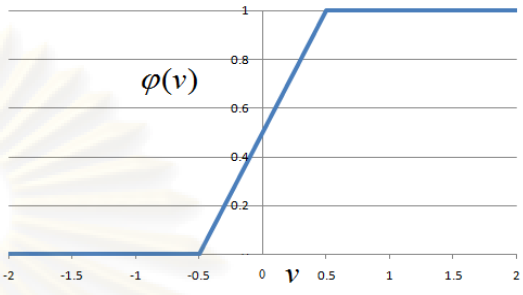


Figure 2.4 (b) The piecewise-linear function.

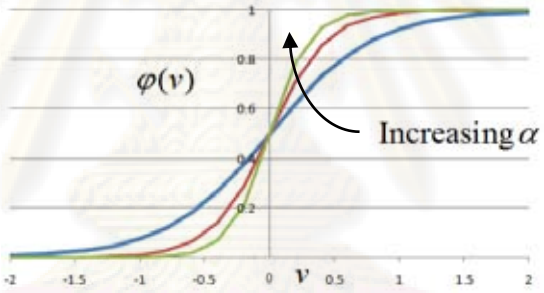


Figure 2.4 (c) The Sigmoid function.

Figure 2.4 Various types of activation functions.

1. Threshold function. The mathematical definition of the activation function illustrated in Figure 2.4 (a) can be described by

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases} \qquad 2.3$$

This type of an activation function is known as the *Heaviside Function* in an engineering field. The output of this function is achievable in two values: getting on the value of 1 when the induced local field (v) is nonnegative value and getting on the value of 0 otherwise.

2. Piecewise-Linear Function. The mathematical definition of the activation function illustrated in Figure 2.4 (b) can be described by

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \qquad 2.4$$

The result obtained from this form of an activation function possibly falls in two areas: non-saturation area, or saturation area. As in equation 2.4, if the value of the induced local field v is in range of $(-\frac{1}{2}, \frac{1}{2})$, the output of function still vary, and the amount of output is equal to the induced local field itself. On the other hand, if the value of the induced local field is larger or less than the restricted range, the output will be set to a value of one or zero, respectively.

3. Sigmoid Function. The S-shape graph illustrated in Figure 2.4 (c) is the most common form of an activation function used in the artificial neural networks. This form of an activation function is defined by

$$\varphi(v) = \frac{1}{1 + \exp(-\alpha v)} \qquad 2.5$$

α is a slope parameter. Adjusting this parameter results in a changing of the slope of the S-shape graph. As shown in Figure 2.4 (c), the output of the sigmoid function is a continuous value between zero and one. When compared with the two previous functions, the altering of the output of the sigmoid function is smoother than the two previous functions. Accordingly, the use of the sigmoid function in the ANN yields a more refined output which makes it more popular.

2.4 Multilayer perceptron network

The artificial neuron illustrated in Figure 2.2 can be mentioned as a perceptron. The perceptron is a single artificial neuron with the adjustable synaptic weights and a bias. It is a simple model used for the classification of patterns. The use of a single perceptron in classifying could achieve a high classification performance on linearly-separable problems. However, it gives a low performance when dealing with nonlinearly-separable problems. The detail about why a single perceptron fail to solve nonlinearly-separable problems can be found in [1]. Commonly, to use perceptron handling real world problems, we connect a single perceptron together to form a

network of perceptrons. There are various designs of the perceptrons networks. The famous one is the multilayer perceptrons (MLP) network as shown in Figure 2.5.

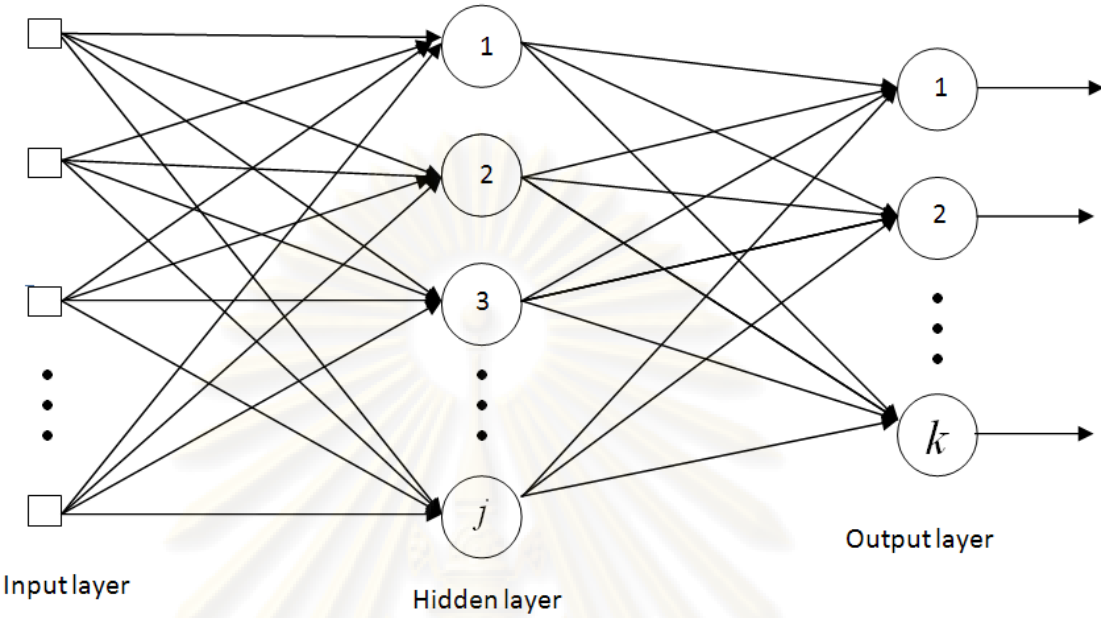


Figure 2.5 The multilayer perceptron network.

This type of a network consists of three basic layers:

1. Input layer. This layer consists of a group of input nodes. A number of the input nodes are equal to a number of members of the input signal. This layer plays role in receiving the input pattern and constituting the input signal to apply to the next layer of network. The input node is sometime called source node. There is no computation in this layer.

2. Hidden layer. This layer consists of a group of hidden nodes which are the perceptrons. The perceptrons receive the input signal from the input layer and perform the computation shown in equations 2.1 and 2.2 yielding the output signal. The output signals obtained from this layer will be sent to the next layer of the network. In addition, this layer of the network can be constructed more than one layer. However, the one-hidden node network is preferred, and it is successful in solving many complicated problems.

3. Output layer. This layer is the last layer of the network. It consists of a group of output nodes. The output signals obtained from the hidden layer act as the input signals in this layer. After receiving the input signal, the output nodes perform the same computation as similar in the hidden nodes yielding the output signal. The

output signal obtained from this layer is determined as the output signal of the network.

According to the three layers of the MLP network, the input signals fed to the network will be sent forward through the entire network, layer by layer. Therefore, the MLP network is sometime called the *feed forward network*. In the next section, the learning process of the network will be described.

2.5 An Overview of Backpropagation Algorithm.

As in the previous explanation, the learning process is a process of adjusting weights. A number of learning algorithms have been developed to apply with the MLP network. The most famous learning algorithm is the backpropagation (BP) algorithm. This algorithm is the supervised learning scheme. It requires a specific target output for each input signal. To understand well in the underlying idea of this algorithm, the concept of the propagation signals should be informed first. The signals propagating through the network can be categorized into two types:

1. Function signal. The signals propagating forward from different layers are called *function signal* or sometime called *forward signal* after its direction of propagation.

2. Error signal. As we described that the backpropagation learning algorithm is performed under the supervised learning scheme which means each input signal has its associated output signal (target signal). The target signals can be either set by the backpropagation algorithm itself or set by user, and the assigned values should be in a possible range of the current-using activation function. Equation 2.6 shows the definition of the error signal.

$$e_k = d_k - y_k \qquad 2.6$$

y_k is the output signal obtained from forward process corresponding to the output node k, d_k is the target signal corresponding to the output node k, and e_k is the error signal corresponding to the output node k, respectively. The error signal will be sent backward through the network to all neurons (perceptrons) in the network, it is sometime called backward signal. The error signal is sent backward to each neuron to use in the weight adjustment process. The amount of weight change depends on a value of the error signal itself. Furthermore, the error signal is also used to determine the performance of learning. A smaller value of error means better learning.

The BP algorithm consists of two main phases: forward phase and backward phase. The forward phase is the processes of receiving the input signal, propagating the signal forward through various layers of the network till yielding the output signal. The backward phase is the processes of computing the error signal and sending the error signal backward to all neurons (perceptrons) in the network.

The BP algorithm is an iterative technique. The forward and backward phases are iteratively performed until the satisfying value of error is met. In the next chapter we will show the full mathematical details of the backpropagation algorithm.

2.6 Information Gain in a Decision Tree Algorithm

The construction of a decision tree is a recursive process. It bases on a divide-and-conquer technique. In this section, how the information gain help in constructing a decision tree and how the finished decision tree classify data are given.

Information gain is a measure used to determine the relevance of the attributes of a data set with respect to the target class. It was used in a decision tree algorithm [9], [10] as a criterion to select a proper attribute for branching nodes of a decision tree. When the decision tree algorithm was applied to handle a numeric data set, each decision node can be considered as a hyperplane that is orthogonal to a standard e_i vector (the vector that represents the i^{th} axis) of a data space. The complete decision tree can be referred to a group of hyperplanes that partition the data space into subspaces. According to this view point, a problem of constructing the decision tree can be viewed as a technique that recursively select the most relevant attribute for constructing a splitting hyperplane. The information gain is involved in a process of determining the most relevant attribute in which the attribute that gives the highest information gain will be selected. The split subspace that contains only one target class is called a pure subspace. The decision tree can easily identify instances in this subspace to be that unique value of the target class. For an impure subspace, the decision tree identifies instance in this subspace by following the majority class. In the next chapter, we will show the full steps of the decision tree construction.

