APPLICATIONS OF NEURAL NETWORKS IN THE BINARY CLASSIFICATION PROBLEM

By

Chan Pak Kei, Bernard

A Thesis
Submitted in partial fulfillment of the requirements for
the degree of Master of Philosophy

Department of Systems Engineering & Engineering Management
The Chinese University of Hong Kong

June 1997

Abstract

The binary classification problem classifies observations into two predetermined groups. Many real-life problems are applications of the binary classification problem. The literature has suggested a single hidden-layer backpropagation feedforward neural network is a consistent universal classifier for the binary classification problem. However, there are still limitations in applying the neural network approach.

In this research, we address two limitations: data dependency and network size. For the data dependency problem, we propose a hybrid model. This approach uses a data handler to separate the data into good and bad data before the data are fed for training or operating the networks. By training two independent networks using two groups of data, we are able to better describe the distribution space of the corresponding data sample with two different functions. The computational result shows that the hybrid approach improves the accuracy of the classification.

The network size is often decided by the trial-and-error test. This makes the application of a neural network very tedious. There are some network architecture altering methods. In this research, we propose some methods that combine the advantages of these network architecture altering methods. By evaluating the performance of generalization ability, time consumption and resulting network size, we find that methods that adopt the properties from pruning and constructive algorithms always produce a better network.

Acknowledgments

I would like to show my deepest gratitude to my advisors, Professor Chun-Hung Cheng and Professor Boon-Toh Low, for their guidances and suggestions. Without their support, this thesis would not be completed.

Moreover, I would like to thank all my colleagues and friends who made my post-graduate life joyful and memorable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The binary classification problem is a special case of the classification problem, and has been continuously studied in the literature. Many practical applications of the binary classification models include bankruptcy data analysis, medical diagnosis, signal processing, etc. Typically, many research studies solve the binary classification problem using different discriminating approaches.

Neural networks are used to solve the binary classification problem in this thesis. Neural networks have been shown to be very effective in solving some practical binary classification applications[1, 2, 3, 4, 5]. Yet, there are several limitations in applying neural networks. In this research, we try to address some of these limitations and improve the performance of neural networks.

The hybrid classification model is developed to minimize the vagueness effect introduced by the noise data. By using two different discriminating functions to

describe the distribution space of the data, the generalization performance can be improved. The model is implemented, and tested using the data of breast mass cytology. It is also tested using the synthetic data, which contains a large amount of noise data, so as to shows that the hybrid classification model performs better in a noisy environment than the conventional method. Furthermore, to study the problem in automatically searching for the suitable network architecture, we tested the network architecture altering methods using the databases of breast mass cytology and tic-tac-toe.

1.2 Classification Approaches

Typically, there are two approaches to study classification problem : supervised learning and unsupervised learning. Supervised learning is a method which is given with certain classes of observations and establishes rules to classify a new observation into one of the existing classes[6]. While, unsupervised learning is a method which establishes the existence of classes for a given set of observations[6]. In this research, we study approaches in supervised learning.

To be more precise, the classification problem using supervised learning requires a finite number N of classes, $c_1, \ldots, c_N$[7], where N is greater than one. Each observation consists of a set of attributes, X, and is assigned to one and only one of the classes according to its attributes. The application domains of the classification problem include character recognition, speech understanding, medical diagnosis, process fault detection, managerial decision making, and financial decision making[1, 2, 3, 4, 7, 8].

In this research, we concentrate on the binary classification problem rather than the multi-class classification problem. The binary classification problem is a subset of the classification problem, where observations are assigned to one of the two categories only(i.e., $N = 2$). The problem is widely applied in different application domains, such as, bankruptcy prediction in financial forecasting, exclusive-OR(XOR) problem in signal processing, risk analysis in managerial decision forecasting and breast-mass identification in medical diagnosis[1, 2, 3, 4, 5, 9].

1.3 The Use of Neural Network

The performance of the classification process depends on how well the discriminating function for the entire classification problem performs. A discriminating function is developed to minimize the misclassification rate, based on some given samples of input and output vector couples that are referred to as "training data set". This discriminating function is then used for classifying new observations into previously defined groups and for testing the generalization performance. However, one should know that an accurate performance on training data set may not intuitively lead to an accurate prediction on the unseen observations(testing data set)[6].

There are three main approaches to tackle the classification problem : statistical approaches, machine learning approaches and neural network approaches. Statistical approaches are the traditional methods in the classification problem. They are generally characterized by having an explicit underlying probability model, which provides a class probability rather than simply a classification[6]. The most widely

used statistical approaches are the Fisher's linear discriminant model[1, 6] and its extensions, multivariate[10] and univariate[11] discriminant analysis. These methods have some restrictions of the linear prediction manner, assumptions of multivariate normal distribution, identical covariance matrices of each class, and known mean vectors, covariance matrices, prior probabilities and misclassification costs[1]. Other statistical approaches include logistic regression and k nearest neighbour.

Machine learning approaches improve their performance automatically in a stand-alone manner by learning process, rather than the programs that built up solely upon the analysis by programmer. Decision tree methods commonly used in machine learning approaches. ID3 and C4.5 are the typical methods[1, 6, 12]. Decision tree methods are the family of symbolic data analysis algorithms. The classification procedure in decision tree is based on recursive partitioning of the sample space[6]. The experiments in [6] show that all decision tree methods always perform the same. And their performance on the multi-modal data will be better when compared to classical statistical methods.

Neural network approaches are also a kind of machine learning approaches. More than a typical machine learning approach, neural network approaches have adopted some statistical techniques and the properties of a stand-alone system of machine learning approaches[6]. These approaches act with a similar behaviour to networks of neurons in brain. Several advantages in applications such as, learning from experience, generalizing from examples, extracting essential information from noisy data, developing solutions fast once appropriate network design is adopted, adaptability to different domains, computational efficiency when operation, and non-linearity prop-

erties that applicable to complex and real world situations can be achieved[13]. Many research studies show that neural network approaches are more reliable than other methods[1, 5, 14].

1.4 Motivations

Although several research studies suggest that neural network approaches have a higher classification ability[1, 5, 14] than many other methods, the predictive capability of these approaches still has potential for further improvement. In the following paragraphs, we discuss some limitations of applying neural networks.

Data Dependency

The training processes of neural networks are very sensitive to the training data set[8]. The training data may affect a network through the different ratio between the number of observations with different classes[5], and the noise. A trained network is more sensitive to some specific classes if the training data set consists of a larger ratio of observations in these classes than the other. And also, although neural network approaches can extract essential information from noisy data, their performance may be affected by having too many noisy data in the training data set[15].

Unknown Network Size

Before a training to a neural network can be processed, we often need to arbitrarily decide the network size first. We often need to perform trial-and-error tests on the

network size. A small network may not achieve the desirable prediction ability, but an oversized network may also suffer from the problem of memorizing and overfitting[6, 12, 16].

Unknown Parameter Setting

The proper choices of the rate parameters, such as step size of gradient method, learning rate, network momentum for neural networks are also important. Too small may lead to long training time. Too large may result in an unstable training and poor solution network[12, 17]. Generally, the choices of the rate parameters depend on the experience and must be varied from different problems. Thus, it is often a time consuming and trial-and-error process.

When to Stop Training

It is difficult to determine when to stop a training. Typically, the training will be terminated when an acceptable misclassification rate on the training data set is obtained. However, the level of this misclassification rate is set arbitrarily. On one hand, a large misclassification tolerance rate will probably lead to a bad solution network. On the other hand, a small misclassification tolerance rate will result in an overfitting solution network.

Initial Weight Dependency

A successful training process that produces a solution network depends on the initial weights[18, 19]. Poor initial weights often make a solution network trapped in

a local minimum and hence achieve a sub-optimal performance or never obtain an acceptable solution network. Also if we start with all the weights equal to zero or any single number, the network will not be trainable[18]. Unfortunately we do not have any rules for setting the initial weights that will definitely lead to a successful training.

Unreadability

The distributed nature of the knowledge representation in a neural network is often unreadable[6, 20]. It is extremely difficult for the user to understand the knowledge that represented by a trained neural network by just studying its connection weights and thresholds[20].

1.5 Organization of Thesis

In this research, we address some limitations of applying neural networks to the classification problem. By addressing the limitations, we are able to improve the performance of neural network approaches. Furthermore, a single hidden-layer back-propagation feedforward neural network is used in this research as it is an universal consistent classifier for the binary classification problem[21, 22, 23, 24, 25]. The remaining chapters in the thesis are organized as follows :

Chapter 2 : Related Work

Chapter 2 introduces the existing work in neural network approaches related to the classification problem. The topology of a neural network, the techniques that contribute to the hybrid classification model, and the previous studies in network architecture altering approaches are explained in this chapter.

Chapter 3 : Hybrid Classification Systems

To deal with the noisy data dependency problem, in neural network approaches, we propose a hybrid classification approach. The hybrid model is composed of fuzzy linear regression with fuzzy intervals analysis(FLRFIA) and neural networks. The FLRFIA works as a data handler and separates the data sample into two groups : good data and the bad data, before the training and operation begin. By training two independent neural networks with these two groups, we can better describe the distribution space of the corresponding data sample with two different discriminating functions. The result shows that this hybrid approach improves the accuracy of the classification.

Chapter 4 : Searching for Suitable Network Size Automatically

Typically, the desirable network size is determined through trial-an-error tests. A certain network size is chosen because it gives the most promising generalization performance in the trial-and-error tests. In spite of the simplicity, it is a very tedious work. Recent studies suggest the use of the network architecture altering approaches : pruning algorithms and constructive algorithms. These methods allow the network

architecture being altered during the training. In Chapter 4, we introduce the modified constructive methods, emphasizing the integration of pruning and constructive methods. In order to evaluate the performance, the fixed architecture network method and pruning methods are also tested and evaluated with respect to the generalization ability, time complexity and the obtained network size.

Chapter 5 : Summary of Thesis

This chapter gives a review, discussion and conclusion to this research. And also, the possible future extensions are included.

Chapter 2

Related Work

2.1 Overview

In this research, the neural network approaches are applied to the binary classification problem. In particular we use a backpropagation feedforward neural network as a classifier. As discussed in the last chapter, there are several limitations in applying a neural network. We propose a data preprocessing approach to deal with the data problem, and approaches to search for the desirable network architecture. In this chapter, we will review a backpropagation feedforward neural network, and the training process. Approaches related to data preprocessing and searching for the desirable network architecture will also be reviewed.

2.2 Neural Network

Unlike conventional data processing techniques which require complex programming, neural networks develop their own solutions to problems. In fact, neural networks are trained rather than programmed[13].

2.2.1 Backpropagation Feedforward Neural Network

A neural network can be represented by nodes, and interconnections associated with weights. Figure2.1 shows the structure of a multilayer backpropagation feedforward neural network[18]. Each node is a computational unit, in which input signals from other units are mapped to output signals by a specific activation function.

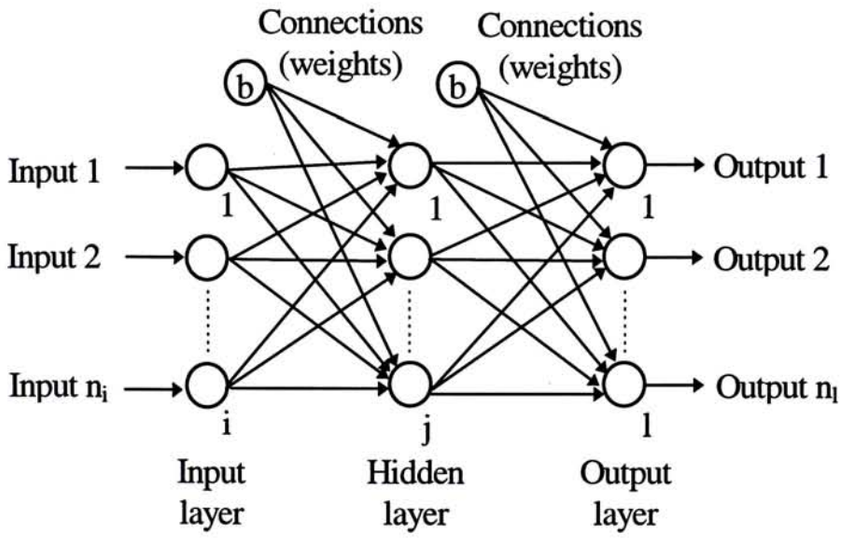

Figure 2.1: Structure of Multilayer Backpropagation Feedforward Neural Network

Before we discuss the training process of a backpropagation feedforward neural network, we define the following terminologies of neural networks.

Network Inputs and Outputs : A network input is a single pattern. Each input pattern has a corresponding value or a set of values that are mapped to a network output. The difference between the network output and the desirable output is the measurement for the generalization performance of a trained network.

Training Data Set : A set of examples are used for learning, that is to fit the weights of a network[21].

Testing Data Set : A set of examples are used to assess the performance of a trained network[21].

Generalization Performance : The generalization performance is the standard of that measures how well a trained network performs. Based on the generalization performance, we can compare the prediction ability between different classification models or topologies.

Weights : The weights are associated with the links between nodes. By varying the weights, a neural network can implement any transformation between its inputs and outputs.

Bias : In each layer, there is a unit whose output is always equal to 1, and that connects to the next layer[26]. The weights on this connection is called biases and is learned in the same way as the other weights[27]. It is an optional term[25]. However, introducing the bias always helps the convergence of training a network.[26]

Activation Function : The activation function is usually nonlinear and bounded. Through different activation functions, different properties of classifiers or function approximators can be estimated.

Network Architecture : There are many different neural network architectures, but one of the most common is the multilayer perceptron or feedforward neural network[13].

Stopping Criteria : Usually, there are two stopping criteria for the training of a neural network : the training misclassification rate and the maximum iteration runs. The training misclassification rate is the classification accuracy of a network being trained. A larger misclassification rate tolerance would lead to a poor classification performance of the training data set. A smaller misclassification rate tolerance would result in a better classification performance of the training data set. The maximum iteration runs is the stopping criterion that controls training time.

2.2.2 Training of a Backpropagation Feedforward Neural Network

The calculation of a backpropagation feedforward neural network can be divided into two parts : feedforward calculation and backpropagation calculation.

Feedforward Calculation

The feedforward calculation is used in both the training process and the operations of a trained network. The feedforward calculation provides a sequence of mapping processes from the input space to the output space. The input data allows for mapping from one layer to the other layer according to the connection weights and the node activation functions. As each connection and all data flow go from the input layer to the output layer, and since there is no feedback loop, the calculation is feedforward in nature. Each activation unit that provides the transformation from inputs to outputs, is shown in Fig.2.2([26]).

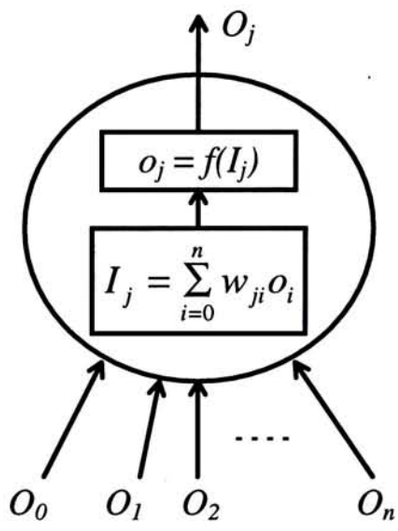

Figure 2.2: Structure of an Activation Unit

The inputs to the unit, j, in a hidden layer are aggregated using

$$\text{Net-input}_j = I_j = \sum_{i=0}^{n} w_{ji} o_i \tag{2.1}$$

The output of the unit, j, is produced by the activation function, $f(I_j)$. The literature

suggest a sigmoid backpropagation neural network is the universal consistent classifier for many types of the binary classification problem[21, 22, 23, 24, 25]. In this research, we use a sigmoid function as an activation function. A sigmoid function (Eq.2.2) shows how the output of hidden unit j is computed. The corresponding outputs of the sigmoid function are shown in Fig.2.3([18]).

$$\text{output}_j = o_j = \frac{1}{1 + \exp(-I_j)} \tag{2.2}$$

A sigmoid function is also called a squashing function. The output of an activation

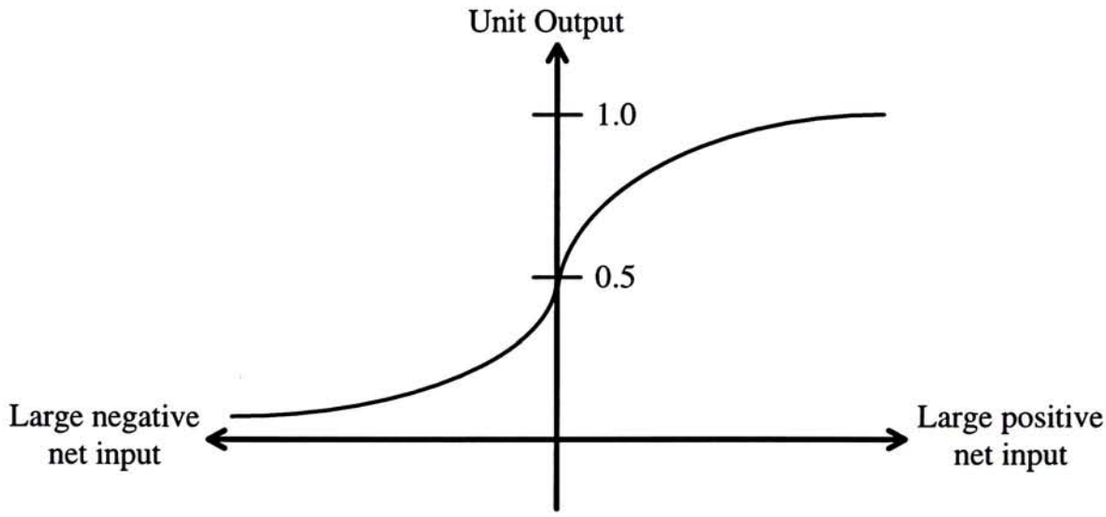

Figure 2.3: Corresponding Output of a Sigmoid Function

function is limited between 0 and 1 corresponding to the net input to the unit. For a large negative net input, the output approaches 0; while for a large positive net input, the output approaches 1. When the net input is 0, the output is 0.5. The output of each output layer, l, is calculated similarly using Eq. 2.3 and 2.4.

$$\text{Net-input}_l = I_l = \sum_{j=0}^{n} w_{lj} o_j \tag{2.3}$$

$$\text{output}_l = o_l = \frac{1}{1 + \exp(-I_l)} \tag{2.4}$$

Through the feedforward calculation, we achieve the sequence of mapping processes from the input space to the output space with cooperating the associated weights and the activation units. An iteration run of a feedforward neural network is then completed and a set of network output for the pattern classification can be obtained. Thus, the backpropagation calculation can be proceeded with these information.

Backpropagation Calculation

The backpropagation calculation is used during the training process. With the help of the feedforward calculation, we carry the processes of weight adjustment and the error propagation. These processes are parts of network's learning or training. The feedforward calculation produces an output vector. This output vector is then compared with the actual (or target) output vector to give an average sum-squared error value. The goal of the training process is to minimize this average sum-squared error over all training data sample[18] so as to find a set of weights which can map the input vectors to corresponding output vectors within a tolerant error level. The backpropagation calculation propagates the error value back and thus performs the weight adjustments accordingly.

The goal of the backpropagation training process is to determine weights and biases that minimize the average sum-squared error over all training data set. Therefore, we construct an error function of sum-squared error of the network classification

performance in Eq.2.5. We divide Eq.2.5 by the total number of the training data set, to get the average sum-squared error.

$$E_p = \frac{1}{2}\sum_{l=1}^{n_l}(t_{pl} - o_{pl})^2 \tag{2.5}$$

In order to determine the step to update the weights, we use Eq.2.6 to compute the gradient. Then we perform the backpropagation.

$$\delta_l = -\frac{\partial E_p}{\partial I_l} \tag{2.6}$$

By solving Eq.2.6, we obtain the expression of error signal in the output layer.

$$\begin{aligned} \delta_l &= f'(I_l)(t_l - o_l) \\ &= (t_l - o_l)o_l(1 - o_l) \end{aligned} \tag{2.7}$$

This error signal is used for updating the weights between the output layer and the previous hidden layer. The updating of weights is shown in Eq.2.8, where $\eta \in [0, 1]$ is the learning rate coefficient.

$$w_{lj}(new) = w_{lj}(old) + \eta\delta_l o_j \tag{2.8}$$

As the weight updating sometimes falls into the local minimum[18], a momentum coefficient, $\alpha \in [0, 1]$, is introduced in Eq.2.9([26]) to make the movement of training avoid being trapped by the local minimum.

$$w_{lj}(new) = w_{lj}(old) + \eta\delta_l o_j + \alpha[\triangle w_{lj}(old)] \tag{2.9}$$

$\triangle w_{lj}(old)$ does not equal to $w_{lj}(old)$, where $\triangle w_{lj}(old)$ is the previous weight change, and $w_{lj}(old)$ are the old weights of the network. The new weights are composed of

the old weights and weight change. The new weight change is computed by the error signal and the momentum of previous weight change.

For the hidden layers, the equation of error signal is slightly different.

$$\begin{aligned} \delta_h &= f'(I_h)\sum_{l=0}^{n_l} w_{lh}\delta_l \\ &= o_h(1-o_h)\sum_{l=0}^{n_l} w_{lh}\delta_l \end{aligned} \tag{2.10}$$

where h indicates the hidden layers. And the weight updating equation for the hidden layers are shown in Eq.2.11.

$$w_{ji}(new) = w_{ji}(old) + \eta\delta_j o_i + \alpha[\triangle w_{ji}(old)] \tag{2.11}$$

2.2.3 Single Hidden-layer Model

In this research, we examine the use of a backpropagation feedforward neural network in the binary classification problem. In particular, we use a single hidden-layer model shown in Fig.2.4. Many researchers have suggested that the model is very applicable to the problem [21, 22, 23, 24, 25].

2.2.4 Data Preprocessing

Data preprocessing for a neural network is a process that converts the raw data into the suitable inputs to a network[13]. Generally, there are several types of data preprocessing : normalization and parameterization[13, 18]. Normalization is the procedure to normalize the numerical inputs to a certain range, usually in the interval between 0 and 1. Parameterization is the procedure to convert the inputs to the

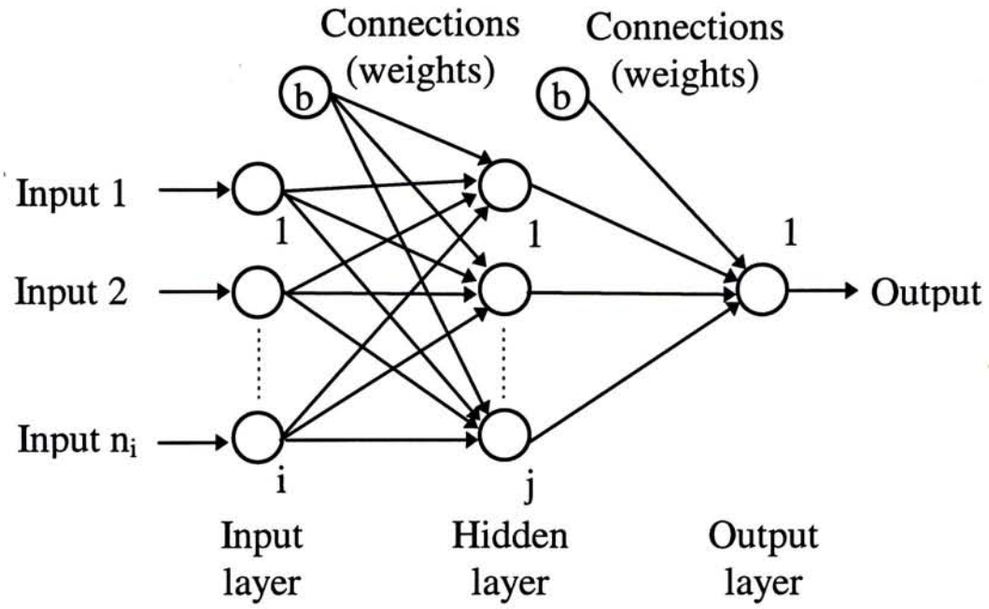

Figure 2.4: Structure of Single Hidden-layer Backpropagation Feedforward Neural Network for Binary Classification Problems

calculated parameters with different methods, such as logarithm, square, Fourier transform, chi-square goodness-of-fit, correlation coefficient, etc. When the raw input data may be textual data, encoding must be proceeded before the input data are fed into the network. There may be some non-contributing input data in the database. For the reason to reduce the time complexity required for the training, one would filter those non-contributing data by the selection of the input data. These preprocessing methods are used for similar purposes, for example, transforming the data into a suitable format for the neural network input, selecting the most relevant data, and minimizing the number of inputs to the neural network.

2.3 Fuzzy Sets

Fuzzy sets and logic were first introduced by Zadeh in 1965[21, ?]. They represent the break-through not only to the studies of uncertainty, but also to the two-valued sets and logic. The classical two-valued sets are called the crisp sets, which are strict binary decisions and assignments[21] that only represent either true or false. The fuzzy sets bring in the new concept of uncertainty beyond using probability theory. They provide measurement tools for the uncertainty and the vagueness in different problem domains. As an observation may not exactly meet sets with a crisp class, it may fall into the transition between membership (certainly belongs to the set) and non-membership (certainly does not belong to the set)[?]. In basic concept of fuzzy sets, the two-valued sets are described as its special case, where an observation only belongs to the membership or the non-membership[?].

2.3.1 Fuzzy Linear Regression Analysis

Fuzzy linear regression analysis, or Tanaka's model was introduced by Tanaka et al. in the early 1980's[28]. It is the extension of linear regression analysis with using fuzzy parameters. There are two objectives for developing fuzzy regression analysis. One deals with the situation when the relationship between the given variable cannot be described by the crisp function. The other handles the data which are fuzzy in nature[?]. There are four possible cases for the nature of the input data(independent variable) and the output data(dependent variable) in formulating the fuzzy linear regression model. They are couples of fuzzy input and output, non-fuzzy input and

fuzzy output, fuzzy input and non-fuzzy output, and non-fuzzy input and output[29]. In this research, we apply the case of non-fuzzy couple of input and output. However, for the binary classification problem, since the output is either 0 or 1(crisp sets), we may consider this case as the couple of non-fuzzy input and fuzzy output, as mentioned above. Similar to the linear regression analysis, the fuzzy linear regression shows the relationship between the dependent and the independent variables by a linear function .

$$Y = A_0 x_0 + A_1 x_1 + \cdots + A_N x_N = \mathbf{Ax} \qquad (\mathbf{x}_0 := 1) \tag{2.12}$$

where $\mathbf{Y}$ is the dependent variable, $\mathbf{x}$ is the vector of the independent variables, and $\mathbf{A}$ is the vector of a fuzzy set on the product space of parameters. The fuzzy parameter $\mathbf{A}$ is represented by the triangular fuzzy numbers :

$$A_j(a_j) = \begin{cases} 1 - \frac{|\alpha_j - a_j|}{c_j} & \text{if } \alpha_j - c_j \le a_j \le \alpha_j + c_j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.13}$$

where $A_j(a_j)$ is the membership function of the fuzzy parameters a_j, α_j and c_j are the center and spread of the fuzzy parameter a_j, respectively. With the expressions of the fuzzy parameters in the form of triangular fuzzy numbers and the extension principle, we can obtain a symmetric triangular characteristic for the independent variable of each training data set[?, 28, 30, 31] :

$$Y(y) = \begin{cases} 1 - \frac{|y - \mathbf{x}^t \boldsymbol{\alpha}|}{\mathbf{c}^t |\mathbf{x}|} & \text{for } \mathbf{x} \neq 0, \\ 1 & \text{for } \mathbf{x} = 0,\ y \neq 0, \\ 0 & \text{for } \mathbf{x} = 0,\ y = 0, \end{cases} \tag{2.14}$$

In order to minimize the total vagueness, we then :

$$\text{MIN} \sum_{j=0}^{N}(c_j \sum_{i=0}^{M}|x_{ij}|) \tag{2.15}$$

where M is the number of training samples. If the membership value of each observation y_i is greater than an imposed threshold :

$$Y(y_i) \geq h \qquad \text{for } i = 1, 2, \ldots, M \tag{2.16}$$

A linear programming problem is constructed :

$$\text{MIN} \sum_{j=0}^{N}(c_j \sum_{i=0}^{M}|x_{ij}|) \tag{2.17}$$

subject to

$$\mathbf{x}_i^t\boldsymbol{\alpha} + \left|L^{-1}(h)\right| \sum_{j=0}^{N} c_j \left|x_{ij}\right| \geq y_i,$$

$$\mathbf{x}_i^t\boldsymbol{\alpha} - \left|L^{-1}(h)\right| \sum_{j=0}^{N} c_j \left|x_{ij}\right| \leq y_i,$$

$$c \geq 0, \alpha \in \Re, x_{i0} := 1,$$

$$0 \leq h \leq 1,$$

$$i = 1, 2, \ldots, M$$

where $|L^{-1}(h)| = 1 - h$, $h \in [0, 1]$, and the choice of the h value influences the widths c_j of the fuzzy parameters. By solving this linear programming model, the fuzzy parameters, $\mathbf{A}$, can be obtained.

2.4 Network Architecture Altering Algorithms

Dealing with the network size of a neural network during the training, we may use two network architecture altering algorithms : pruning algorithms and constructive

algorithms. These methods prefer a small solution network than a large one. A small solution network provides the advantages of less computational cost, less storage space and being easier to interpret what the trained network is doing[6].

2.4.1 Pruning Algorithms

Pruning algorithm reduces the size of a neural network by cutting down the unnecessary units or weights[9, 16, 32, 33]. It starts from an oversized network and is often used as a post-processing to obtain a suitable network architecture. The following issues summarize some pruning algorithms[32]:

Magnitude Based Pruning

Magnitude based pruning algorithm starts with an oversized network. After each training, it removes the smallest weight of the whole network. This method often reduces the number of retraining cycles[34] and is the simplest way in pruning algorithms. However, it may often lead to the elimination of the wrong weights[35].

Optimal Brain Damage(OBD)

This method physically decreases the capacity of the model in order to limit overfitting[36]. Similar to other pruning methods, it trains the oversized network to minimize the classification error of the training data set. The pruning is performed to the unit which gives the smallest saliency. The saliency that is computed for each hidden unit indicates the change of the error function when pruning is applied to the hidden unit. The corresponding approximation of the error function with respect to

weights is in terms of Taylor series. As the first derivatives vanish and all the higher order terms decrease, only a Hessian matrix, H, is considered. This Hessian matrix is obtained by considering all the second order derivatives in the error function with respect to weights, $\frac{\partial^2 E}{\partial w^2}$. OBD assumes that H is a diagonal matrix by the diagonal approximation; thus, simplifies the calculations. It is used to estimate the increase in the training error, the saliency, when removing certain weights[37].

Optimal Brain Surgeon(OBS)

This is an extension method to OBD. Unlike OBD that performs poorly when the problem on hand leads to a non-diagonal Hessian matrix, H[35], OBS computes the full Hessian matrix rather than estimating it as a diagonal matrix. Hence, a more accurate approximation of the error function can be obtained. However, OBS requires an expensive computation. The computation is the result of the inverse of the Hessian matrix for deducing saliency and weight change for every link. In OBS, all the weights are updated before the next iteration for searching the new saliency and weight change is proceeded.

Skeletionization

Similar to OBD and OBS, skeletionization removes hidden units by considering the effects of removing a hidden unit to the change of the error function. In general, it computes a measure of relevance that identifies which hidden units are most critical to the performance, and removes the least relevance one in order to construct a skeleton of the network[38]. The measure of relevance is the estimation of the difference, ρ,

between the error of the network, E, on the training data set that with and without the corresponding hidden unit.

$$\rho_i = E_{\alpha_i=0} - E_{\alpha_i=1} \tag{2.18}$$

where α_i is the attention strength of the unit, i, which controls the flows from the output of one unit to its succeeding layer. $\alpha_i = 0$ corresponds to hidden unit i that is removed from the network, while $\alpha_i = 1$ corresponds to hidden unit i that remains in the network. With the approximation of ρ_i, the selection of pruning a unit which gives the least relevance to the network can be made. Skeletonization algorithm can also be performed to the input units to suppress their influences. Since the removing of a unit is made by setting $\alpha = 0$, so that a network is able to recover easily when a unit has been removed[38].

Non-contributing Units

Pruning non-contributing units is simpler than the other pruning algorithms(except the magnitude based pruning), and provides a satisfactory result. The pruning approach for non-contributing units investigates the output of each hidden unit for the whole training set. A unit is considered a non-contributing unit when its output does not change for the whole input patterns, duplicates or inversely duplicates the output of another unit in the same layer[9, 16, 32].

2.4.2 Constructive/Growing Algorithms

A constructive algorithm, contrary to a pruning algorithm, searches for the solution network from a minimal network size. The advantage of a constructive algorithm is derived from the simplicity in defining an initial network and the preference of a small architecture solution network[17]. There are some well established constructive methods which try to search for suitable network size in the way of single unit learning. Readers may refer to [6], [17] and [26] for more detailed discussion of the following approaches :

Tower Algorithm

Tower algorithm adds a new unit to the least hidden layer in a network. Weights of trained units are frozen before a new unit is added. A newly added unit is fed with all the input values of the training data, the output from the unit that is most recently trained. The network will keep on growing until no further improvement can be obtained. Otherwise, the last added unit will be removed and the training is finished with the solution network obtained before the last growing.

Tiling Algorithm

Tiling algorithm constructs a strictly layered network. The inputs of a unit come from the outputs of immediate previous layer only. Each hidden layer contains a master unit that must performs better than the previous layer. New hidden unit is added to the same layer as the ancillary unit that help the layer to be faithful. A faithful layer means each succeeding layer has a different representation for the

inputs, and no two training samples with different class have the same representation in any layer[26]. The training stops when all training data set are classified correctly.

Upstart Algorithm

Upstart algorithm starts with a single unit network. The misclassification of the network will cause the reinforcements to the unit. For example, if the class is +1, but the output is -1, a positive reinforcement will be given by adding a specific unit. On the contrary, a negative reinforcement will be given by adding a specific unit. When the addition of reinforcement units results in incorrectly classification, the new reinforcement units for the previous reinforcement units will be added. The training stops when all training data set are classified correctly.

Dynamic Node Creation(DNC)

DNC algorithm adds one hidden unit at a time and always in the same hidden layer. After a new hidden unit is added, the whole network must be retrained. The information of the previous training will probably lose when a new hidden unit is added. And, the computational cost will be increased drastically as the network enlarges. However, it is a very simple algorithm.

Projection Pursuit Regression(PPR)

Some constructive algorithms are based on the statistical technique PPR. A new unit is added to the same layer one at a time. However, the retraining of the whole network is not required. After a new hidden unit is added, the input-to-hidden weights,

parameters associated with the hidden unit activation function and the hidden-to-output weights are to be trained. They are usually trained separately. While one of the parameters is being trained, the other two are frozen. The training of the new unit starts with the input-to-hidden weights first, then the parameters associated with the activation function and the hidden-to-output weights. This training process stops when there is no further improvement of the performance.

Cascade-Correlation Algorithm(Cascor)

Cascor starts with a small network and adds a hidden unit one at a time. The newly added hidden units build a multilayer network. The inputs to the new unit are connected to all inputs of the training data and the outputs of other previous units. When the network has no further improvement in the training error and the error is not significantly small enough to terminate the training, a new unit is added to the network. Cascor network adds a new hidden unit which has been trained to maximize the correlation, S, between the new unit's output and the residual classification error of the output unit:

$$S = \sum_o \left| \sum_p (V_p - \overline{V})(E_{po} - \overline{E_o}) \right| \tag{2.19}$$

where V_p is the new unit's output of training pattern p, $\overline{V}$ is the average to the output of new units, E_{po} is the residual error observed at output unit o for training pattern p, $\overline{E_o}$ is the average to the residual error. In order to maximize S, we obtain a partial derivative of the error with respect to the incoming weights of the new unit :

$$\frac{\partial S}{\partial w_i} = \sum_{p,o} \sigma_o (E_{po} - \overline{E_o}) f'_p I_{ip} \tag{2.20}$$

where σ_o is the sign of the correlation between the new unit's output and the output of network, f'_p is the derivative of training pattern p of the new unit's activation function with respect to the sum of its inputs, and I_{ip} is the input of training pattern p to the new unit. Once we have obtained the weights of the new unit that maximize the correlation, the weights are frozen and the retraining to the output unit is then proceeded with the one additional new input to it.

2.5 Summary

As mentioned in the last chapter, we address to the general limitations in applying a single hidden-layer backpropagation feedforward neural network to the binary classification problem : the data dependency and the unknown network size. The related literature in handling such problems were reviewed.