CHAPTER III
MATHEMATICAL MODEL AND ALGORITHM

3.1 Backpropagation Algorithm

As we described in the previous chapter, the most famous learning algorithm used to train the MLP network is the backpropagation (BP) algorithm. The learning process can be viewed as a method of adjusting weights to create a specific output signal associating to a particular input signal. Equation 3.1 shows the computation used in the weight adjustment process.

$$w_{new} = w_{old} + \Delta w \qquad 3.1$$

w_{old} is a weight before adjusting, w_{new} is a new weight obtained from adjusting and Δw is a degree of weight change. The delta weight can be either positive or negative value depending on to increase or decrease the old weight (w_{old}). Table 3.1 shows a sample data set.

Table 3.1 Sample data set.

Number	Feature1	Feature2	Feature3	Class
1	1.0	0.5	0.0	A
2	0.7	0.6	0.2	A
3	0.7	0.5	0.7	B
4	0.5	0.6	1.0	B

This data set was created to illustrate how to construct a multilayer perceptron network to handle the real data set. The sample data set contains three attributes (feature 1, 2 and 3) and two classes (A, B). The target signal for class A was set as a vector of two dimensions with a value of <1, 0>. For class B, the output target was set as a vector of two dimensions with a value of <0, 1>. Suppose we have *n* classes of data, the output targets will be the vectors of *n* members as following: <1, 0, …, 0> for the 1st class, <0, 1, …, 0> for the 2nd class and going on with this pattern. Hence, we get <0, 0, …,1> for the nth class. This is the way commonly used to establish the target signals.

We have the data set whose each input signal consists of three members (not including the class label attribute), and there are two classes of data. Hence, the design of the multilayer perceptron network should be as the one shown in Figure 3.1.

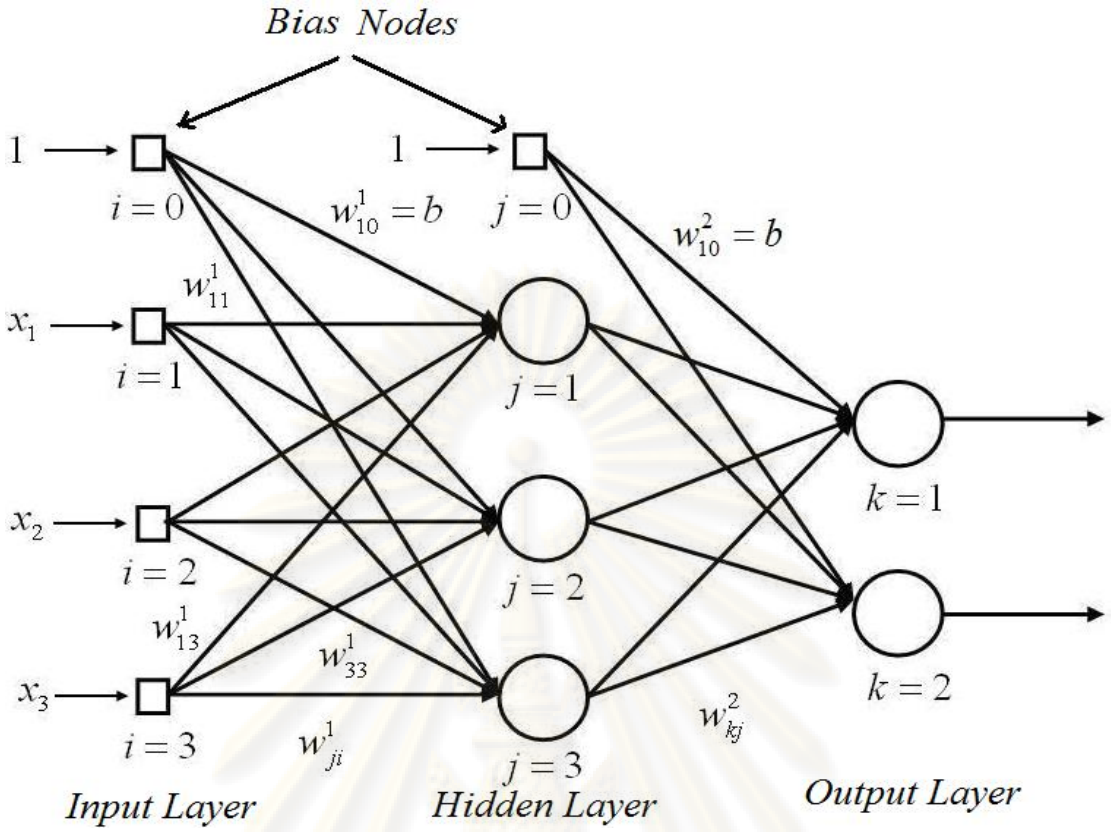


Figure 3.1 The design of the multilayer perceptron network.

In Figure 3.1, the network consists of three layers. The input layer consists of three input nodes and another one bias node. A number of input nodes are designed to fit a size of the input signal (a vector with 3 members). The hidden layer contains three hidden nodes and one bias node. This layer of the network is allowed to have more than one layer, and a number of hidden nodes in each layer are adjustable. Finally, the output layer consists of two output nodes. A number of output nodes are equal to a number of classes.

The definitions of each notation are as follows: x_i refers to each element of the input signal, w is an adjustable weight of the neuron where the related superscript is a group of weights, and the related subscript is an order of the associated neuron nodes. For example w_{11}^1 is an adjustable weight which belongs to group one and corresponds to the connection between the first hidden node and the first input node.

There are two groups of weights in the network: the first group is a group of weights between the input layer and the hidden layer, the second group locates

between the hidden layer and the output layer. Both groups can be viewed in a matrix form as shown below.

$$\begin{bmatrix} w_{10} & w_{11} & w_{12} & w_{03} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Figure 3.2 Matrix form of the first group of the adjustable weights.

$$\begin{bmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \end{bmatrix}$$

Figure 3.3 Matrix form of the second group of the adjustable weights.

The data in the first column of both groups are biases. This first group of weights is sometimes called the hidden-layer weights, and the second group of weights is sometimes called the output-layer weights. The bias nodes were assigned by a value of 1 which makes the multiplications between the inputs and the biases equal to values of the biases themselves. During the learning process, all of weights in both groups are adjusted by using equation 3.1. In accordance with equation 3.1, the backpropagation algorithm can be considered as a method to find appropriate delta weights (Δw) for all weights in the network. When applying Δw to the old weights (w_{old}), the network with a new group of weights will produce a smaller value of error. As described in the previous chapter, the backpropagation algorithm consists of two main phases: the forward phase and the backward phase. More precise details of these two phases are described in the following sections.

3.1.1 Forward phase computation

The forward phase of backpropagation algorithm begins from receiving an input signal and following by the computation of the hidden layer and the computation of the output layer until yielding the output signal. The first computation occurs in the hidden layer, there is no any computation in the input layer. Equation 3.2 shows the computation of the induced local filed of each neuron in the hidden layer.

$$v_j = \sum_{i=1}^{n} x_i w_{ji} + w_{j0} \qquad 3.2$$

v_j refers to the induced local field corresponding to the hidden node j, x_i refers to a member of the input signal corresponding to the input node i, w_{ji} is an adjustable

weight corresponding to the hidden node *i* and the input node *j*, and w_{j0} is a bias of the hidden node *j*.

In equation 3.2, each member of the input signal is multiplied by an associated weight. Then the results are summed together, and added by a bias to produce an induced local field (v_j). The induced local field will be used to calculate the output signal of the hidden neuron. Equation 3.3 shows the computation of the output signal.

$$y_j = \varphi(v_j) \tag{3.3}$$

$\varphi(v_j)$ is an activation function where the input of function is an induced local field obtained from equation 3.2, and y_j is an output signal of the hidden node *j*.

After getting all output signals from the hidden nodes, these output signals will be transmitted to the next layer of the network. Equation 3.4 shows the computation of the induced local field of the output neurons (output nodes).

$$v_k = \sum_{j=1}^{n} y_j w_{kj} + w_{k0} \tag{3.4}$$

v_k is the induced local field corresponding to the output node *k*, y_j is an output signal of the hidden node *i* or an input signal of the output node *j*, w_{kj} is an adjustable weight corresponding to the output node *k* and the hidden node *j*, and w_{k0} is a bias of the output node *j*.

In equation 3.4, the computation is similar to the computation of the hidden neuron where the output signals of the hidden neurons act as the input signals in this layer. Similarly, the output signal of the output neuron can be calculated by the same way.

$$y_k = \varphi(v_k) \tag{3.5}$$

$\varphi(v_k)$ is an activation function where the input of function is an induced local field obtained from equation 3.4, and y_k is an output signal of the output node *k* (output neuron *k*). The output signals obtained from the output layer are determined as the real output signal of the network.

3.1.2 Backward Phase Computation

In the backward phase of the backpropagation algorithm, an error signal will be sent backward to all neurons in a network. The purpose of sending the error signal backward is to find the effect to the error of each weight. Moreover, the size of a related delta weight in equation 3.1 depends on amount of this effect. In fact, the key

of the backward phase is to find the appropriate delta weights (Δw) for all weights in the network. When applying these delta weights to the weights (w_{old}), the network with a new group of weights (w_{new}) will produce a smaller value of error. Equation 3.6 shows the definition of the error signal corresponding to any particular output node.

$$e_k = d_k - y_k \tag{3.6}$$

d_k is the output target for the output node k, y_k is the output signal from the output node k and e_k is the error signal corresponding to the output node k.

Equation 3.6 shows the produced error corresponding to each output node. In order to estimate the error of network, we use the cost function ($C(\bullet)$) which is defined by

$$C(<e_1, e_2, ..., e_n>) = \frac{1}{2}\sum_{k=1}^{n} e_k^2 \tag{3.7}$$

From equation 3.7, the input of the cost function is a vector of errors. The related subscript is an order of the output node. Equation 3.7 is the computation of the error associated with only one input signal. Adding an order of the input signal to equation 3.7, we get

$$E(n) = \frac{1}{2}\sum_{k=1}^{n} e_k^2(n) \tag{3.8}$$

n is an order of input signals in a training set. Consequently, $E(n)$ is the network's error with respect to the n^{th} input signal. In order to compute the network's error with respect to all input signals in the training set, we use a mean square error (E_{av}) which is defined by

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{3.9}$$

N is a number of input signals in a training set.

The degree of a delta weight applied to any single weight is determined by the amount of an effect to the error of the weight itself. This effect is determined by a rate of change of the error with respect to a rate of change of that particular weight. Accordingly, we use the differentiation of the error ($E(n)$) with respect to the particular weight as shown in equation 3.10 to find an appropriate value of delta weight.

$$\Delta w_{kj} = -\eta \frac{\partial E(n)}{\partial w_{kj}} \tag{3.10}$$

We called equation 3.10 a delta rule. It shows that the degree of delta weight (Δw) depends on the differentiation. *E(n)* is the error corresponding to the nth input signal. w_{kj} is the adjustable weight corresponding to the output node *k* and the hidden node *j*. Δw_{kj} is the delta weight used to apply to the associated weight (w_{kj}). η is a learning rate which plays role in controlling a degree of Δw_{kj},. Its value lies in range of [0,1]. The network's error $E(n)$ can be expressed as a function of the weight (w_{kj}). Graphs illustrating the relation between $E(n)$ and w_{kj} can be plotted as shown in Figure 3.4.

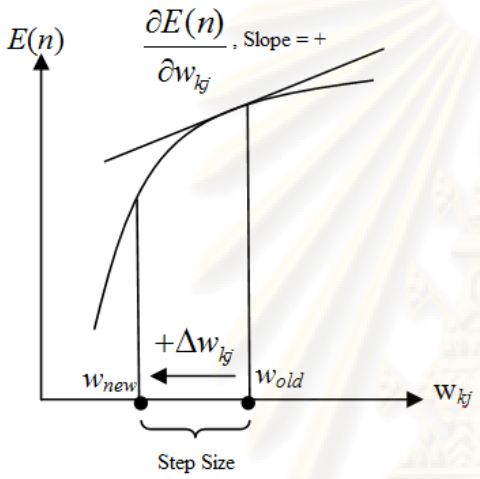


Figure 3.4 (a) In case of increasing function

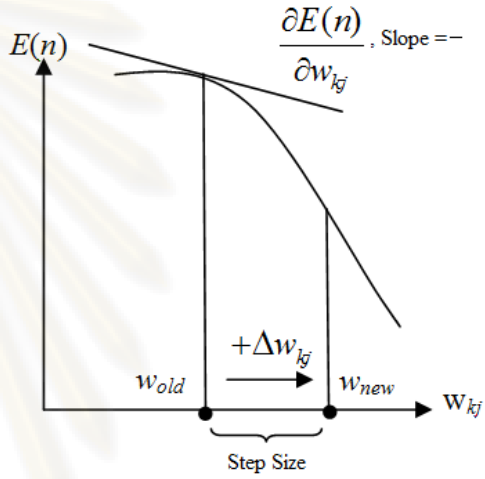


Figure 3.4 (b) In case of decreasing function

Figure 3.4 The graphical relations between $E(n)$ and w_{kj}.

As shown in Figure 3.4, adding minus sign to $\eta \frac{\partial E(n)}{\partial w_{kj}}$ ensures that a direction of the delta weight is opposite to the increasing direction of the error. Furthermore, the weight adjustment process can be considered as a process of moving weights in a direction that decreases the error. The step size of the move is controlled by the learning rate (η). To find the solution of equation 3.10 we start from equation 3.8, differentiating both sides with respect to $e_k(n)$ then we get

$$\frac{\partial E(n)}{\partial e_k(n)} = e_k(n) \qquad 3.11$$

(n) denotes the correspondence to the nth input signal. Differentiating both side of equation 3.6 with respect to $y_k(n)$, we get

$$\frac{\partial e_k(n)}{\partial y_k(n)} = -1 \qquad 3.12$$

Differentiating equation 3.5 with respect to $v_k(n)$, we get

$$\frac{\partial y_k(n)}{\partial v_k(n)} = \varphi_k'(v_k(n)) \tag{3.13}$$

Differentiating equation 3.4 with respect to $w_{kj}(n)$, we get

$$\frac{\partial v_k(n)}{\partial w_{kj}(n)} = y_j(n) \tag{3.14}$$

The use of equations 3.11 to 3.14 in 3.10 yields

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi'(v_k(n))y_j(n) \tag{3.15}$$

Equation 3.15 shows the solution of $\frac{\partial E(n)}{\partial w_{kj}(n)}$. Replacing this solution in equation 3.10, we get

$$\Delta w_{kj} = \eta e_k(n)\varphi'(v_k(n))y_j(n) \tag{3.16}$$

Equation 3.16 can be rewritten in another equivalent form which intends to show the delta weight Δw_{kj} in a term of a local gradient ($\delta_k(n)$).

$$\Delta w_{kj} = \eta \delta_k(n) y_j(n) \tag{3.17}$$

$\delta_k(n)$ is the local gradient of the output node *k* which is defined by

$$\begin{aligned} \delta_k(n) &= -\frac{\partial E(n)}{\partial v_k(n)} \\ &= \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \\ &= e_k(n)\varphi_k'(v_k(n)) \end{aligned} \tag{3.18}$$

Equation 3.17 shows the computation of the delta weight used to update the output layer weights. According to equation 3.17, *k* refers to the related output node and *j* refers to the related hidden node, respectively. The computation of the delta weights for updating the hidden layer weights is similar to equation 3.17, except the local gradient which is computed differently.

$$\Delta w_{ji} = \eta \delta_j(n) y_i(n) \tag{3.19}$$

Equation 3.19 shows the computation of delta weights used to update the hidden layer weights. *j* refers to the related hidden node, *i* refers to the related input node. The local gradient of hidden node $\delta_j(n)$ is computed by

$$\begin{aligned}\delta_j(n) &= -\frac{\partial E(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial y_j(n)}\varphi_j'(v_j(n))\end{aligned} \tag{3.20}$$

From equation 3.8 we have

$$E(n) = \frac{1}{2}\sum_{k=1}^{n} e_k^2(n) \tag{3.21}$$

k refers to the k^{th} output node, n denotes the correspondence to the n^{th} input signal.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_{k=1}^{n} e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} \tag{3.22}$$

Applying chain rule for the partial derivative $\partial e_k(n)/\partial y_j(n)$,we get

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_{k=1}^{n} e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.23}$$

From equations 3.5 and 3.6

$$\begin{aligned}e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n))\end{aligned} \tag{3.24}$$

Hence

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{3.25}$$

From equation 3.4,

$$v_k = \sum_{j=1}^{n} y_j w_{kj} + w_{k0} \tag{3.26}$$

Hence

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.27}$$

By using equations 3.25 and 3.27 in equation 3.23, we get

$$\begin{aligned}\frac{\partial E(n)}{\partial y_j(n)} &= -\sum_{k=1}^{n} e_k(n)\varphi_k'(v_k(n))w_{kj}(n) \\ &= -\sum_{k=1}^{n} \delta_k(n)w_{kj}(n)\end{aligned} \tag{3.28}$$

By using equation 3.28 in equation 3.20, we get the local gradient of hidden node as

$$\delta_j(n) = -\varphi_j'(v_j(n))\sum_{k=1}^{n}\delta_k(n)w_{kj}(n) \qquad 3.29$$

In order to compute the local gradient for both output node and hidden node, it needs the first derivative of an activation function. The derivative form of the activation function will be shown in the next section.

The backpropagation algorithm is an iterative technique. All data (input signal) in a training set will be presented to the network one by one. When all input signals are presented to the network, we called an epoch. The backward phase can be performed in two ways: sequential mode and batch mode. Sequential mode performs the backward phase instantaneously after the forward phase of each input signal (update one by one). Batch mode performs backward phase after all input signals were presented to the network (update epoch by epoch). The sequential mode is more popular than the batch mode, particularly for solving pattern classification problem, for two reasons [1]. First, the algorithm is easy to implement. Second, it provides an effective solution to large and difficult problems.

The learning ability of the network is determined by the error produced during learning. The network's error determine by using the mean squared error shown in equation 3.9. To stop learning process, the error should converge to a sufficiently small value. In practice, one simple and effective stopping criterion used to determine the convergence of the network's error is the absolute rate of change of the mean squared error. This stopping criterion bases on an idea that when the absolute rate of change of the mean squared error per epoch is sufficiently small, the error is assumed to be converged. The rate of change of error is typically considered to be small enough if it lies in the range of 0.1% to 1% per epoch, sometimes a small value like 0.01% is used. The absolute rate of change of the mean squared error per epoch is defined by

$$\left|\frac{E_{av}(m-1)-E_{av}(m)}{E_{av}(m-1)}\right|\times 100 \le \text{a small value defined by user} \qquad 3.30$$

(the popular one is 0.01)

m denotes the m^{th} epoch of the learning process.

3.1.3 Backpropagation algorithm conclusion

As we described, the backward phase of the backpropagation algorithm requires the first derivative of the activation function. In this section we first show the derivative form of the logistic function which is commonly used in the multilayer perceptron network. The logistic function is defined by

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))} \quad a>0 \text{ and } -\infty < v_j(n) < \infty \qquad 3.31$$

$v_j(n)$ is the input of function. It is a linear combination of the input signals (induced local field) corresponding to the neuron node j and the n^{th} input.