Chapter 3

Hybrid Classification Systems

3.1 Overview

Although several research studies suggest that the neural network approach has a more accurate classification ability[1, 5, 14] than most of the other approaches, the accuracy of prediction of the neural network approach has potential for further improvement. For example, Han[8] indicates that the relative performance of different classification techniques may depend on the data conditions. Thus, this is a problem when using the neural network. The generalization of the neural network depends on the distribution of the training data set[14]. For the training data set with fewer noise data, the generalized network may have a good performance. However, when the data set consist of more noise data, the neural network will be generalized more depending on the noise data; as a result, the misclassification rate of the noise free data, which should be easier to be classified, increases.

The objective of this study is to propose a way to improve the accuracy of the

neural network. Rather than developing a new architecture and algorithm to achieve a better performance, we use a hybrid model instead. Furthermore, the experiment results of applying our model and the conventional backpropagation neural network to the breast cytology diagnosis shows that these neural network based methods are capable of being reliable decision support systems for the medical diagnosis.

The hybrid model developed in this study comprises of two phases. In phase I of the model, fuzzy regression method with fuzzy interval analysis is applied. In phase II of the model two simple backpropagation neural network constructions as the final classification engine are provided.

Both the models are implemented, and tested using the data of breast mass cytology. The class sets are benign and malignant and each element of the class sets consists of nine cytological characteristics of benign or of malignant breast fine-needle aspirates; however, no single characteristic alone or presently described class distinguishes between benign and malignant samples[3]. The previous studies to solve this problem are multisurface method of pattern separation[3], decision tree[3], and mathematical programming method[4]. Therefore, applying the neural network based methods to the breast mass cytology classification problem is innovative.

By using the fuzzy linear regression with the fuzzy interval analysis, we separate the training data into two groups based on the fuzzy interval. The separated training data sets are used to generalize two neural networks accordingly. With two neural networks, we formulate two different functions to describe the distribution space of the data, rather than using one to generate one function to do so. In our experiment, our model is compared with the conventional backpropagation neural network. The

result shows that using two different functions to describe the distribution space of the observations promises a more accurate classification result.

The chapter is organized as follows. In Section 3.2, Tanaka's model, the modified Tanaka's model with fuzzy interval analysis, and the multilayer feedforward backpropagation neural network are introduced. In Section 3.3 the sample data and the methodology used are described. In Section 3.4 the details for the construction of our model are explained. In Section 3.5 the results and the comparison between our model and the conventional backpropagation neural network are reported. Finally, in Section 3.6 a conclusion on the experiment are provided.

3.2 Literature Review

3.2.1 Fuzzy Linear Regression(FLR) with Fuzzy Interval Analysis

The fuzzy linear regression approach was introduced by Tanaka et al.[28, 39] in order to deal with a vague phenomenon. The assumption of Tanaka's model is that the input and output data of fuzzy linear model are fuzzy, the relationship between the input and output data is given by a fuzzy function, and the distribution of the data is possibilistic[30]. The fuzzy linear regression has been applied to forecasting in an uncertain environment for finding an interrelationship between the linear interval model and the output intervals of the given data. Tanaka's model[30, 31] assumes a

linear function as follow :

$$Y = A_0 x_0 + A_1 x_1 + \cdots + A_N x_N = \mathbf{Ax} \qquad (\mathbf{x}_0 := 1) \tag{3.1}$$

where $\mathbf{Y}$ is the dependent variable, $\mathbf{x}$ is the vector of the independent variables, and $\mathbf{A}$ is the vector of a fuzzy set on the product space of parameters.

The fuzzy parameters A_j are represented in the form of triangular fuzzy numbers :

$$A_j(a_j) = \begin{cases} 1 - \frac{|\alpha_j - a_j|}{c_j} & \text{if } \alpha_j - c_j \leq a_j \leq \alpha_j + c_j \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $A_j(a_j)$ is the membership function of the fuzzy set of a_j, α_j is the center, and c_j is the spread of the fuzzy number.

Applying the extension principle :

$$Y(y) = \begin{cases} 1 - \frac{|y - \mathbf{x}^t \boldsymbol{\alpha}|}{\mathbf{c}^t |\mathbf{x}|} & \text{for x} \neq 0, \\ 1 & \text{for x} = 0, \text{ y} \neq 0, \\ 0 & \text{for x} = 0, \text{ y} = 0, \end{cases} \tag{3.3}$$

In order to minimize the total vagueness, we then :

$$\text{MIN} \sum_{j=0}^{N} (c_j \sum_{i=0}^{M} |x_{ij}|) \tag{3.4}$$

where M is the number of training samples.

If the membership value of each observation y_i is greater than an imposed threshold :

$$Y(y_i) \geq h \qquad \text{for } i = 1, 2, \ldots, M \tag{3.5}$$

A linear programming problem is constructed :

$$\text{MIN} \sum_{j=0}^{N} (c_j \sum_{i=0}^{M} |x_{ij}|) \tag{3.6}$$

subject to

$$\mathbf{x}_i^t\boldsymbol{\alpha} + \left|L^{-1}(h)\right| \sum_{j=0}^{N} c_j \left|x_{ij}\right| \geq y_i,$$

$$\mathbf{x}_i^t\boldsymbol{\alpha} - \left|L^{-1}(h)\right| \sum_{j=0}^{N} c_j \left|x_{ij}\right| \leq y_i,$$

$$c \geq 0, \alpha \in \Re, x_{i0} := 1,$$

$$0 \leq h \leq 1,$$

$$i = 1, 2, \ldots, M$$

where $|L^{-1}(h)| = 1 - h$, $h \in [0, 1]$, and the choice of the h value influences the widths c_j of the fuzzy parameters.

Tanaka's model can be used to analyze the interval of the dependent variable y, however, the drawback is that a few values may dominate the estimation of the bounds of the crisp interval. Therefore, the model is very sensitive to outliers[30]. In, 1993, Peters[30] provided a modification to Tanaka's model. In this model, the bounds of the interval are assumed to be fuzzy rather than crisp, so that each of the dependent data y has a membership degree of belonging to the interval. Peters' fuzzy linear programming model is formulated as follows:

$$\text{MAX } \overline{\lambda} = \frac{1}{M} \sum_{i=1}^{M} \lambda_i \tag{3.7}$$

subject to

$$\begin{aligned}
(1 - \overline{\lambda})p_0 - \sum_{i=0}^{M}\sum_{j=0}^{N} c_j \left|x_{ij}\right| &\geq -d_0, \\
(1 - \lambda_i)p_i + \mathbf{x}_i^t\boldsymbol{\alpha} + \sum_{j=0}^{N} c_j \left|x_{ij}\right| &\geq y_i, \\
(1 - \lambda_i)p_i - \mathbf{x}_i^t\boldsymbol{\alpha} + \sum_{j=0}^{N} c_j \left|x_{ij}\right| &\geq -y_i,
\end{aligned}$$

$$
\begin{aligned}
-\lambda_i &\geq -1, \\
\lambda_i, c &\geq 0, \alpha \in \Re, x_{i0} := 1, \\
\left|L^{-1}(h)\right| &: = 1; i = 1, 2, \ldots, M
\end{aligned}
$$

where λ represents the membership degree to which the solution belongs to the set "good solution". λ can be determined by a trade-off between the objective function and the equation of the "worst" datum y[30]. p_i is the width of the "tolerance interval".

A high value of p_0 and low values of p_i leads to a wide interval; while a low value of p_0 and high values of p_i leads to a narrow interval. d_0 represents the desired value of the objective function. The suggested value of d_0 is 0, since the total vagueness is desired to be 0. The objective function MAX $\overline{\lambda} = \frac{1}{M} \sum_{i=i}^{M} \lambda_i$ allows for the training data of each datum to compensate the model. By the weight factor $\frac{1}{M}$, each training datum influences the regression function.

3.3 Data Sample and Methodology

For this study, the data sample consists of patients' breast mass cytology information who were admitted for breast cancer diagnoses at the University of Wisconsin(Madison) form 1989 to 1992[2, 3, 4, 40]. There are 699 instances in the database, out of which 683 instances are chosen for our experiment. Sixteen instances are removed due to missing values. Each instance comprises of 9 attributes and 1 class value. These 9 attributes required for Fine-Needle Aspirate (FNA) testing were taken from each patient's breast[4]. These attributes are shown in Table 3.1.

Attribute	**Domain**
Clump Thickness	1-10
Uniformity of Cell Size	1-10
Uniformity of Cell Shape	1-10
Marginal Adhesion	1-10
Single Epithelial Cell Size	1-10
Bare Nuclei	1-10
Bland Chromatin	1-10
Normal Necleoli	1-10
Mitoses	1-10

Table 3.1: Attribute descriptions

Each attribute is a scalar observation. The range of the attribute values shows the likelihood of malignancy; the larger the value, the greater the likelihood of malignancy. The class value tells us whether the symptom is benign or malignant for the corresponding instance. In the chosen instances, there are 444 instances belonging to the class of benign breast masses and 239 instances belonging to the class of malignant breast masses.

In each training sample set, 200 instances (100 benign and 100 malignant) are randomly selected. For each testing sample set, 169 instances (84 benign and 85 malignant) are randomly selected. Forty-five couples of training and testing data sample sets are generated from the 683 instances, and tested by our model.

In our experiment, **L**inear, **In**teractive, **D**iscrete **O**ptimizer (LINDO)[41] is used

to solve the Linear Programming problem. LINDO is an optimization modelling system. It has the advantages in that it allows for quick formulation, modification, and solution for the linear programming problem. Also, a single hidden-layer feedforward backpropagation neural network is used as the basic classifier. The process flowchart of the methodology used for developing our model in this study is depicted in Figure 3.1.

3.4 Hybrid Model

3.4.1 Construction of Model

The basic model is shown in Figure 3.2. The first part of the model separates the data sample into two groups. The second part provides two independent neural networks as the classifiers and classifies the two groups of data sample into two distinct classes.

Phase I : Fuzzy interval analysis by fuzzy linear regression

Rather than finding a best fitted interval in which all training data can be contained, we apply the fuzzy linear regression with fuzzy interval model as a data handler. The class variable y_i, and the attribute variables x_{ij} of each instance i in the training data set are used for composing the fuzzy linear programming model (3.7). By solving this fuzzy linear programming model, $\overline{\lambda}$, λ_i, α_j and c_j are obtained. Since

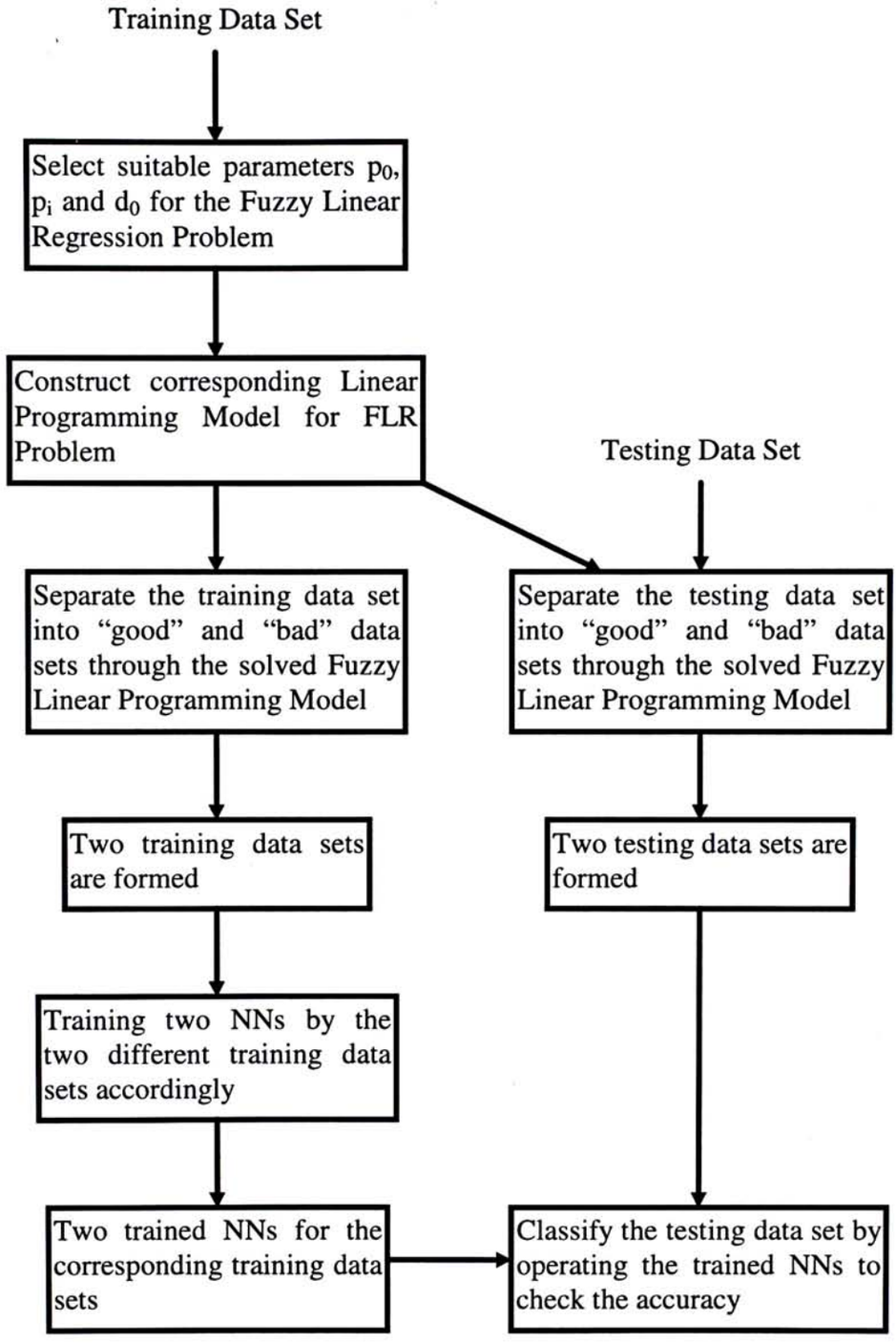

Figure 3.1: Process flow chart of the hybrid model

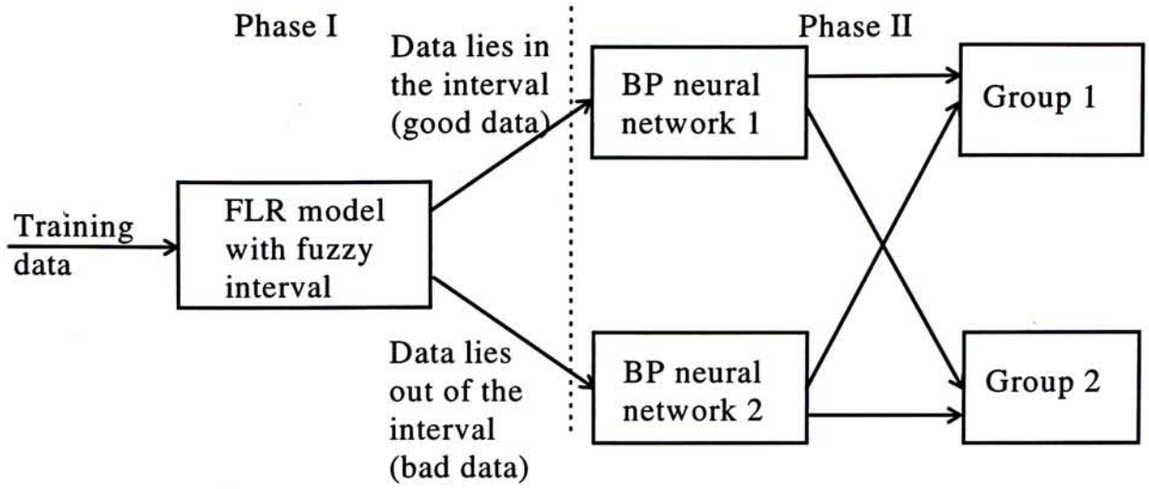

Figure 3.2: Construction of the Hybrid Model

we assume the fuzzy characteristic of the bounds of the interval, each datum has its own certain degrees of belief(memberships degree) of lying into the interval.

The solved fuzzy linear programming model shows that one group, representing those data with instances of membership degree of $1(\lambda = 1)$, lies inside the interval; while the other group, with instances of membership degree smaller than $1(\lambda < 1)$, lies outside the interval. In other words, the instances with the second group are classified as noise data. Figure 3.3 shows that λ_i may tend to $-\infty$ when data is infinitely far away from the boundary of the interval[30]. The separated training data sets are then used to generalize the backpropagation neural networks in the phase II model.

Since the phase II model performs the classification procedure independently for two groups of data, the testing data set have to be processed in the data handling step before they can be tested in the phase II model. To determine which data falls into the interval or lies outside the interval, we only needed to test the feasibility of the

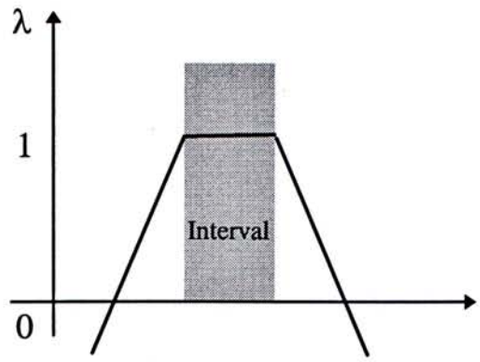

Figure 3.3: Membership function

fuzzy linear programming model with the testing data set. As a result, two groups of testing data set are obtained in the same manner as the separated training data sets. By using the fuzzy linear regression with fuzzy interval analysis, the training and testing data sets are then separated into a group of "good" data and a group of "noise" data, in order to perform further training and testing processes, respectively.

We control the width of the interval as we desire by choosing different values of d_0, p_0 and p_i. In other words, we control the size of the two training and two testing data groups. The wider the intervals, the more the data would lie inside.

Phase II : Backpropagation neural network model (BPNN)

A three layers (an input layer, a hidden layer, and an output layer) feed-forward backpropagation neural network is used. There is only one node in the output layer. For the breast cancer experiment, those outputs smaller than 0.5 are classified as group benign or "0"; while, those outputs greater than or equal to 0.5 are classified as group malignant or "1". As mentioned in the previous sections, the two training data groups are fed into two BPNN models separately. One of the BPNN models

is trained with the training data set which lie inside the interval, and the other one is trained with the outliers. Figure 3.4 shows the basic construction of the neural network we use in our experiment.