Differentiating equation 3.31 with respect to $v_j(n)$, we get

$$\varphi_j'(v_j(n)) = \frac{a\exp(-av_j(n))}{\left[1+\exp(-av_j(n))\right]^2} \qquad 3.32$$

As described in equation 3.2 that $y_j = \varphi_j(v_j(n))$, hence we can express $\varphi_j'(v_j(n))$ as

$$\varphi_j'(v_j(n)) = ay_j(n)[1-y_j(n)] \qquad 3.33$$

Consider the local gradient of the output node k which is defined by

$$\delta_k(n) = e_k(n)\varphi_k'(v_k(n)) \qquad 3.34$$

The error of the output node k is defined by

$$e_k(n) = d_k(n) - y_k(n) \qquad 3.35$$

By using equation 3.33 and equation 3.35 in equation 3.34, we get

$$\delta_k(n) = a[d_k - y_k(n)]y_k(n)[1-y_k(n)] \qquad 3.36$$

Now, consider the local gradient of hidden node j which is defined by

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) \qquad 3.37$$

By using equation 3.33 in equation 3.37, we get

$$\delta_j(n) = ay_j(n)[1-y_j(n)]\sum_k \delta_k(n)w_{kj}(n) \qquad 3.38$$

The back propagation algorithm basically consists of five main processes as follows:

1. The process of initialization. Randomly initialize weights and biases for the network.

2. The presentation of training example. Randomly select a training example (input data or input signal) to present to the network.

3. The forward phase computation.

3.1 Compute the induced local field $v_j(n)$ of all hidden nodes by

$$v_j(n) = \sum_{i=1}^{m} w_{ji}(n) y_i(n) + w_{j0} \qquad 3.39$$

n denotes the correspondence to the n^{th} input signal, y_i is the i^{th} element of the input signal which corresponds to the i^{th} node, w_{ji} is the weight which corresponds to the hidden node j and the input node i, w_{j0} is a bias of the hidden node j, and m is a number of the input nodes in the input layer of the network, not including the bias node.

3.2 Compute the output signals of all hidden nodes j by

$$y_j = \varphi_j(v_j(n)) \qquad 3.40$$

y_j is the output signal of the hidden node j, φ_j is the activation function of the hidden node j, and $v_j(n)$ is the induced local field of the hidden node j.

3.3 Compute the induced local field $v_k(n)$ of all output nodes by

$$v_k(n) = \sum_{j=1}^{m} w_{kj}(n) y_j(n) + w_{k0} \qquad 3.41$$

y_j is the output signal of the hidden node j, w_{kj} is the weight which corresponds to the output node k and the hidden node j, w_{k0} is the bias of the output node k, and m is a number of the hidden nodes in the hidden layer of the network, not including the bias node.

3.4 Compute the output signals of all output nodes k by

$$y_k = \varphi_k(v_k(n)) \qquad 3.42$$

y_k is the output signal of the output node k, φ_k is the activation function of the output node k, and $v_k(n)$ is the induced local field of the hidden node k.

3.5 Compute the error signals of all output nodes by

$$e_k(n) = d_k(n) - y_k(n) \qquad 3.43$$

e_k is the error signal which corresponds to the output node k, d_k is the target signal which corresponds to the output node k, and y_k is the output signal from the output node k.

4. The backward phase computation.

4.1 Compute the local gradients of all output nodes by

$$\delta_k(n) = e_k(n)\varphi_k'(v_k(n)) \qquad 3.44$$

$\delta_k(n)$ is the local gradient of the output node k, $e_k(n)$ is the error signal which corresponds to the output node k, $\varphi_k'(\bullet)$ is the first derivative of the activation function of the output node k, and $v_k(n)$ is the induced local field of the output node k.

4.2 Compute the local gradients of all hidden nodes by

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k^m \delta_k(n)w_{kj}(n) \qquad 3.45$$

$\delta_j(n)$ is the local gradient of the hidden node j, $\varphi_j'(v_j(n))$ is the first derivative of the activation function of the hidden node j, $\delta_k(n)$ is the local gradient of the output node k, $w_{kj}(n)$ is the weight which corresponds to the output node k and the hidden node j, and m is a number of the output nodes in the output layer.

4.3 Updating weights.

4.3.1 Update the adjustable weights of the output nodes by

$$w_{kj}(n+1) = w_{kj}(n) + \alpha(\Delta w_{kj}(n-1)) + \eta\delta_k(n)y_j(n) \qquad 3.46$$

w_{kj} is the weight which corresponds to the output node k and the hidden node j, $w_{k\,0}$ is the bias, α is a momentum (see Notes 1 and 2), Δw_{kj} is the delta weight corresponding to the output node k and the hidden node j, Δw_{k0} is used for updating the bias, η is a learning rate, δ_k is the local gradient of the output node k, and y_j is the output value of hidden node j. In case of $j = 0$, it is the signal from the bias node which is equal to the bias itself.

4.3.2 Update the weights of the hidden nodes by

$$w_{ji}(n+1) = w_{ji}(n) + \alpha(\Delta w_{ji}(n-1)) + \eta\delta_j(n)y_i(n) \qquad 3.47$$

n, α, η are defined as similar as in equation 3.46, w_{ji} is the weight corresponding to the hidden node j and the input node i, $w_{j\,0}$ is the bias, Δw_{ji} is the delta weight corresponding to the hidden node j and the input node i, Δw_{j0} is used for updating the bias, δ_j is the local gradient of the hidden node j, and y_i is the input value corresponding to the input node i. In case of $i = 0$ it is the signal from the bias node.

5. The process of examining the error of the network.

The second, third and fourth processes described above will be repeated until all data (input signal) in the training set are presented to the network (reached an epoch of learning). Then the process of examining the error will start.

5.1 Compute the mean squared error by

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \qquad 3.48$$

5.2 Compute the absolute rate of change of the mean squared error by

$$\left|\frac{E_{av}(m-1) - E_{av}(m)}{E_{av}(m-1)}\right| \times 100 \leq The\ stop\ criterion. \qquad 3.49$$

m is the m^{th} epoch of learning process. If the conditioned-equation 3.49 is true then terminate the learning process; otherwise, repeat steps 2, 3 and 4 until the stopping criterion is met. Sometimes the learning process is terminated by both stopping criterion and a specific number of epochs. This manner ensures that the learning process will be stop anyway. It is possible that the stopping criterion is unable to be met.

Notes 1: equations 3.46 and 3.47 are similar to equation 3.1 ($w_{new} = w_{old} + \Delta w$) by $w(n+1)$ is equal to w_{new}, $w(n)$ is equal to w_{old}, and $\eta\delta(n)y(n)$ is equal to Δw. However, there is a different term which is $\alpha(\Delta w(n-1))$, the previous weight ($\Delta w(n-1)$) multiplied by a momentum (α). Including this term has a stabilizing effect to the convergence of the error, see [1] for full detail of how the momentum works.

Notes 2: The Momentum and the learning rate have to be set in a range of [0,1].

3.2 Decision Tree Algorithm

Decision tree algorithm is a divide-and-conquer technique. Its construction processes can be performed recursively. Information gain is employed to find a proper attribute for branching nodes of a tree in each round of computation. Suppose we have a training data as shown in Table 3.2.

Table 3.2 The Weather data set (nominal version).

Number	Outlook	Temperature	Humidity	Windy	Class
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
3	overcast	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	No
7	overcast	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
12	overcast	Mild	High	TRUE	Yes
13	overcast	Hot	Normal	FALSE	Yes
14	Rainy	Mild	High	TRUE	No

The data set consists of four attributes, two classes of data (Yes or No) and fourteen data. In order to find the information gain of each attribute. We first find the information value of the data set. The information value is used to measure the purity of a data set in a unit called bit. For example,

$$info(\text{weather data set}) = info\,(9,5) \tag{3.50}$$

9 is a number of *Yes* and 5 is a number of *No* in the data set, respectively. For any positive or zero integers p and q, $info(p,q)$ is defined by