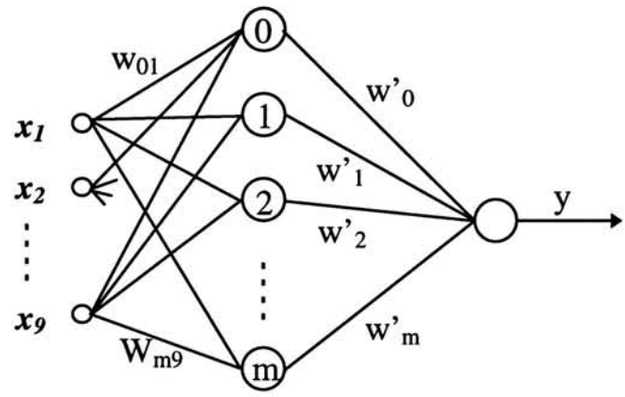

Figure 3.4: Single hidden-layer, single output neural network

3.5 Experimental Results

3.5.1 Experimental Results on Breast Cancer Database

We randomly generate 45 couples of training and testing data samples. Each training data sample formulates its own fuzzy linear programming model. From this model, the training and testing data sample are separated into two groups, the data which lies in and out of the interval. Two different groups of training data sample are then fed into two independent NNs in order to perform the training and generalization process. Two trained NNs are then tested with the corresponding testing data sample. Then, we compare the accuracy of our model with the conventional method which

trained a NN with the whole set of training data sample. Table A.1 shows the comparison of the accuracy (in percentage) between our model (Hybrid Model) and the conventional method (Batch Process).

Trial	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	90.28	91.55
2	90.18	91.64
3	90.41	91.85
4	90.34	91.85
5	90.57	92.06
Average	*90.36*	*91.79*

Table 3.2: Average prediction accuracy on 5 trials

The values(that are in bold face) in Table A.1 to A.5 indicate instances where the hybrid method has generated equal or better predictions than the conventional method. The overall average prediction accuracy as shown in Table 3.2, of our model is 91.79% which is 1.43% more accurate than the conventional one. Although there are some samples in the conventional method that have a better prediction result, it can be argued that our model promises a better prediction ability since the accuracy difference between our model and the conventional method for these samples is relatively small. And the largest difference that our model outperforms the conventional method is 9.47%, while the largest difference that our model perform poor than the conventional method is only 1.77%.

Now let's look at a specific illustration of phase I. When we analyze a specific sample difference, the following conclusions can be made. For example, sample 2 in experimental trial 1, in our model experience a 5.33% improvement. To explain the process of the phase I model, we analyze the distribution of the dependent variable value of the training and testing data sample in sample 2. Figure 3.5 shows the dependent variable value of each training datum. These values are calculated by Eq. (3.1), where **A** is obtained by solving Eq. (3.7) and by substituting **A** with α(center of the fuzzy number). Similarly, the dependent variable value of each testing datum is obtained in Figure 3.8 by substituting the preexisted centers of **A**.When the training and testing data samples are fed into the phase I model, those data samples are then separated into two groups accordingly. The dependent variable value of those separated data samples are shown in Figures 3.6, 3.7, 3.9, 3.10, respectively. When comparing Figure 3.6 and 3.7, we can observe the differences between the data sample which lie inside and outside the interval. In Figure 3.6, the data lie around 0 or 1 without large differences. However, in Figure 3.7, the data fluctuates around 0 or 1 with very large variations. The same situation occurs in the testing data sample, too. These results show that the dependent variable values which are close to the crisp output [0, 1], are more believed to lie inside the interval; otherwise, they are believed to lie outside the interval.

According to the separated groups of the training data samples, two independent NNs can be trained; as a result, there are two different functions to describe the distribution space of the training data sample. We can observe that the advantage

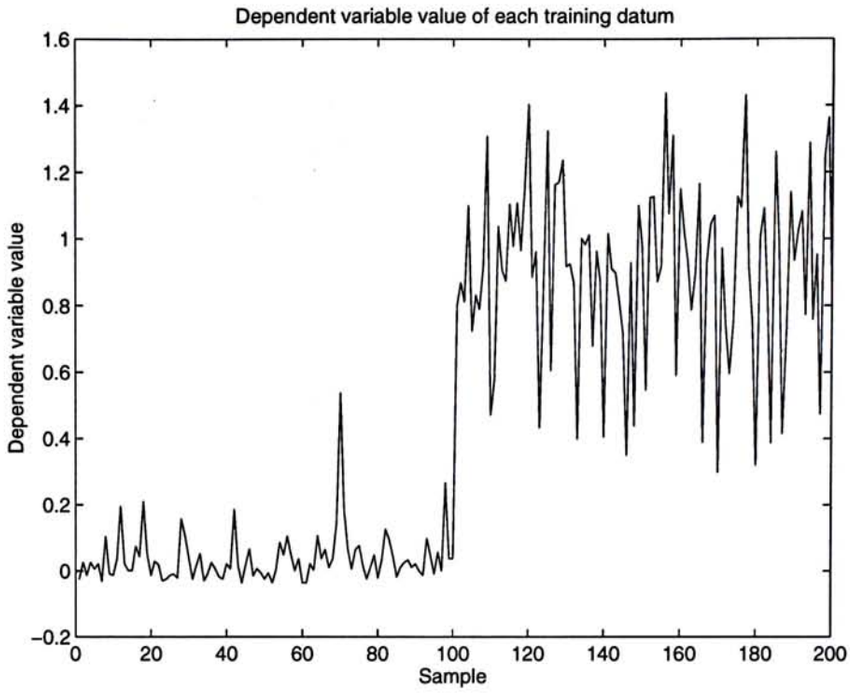

Figure 3.5: Dependent variable value of each training datum

of using two functions over using one function(conventional method) is that they can describe a more complex distribution space as a whole. However, the drawback of our model is that it is more time consuming than the conventional one.

3.5.2 Experimental Results on Synthetic Data

In order to show the improvement on the generalization performance of the hybrid classification model in a noisy environment, we synthetically create a database with a large amount of noise, and compare the generalization performance between the conventional method and the hybrid classification model. The database is a binary classification database. Each instance has 9 attributes and 1 class value. There are 883 instances (544 instances belong to class 0, 339 instances belong to class 1). In each training sample set, 200 instances are randomly selected. For each testing sample set,

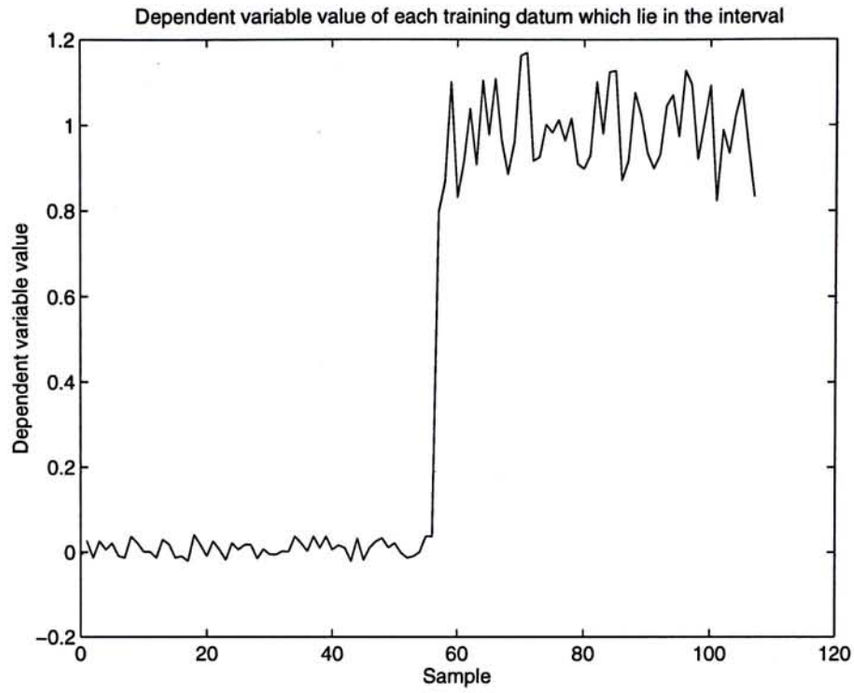

Figure 3.6: Dependent variable value of each training datum which lie inside the interval

169 instances are randomly selected. We test on three kinds of training and testing couple. The first one is composed with equal amount of good data and noise data, the second one is composed with 10% more noise data, and the last one is composed with 20% more noise data. In each test, forty-five couples of training and testing data sample sets are generated from the 883 instances, and tested by our model.

The values(that are in bold face) in Table A.6 to A.8 indicate instances where the hybrid method has generated equal or better predictions than the conventional method. In Table A.6, the accuracy comparison with respect to sample data sets of equal amount of good and noise data, it shows that the hybrid model experiences a 2.85% more accurate on average. For the comparison result on those sample data sets

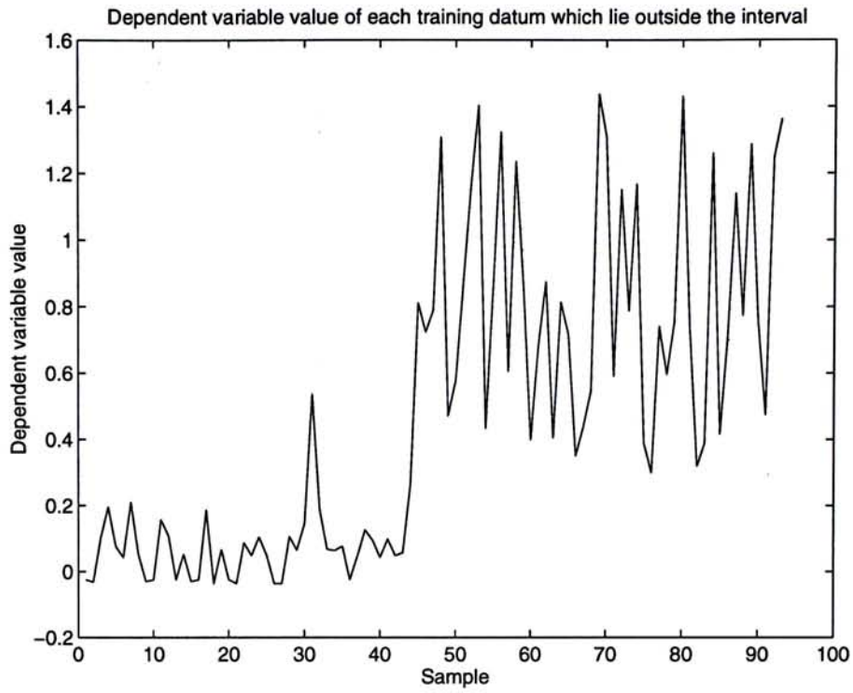

Figure 3.7: Dependent variable value of each training datum which lie outside the interval

composed with 10% more noise data than the good data, Table A.7 shows the hybrid model experiences a 5.11% more accurate on average. When the sample data sets with 20% more noise data are used, it shows a 7.78% accuracy improvement in Table A.8. The summary of the average prediction accuracy on these three experiments are showed in Table 3.3. From the results, we observe that the generalization performance of the hybrid model is better than the conventional method as the data contain more noise.

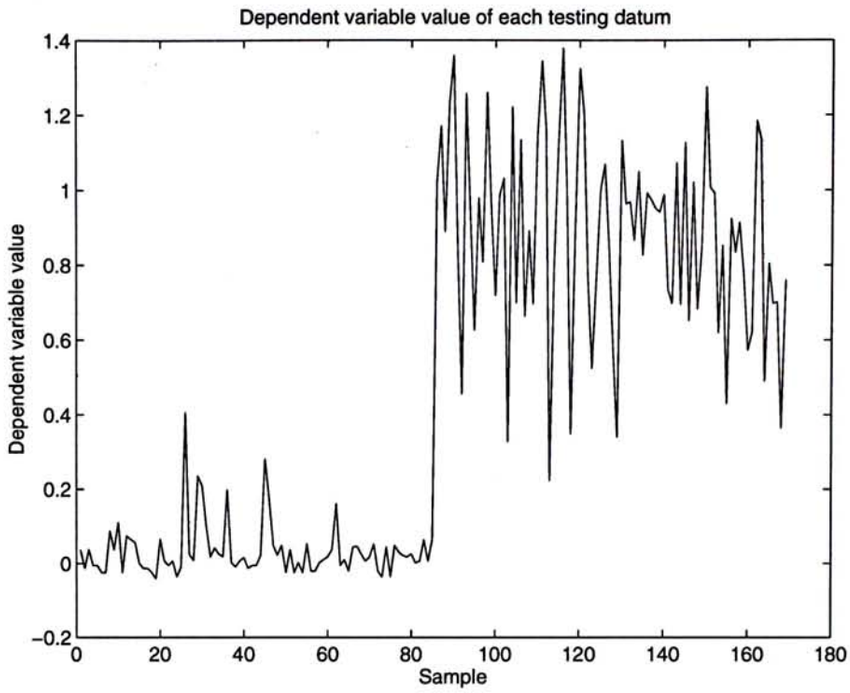

Figure 3.8: Dependent variable value of each testing datum

3.6 Conclusion

This chapter has proposed a hybrid binary classification model. Phase I is used to find a fuzzy interval so as to separate the training data into two groups, i.e. whether the data lies inside or outside the interval. The objective of phase I is to minimize the effect of the vagueness data in the training data, and to separate the certain

Additional Noise	$BatchProcess(\%)$ $(Conventional)$	$HybridMethod(\%)$	$Difference(\%)$
0%	74.42	77.26	*2.85*
10%	71.65	76.76	*5.11*
20%	68.78	76.60	*7.78*

Table 3.3: Average prediction accuracy on 5 trials

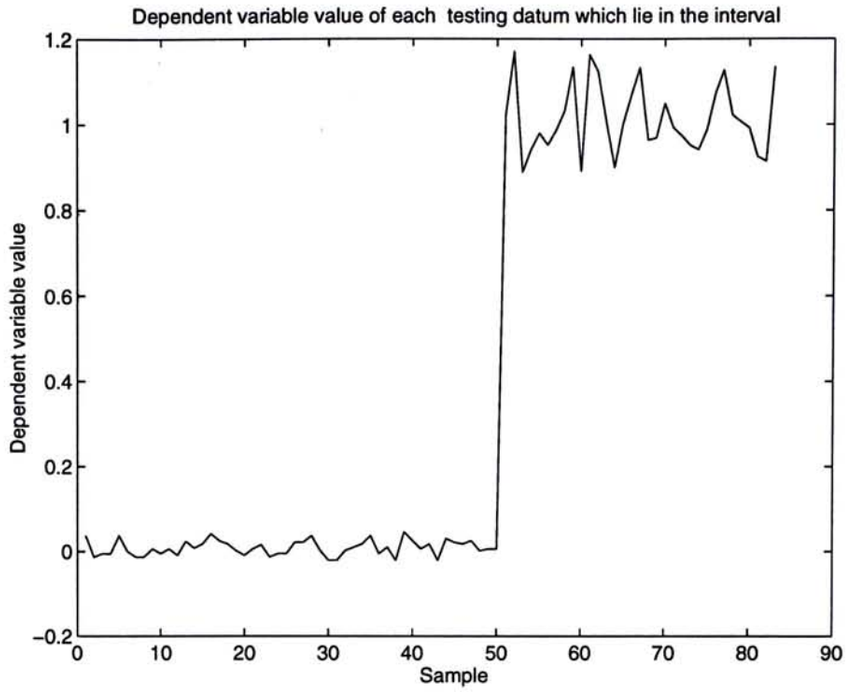

Figure 3.9: Dependent variable value of each testing datum which lie inside the interval

data and vagueness data into two groups. According to the fixed parameters and the found unknowns in the FLR with fuzzy interval model, the testing data sample is then separated into two groups also. In phase II, two single hidden-layer BPNN models are used to build up the classification engines.

The two independent NNs allow us to formulate two different non-linear discriminant functions to classify the data. The conventional method uses one NN to describe the distribution space of the data. Although, a NN can be used to formulate a highly non-linear function, it is hard to describe a very complex distribution space. The hybrid model provides us with two independent functions to describe the distribution space of the data sets, therefore, the ability of describing the distribution space is im-

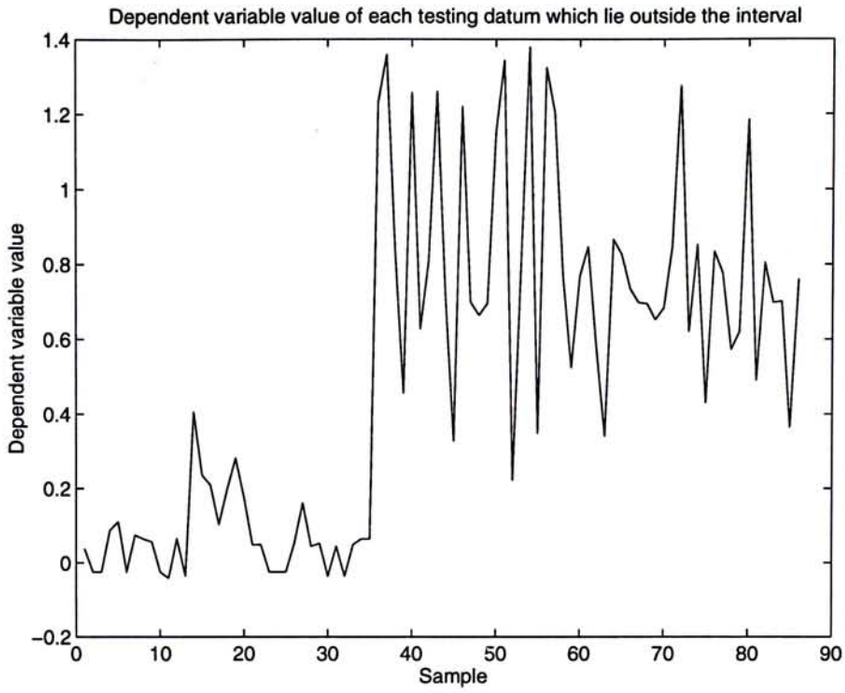

Figure 3.10: Dependent variable value of each testing datum which lie outside the interval

proved. For future studies, one may want to change the phase II classification engines by using other approaches and techniques to see if the accuracy is further improved. In our study, the phase II model uses the conventional BPNN. The results show that, when using the BPNN approach to act as the classification engine, the phase I model plays an important role in the accuracy improvement.

We finally note that the accuracy with which benign and malignant breast mass are diagnosed using the hybrid model is an illustrative example of the usefulness of the proposed method in solving other medical diagnostic and decision making problem.

Chapter 4

Searching for Suitable Network Size Automatically

4.1 Overview

A neural network is a structure of autonomous nodes interconnected by one-directional links. Weights(that are determined during a training process) are associated with the links. Input signals are aggregated according to weights on the links and transformed via an activation function to produce output signals.

One of the problems in applying a neural network pertains to determining its right architecture. For example, in a three-layered neural network, it is difficult to decide the right number of hidden nodes. When a neural network has too many hidden nodes, it tends to memorize the training patterns rather than to generalize the prediction ability. On the other hand, a neural network may not achieve the desirable prediction ability if it has too few hidden nodes.

The suitable network size is also usually unknown. In general, we have to start a training process with an arbitrary and often oversized network. Then we perform several tests with different network sizes. Finally, we choose the most promising network size. The criteria for choosing a network usually include the ability of each considered network size on the generalization performance, and the level of ease to obtain a solution network in a specific problem.

The conventional approach, discussed above, is very tedious and time consuming. In the literature, there are two main approaches to deal with the problem. They are pruning algorithms and constructive algorithms. Pruning algorithms reduce the size of a neural network by cutting down the unnecessary units or weights[9, 16, 32, 33]. They start from an oversized network and are often used as a post-processing to obtain a suitable network architecture. Constructive algorithms, contrary to pruning algorithms, search for the solution network from a minimal network size. The advantage of constructive algorithms is derived from the simplicity in defining an initial network and the preference of a small architecture solution network[17]. Most of the recent studies in constructive algorithms concentrate on expanding the network architecture layer by layer and adding new hidden units one at a time. In this research, we will mainly consider determining the number of hidden units in the same layer.

One of the motivations in this research is that we want to find effective and efficient methods to construct a suitable architecture for a single hidden layer backpropagation feedforward neural network automatically. The methods should be flexible to help us find tailor-made network architectures for different applications without requiring a wild guess on the network size. By evaluating the performance based on generalization

accuracy, the time complexity, and the obtained network sizes with some different algorithms, we may obtain ideas in searching for the suitable network architecture for a problem while a learning process proceeds.

In this research, the databases of breast cancer mass cytology test results[2, 3, 4, 40] and tic-tac-toe patterns are used to evaluate the performance of different algorithms. We will first review the methods of pruning and constructive algorithms which have been implemented and used for the evaluation in Section 4.2. Then, in Section 4.3, our methodology, and the data samples will be described. The setup of experiments and the experimental results are given in Section 4.4. Finally, in the last section, brief discussions and conclusion are provided.

4.2 Literature Review

4.2.1 Pruning Algorithm

Pruning is one of the most popular ways to find a small suitable architecture[34] for a neural network. A pruning algorithm removes some hidden units or weights in an oversized network to produce a solution network[17]. There are a variety of removing procedures, such as magnitude based pruning, optimal brain damage, optimal brain surgeon, skeletionization, and non-contributing units[32].

In this research, we are concerned with the pruning approach of the non-contributing units and its modification, which is simpler than the other pruning algorithms, but provides a satisfactory result. The pruning approach of non-contributing units inves-

tigates the output of each hidden unit for the whole training set. A non-contributing unit has its output that does not change for the whole input patterns, or when it duplicates or inversely duplicates the output of another unit in the same layer[9, 16, 32]. The pruning algorithms studied here are proposed in [9] and [16].

One Pass Pruning

This approach was introduced in [16] to perform pruning to a solution network, where a solution network is a trained network that provides a satisfactory classification ability. When a solution network has been obtained, two categories of hidden units will be removed : the duplicated units and the non-contributing units. The first category of units to be identified for removing is a hidden unit has an output for all input patterns which is the same as, or opposite to, the output of another hidden unit in the same layer. Once these two hidden units are identified, one of them will be removed as they are duplicate or inversely duplicate of each other. The second pruning procedure identifies hidden units that do not contribute to the overall solution network. In other words, since the non-contributing units do not provide any classification information, their removal will not affect the generalization ability of the pruned network[16]. The non-contributing units often classify all input patterns into the same group.

Prune and Retrain

In [9], the characteristics of the excessive hidden units in an oversized solution network are studied. Similar to the pruning algorithm in [16], the pruning procedure

will be performed after a solution network is achieved. There are four categories of excessive hidden units: excessive non-contribution, excessive duplication, excessive inversely duplication, and excessive inadequacy. Once these excessive units have been pruned, retraining of the pruned network will be performed until no more excessive hidden units are found in the solution network. The excessiveness of a hidden unit means that a unit may be removed without affecting the generalization performance of the network. In [9], the four categories of excessive units are defined as follows :

Excessive Non-contributing unit : An excessive non-contributing unit would give similar output for all input patterns, o_{pj}. This unit classifies those input patterns that lie on one side of the decision hyperplanes, so that no discrimination information would be given by this unit. An excessive non-contributing unit can be detected by the followings :

$$\text{excessive-noncontributing}(j) = \text{noncontributing}(j) \text{ AND } [\text{closeness}(o_{pj}) \leq \epsilon_1]$$

where noncontributing(j) is defined as

$$\text{noncontributing}(j) = [(o_{pj} \geq 0.5) \text{ OR } (o_{pj} < 0.5), \forall p$$

and closeness(o_{pj}) is defined as

$$\text{closeness}(o_{pj}) = \max_{p_1,p_2}(|o_{p_1 j} - o_{p_2 j}|)$$

Excessive Duplicated unit : It is a unit that has its weight vector that converges to the same to another unit in the same layer. It can be detected by :

$$\text{excessiv-duplicated}(i,\ j) = \text{duplicated}(i,\ j) \text{ AND } [\text{diff}(o_{pi}, o_{pj}) \leq \epsilon_2]$$

where duplicated(i, j) is defined as

$$\text{duplicated}(i,\ j) = [(o_{pi} - 0.5)(o_{pj} - 0.5) > 0,\ \forall p]$$

which detects the two units have the similar classification ability to all training patterns. And diff(o_{pi}, o_{pj}) is defined as

$$\text{diff}(o_{pi}, o_{pj}) = \max_{p}(|o_{pi} - o_{pj}|)$$

in order to show the difference between the outputs of the two duplicated units in all patterns. If the maximum difference is less than the tolerant threshold ϵ_1 and the two units duplicate each other, the hidden units are identified as the excessive duplicated unit. The classification ability of the two duplicated units can be said to be almost the same, if the maximum difference is close to zero.

Excessive Inversely-duplicated unit : The inversely-duplicated unit are similar to the duplicated unit described above. The transformed output of the input patterns of an inversely-duplicated unit have opposite properties to another unit in the same layer. It can be detected by

$$\begin{aligned} &\text{excessive-inversely-duplicated}(i,\ j) \\ = \ &\text{inversely-duplicated}(i,\ j) \text{ AND } [\text{diff}(inv(o_{pi}), o_{pj}) \leq \epsilon_3] \end{aligned}$$

where inversely-duplicated(i, j) is defined as

$$\text{inversely-duplicated}(i,\ j) = [(o_{pi} - 0.5)(o_{pj} - 0.5) < 0,\ \forall p]$$

and $inv(o_{pj})$ is defined as

$$inv(o_{pj}) = 1 - o_{pj}$$

Excessive Inadequate unit : The excessive inadequate unit performs partially correct classification to the input patterns, as a result, the generalization performance would be degraded. Therefore, this kind of node should be detected and removed from the solution network. There are two conditions to satisfy the detection rule of excessive-inadequate units : same-merge and not-mix-merge. As the separated same class patterns will be merged by the other units, this process is called the same-merge process. However, one should beware that the removal of an inadequate unit that performs the bad classification to the same class patterns should not result to the situation of merging different classes patterns, therefore, the situation of not-mix-merge must also be satisfied. The detection rule of excessive-inadequate is showed as follow :

$$\text{excessive-inadequate}(j) = \text{not-mix-merge}(j) \text{ AND same-merge}(j)$$

where not-mix-merge(j) is defined as

if (p_1 and p_2 are of the different classes) AND $[(o_{p_1 j} - 0.5)(o_{p_2 j} - 0.5) < 0]$

THEN not-mix-merge(j)

$= [(o_{p_1 i} - 0.5)(o_{p_2 i} - 0.5) < 0 \text{ OR } |o_{p_1 i} - o_{p_2 i}| < \epsilon_4 \text{ for some } i \neq j]$

and same-merge(j) is defined as

if (p_1 and p_2 are of the same classes) AND $[(o_{p_1 j} - 0.5)(o_{p_2 j} - 0.5) < 0]$

THEN same-merge(j)

$= [(o_{p_1 i} - 0.5)(o_{p_2 i} - 0.5) < 0 \text{ AND } |o_{p_1 i} - o_{p_2 i}| < \epsilon_5 \ \forall i \neq j]$

$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5$ are the trade-off thresholds between the network size and its generalization performance of different pruning conditions. More excessive units would be identified and pruned with larger thresholds, and also, the generalization performance would be affected as well.

4.2.2 Constructive Algorithms (Growing)

Apart from pruning algorithms, constructive algorithms are the methods to search for the suitable architecture from a minimal network. These methods expand the network architecture from layer to layer or in the same layer by introducing new hidden unit. Different constructive algorithms have their unique expanding topologies. In this study, we consider the constructive algorithms for a single hidden layer neural network model, and therefore, the unit expansion will be performed at the same layer. Our model is a modification of SplitNet[42]

The output of a hidden unit for different input patterns during a learning process, may be classified into three possible states. They are the present (yes state), not present (no state) and maybe state. The hidden unit is said to be a "yes" state for an input pattern when the input to the sigmoid hidden unit is greater than 1.4 (the output to sigmoid hidden unit is greater than 0.8), where a sigmoid hidden unit is an activation function that limits the output values between 0 and 1 as shown in Eq.4.1.

$$output = \frac{1}{1 + \exp(-input)} \tag{4.1}$$

The hidden unit is said to be a "no" state when the input is less than -1.4 (the output is less than 0.2). The "maybe" state is defined for the hidden unit which has the input

between -1.4 and 1.4 for a input pattern (the output lies between 0.2 and 0.8). The reasons of using the input to the sigmoid hidden unit rather than the output are due to the faster computational speed and more accurate cutoff values. The SplitNet splits a hidden unit into two units when the ratio of the number of "maybe" state to the total number of input patterns is greater than 0.6 in the last 50 iterations, as shown in the following equation

$$\frac{\text{number of maybe state}}{\text{total number of input patterns}} > 0.6, \tag{4.2}$$

SplitNet is a constructive algorithm designed for a fully interconnected neural network with only one output unit. It was tested with the parity problem in [42]. A parity problem is a binary classification problem that classifies the number of zeros in a string of binary digits, where the output is 1 if there is an odd number of zeros; otherwise the output is 0. The performance of SplitNet was compared with the fixed architecture network in [42], and shown that it performed better in terms of improvement in the time complexity (where the time complexity was measured according to the required iteration runs).

4.2.3 Integration of methods

The combination of constructive algorithm (Cascade Correlation, Cascor) and pruning algorithm (Optimal Brain surgeon, OBS) in order to control the growth of an expanding network architecture was introduced in [33]. Cascor is a constructive algorithm to build a feed-forward network that begin with a small network. New hidden units are added one by one to create a multi-layer architecture[6]. The optimal

brain surgeon is a pruning algorithm to remove unimportant weights from a solution network based on the information from all second derivatives of the error function[32, 33, 35]. This approach is to control the growth of Cascor in which each new hidden unit is pruned before it is added to the network[33]. The architecture of the solution network is smaller than the one found solely by the Cascor method and has the advantage of avoiding the overfitting problem.

However, this method results in a multi-layer feed-forward network, which is different from our consideration, a single hidden layer feed-forward network. Therefore, we use this related work as our survey rather than as a comparison methods.

4.3 Methodology and Approaches

In this study, we want to find a flexible method to obtain suitable network architectures for different problem domains using single hidden layer backpropagation feedforward neural networks. The recent methods to decide the network size are the pruning algorithms, constructive algorithms, trial-and-error. As pruning algorithms and constructive algorithms have their limitations, an approach that combines both pruning and constructive algorithms would be a possible solution.

4.3.1 Growing

When we reviewed the model of SplitNet[42], we found that the growth of the hidden unit might lead to the infinite growing of the duplicated units. Also, we observed that the network would often fall into the local minimum when all the

inputs to every hidden unit are all greater than 1.4 ("yes" state) or lesser than -1.4 ("no" state). In other words, all the input patterns are classified into the same group by these hidden units. This situation is similar to the network that is composed of non-contributing units only. In order to deal with this situation, we will introduce a new hidden unit to the network, only if the network satisfies Eq. 4.3 for each review period.

$$\begin{aligned} &\text{Grow\_new\_unit} \\ &\text{IF (number of yes-state}_i = \text{total number of input patterns} \\ &\text{AND number of no-state}_i = \text{total number of input patterns)} \ \forall i \end{aligned} \tag{4.3}$$

where i denotes the number of hidden units. Fig.4.1 shows the process flow of the growing method. Since a larger network architecture is constructed after growing, the enlarged network may need longer time to learn the training patterns than a smaller one. For this reason, as the network grows, the review on the need of growing will be made after a longer training iterations has been proceeded. To prevent the infinite training occurrence, the network will stop training if there is no growing to the network after a certain number iterations have been proceeded.

4.3.2 Combinations of Growing and Pruning

The same problem of excessive units would occur in the solution network found by constructive algorithms, since the constructive algorithms only consider when to expand but may add unnecessary units to the network. The excessive units have

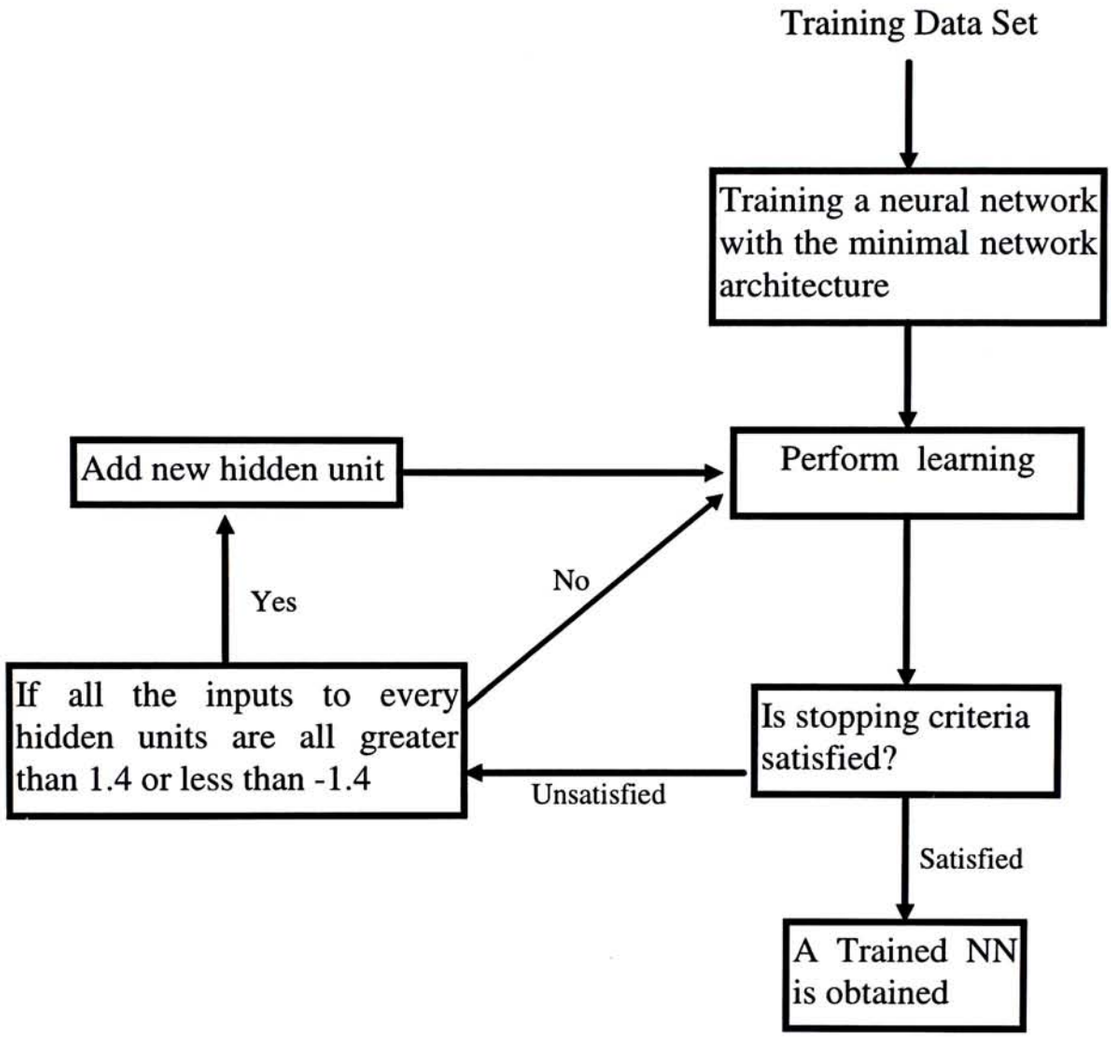

Figure 4.1: Process Flow of the Growing Method

problems such as the computation problem and the storage problem. In the following, we propose two possible ways to solve these problems.

Growing then Pruning

This approach makes use of the pruning with retraining method discussed in Section II. After the solution network is obtained from a growing approach, a pruning process is incorporated to remove the non-contributing units, duplicated units, inversely-duplicated units, and inadequate units from the solution network. The pruned solution network will be retrained if the classification performance are degraded or unable to achieve a certain level. Fig.4.2 shows the process flow.