$$info(p,q) = entropy(\frac{p}{p+q}, \frac{q}{p+q}) \tag{3.51}$$

where

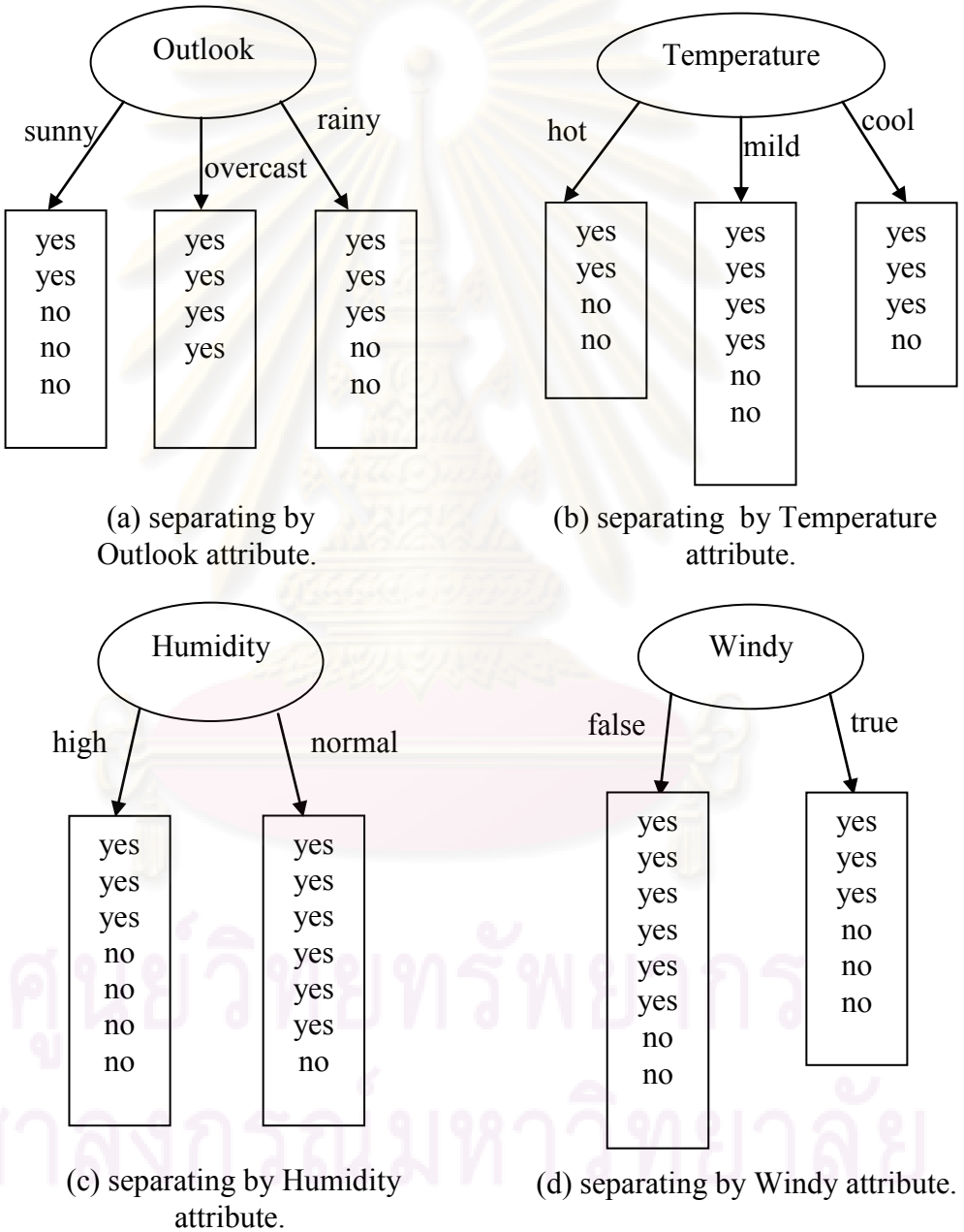
$$entropy(\frac{p}{p+q}, \frac{q}{p+q}) = -\frac{p}{p+q}\log(\frac{p}{p+q}) - \frac{q}{p+q}\log(\frac{q}{p+q}) \tag{3.52}$$

From equation 3.51 and equation 3.52, hence

$$info(9,5) = entropy(\frac{9}{14}, \frac{5}{14}) \tag{3.53}$$

$$entropy(\frac{9}{14},\frac{5}{14}) = -\frac{9}{14}\log(\frac{9}{14}) - \frac{5}{14}\log(\frac{5}{14}) \quad 3.54$$
$$= 0.940 \ bits$$

The data set with a smaller information value is determined as a purer data set. The information value can be applied to determine the purity of data set with respect to only one attribute. Figure 3.5 shows subsets obtained from separating the Weather data set by its attributes.



(a) separating by Outlook attribute.

(b) separating by Temperature attribute.

(c) separating by Humidity attribute.

(d) separating by Windy attribute.

Figure 3.5 The separation of data set with respect to each attribute of the Weather data set.

In order to compute the information value with respect to the outlook attribute, the information values with respect to each possible value of the outlook attribute are required.

The information value with respect to *outlook* = *sunny* is computed by

info(2,3) = 0.971 *bits*

2 is a numbers of *yes* and 3 is a numbers of *no* with respect to outlook = to sunny.

Similarly, the information values with respect to *outlook* = *overcast* and *outlook* = *rainy* are computed by

info(4,0) = 0.0 *bits*

info(3,2) = 0.971 *bits*

The information value of the outlook attribute can be calculated as follows

$$info(outlook) = \frac{5}{14} info(sunny) + \frac{4}{14} info(overcast) + \frac{5}{14} info(rainy) \qquad 3.55$$

$$= \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0.0 + \frac{5}{14} 0.971$$

$$= 0.693 bits$$

According to equation 3.55, the fraction $\frac{5}{14}$ multiplied by *info*(sunny), 5 is a number of data that *outlook* = *sunny*, and 14 is a number of all data in a data set. For the other fractions $\frac{4}{14}$ and $\frac{5}{14}$ can be explained in the same way.

The information values of the other attributes can be computed similarly. The results are as follows.

info(temperature) = 0.911 *bits*

info(humidity) = 0.788 *bits*

info(windy) = 0.892 *bits*

The information gain with respect to attribute A is defined by

gain(A)= *info*(data set) - *info*(A) 3.56

Hence, *gain*(outlook) is computed by

gain(outlook) = *info*(weather data set) - *info*(outlook)

= 0.940-0.693

= 0.247 *bits*

The information gains of others are as follows:

gain(temperature) = 0.029 *bits*

gain(humidity) = 0.152 *bits*

gain(windy) = 0.048 *bits*

The attribute that yields the highest information gain is the outlook attribute. This indicates that separating the data set by the outlook attribute yields the purest subset. Hence it is proper to be used in a branching process.

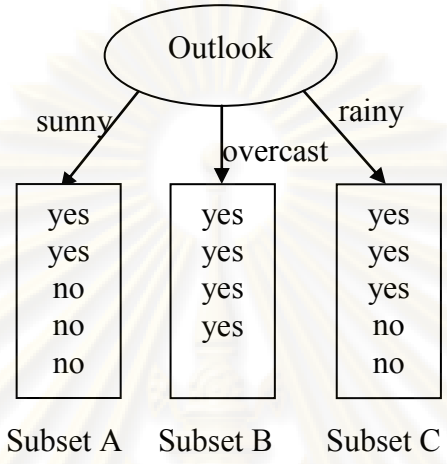


Figure 3.6 Subsets (A, B and C) obtained from separating the Weather data set by the outlook attribute.

Three possible values of the outlook attribute were used to split the Weather data set into three subsets as shown in Figure 3.6. Now, the decision tree consists of one root node with three branches. To create other child nodes, the process of finding the appropriate attribute will be performed again on each of the three separated subsets A, B and C.

The manner of constructing a decision tree described above will be performed continually until the stopping criterion is met. There are two stopping criterions used to terminate the decision tree algorithm.

1. Stop when the separated subsets are pure (there is only one class) such as the subset B in Figure 3.6. There is only class *yes* in this subset, so there is no any further process to perform on this branch.

2. Stop when there is no attribute left to be considered. Any used attribute will not be considered as the candidate attribute again in the process performed on its child nodes. In Figure 3.6, the outlook attribute will not be considered as the candidate attribute in the process performed on the subset A and C.

In pattern classification problem, most data sets contain numeric attributes. In order to extend the information gain to numeric data, the process is slightly different.

In numeric attribute, the data set splitting is restricted to be a binary split and not allowed to split data of the same class. Table 3.3 shows the numeric version of the Weather data set.

Table 3.3 Weather data set (numeric version)

Instance	Outlook	Temperature	Humidity	Windy	Class
1	sunny	85	85	FALSE	No
2	sunny	80	90	TRUE	No
3	overcast	83	86	FALSE	Yes
4	Rainy	70	96	FALSE	Yes
5	Rainy	68	80	FALSE	Yes
6	Rainy	65	70	TRUE	No
7	overcast	64	65	TRUE	Yes
8	sunny	72	95	FALSE	No
9	sunny	69	70	FALSE	Yes
10	Rainy	75	80	FALSE	Yes
11	sunny	75	70	TRUE	Yes
12	overcast	72	90	TRUE	Yes
13	overcast	81	75	FALSE	Yes
14	Rainy	71	91	TRUE	No

This data set contains two nominal attributes: Temperature and Humidity. Now, we will show the computation of *gain*(temperature).

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	no	yes	yes	no
						yes	yes				

The temperature data are sorted by value, and the repeated values (72, 75) are collapsed together. There are 8 possible splitting areas as marked by arrows. A splitting point can be any value lying in the boundary of the splitting areas. Such as in the first splitting area which lies in range of [64, 65], the splitting point can be easily chosen by using a mean value of its boundary which is 64.5. So, according to the temperature attribute, we have 8 possible splitting points. These splitting points are candidates for the proper splitting point. The information value of each splitting point is computed based on the information value of its left side and right side as follows

$$info(temperature) = \frac{p}{p+q} info(left\ side) + \frac{q}{p+q} info(right\ side) \qquad 3.57$$

p is a number of data on the left side and q is a number of data on the right side. The splitting point that gives the lowest information value will be determined as the

information value of the temperature attribute. Similarly to the nominal attribute, the information gain of the numeric attribute is computed by

gain(the selected attribute) = *info*(weather data set) – *info*(the selected attribute) 3.58

For example, *gain*(temperature) can be computed as follows. First step is the computation of *info*(temperature). As we have eight possible splitting points, so we will get eight possible information values. The computation of the information value of the fourth splitting point which is 71.5 is computed by.

info(left side of 71.5) = *info*(4,2)

4 is a number of *yes* and 2 is a number of *no* on the left side of the splitting point.

Similarly, the information of the right side is computed by

info(right side of 71.5) = *info*(5,3)

Hence

$$info(71.5) = \frac{6}{14} info(4,2) + \frac{8}{14} info(5,3)$$
$$= 0.939 \ bits$$

After getting all information values, the lowest one will be used to represent the *info*(temperature). When getting the *info*(temperature), now we can compute the *gain*(temperature) by using equation 3.58. The *info*(weather data set) is computed in the same way as describe in the computation of the nominal attribute.

Note: the important differences between a numerical splitting and a nominal splitting are a number of splitting. The numerical splitting is restricted to be a binary split while a number of splitting of the nominal splitting depends on a number of the possible values of that nominal attribute. Additional, the used numeric attribute can be used again in the process of finding the appropriate attribute performed on its child nodes while we cannot do this in a nominal attribute.