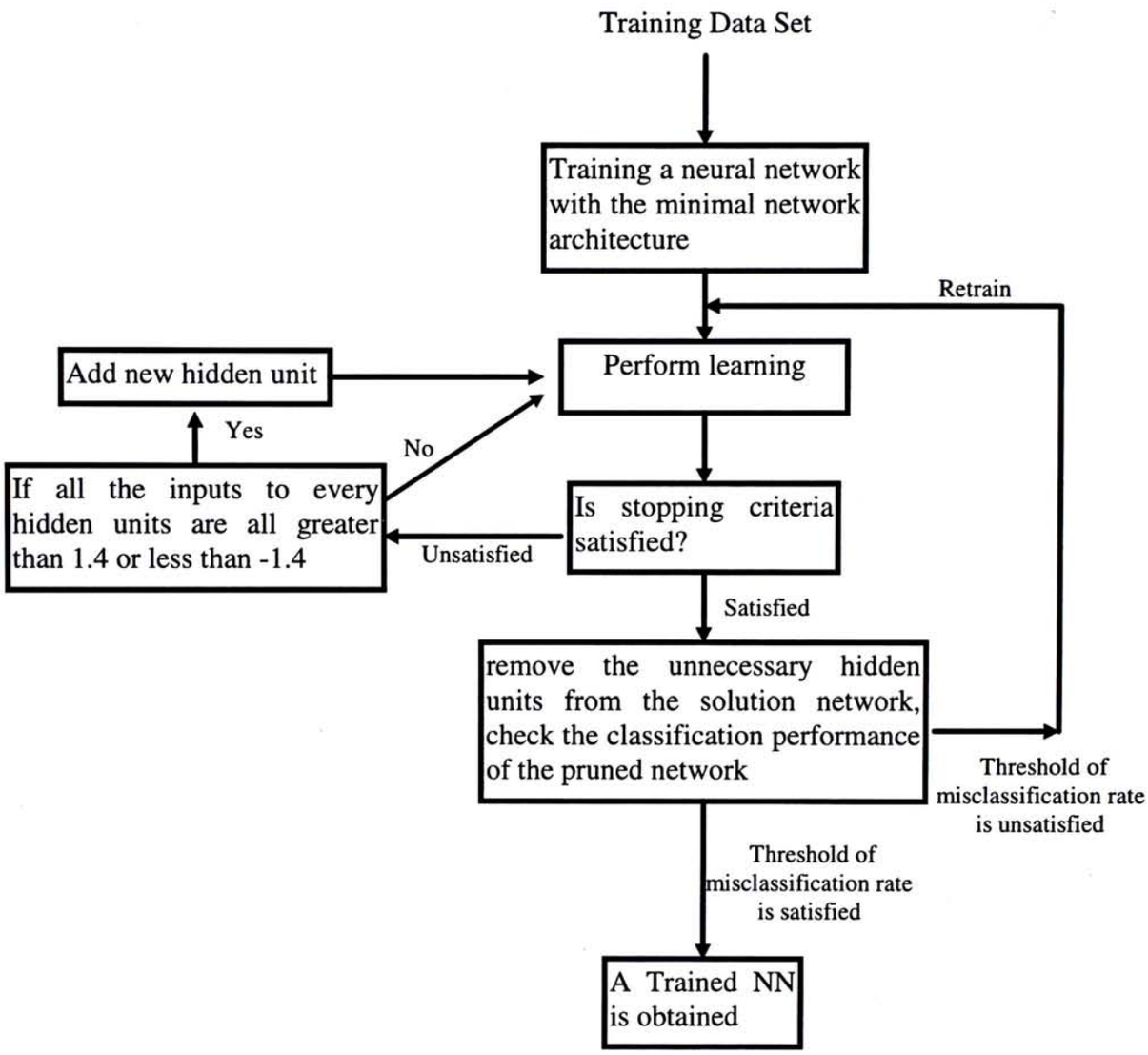

Figure 4.2: Process Flow of Growing then Pruning Method

Growing while Pruning

The previous method is not flexible enough to obtain the minimal solution network architecture as many unnecessary units are included in the network during a learning process. When a network grows to a certain size, the computation cost will be increased. If a network contains many unnecessary units, the disadvantages of large computation overhead become apparent. Therefore, if the unnecessary units can be managed during the learning and growing processes, we can effectively control computational requirement. Fig.4.3 shows the process flow of a better method.

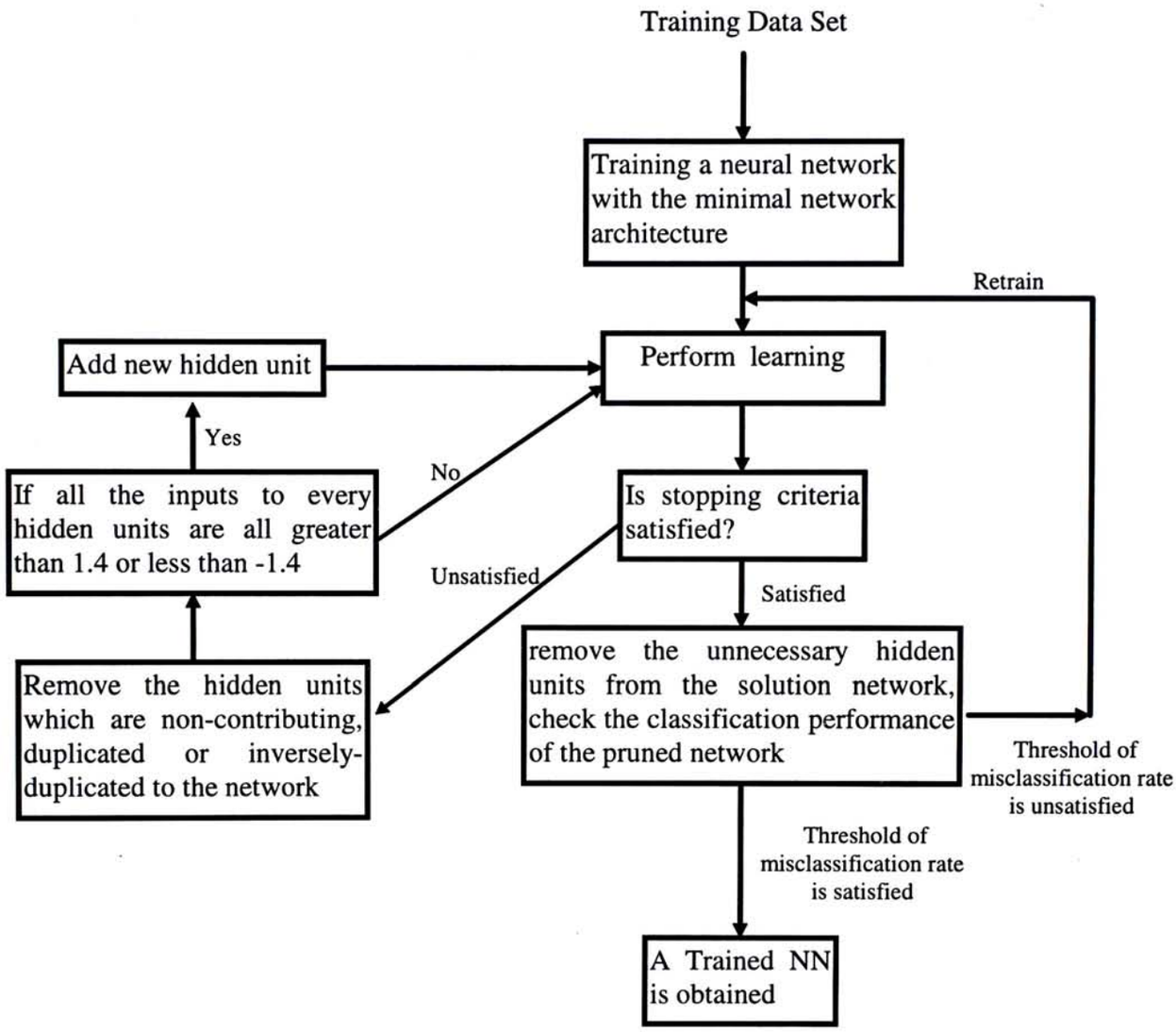

Figure 4.3: Process Flow of Growing while Pruning Method

When the growing proceeds, the network may fall into a local minimum when all the units are non-contributing units. And also, when the duplicated or inversely-duplicated units are not significant to the learning process as the discriminant information are duplicated. Therefore, the removal of those excess non-contributing units, duplicated units and inversely-duplicated units before the growing proceeds can reduce the computational costs as a whole. As a result, we can always obtain a nearest minimal architecture for training data set once the solution network is obtained. We do not consider the inadequate units in the pruning that proceeds before growing because the inadequate units always provide unique information to the learning function unless they are duplicated by the other units.

In summary, we evaluate three proposed methods, growing method, growing then pruning method, and growing while pruning method in the following section. Three developed methods by previous literature are used for comparisons in order to show the different performance by different methods. The three developed methods are fixed architecture network, one pass pruning method, and pruning with retraining method.

4.4 Experimental Results

In order to compare the overall performance of different models, the evaluations on the generalization performance or prediction accuracy, time complexity required to obtain the solution network and the network size of the solution network are investigated. We compare the performance of the conventional trial-and-error method

with oversized network, the pruning methods, the growing method, and the methods integrating pruning and growing algorithms.

In our experiments, we use two databases: breast mass cytology information, and encoded tic-tac-toe board configurations. The first database is used in the last chapter. This database contains "noise" data. The second database does not contain "noise" data.

4.4.1 Breast-Cancer Cytology Database

This data sample contains patients' breast mass cytology information. These patients were admitted for breast cancer diagnoses at the University of Wisconsin(Madison) from 1989 to 1992[2, 3, 4, 40]. There are 699 instances in the database. We use 683 instances for our experiment. Sixteen instances are removed due to missing values. Each instance comprises 9 attributes and 1 class value. These 9 attributes were collected when Fine-Needle Aspirate (FNA) testing were taken from each patient's breast mass[4]. These attributes are shown in Table 4.1.

Each attribute is a scalar observation. An attribute value shows the likelihood of malignancy; the larger the value, the greater the likelihood of malignancy. The class value tells us whether the symptom is benign or malignant for the corresponding instance. In total, 444 instances are in the class of benign breast masses and 239 instances are in the class of malignant breast masses.

In each training sample set, 200 instances (100 benign and 100 malignant) are randomly selected. For each testing sample set, 169 instances (84 benign and 85

Attribute	**Domain**
Clump Thickness	1-10
Uniformity of Cell Size	1-10
Uniformity of Cell Shape	1-10
Marginal Adhesion	1-10
Single Epithelial Cell Size	1-10
Bare Nuclei	1-10
Bland Chromatin	1-10
Normal Necleoli	1-10
Mitoses	1-10

Table 4.1: Attribute descriptions of Breast-cancer Database

malignant) are randomly selected. Forty-five couples of training and testing data sample sets are generated from the 683 instances and are used to test different models. Since the solution networks rarely contain one hidden unit, we set the number of initial hidden units to be 2. Moreover, the number of hidden units for the oversized network is set to 18 by trial-and-error. This oversized network provides a promising solution.

Generalization Performance

Table B.1 contains the result of generalization performance of each of the following methods:

(1) : Fixed Architecture Neural Network

(2) : One Pass Pruning

(3) : Pruning with Retraining

(4) : Growing (trial 1)

(5) : Growing (trial 2)

(6) : Growing then Pruning (trial 1)

(7) : Growing then Pruning (trial 2)

(8) : Growing while Pruning (trial 1)

(9) : Growing while Pruning (trial 2)

where trial 1 and 2 mean the two tests on the same training data set with the same method.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Average Accuracy	90.20	90.26	91.29	91.09	91.03	91.31	91.91	91.43	91.40
Median	90.53	90.53	91.12	91.72	91.72	91.12	91.72	91.72	91.12
Minimum	81.07	81.07	85.21	82.84	84.62	85.21	85.21	84.02	86.39
Maximum	95.27	95.86	96.45	95.86	95.27	96.45	96.45	96.45	95.86
Standard Deviation	2.82	2.90	2.49	2.61	2.46	2.54	2.53	2.76	2.41

Table 4.2: Statistics of Accuracy of Different Algorithms : Breast-cancer database

From Table 4.2, we observe that the average generalization performances of the fixed architecture and one pass pruning methods are about 1% less accurate than the

other four methods. These show the generalization ability of a smaller network architecture may have a better performance than a larger one on average. Furthermore, the performance of the pruned network with retraining process is always better than the one without retraining process. Therefore, when we apply pruning to the growing method, we consider the one with retraining process.

Time Complexity

Table B.2 contains the time complexity required by each method

(1) : Fixed Architecture Neural Network

(2) : Growing (trial 1)

(3) : Growing (trial 2)

(4) : Growing then Pruning (trial 1)

(5) : Growing then Pruning (trial 2)

(6) : Growing while Pruning (trial 1)

(7) : Growing while Pruning (trial 2)

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Average Time Consuming	158.77	273.19	447.08	1269.85	494.37	19.18	20.00
Median	36.37	26	63.62	158.43	111.53	4.64	7.83
Minimum	1.52	0.27	0.93	1.47	2.85	0.60	0.54
Maximum	1070.80	2356.9	5904.12	8144.93	2527.67	243.06	208.02
Standard Deviation	230.66	542.16	994.44	2243.46	738.29	42.18	34.52

Table 4.3: Statistics of Time Complexity Required of Different Algorithms : Breast-cancer database

We measure the time complexity by clock time. It is different from some researchers who use the learning iterations. We think that the overhead of those prun-

ing and growing methods that analyze the network during a learning process must also be considered. In Table 4.3, the least time consuming method on average is growing while pruning, followed by the fixed architecture network, growing, and the growing then pruning, respectively. Since the two pruning methods are performed on the fixed architecture network, the time consumed by those pruning methods would only be a little longer than the fixed architecture network; and therefore, we will not consider the time consumed of solely appending the pruning methods to the fixed architecture network. It should be noted that the minimum time consumed by the growing, and the two growing and pruning integrated methods are always less than the one by the fixed architecture network. This is because the time spent on a small solution network is less than the time spent on an oversized network as the computation cost can be greatly reduced in a small network. As the network grows larger and larger, the computational cost and time consumed will increase. Once the network size approaches to the size of the fixed architecture network, the time consumed is likely to be larger than the fixed architecture one as the time to search of the solution space increases. The time consumed by the growing then pruning method is larger than the growing method, since the pruning is an computational overhead to the growing method. However, we can observe that the time consumed by the growing while pruning method give a definitely time advantage. Due to the pruning of the unnecessary units from the network during the searching of the solution space, the network can be remained with a small suitable size when the learning proceeds.

Network Size

Table B.3 contains the network size obtained by each method.

(1) : One Pass Pruning

(2) : Pruning with Retraining

(3) : Growing (trial 1)

(4) : Growing (trial 2)

(5) : Growing then Pruning (trial 1)

(6) : Growing then Pruning (trial 2)

(7) : Growing while Pruning (trial 1)

(8) : Growing while Pruning (trial 2)

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Average Network Size	5.44≈6	1.51≈2	7.29≈8	8.78≈9	1.22≈2	1.22≈2	1.60≈2	1.58≈2
Median	5	1	6	7	1	1	2	1
Minimum	1	1	2	2	1	1	1	1
Maximum	14	3	19	29	2	2	3	3
Standard Deviation	4.07	0.65	4.19	5.20	0.42	0.42	0.53	0.65

Table 4.4: Statistics of Network Size of Different Algorithms : Breast-cancer database

The statistics on the size of the solution networks obtained by those methods is presented in Table 4.4. It shows that the average smallest network architectures are always provided by the methods pruning with retraining, those growing methods with pruning process. The one pass pruning method gives a second biggest average network architecture, since the one pass pruning method does not consider the removal of the inadequate units. The growing method gives the biggest average network architecture because the excessive units will not be removed. If we compare the aver-

age network architecture first obtained (no pruning performed) between the growing method and the growing with pruning method, the result would be very close, as the second method is built up on the foundation of the growing method. No matter what the difference size achieved by those network architecture altering methods are, we observed that these kinds of solution networks are much smaller than the fixed architecture one, 18 hidden units. From Table 4.4, we can observe that the maximum obtained network architecture in (3) and (4) are 19 and 29, by which we know that sometimes we need a larger network architecture for a specific problem space. To deal with such kind of problem, the suitable methods should be the growing based methods.

4.4.2 Tic-Tac-Toe Database

This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first[40]. The target concept is "win for x", where it is true when "x" has one of 8 possible ways to create a "three-in-a-row". There are 958 instances which are the legal tic-tac-toe endgame boards, out of which 626 instances are the group of "win for x" and 332 instances are the group of "not win for x". The 9 attributes each corresponds to one tic-tac-toe square and have been shown in Table 4.5, where x = player x has taken, o = player o has taken and b = blank.

The two classes are defined as 1 ("win for x") and 0 ("not win for x"). The attribute information {x, o, b} is represented by integer set {2, 0, 1} respectively as

Attribute	Domain
top-left-square	{x, o, b}
top-middle-square	{x, o, b}
top-right-square	{x, o, b}
middle-left-square	{x, o, b}
middle-middle-square	{x, o, b}
middle-right-square	{x, o, b}
bottom-left-square	{x, o, b}
bottom-middle-square	{x, o, b}
bottom-right-square	{x, o, b}
Class	{1, 0}

Table 4.5: Attribute descriptions of Tic-Tac-Toe Database

the input for the network.

In each training sample set, 400 instances (200 in each group) are randomly selected. For each testing sample set, 200 instances (100 in each group) are randomly selected. Forty-five couples of training and testing data sample sets are generated from the 958 instances, and are used to test different models. The number of initial hidden units is set to 1 as the network sometimes obtains its solution network in this database. Moreover, the number of hidden units for the oversized network is set to 7 by trial-and-error thus always providing a promising solution network.

Generalization Performance

Table B.4 contains the result of generalization performance of each method.

(1) : Fixed Architecture Neural Network

(2) : One Pass Pruning

(3) : Pruning with Retraining

(4) : Growing (trial 1)

(5) : Growing (trial 2)

(6) : Growing then Pruning (trial 1)

(7) : Growing then Pruning (trial 2)

(8) : Growing while Pruning (trial 1)

(9) : Growing while Pruning (trial 2)

From Table 4.6, we find that the average generalization performances can be divided into four groups. The most accurate methods are the growing methods with pruning process introduced. The second are the growing method and the pruning with retraining method. Then the fixed architecture network and the one pass pruning method are followed. The one pass pruning gives an unsatisfied generalization

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Average Accuracy	88.23	83.21	89.93	89.16	90.73	91.94	90.74	91.03	90.81
Median	88.50	85.00	91.00	89.00	91.00	93.00	91.00	91.50	91.00
Minimum	78.50	50.00	82.00	81.50	81.00	85.50	78.00	80.50	83.00
Maximum	96.00	96.00	96.00	95.50	99.50	97.00	98.50	97.00	98.00
Standard Deviation	4.52	8.79	3.92	3.59	3.71	2.90	4.94	3.47	3.71

Table 4.6: Statisitics of Accuracy of Different Algorithms : Tic-Tac-Toe database

performance which is caused by the degraded solution network after pruning process is performed. Again, this shows that the pruning with retraining method would be more reliable than the one pass method. The statistics in Table 4.6 also shows that the network architecture altering methods except the one pass pruning method always performs better than the fixed architecture network.

Time Complexity

Table B.5 contains the time complexity required by each method.

(1) : Fixed Architecture Neural Network

(2) : Growing (trial 1)

(3) : Growing (trial 2)

(4) : Growing then Pruning (trial 1)

(5) : Growing then Pruning (trial 2)

(6) : Growing while Pruning (trial 1)

(7) : Growing while Pruning (trial 2)

Table 4.7 shows the statistics of time complexity required by those methods other than the pruning methods. The statistics gives a similar result to the breast-cancer database, except that the average time complexity required by the fixed architecture

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Average Time Consuming	165.67	68.83	144.13	264.53	124.29	54.18	46.91
Median	124.30	44.36	47.28	60.74	49.17	45.40	43.38
Minimum	56.31	13.97	14.88	15.32	19.89	15.72	17.14
Maximum	791.02	373.14	2442.05	5736.17	672.73	278.36	119.20
Standard Deviation	125.51	72.72	378.93	861.07	161.64	43.02	23.59

Table 4.7: Statistics of Time Complexity Required of Different Algorithms : Tic-Tac-Toe database

network is longer than the growing method in tic-tac-toe database. Since the required network size for the tic-tac-toe classification problem is relatively small, the smaller solution network can be obtained by the growing method. The smaller the network size during the learning stages, the less computational costs and time complexity there will be. The growing while pruning method shows again that we can obtain a better performance in time saving over the other methods.

Network Size

Table B.6 contains the network size obtained by each method.

(1) : One Pass Pruning

(2) : Pruning with Retraining

(3) : Growing (trial 1)

(4) : Growing (trial 2)

(5) : Growing then Pruning (trial 1)

(6) : Growing then Pruning (trial 2)

(7) : Growing while Pruning (trial 1)

(8) : Growing while Pruning (trial 2)

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Average Network Size	3.00	3.00	2.31≈3	3.02≈3	1.71≈2	1.89≈2	1.73≈2	1.69≈2
Median	3	3	2	2	2	2	2	2
Minimum	1	1	1	1	1	1	1	1
Maximum	5	5	7	11	4	4	4	3
Standard Deviation	1.37	1.40	1.24	1.96	0.75	0.77	0.77	0.63

Table 4.8: Statistics of Network Size of Different Algorithms : Tic-Tac-Toe database

The statistics of the solution network size obtained by different methods is showed in Table 4.8. It is quite different from the result of breast-cancer database showed in Table 4.4. In tic-tac-toe database, the average network size obtained by each network architecture altering methods is very close. Since the tic-tac-toe database is noise-free database and the patterns are quite simple, we can use a quite small network architecture to obtain the desirable discriminant ability. However, we can observe

that the larger maximum network size is also obtained by the growing method, 7 and 11 hidden units respectively in columns (3) and (4). This means that although we can achieve the solution network with the network architecture altering methods, sometimes we still need to use a larger network to search for the solution space.

The following are the descriptions of Table 4.9-4.11:

	(1)	(2)	(3)	(4)	(5)	(6)
Breast-cancer	90.20	90.26	91.29	91.06	**91.61**	91.42
Tic-Tac-Toe	88.23	83.21	89.93	89.95	**91.34**	90.92

Table 4.9: Summaries of Average Accuracy of Different Database

	(1)	(4)	(5)	(6)
Breast-cancer	158.77	360.14	882.11	**19.59**
Tic-Tac-Toe	165.67	106.48	194.41	**50.19**

Table 4.10: Summaries of Average Time Complexity Required of Different Database

	(2)	(3)	(4)	(5)	(6)
Breast-cancer	5.44≈6	**1.52≈2**	8.04≈9	**1.22≈2**	**1.59≈2**
Tic-Tac-Toe	3.00	3.00	2.67≈3	**1.80≈2**	**1.71≈2**

Table 4.11: Summaries of Average Netowrk Size of Different Database

Tables 4.9-4.11 summarize of the average performance on the accuracy, time complexity and the network size achieved by different methods :

The tables show that the growing while pruning method is flexible and reliable in obtaining an accurate and suitable-sized network, although it is only the second best in terms of generalization performance.

(1) : Fixed Architecture Neural Network

(2) : One Pass Pruning

(3) : Pruning with Retraining

(4) : Average of Growing (trial 1) & Growing (trial 2)

(5) : Average of Growing then Pruning (trial 1) & Growing then Pruning (trial 2)

(6) : Average of Growing while Pruning (trial 1) & Growing while Pruning (trial 2)

4.5 Conclusion

In this research, we want to find a flexible and reliable approach to obtain a suitable network architecture for a problem domain automatically. By evaluating on the performance in the generalization performance, time consuming and network size of some different network architecture altering methods and the fixed network architecture method, we find that the growing based methods always provide a better generalization performance than the other. From the experimental results, we find that the fixed architecture method and the pruning method to the fixed architecture network do not provide an overall satisfactory performance. Moreover, these methods need to perform the trial-and-error test on the network architecture before they proceed learning. Therefore the pruning methods are not a flexible way to obtain the suitable network architecture. However, the growing method and the growing then pruning method often take a longer time to obtain a solution network than the other. And also, the growing method may produce a network with the unnecessary units.

The growing while pruning method avoids the problems of the others according to the performance on the generalization ability, time consuming and network size obtained. The results show that this method gives a promising performance, such

that we can obtain the suitable network architecture by this flexible and reliable approach. As the growing while pruning method proceeds the removals to excessive non-contributing, duplicated and inversely-duplicated units before the growing analysis, the non-contributing and duplicated information can be ignored. As a result, the network can be always kept in a small but information unique network architecture; in the mean time, the network can remain in the economic computational state so as to minimize the time consuming for the whole process. Afterward, the last pruning process will be performed to the obtained solution network, so that the unnecessary units will not appear in the final solution network.

From our experiments, we determined that since the constructive algorithms prefers the smaller solution network, therefore, constructive algorithms are the most flexible in obtaining the suitable network size. However, to deal with the unnecessary units in the solution network, the pruning algorithms should be introduced also to the solution network. If the time consuming issue is considered, the removals to those units that do not provide unique information should be involved, such that the computation cost can be reduced. In this chapter, we find that growing while pruning is one of the possible ways to achieve above flexibility and reliability requirements, and would be a possible direction to assisting other constructive algorithms and pruning algorithms.

Chapter 5

Conclusion

5.1 Recall of Thesis Objectives

In this research, we address for two limitations in using neural network as a classifier for the binary classification problem : the data dependency and the unknown network size.

Typically, training to a neural network is very data dependent. The generalization performance of a solution network is affected by the distribution of the training data set. Different ratio between the number of classes and the level of noisy data involved would provide different classification ability. Although, many literature suggest that introducing some noisy data can give an ability of immunity to the noisy data when a solution network is operated with the unseen data, the great amount of noisiness in the practical environment would lead to a poor generalization performance. Therefore, one of the objectives in this research is to find a way to minimize the effect of this vagueness from a large amount noisy data.

The choice of network size is a usual problem to the users of neural network. Training always involves trial-and-error tests to find a promising network size provides a good generalization performance. A small network may not achieve the desirable prediction ability, but an oversized network may suffer from the problems of memorizing and overfitting. However, the trial-and-error method often suggests an oversized network for the training. In order to avoid this tedious trial-and-error tests, we look for other methods that are flexible and reliable in deciding the network size.

5.2 Summary of Achievements

5.2.1 Data Preprocessing

Training of a network is a data dependent process. When neural networks are applied to practical environments, the embedded vagueness in the data would affect the generalization performance of neural networks. In chapter 3, we propose a hybrid binary classification model that is composed of fuzzy linear regression with fuzzy interval analysis(FLRFIA) and single hidden-layer backpropagation feedforward neural network(BPNN). To minimize the effect of the vagueness data, FLRFIA acts as a data handler that separates the data into two groups : data lie inside or outside the interval. The data lie inside the interval are identified as the certain data, while the data lie outside the interval are identified as the vagueness data. As the training data set is separated into two groups, two signal hidden-layer BPNN models are used. One is used for classifying the certain data, and the other is used for classifying the

vagueness data.

The experimental results show that the proposed hybrid model performs better than using a single hidden-layer BPNN in the binary classification problem. Since the main classifiers in the hybrid model are the same as the conventional BPNN, we conclude that the data handler, FLRFIA, plays an important role in the improvement of the generalization performance.

On the other hand, before we can separate the data into the certain data and the vagueness data, the linear programming model must be constructed and solved for the entire training data. This procedure is quite a time consuming process.

5.2.2 Network Size

In chapter 4, we study the problem of searching a desirable network size for the single hidden-layer BPNN in the binary classification problem. In this research, we use three growing based methods for the single hidden-layer BPNN, growing method, growing then pruning method, and growing while pruning method. These growing based methods find the better generalization performance improvement when compared with the fixed network architecture method and pruning methods. The growing while pruning method provide the best overall performance among the other methods that considered in this research. The experimental results show that it is possible to find a flexible and reliable method by using both the constructive and pruning algorithms to compensate the limitations of each other.

5.3 Future Works

In this research, we consider the binary classification problem only. To generalize our research, the extension to the multi-class classification problem could be studied.

The hybrid classification model we proposed in chapter 3 is to minimize the effect of the vagueness data to the network. The proposed model achieves this objective by applying FLRFIA. In general, we can use other methods to reduce the complexity and time taken to construct and solve the linear programming model. For example, we may consider taking n samples of training data set. For each sample, we train a network. The prediction will be jointly by all the networks. The decision can be made by majority votes.

There are many different network size models available in the literature. In this research, we show that some combinations of models may provide better size networks. Many other combinations may be considered. A comprehensive evaluation of these combined models will be interesting.

Appendix A

Experimental Results of Ch.3

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**91.12**	**91.12**
2	**86.98**	**92.31**
3	**87.57**	**89.35**
4	**92.90**	**93.49**
5	**86.39**	**87.57**
6	**88.17**	**89.94**
7	**88.76**	**89.35**
8	**92.90**	**95.27**
9	**91.72**	**91.72**
10	94.67	94.08
11	**94.67**	**95.86**
12	**88.17**	**91.12**
13	**89.35**	**90.53**
14	**88.76**	**93.49**
15	90.53	89.94
16	**88.76**	**88.76**
17	**92.90**	**94.67**
18	**85.80**	**88.17**
19	**89.94**	**91.72**
20	**90.53**	**91.12**
21	**89.35**	**91.12**
22	**90.53**	**91.12**
23	**87.75**	**88.17**

Table A.1: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**88.76**	**91.72**
25	**87.57**	**88.17**
26	**94.08**	**94.67**
27	**87.57**	**89.94**
28	**86.98**	**88.17**
29	**93.49**	**93.49**
30	**87.57**	**87.57**
31	92.90	92.31
32	89.94	89.35
33	**90.53**	**93.49**
34	**91.72**	**92.90**
35	**94.08**	**94.08**
36	94.08	93.49
37	**90.53**	**91.12**
38	**89.94**	**92.90**
39	**92.90**	**93.49**
40	**86.39**	**86.98**
41	**86.98**	**88.76**
42	**91.72**	**93.49**
43	**94.08**	**95.86**
44	**91.12**	**94.08**
45	**91.72**	**93.49**
Average	*90.28*	*91.55*

Table A.1: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**87.76**	**91.12**
2	**90.53**	**91.72**
3	89.35	88.76
4	**94.08**	**94.67**
5	**86.98**	**88.76**
6	90.53	89.94
7	**87.57**	**89.35**
8	**86.98**	**96.45**
9	**91.72**	**92.31**
10	**93.49**	**94.08**
11	**90.53**	**90.53**
12	**91.12**	**92.90**
13	**91.12**	**93.49**
14	**81.07**	**87.57**
15	**85.80**	**86.98**
16	**90.53**	**92.31**
17	**94.08**	**95.86**
18	**90.53**	**94.08**
19	**92.31**	**95.27**
20	**89.35**	**89.94**
21	90.53	89.94
22	**86.98**	**88.17**
23	**89.35**	**91.72**

Table A.2: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**86.39**	**87.57**
25	**94.67**	**94.67**
26	**86.98**	**89.35**
27	**86.39**	**89.35**
28	**92.90**	**93.49**
29	89.94	89.35
30	**95.27**	**95.86**
31	**92.31**	**93.49**
32	**89.94**	**91.12**
33	**91.72**	**92.90**
34	**88.17**	**90.53**
35	89.35	88.76
36	**93.49**	**94.67**
37	**86.98**	**88.76**
38	**91.72**	**92.31**
39	89.35	88.17
40	**90.53**	**92.90**
41	90.53	89.94
42	**88.76**	**91.72**
43	**92.90**	**93.49**
44	**93.49**	**95.27**
45	**94.08**	**94.08**
Average	*90.18*	*91.64*

Table A.2: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**89.94**	**92.90**
2	**91.12**	**92.31**
3	89.35	88.17
4	**95.27**	**95.86**
5	**86.39**	**88.17**
6	**89.35**	**89.94**
7	**87.57**	**89.94**
8	**91.72**	**97.63**
9	**89.94**	**90.53**
10	**93.49**	**94.08**
11	91.12	89.35
12	**88.76**	**91.72**
13	**93.49**	**94.67**
14	**84.62**	**86.39**
15	**86.39**	**87.57**
16	**91.72**	**92.90**
17	**92.31**	**95.27**
18	**89.35**	**92.90**
19	**92.90**	**95.27**
20	**89.94**	**91.72**
21	91.12	90.53
22	**86.98**	**89.35**
23	**88.76**	**89.94**

Table A.3: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**85.80**	**88.76**
25	95.86	94.67
26	**88.76**	**89.94**
27	**85.80**	**89.35**
28	**92.90**	**92.90**
29	**90.53**	**90.53**
30	**92.31**	**93.49**
31	**91.72**	**94.08**
32	**91.12**	**91.72**
33	**89.94**	**92.90**
34	**89.94**	**92.31**
35	89.35	88.76
36	**92.90**	**94.67**
37	**86.39**	**87.57**
38	**92.31**	**93.49**
39	**88.76**	**90.53**
40	**91.12**	**92.31**
41	**89.94**	**90.53**
42	**91.72**	**94.67**
43	**92.90**	**94.08**
44	**92.90**	**94.67**
45	**94.08**	**94.08**
Average	*90.41*	*91.85*

Table A.3: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**90.53**	**94.67**
2	**91.12**	**94.08**
3	**88.76**	**89.35**
4	**94.08**	**94.08**
5	**84.62**	**86.39**
6	**89.35**	**89.94**
7	**87.57**	**88.17**
8	**92.90**	**95.86**
9	**92.31**	**92.31**
10	**91.72**	**92.90**
11	**91.12**	**91.72**
12	**88.17**	**89.35**
13	**92.31**	**92.90**
14	**86.98**	**89.35**
15	**86.39**	**88.76**
16	**90.53**	**92.90**
17	**92.31**	**94.67**
18	**91.72**	**94.67**
19	**92.90**	**94.08**
20	**90.53**	**92.90**
21	**89.94**	**91.12**
22	**88.17**	**91.72**
23	**90.53**	**91.72**

Table A.4: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**86.39**	**87.57**
25	**95.27**	**95.27**
26	**88.17**	**88.17**
27	**82.25**	**87.57**
28	**89.35**	**91.12**
29	**88.17**	**88.17**
30	**92.90**	**94.67**
31	**92.31**	**95.27**
32	**91.72**	**91.72**
33	**90.53**	**94.08**
34	**91.12**	**92.31**
35	**88.17**	**89.35**
36	**93.49**	**94.67**
37	**86.98**	**88.17**
38	**91.72**	**92.31**
39	**89.94**	**90.53**
40	91.72	91.12
41	**89.94**	**90.53**
42	**91.72**	**94.08**
43	**92.90**	**94.67**
44	**92.31**	**94.67**
45	**93.49**	**93.49**
Average	*90.34*	*91.85*

Table A.4: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**90.53**	**93.49**
2	**89.94**	**91.72**
3	89.35	88.76
4	**93.49**	**94.08**
5	**86.39**	**88.76**
6	**88.76**	**88.76**
7	**87.57**	**88.76**
8	**91.72**	**97.04**
9	**90.53**	**91.12**
10	**92.90**	**95.27**
11	**92.31**	**91.12**
12	**88.76**	**91.72**
13	**92.90**	**94.67**
14	**84.62**	**86.98**
15	**88.17**	**89.35**
16	**91.72**	**93.49**
17	**92.31**	**94.67**
18	**91.72**	**92.90**
19	**92.90**	**95.27**
20	**92.90**	**95.27**
21	**91.12**	**91.12**
22	**88.17**	**91.12**
23	**90.53**	**92.31**

Table A.5: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**86.98**	**87.57**
25	**95.27**	**95.27**
26	**88.17**	**88.76**
27	**86.98**	**89.35**
28	92.90	92.31
29	**88.17**	**88.76**
30	**95.27**	**95.86**
31	**89.94**	**95.27**
32	**88.76**	**88.76**
33	**91.12**	**94.08**
34	**91.12**	**92.31**
35	**89.35**	**89.94**
36	**92.90**	**93.49**
37	**86.39**	**87.57**
38	**92.31**	**92.90**
39	**91.12**	**91.72**
40	**89.35**	**90.53**
41	**91.12**	**91.72**
42	**90.53**	**94.86**
43	**94.67**	**95.86**
44	**91.72**	**94.67**
45	**92.31**	**93.49**
Average	*90.57*	*92.06*

Table A.5: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	67.46	66.27
2	**78.70**	**82.84**
3	73.96	73.37
4	**77.51**	**77.51**
5	**72.48**	**79.29**
6	**69.23**	**78.11**
7	**76.92**	**77.51**
8	**77.51**	**78.70**
9	79.29	78.70
10	**77.51**	**78.11**
11	**72.78**	**75.74**
12	**73.96**	**78.11**
13	**68.64**	**79.29**
14	**78.11**	**82.25**
15	**78.70**	**79.88**
16	75.74	75.15
17	**66.86**	**76.92**
18	**79.88**	**82.25**
19	**76.33**	**79.29**
20	80.47	79.29
21	**78.70**	**79.88**
22	**68.05**	**76.92**
23	**69.23**	**71.60**

Table A.6: Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**68.64**	**76.33**
25	75.74	74.56
26	70.41	69.23
27	**73.96**	**76.33**
28	**70.41**	**77.51**
29	**65.09**	**73.37**
30	**78.11**	**82.84**
31	**85.21**	**86.39**
32	**71.01**	**72.78**
33	**72.78**	**79.88**
34	**72.78**	**73.37**
35	**74.56**	**78.70**
36	**82.25**	**82.84**
37	**76.92**	**78.70**
38	**69.23**	**72.78**
39	**78.70**	**78.70**
40	**66.86**	**72.19**
41	**71.60**	**71.60**
42	**78.11**	**78.11**
43	**74.56**	**77.51**
44	79.29	78.70
45	**74.56**	**77.51**
Average	*74.42*	*77.26*