3.3 The information gain as a tool to identify the difficult-to-classify and easy-to-classify data

In this research, we use the information gain as a rapid tool to distinguish between difficult-to-classify and easy-to-classify data. We apply a new parameter to control a minimum number of data in subspaces called minimum instances per area. If a number of data in subspace obtained from splitting by a hyperplane is less than this

parameter, the process will remove that hyperplane and stop the constructing process on that subspace. Figure 1 shows the graphical view of the applied hyperplanes on the Iris data set [15].

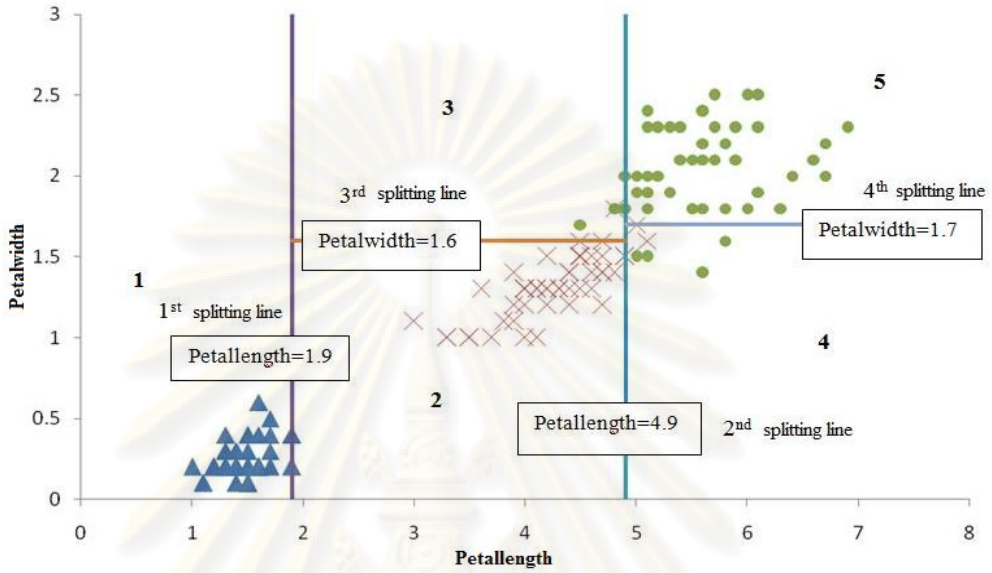


Figure 3.7. The graphical view of the applied hyperplanes

For the purpose of demonstration in 2-dimensions, we have removed two out of four attributes of the Iris data set. Three different symbols represent the different classes. As shown in Figure 1, data space is partitioned by hyperplanes into five subspaces. Subspace number 1, 2 and 5 are pure subspaces. Data fallen in this areas are correctly classified corresponding to its target class. We call data located in these subspaces as easy-to-classify data. For subspace number 3 and 4, they are impure subspaces. The data fallen in these areas are more difficult to classify when compared with the data located in the pure subspaces. How difficult to classify these data depends on the impurity of the subspaces they fallen in. We include the entropy measurement (for more detail of the entropy, see [10] and [17]) to determine the level of impurity. Furthermore, we add a parameter called impurity threshold to use for identifying the difficult-to-classify data from the impure subspaces. If the entropy of the current-determined subspace is higher than the impurity threshold, we identify data located in this area as the difficult-to-classify data.

3.4 Duplicate-sampling of difficult-to-classify scheme

In each epoch of the standard BP learning, each datum will be used only once to present to the network. Our proposed scheme emphasizes the learning of the

difficult-to-classify data by presenting them to the network more often than usual. Increasing the presentation of the difficult-to-classify data is performed by duplicating them before starting the learning process. Consequently, the complete process of training the network by the BP learning with the proposed scheme can be illustrated as follows:

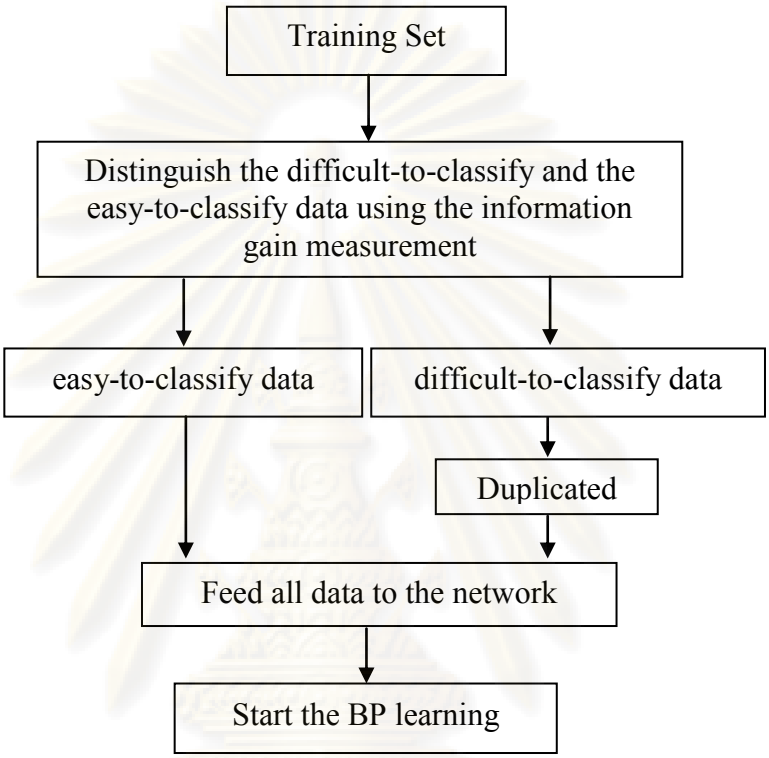


Figure 3.8 The flow chart of training the MLP network by the BP with the duplicate-sampling of difficult to classify scheme.

CHAPTER IV
THE EXPERIMENT RESULTS

The experiments were divided into three categories, depending on their different objectives. The first experiment was set up to compare the computational time and the classification performance between the MLP network trained by the standard BP algorithm and the MLP network trained by the BP algorithm with duplicate-sampling of difficult-to-classify scheme. The second experiment was set up to show an effect to the classification performance from adjusting parameters of the proposed scheme. Finally, the third experiment was set up to compare the classification performance of the MLP network trained by the proposed scheme with the well-known classification algorithms, including support vector machine, RBF network, J48 decision tree, NaïveBayes and Nearest Neighbor.

Data sets used in these experiments were taken from UCI Repository [15]. The details of all data sets are shown in Table 4.1.

Table 4.1 Details of data sets used in the experiments.

Name	Data	Features	Classes	Training set/Testing set
Liver	345	6	2	227/118
Diabetes	768	8	2	506/262
Iris	150	4	3	99/51
Heart-Statlog	270	13	2	178/92
Ionosphere	351	34	2	231/120
Vehicle	846	18	4	558/288
Balance-scale	625	4	3	412/213
Vowel	990	10	11	653/337

4.1 MLP Network Trained by the BP Algorithm compare with MLP Network Trained by the Duplicate-sampling of difficult-to-classify scheme

In this research, we concentrate on the classification of a numeric data set. So, we removed two nominal attributes from the Vowel data set before using it. The data sets were split into training and test set in a ratio of 2:1. We performed two processes to each experiment.

1. Process of finding an optimal number of hidden nodes. This process aim to find the best number of hidden nodes for each data set. We construct the one-hidden layer MLP networks with a various number of hidden nodes ranging from one to thirty. Then, we train all of them, using the training set, with the

standard BP algorithm by fixing the parameters as follows: a constant of the logistic function = 0.5, learning rate = 0.3 and number of epochs = 1000. The number of hidden nodes that gives the highest accuracy on the test set is determined as the optimal number for the corresponding data set. The results of this process are shown in Table 4.2.

Table 4.2 The optimal number of hidden nodes corresponding to each data

Data set	Number of optimal hidden nodes
Liver	6
Diabetes	1
Iris	2
Heart-Statlog	9
Ionosphere	2
Vehicle	14
Balance-scale	12
Vowel	23

2. Process of training and testing. We used an optimal number of hidden nodes from the first process to construct two copies of the networks. One was trained by the standard BP algorithm, and another one was trained by the BP algorithm with the duplicate-sampling of difficult-to-classify scheme. We temporarily stopped the learning process at various points, every 250 epochs. At each stopping, we applied the test set to the network. The learning process was completely stopped when the classification accuracy showed no improvement while the network's error on training set was dropping. The results of this process were shown in Figure 4.1.

The parameters of the proposed scheme were set as follows: a number of duplication = 1, minimum instance per area = 4 and the impurity threshold = 0.2. According to the more popular sequential mode of BP algorithm over the batch mode in solving pattern classification problem [1], so we used the sequential mode in our experiment.

For figure 4.1, the x-axis denotes a number of epochs, and the y-axis denotes a percentage of accuracy on the test set. The continuous line represents the accuracy of the network trained by the standard BP, and the dotted line represents the accuracy of the standard BP with the duplicate-sampling of difficult-to-classify scheme.

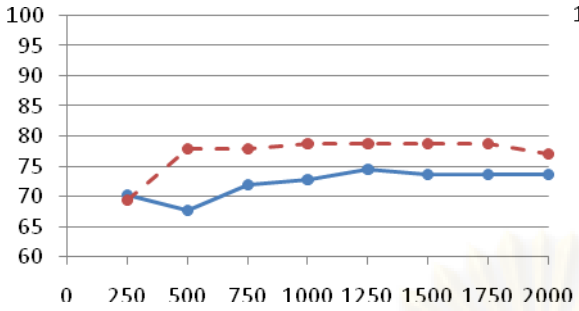


Figure 4.1.1 Liver data set result.

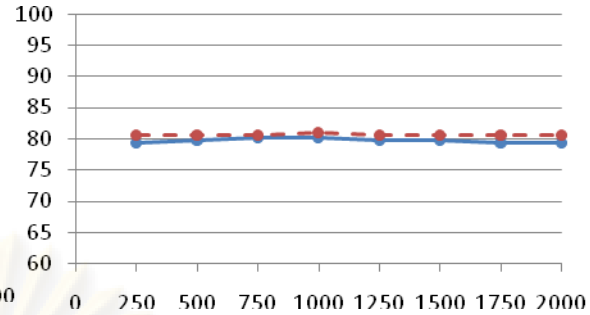


Figure 4.1.2 Diabetes data set result.

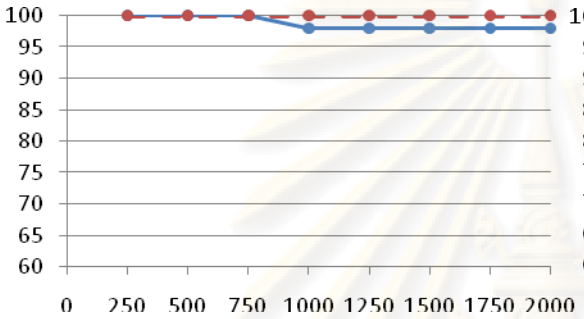


Figure 4.1.3 Iris data set result.

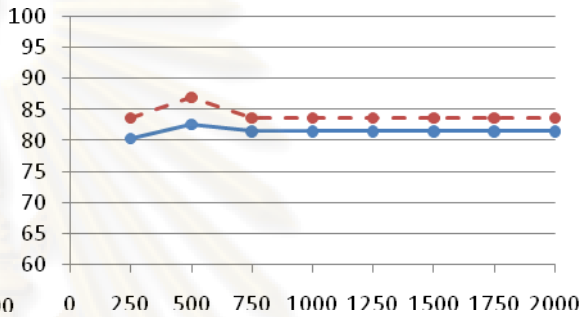


Figure 4.1.4 Heart-Statlog data set result.

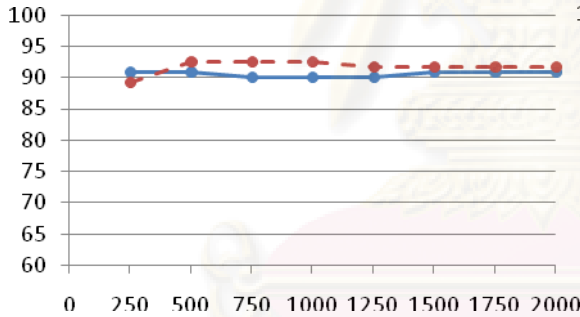


Figure 4.1.5 Ionosphere data set result.

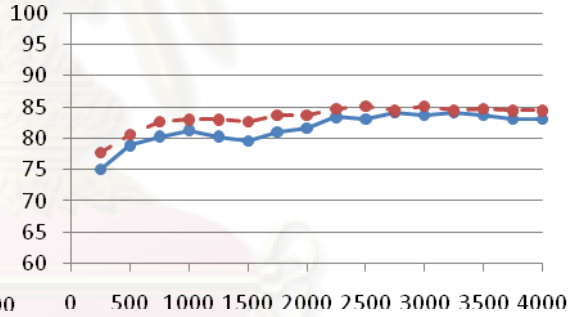


Figure 4.1.6 Vehicle data set result.

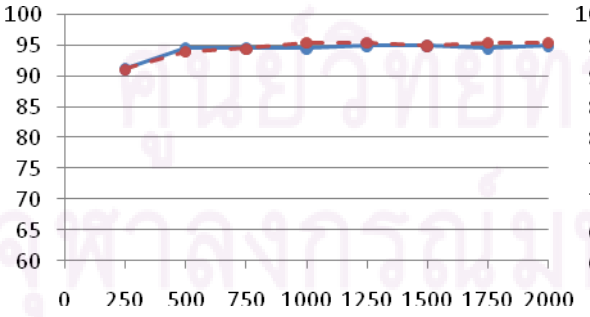


Figure 4.1.7 Balance Scale data set result.

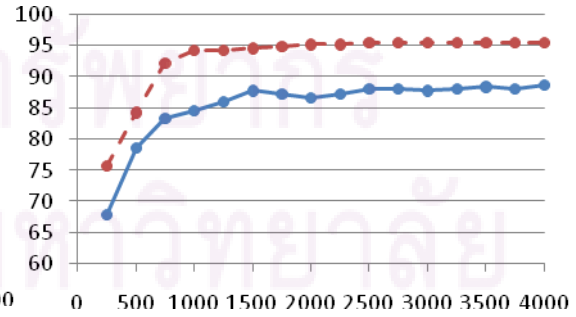


Figure 4.1.8 Vowel data set result.

Figure 4.1 The experimental results.

We examine the highest performance of the standard BP compared with the BP with the proposed scheme. If the BP with the proposed scheme achieves the same or higher accuracy by requiring less number of epochs, we assume that it shows a computational time improvement. Since the number of data per epoch of the BP with the proposed scheme is larger than the standard BP, we will assume the improvement when the actual computation time also shows the improvement. The comparisons of the performance of both techniques are shown in Table 4.3 and 4.4. Table 4.3 shows the comparison of the best accuracy of the standard BP and the similar or higher accuracy of the BP with the proposed scheme. According to this table, it shows the computational time improvement occurring on 6 data sets: Liver, Diabete, Heart-Statlog, Vehicle, Blance-Scale and Vowel. Only two data sets (Iris and Ionosphere) use a little longer computational time. Table 4.4 shows the comparison of the best classification performance of both techniques. From this table, it shows the classification performance improvement occurring on 7 data sets, except the Iris data set which shows the same accuracy at 100%. In table 4.5, the details of the number of difficult-to-classify data and time used in identifying them are shown corresponding to each data set.

Table 4.3 The comparison of the best accuracy of BP and the similar or higher accuracy of the BP with the propose scheme.

	Standard BP			**BP with the proposed scheme**		
	Best accuracy	**Epochs**	**Actual time**	**Similar or higher accuracy**	**Epochs**	**Actual time**
Liver	74.58	1250	9.05	77.97	500	5.31*
Diabetes	80.15	750	11.047	80.53	250	5.266*
Iris	100	250	0.938	100	250	1.141
Heart-Statlog	82.61	500	23.172	83.7	250	18.06*
Ionosphere	90.83	250	30.28	89.17	250	31.4
Vehicle	84.03	2750	794.24	83.68	1750	649.35*
Balance-Scale	94.84	1250	97.813	95.31	1000	85.49*
Vowel	89.61	5000	2018.78	92.28	750	607.7*

Note:

1. * denotes the improvement of computational time.
2. Time is measured in seconds.
3. The actual time of the BP with the proposed scheme is already included the time used in finding the difficult-to-classify data.

Table 4.4 The comparison of the best accuracy of BP and the BP with the proposed scheme.

	Standard BP			BP with the proposed scheme		
	Best accuracy	**Epochs**	**Actual time**	**Best accuracy**	**Epochs**	**Actual time**
Liver	74.58	1250	9.05	78.81*	1000	10.17
Diabetes	80.15	750	11.047	80.92*	250	5.266
Iris	100	250	0.938	100	250	1.141
Heart-Statlog	82.61	500	23.172	86.96*	500	35.9
Ionosphere	90.83	250	30.28	92.50*	500	62.79
Vehicle	84.03	2750	794.24	85.07*	2500	916.43
Balance-Scale	94.84	1250	97.813	95.31*	1000	85.49
Vowel	89.61	5000	2018.78	95.55*	2500	2007.36

Note: * denotes the improvement of classification performance.

Table 4.5 The details of the number of difficult-to-classify data and the used time.

	Number of difficult-to-classify data		Used time
Liver	90	(39.65% of training data)	0.2
Diabetes	146	(28.85% of training data)	0.62
Iris	14	(14.14% of training data)	0.03
Heart-Statlog	42	(23.60% of training data)	0.02
Ionosphere	19	(8.23% of training data)	0.81
Vehicle	190	(34.05% of training data)	1.58
Balance-Scale	135	(32.77% of training data)	0.22
Vowel	402	(61.56% of training data)	1.85

4.2 Parameters Adjustment of Duplicate-Sampling of Difficult-to-Classify Scheme

This section aims to show the effect to the classification performance from adjusting the parameters of the proposed scheme. The duplicate-sampling of difficult-to-classify scheme consists of three adjustable parameters which are a minimum instance, a purity threshold and a duplication rate. The definitions of each parameter are defined as follows:

1. The minimum instance is a least number of data allowed to be split into a subspace by the plane of the duplicate sampling scheme.

2. Purity threshold is a least information value criterion used to determine whether the data in the mixed sub-domain are difficult-to-classify or not. If the considered subspace has a higher or equal information value when comparing with the threshold, the data in that subspace will be determined as the difficult-to-classify data.

3. The duplication rate is a number of duplications of the difficult-to-classify data.

We performed this experiment by analyzing the parameters one by one. During one parameter is being varied, the others are fixed. The results obtained from this experiment will be compared with the results in the previous experiment which we used the default parameter setting (minimum instance = 4, purity threshold = 0.2 and duplication rate = 2). Varying parameter was performed by:

1. The minimum instance was varied from 4 to 20, increasing by 4.
2. The purity threshold was varied from 0.2 to 1, increasing by 0.2.
3. The duplication rate was varied from 1 to 5, increasing by 1.

The Ionosphere data is selected to use in this task, and the obtained results are shown in Table 4.6.

Table 4.6 The classification performances obtained from varying the minimum instance.

Number of Epochs	**Minimum Instance**				
	4(default)	**8**	**12**	**16**	**20**
250	89.17	93.33	94.17*	90.83	88.33
500	92.5*	93.33	92.50	91.67	90.00
750	92.5*	93.33	91.67	90.83	93.33
1000	92.5*	93.33	91.67	93.33*	95.00
1250	91.67	93.33	91.67	92.50	94.17*
1500	91.67	94.17*	91.67	91.67	94.17*
1750	91.67	94.17*	91.67	91.67	94.17*
2000	91.67	94.17*	91.67	91.67	94.17*
Percentage of Difficult-to-classify data	8.23%	29.00%	38.53%	43.72%	46.75%

Note:

1. the ones marked by * refer to the highest value related to each varied value of the minimum instance.
2. the percentage of difficult-to-classify data are estimated with respect to a number of data in a training set.

Table 4.7 The classification performances obtained from varying the purity threshold.

Number of Epochs	Purity Threshold				
	0.2(default)	0.4	0.6	0.8	1
250	89.17	89.17	89.17	86.67	88.33
500	92.5*	92.5*	92.5*	90	91.67*
750	92.5*	92.5*	92.5*	90	91.67*
1000	92.5*	92.5*	92.5*	90.83*	91.67*
1250	91.67	91.67	91.67	90.83*	91.67*
1500	91.67	91.67	91.67	90	91.67*
1750	91.67	91.67	91.67	90	91.67*
2000	91.67	91.67	91.67	90	91.67*
Percentage of Difficult-to-classify data	8.23%	8.23%	8.23%	3.46%	1.73%

Note: the ones marked by * refer to the highest value related to each varied value of the purity threshold.

Table 4.8 The classification performances obtained from varying the duplication rate.

Number of Epochs	Duplication Rate				
	1(default)	2	3	4	5
250	89.17	87.5	89.17	89.17	86.67
500	92.50*	90*	86.67	86.67	89.17
750	92.50*	89.17	87.5	87.5	88.33
1000	92.50*	89.17	86.67	86.67	90.83*
1250	91.67	89.17	88.33	88.33	90.83*
1500	91.67	89.17	90*	90*	89.17
1750	91.67	89.17	90*	90*	88.33
2000	91.67	89.17	90*	90*	87.5

Note: the ones marked by * refer to the highest value related to each varied value of the duplication rate.

According to Table 4.6, by increasing the minimum instance, a number of the difficult-to-classify data will be increased. From the obtained results, they show that when a number of difficult-to-classify data increases, the classification performance of the trained network seem to be improved.

According to Table 4.7, the obtained results show that when the purity threshold is increased, a number of difficult-to-classify data will be decreased and the classification performance is subsequently decreased.

According to Table 4.8, when the duplication rate is increased, the classification performance is dropped. A high number of duplication rate could change the distribution of the training data and yields as a worse predictive performance of the network.

We would like to present the parameters of the duplicate-sampling of difficult-to-classify scheme as a tool to increase or decrease a number of difficult-to-classify data. There is no a certain number of difficult-to-classify data to guarantee the improvement of classification performance. However, adjusting theses parameters could help validate the classification performance of a trained network until the satisfactory performance is met.

4.3 Comparing with the Well-Known Algorithms

The algorithms involved in benchmarking are the normalized Gaussian radial basis function network, the C4.5 decision tree, Naive Bayes classifier using estimator classes, Nearest-neighbor-like algorithm, a support vector machine trained by the John Platt's sequential minimal optimization algorithm. To implement these algorithms, we used Weka software (version 3.4). For the C4.5 decision tree, the Naïve Bayes classifier and the Nearest-neighbor algorithm we used the default parameters setting. For the RBF network we set a number of clustering equal to a number of classes in the data sets. Finally, for the SVM we varied the exponent used in a polynomial kernel function (from 1to10, increasing by one), then selected the best performance. The experimental results are shown in Table 4.9.

Note: the training and test sets used in the experiment are same as the ones used in the previous experiments.

Table 4.9 The classification performances obtained from various classification algorithms.

Data Sets	RBF Network	J48-Tree	Naïve Bayes	Nearest Neighbor	SVM	MLP trained by the BP	MLP trained by the proposed scheme
Liver	74.57	72.03	77.12	65.25	61.02	74.58	78.81*
Diabetes	77.10	76.72	77.48	74.81	79.77	80.15	80.92*
Iris	100.00*	98.04	98.04	98.04	98.04	100.00*	100.00*
Heart-Statlog	86.96*	72.83	85.87	76.09	84.78	82.61	86.96*
Ionosphere	94.17	87.50	86.67	95.00*	92.50	90.83	94.17
Vehicle	68.06	70.83	40.28	54.86	79.51	84.03	85.07
Balance-Scale	97.65	92.02	91.08	100.00*	95.31	94.84	95.31
Vowel	86.35	77.15	67.66	83.09	93.47	89.61	95.55*

Note: * refer to the best classification performances related to each data set.

From the results shown in Table 4.9, the MLP network trained by the Duplicate-sampling of difficult-to-classify scheme achieves the highest accuracy in five data sets. However, maybe, the classification performances obtained from the benchmarked algorithms are not their best. But, we would like to point out that the MLP network trained by the Duplicate-sampling of difficult-to-classify scheme is better than the standard BP algorithm and achieves a high level of classification performance.

CHAPTER V
CONCLUSION

In the structure of the MLP network, each of non-linear neuron can be referred to a hyperplane in a decision space [1]. According to this view point, the trained MLP network classifies the unseen data by examining their positions in the decision space compared with the positions of the hyperplanes. The BP learning can be considered as a method to iteratively adjust positions of the hyperplanes. The purpose of adjustment is to make those hyperplanes separate the training patterns drawn from different classes. In this research, we propose a technique to reduce the computational time of the BP learning called duplicate-sampling of difficult-to-classify scheme.

The proposed scheme bases on a concept of difficult-to-classify and easy-to-classify data. The difficult-to-classify data in our view point are the data that requires more number of epochs to be recognized by the network when compared with the easy-to-classify data. We utilize the information gain to distinguish these two types of data. Then, we duplicate the difficult-to-classify data while do nothing to the difficult-to-classify data. Consequently, during BP learning, the difficult-to-classify data will be emphasized by the network.

The experiments were performed on eight data sets taken from UCI repository. According to the obtained results, the experiments performed on six out of eight data sets (Liver, Diabete, Heart-Statlog, Vehicle, Blance-Scale and Vowel) show the learning time improvement. For the classification performance, the improvement occurs on seven data sets, except the Iris data set which shows the same accuracy at 100%. Our conclusions on the duplicate-sampling of difficult-to-classify scheme are as follows:

1. The proposed technique makes the BP algorithm require less number of epochs in learning.
2. The MLP network trained by the BP algorithm with the duplicate-sampling of difficult-to-classify scheme achieves a higher or equal predictive ability when compared with the standard BP.

Future Work

In this research, we restrict our study to numeric data. However, in the real world problems, many data sets contain nominal attributes. The extending of the duplicate-sampling of difficult-to-classify scheme to the nominal attributes is an interesting direction to be addressed. Furthermore, the study of applying the proposed scheme to other classifiers as well as a problem of using a new measure such as the gain ratio, gini index etc., to distinguish data are interesting issues to be investigated.

REFERENCES

[1] Haykin S. Neural networks: A comprehensive foundation. 2nd ed. Upper Saddle River, NJ.: Prentice-Hall, 1999.

[2] Jacobs, R. A. Increased rates of convergence through learning rate adaptation. Neural networks 1(1998): 295-307.

[3] Solomon, R., and Van Hemmen, J. L. Accelerating backpropagation through dynamic self-adaptation, Neural networks 9(1996): 589-601.

[4] LeCun, Y. Efficient learning and second-Order methods, A Tutorial at NIPS 93. Denver, 1993.

[5] Stone, M. Cross-validation: A review, Mathematische operationforschung statistischen: Serie statistics 9(1978): 127-139.

[6] Morgan, N., and Bourlard, H. Continuous speech recognition using multilayer percptrons with hidden Markov models, IEEE International Conference on Acoustics, Speech, and Signal Processing. 1(1990): 413-416.

[7] Suresh, S., and Omkar, S. N., and Mani, V. Parallel implementation of back-proppagation algorithm in networks of workstation, IEEE Trans, on Parallel and Distributed Systems. 24-34 Jan., 2005.

[8] Thammano, A., and Meengen, A. A New evolutionary neural network classifier, 9th PAKDD Conference. 18-20 May., Hanoi: Vietnam, 2005.

[9] Quinlan, J. R. Machine learning. Vol. 1. Sydney: University of Sydney, 1975.

[10] Quinlan, J. R. C4.5 programs for machine learning. San Mateo, CA.: Morgan Kaufmann, 1992.

[11] Kent, J. T. Information gain and a general measure of correlation, Biometrika. 70,1 (1983): 163-173.

[12] Mackay, D. J. C. Information-based objective functions for active data selection, Neural computation. 4(1992): 590-604.

[13] Borland, L., and Plastino, A. R., and Tsallis C., Information gain within nonextensive thermostatistics, J.Math.Phys. 39(1998), 6490-6501.

[14] Han, j., and Kamber, M. Data mining concepts and techniques. Morgan Kaufmann, 2001.

[15] Blake, C. L., and Merz, C. J. UCI repository of machine learning database [online] Available from: http://www.ics.uci.edu/~mlearn/MLRepository.html [15/01/2007].

[16] Wesley, H. I. Matlab Supplement to Fuzzy and Neural Approaches in Engineering. John Wiley & Sons, 1997.

[17] Witten, I. H., and Frank, E., Data Mining Practical Machine Learning Tools and Techniques. Elsevier Publishing Company, 2005.

CURRICULUM VITAE

Mr. Parinya Weangsamoot was born in November 12, 1980, in Roi-Et, Thailand. He received a bachelor degree in Medical Science from the Department of Medical Science, Faculty of Medicine, Khonkean University, Thailand in 2002.