Table A.6: (cont'd) Accuracy comparison of prediction ability

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**65.09**	**75.15**
2	**76.33**	**78.70**
3	**71.60**	**75.74**
4	**75.15**	**76.92**
5	**70.41**	**76.33**
6	**66.86**	**73.96**
7	**73.96**	**76.33**
8	**66.86**	**79.29**
9	**75.15**	**78.70**
10	**70.41**	**80.47**
11	**72.78**	**76.33**
12	**74.56**	**75.74**
13	**69.82**	**80.47**
14	**72.78**	**77.51**
15	**78.11**	**78.11**
16	**68.05**	**69.23**
17	**71.01**	**77.51**
18	**77.51**	**78.11**
19	**71.60**	**72.19**
20	**74.56**	**79.29**
21	**73.96**	**79.29**
22	**71.06**	**73.37**
23	**69.82**	**76.92**

Table A.7: Accuracy comparison of prediction ability with 10% additional noise

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**70.41**	**74.56**
25	**74.56**	**76.33**
26	**68.05**	**71.60**
27	**64.50**	**75.74**
28	**69.23**	**75.74**
29	**63.31**	**72.78**
30	**72.19**	**75.74**
31	**78.11**	**82.25**
32	**65.68**	**75.15**
33	**76.33**	**77.51**
34	**67.46**	**75.74**
35	**73.96**	**78.70**
36	**73.96**	**80.47**
37	**71.60**	**79.29**
38	**66.86**	**75.74**
39	**77.51**	**80.47**
40	**69.23**	**72.78**
41	**71.60**	**80.47**
42	**76.92**	**78.70**
43	**62.72**	**76.33**
44	77.51	76.33
45	**75.15**	**76.33**
Average	*71.65*	*76.76*

Table A.7: (cont'd) Accuracy comparison of prediction ability with 10% additional noise

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
1	**63.31**	**73.37**
2	**69.23**	**75.74**
3	**65.68**	**76.92**
4	**75.74**	**80.47**
5	**68.64**	**77.51**
6	**69.23**	**78.70**
7	**66.86**	**76.92**
8	**72.19**	**79.88**
9	**73.96**	**76.92**
10	**75.15**	**79.88**
11	**66.27**	**76.33**
12	**65.68**	**72.19**
13	**71.60**	**80.47**
14	**72.19**	**78.11**
15	**73.37**	**82.45**
16	**66.27**	**71.60**
17	**66.86**	**79.29**
18	**73.37**	**79.29**
19	**69.23**	**74.56**
20	**69.82**	**76.92**
21	**66.86**	**75.74**
22	**66.27**	**71.01**
23	**65.68**	**73.96**

Table A.8: Accuracy comparison of prediction ability with 20% additional noise

Sample	*BatchProcess*(%) (*Conventional*)	*HybridMethod*(%)
24	**69.82**	**73.96**
25	**70.41**	**76.33**
26	**62.72**	**71.01**
27	**66.27**	**72.78**
28	**70.41**	**75.15**
29	**59.76**	**65.68**
30	**71.60**	**79.88**
31	**75.15**	**81.07**
32	**71.60**	**78.11**
33	**68.05**	**76.92**
34	**65.68**	**75.74**
35	**67.46**	**78.11**
36	**73.37**	**80.47**
37	**69.23**	**79.29**
38	**62.72**	**77.51**
39	**68.05**	**74.56**
40	**67.46**	**74.56**
41	**72.19**	**79.29**
42	**68.64**	**78.11**
43	**59.76**	**72.19**
44	**72.19**	**79.29**
45	**69.23**	**76.92**
Average	*68.78*	*76.60*

Table A.8: (cont'd) Accuracy comparison of prediction ability with 20% additional noise

Appendix B

Experimental Results of Ch.4

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1	88.76	91.72	90.53	89.94	89.35	90.53	90.53	91.72	91.72
2	90.53	90.53	92.31	89.94	90.53	91.72	92.31	92.31	92.31
3	89.35	89.35	91.12	88.17	90.53	88.17	90.53	88.76	90.53
4	94.08	92.90	95.27	94.08	93.49	95.27	94.67	94.67	95.27
5	86.98	86.98	86.98	82.84	86.39	85.21	85.21	84.62	87.57
6	90.53	90.53	89.94	88.76	92.31	90.53	88.17	90.53	89.35
7	87.57	87.57	88.17	89.35	88.76	89.35	88.76	89.35	88.76
8	86.98	86.98	93.49	91.12	91.72	91.72	94.08	93.49	94.08
9	91.72	91.12	91.12	92.90	91.72	91.12	91.12	91.12	91.12
10	93.49	93.49	94.67	92.90	93.49	94.67	95.27	91.72	94.08
11	90.53	90.53	90.53	91.12	89.35	91.12	92.90	90.53	92.31
12	91.12	91.12	90.53	89.94	92.90	88.76	90.53	91.72	89.94
13	91.12	91.12	94.08	94.67	94.67	95.27	95.86	94.67	94.08
14	81.07	81.07	90.53	92.31	86.39	88.76	90.53	84.02	91.12
15	85.80	85.21	88.76	85.80	87.57	88.17	88.76	90.53	86.39
16	90.53	90.53	93.49	91.72	91.72	92.90	93.49	93.49	91.72
17	94.08	93.49	91.12	92.31	92.31	92.90	96.45	95.27	95.86
18	90.53	90.53	91.72	91.72	90.53	92.90	91.72	91.12	91.72
19	92.31	92.90	94.67	92.31	92.31	92.90	92.31	94.67	95.27
20	89.35	91.12	92.90	91.72	91.12	90.53	93.49	91.72	89.35
21	90.53	90.53	91.72	92.31	90.53	91.12	90.53	91.72	92.31
22	86.98	86.98	89.94	88.17	89.94	90.53	89.35	86.98	89.35
23	89.35	89.94	89.35	90.53	89.94	89.94	90.53	90.53	88.76

Table B.1: Accuracy of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
24	86.39	86.39	85.21	86.39	84.62	86.98	86.98	85.21	86.39
25	94.67	95.86	94.67	95.86	95.27	96.45	96.45	95.86	95.86
26	86.98	86.98	89.94	88.17	89.35	86.39	89.94	91.12	89.94
27	86.39	86.39	86.39	88.76	86.98	88.76	89.35	88.76	89.94
28	92.90	92.90	91.12	92.31	91.12	93.49	94.08	92.90	90.53
29	89.94	86.39	89.94	91.12	89.35	92.31	91.12	91.72	89.94
30	95.27	94.67	96.45	95.27	94.08	94.67	94.08	96.45	92.31
31	92.31	92.31	92.90	92.31	91.72	90.53	91.72	90.53	90.53
32	89.94	89.94	90.53	91.12	91.72	90.53	91.72	91.12	91.12
33	91.72	91.72	92.90	89.94	92.90	92.90	92.31	91.72	91.72
34	88.17	87.57	89.35	92.31	91.72	92.90	94.08	92.90	92.31
35	89.35	89.35	88.76	88.76	89.94	89.94	91.72	91.72	89.94
36	93.49	93.49	92.90	94.08	91.72	92.31	93.49	93.49	93.49
37	86.98	88.17	88.17	87.57	86.39	86.98	87.57	88.17	86.39
38	91.72	92.90	92.90	91.72	92.90	94.67	94.67	91.12	92.90
39	89.35	88.76	88.76	90.53	89.35	90.53	92.31	89.35	89.94
40	90.53	90.53	92.31	91.12	94.08	94.08	94.08	91.72	91.12
41	90.53	89.35	89.35	91.72	91.72	89.35	91.12	91.12	91.12
42	88.76	90.53	89.94	91.72	91.12	91.72	89.94	89.94	91.72
43	92.90	93.49	94.67	95.27	95.27	91.72	93.49	93.49	93.49
44	93.49	93.49	93.49	94.08	93.49	92.90	94.08	95.27	94.08
45	94.08	94.08	94.67	94.08	94.08	94.67	94.67	95.27	95.27

Table B.1: (cont'd) Accuracy of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	5.76	1873.00	510.46	5577.71	13.70	87.30	17.27
2	326.63	140.26	5904.12	58.35	29.35	243.06	4.66
3	32.36	357.65	4.20	6.75	2148.54	2.98	12.61
4	400.64	36.56	16.79	46.11	1284.00	3.43	90.57
5	5.36	2.59	102.47	63.18	170.17	4.90	4.39
6	11.40	80.58	7.27	104.59	217.84	135.37	74.02
7	36.37	49.34	47.18	428.30	652.53	18.54	2.43
8	38.95	10.03	22.85	759.70	822.80	35.73	2.03
9	3.46	25.38	429.27	5.88	1715.43	2.02	2.48
10	7.43	26.00	518.17	462.82	12.00	1.79	2.41
11	7.17	18.34	974.67	620.23	1999.01	13.15	37.53
12	44.03	403.21	47.19	315.09	2.85	23.77	14.38
13	3.19	87.98	6.75	7.56	116.28	1.93	2.75
14	297.48	0.27	63.62	1648.10	326.03	4.99	1.73
15	17.31	3.30	259.51	4.20	17.44	77.27	29.85
16	14.04	485.96	851.78	6904.50	19.99	10.73	20.63
17	17.18	3.75	50.80	849.19	8.19	3.13	3.46
18	500.47	287.00	996.98	18.16	25.37	1.81	11.00
19	27.33	17.09	3.49	93.63	3.60	4.88	4.39
20	2.64	32.15	1122.08	1.47	14.88	2.02	208.02
21	71.54	1804.45	41.68	2057.37	1084.94	2.40	48.82
22	7.58	15.85	2871.24	6.13	33.25	22.09	8.02
23	1070.80	451.31	2123.63	2851.02	474.6	2.87	3.83

Table B.2: Time Complexity Required of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)
24	174.81	2356.90	13.61	8018.11	185.27	2.04	3.13
25	160.43	4.80	517.68	8144.93	137.90	2.66	7.83
26	296.75	0.36	5.84	2099.04	714.96	6.07	2.00
27	172.50	23.17	22.64	7215.77	37.76	3.26	11.95
28	92.87	225.41	245.59	130.47	111.53	2.26	45.12
29	230.35	348.49	113.42	226.68	894.30	4.64	7.48
30	685.33	0.84	52.67	1202.60	2502.92	2.27	8.58
31	30.89	2.24	2.09	32.74	38.58	7.79	39.62
32	26.94	2.23	1.74	5.22	2.96	4.64	5.34
33	5.19	53.31	2.48	13.59	70.87	2.02	8.85
34	7.92	583.08	252.41	354.78	2207.10	2.70	2.12
35	1.52	7.70	73.50	318.53	81.46	0.60	9.93
36	129.76	17.02	378.96	71.49	20.87	8.01	0.60
37	101.53	0.58	314.65	153.05	2527.67	8.34	3.31
38	393.59	87.80	0.93	742.54	457.62	33.10	2.33
39	3.27	2.72	629.09	12.88	7.02	8.93	19.99
40	18.10	5.19	241.50	129.65	348.60	1.83	21.99
41	214.59	430.16	1.51	114.18	603.79	1.35	39.60
42	164.15	311.20	18.03	3.01	8.69	16.59	43.63
43	563.51	0.59	187.31	158.43	67.56	2.56	0.54
44	694.20	2.89	3.85	3036.37	21.28	25.98	6.38
45	27.12	1614.62	62.87	2069.17	5.08	9.30	2.23

Table B.2: (cont'd) Time Complexity Required of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	9	3	18	10	1	1	2	2
2	1	1	9	29	1	2	3	1
3	9	2	11	4	2	1	1	1
4	8	1	7	6	2	1	2	1
5	10	3	4	7	1	2	2	2
6	5	1	7	5	1	1	2	2
7	5	2	7	7	1	1	2	1
8	3	2	5	6	2	1	1	1
9	6	2	6	11	2	1	1	2
10	8	3	7	10	1	1	2	1
11	7	1	5	16	1	1	2	2
12	1	1	11	7	1	1	2	2
13	9	2	8	5	1	1	1	2
14	5	2	2	8	1	1	1	2
15	6	1	4	10	1	2	2	2
16	8	1	10	12	1	2	2	1
17	3	2	4	7	1	1	2	1
18	13	2	9	16	2	1	2	3
19	2	1	6	3	2	2	2	2
20	13	2	6	17	1	1	1	3
21	4	1	16	7	1	1	1	1
22	3	2	5	21	1	1	2	1
23	1	1	10	18	2	1	2	3

Table B.3: Network Size of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
24	1	1	19	6	1	1	2	2
25	4	1	4	12	1	2	2	2
26	1	1	2	5	1	1	1	2
27	1	1	6	6	1	1	1	1
28	1	1	9	10	1	1	2	1
29	4	1	10	8	1	1	2	1
30	13	3	3	7	1	1	1	3
31	14	1	3	4	2	1	1	2
32	4	1	4	3	1	2	2	1
33	2	1	7	3	1	1	2	1
34	8	1	13	10	1	1	1	2
35	7	1	5	8	2	2	1	2
36	1	1	6	10	2	2	1	2
37	5	2	2	10	1	1	2	1
38	1	1	8	2	1	1	1	2
39	9	2	4	12	1	1	1	1
40	13	2	4	9	1	1	2	1
41	1	1	10	3	1	1	1	1
42	2	2	9	6	1	2	1	1
43	1	1	2	8	1	1	2	1
44	1	1	4	4	1	1	2	1
45	12	2	17	7	1	1	1	1

Table B.3: (cont'd) Network Size of Different Algorithms : Breast-cancer database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1	87.50	89.00	89.00	89.00	88.50	94.00	87.00	88.50	90.50
2	86.50	86.50	86.50	91.50	93.50	94.50	95.50	90.00	88.00
3	87.50	93.50	92.50	82.00	86.00	87.00	93.50	94.50	86.50
4	83.00	82.50	85.00	88.00	90.00	95.50	91.00	86.00	91.00
5	88.50	88.00	92.50	94.50	91.00	93.50	90.50	96.50	96.50
6	92.50	93.50	92.50	89.50	86.50	94.00	89.50	92.50	97.50
7	96.00	91.00	96.00	92.50	89.00	93.00	96.50	91.00	92.50
8	89.00	87.00	89.00	86.50	97.00	95.00	88.50	88.00	88.00
9	84.00	84.00	84.50	89.50	84.50	93.50	81.50	91.00	92.00
10	87.00	82.50	90.00	93.50	91.50	93.50	98.00	95.50	91.00
11	83.50	85.50	86.50	85.00	96.00	91.50	84.00	92.00	84.50
12	92.00	80.00	92.00	87.00	87.50	88.50	88.00	91.00	92.00
13	85.00	79.50	91.00	86.50	85.50	89.50	89.00	88.00	83.00
14	78.50	79.00	82.00	89.00	89.50	91.50	91.50	89.50	87.50
15	84.00	81.50	92.50	88.00	92.00	94.00	98.00	93.00	87.50
16	93.00	83.50	93.00	91.00	88.00	94.50	94.50	97.00	92.00
17	91.00	85.50	91.00	93.50	99.50	95.50	98.50	95.00	97.50
18	96.00	96.00	96.00	85.50	93.00	88.00	98.00	93.00	92.00
19	87.00	85.00	89.00	86.00	93.00	87.00	87.00	91.00	86.50
20	95.00	95.00	95.00	92.50	89.50	92.00	96.00	92.50	88.50
21	88.50	88.50	88.50	84.00	86.50	96.00	83.50	91.50	83.50
22	87.00	87.00	87.50	92.50	97.00	91.00	93.50	97.00	93.50
23	83.50	85.00	85.00	93.00	86.50	86.50	84.00	87.50	90.00

Table B.4: Accuracy of Different Algorithms : Tic-Tac-Toe database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
24	88.50	87.00	90.00	87.00	88.00	89.00	87.00	88.50	88.00
25	80.50	63.50	82.50	86.50	86.50	85.50	90.00	92.00	89.50
26	88.00	85.00	89.00	88.00	91.00	88.00	89.00	88.00	91.00
27	93.50	81.50	93.50	89.50	95.00	86.00	89.50	84.00	96.50
28	80.50	80.00	86.50	91.00	89.00	90.50	93.50	89.00	88.00
29	89.50	88.50	85.50	89.00	94.00	91.00	90.50	94.50	93.50
30	79.00	77.50	82.00	85.50	90.50	95.00	91.00	89.50	92.50
31	81.50	81.50	84.00	93.00	90.50	97.00	90.50	88.50	93.50
32	89.00	89.00	95.50	93.50	89.00	92.00	78.00	87.00	90.50
33	94.00	71.00	94.00	84.50	90.50	93.50	94.00	92.50	92.50
34	93.00	58.50	93.00	94.50	93.00	93.00	95.00	92.50	90.50
35	88.00	75.50	92.50	89.00	93.00	95.00	93.50	94.00	91.50
36	95.00	50.00	95.00	81.50	91.50	93.50	89.50	94.50	96.00
37	92.00	93.50	92.00	88.50	87.00	92.00	95.00	92.00	92.50
38	90.50	79.50	91.50	91.50	92.50	92.50	81.00	80.50	96.00
39	84.50	79.00	84.50	82.50	81.00	89.50	86.00	84.00	84.50
40	92.50	83.50	92.50	92.50	92.00	93.50	94.50	91.00	89.50
41	89.50	81.50	89.50	95.50	91.50	90.50	91.00	91.50	89.00
42	85.50	86.50	87.50	94.00	96.00	93.00	94.00	94.00	87.00
43	87.50	91.00	92.00	91.00	93.50	94.00	91.50	94.00	93.50
44	90.50	91.00	95.00	88.50	95.50	90.00	98.50	92.50	98.00
45	92.50	82.00	93.00	85.50	91.50	94.50	84.00	91.00	91.50

Table B.4: (cont'd) Accuracy of Different Algorithms : Tic-Tac-Toe database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	114.61	227.76	19.62	958.60	51.00	53.67	29.31
2	81.61	54.03	16.91	176.86	66.16	21.30	19.86
3	202.10	41.55	50.45	23.95	620.89	135.63	22.25
4	72.56	14.13	28.49	15.32	47.05	49.21	29.45
5	156.63	27.74	72.53	76.13	45.33	57.11	43.38
6	143.29	373.14	40.48	5736.17	419.59	86.24	25.18
7	92.42	15.25	93.92	29.99	33.11	105.98	50.19
8	277.89	107.72	41.93	68.53	68.83	50.21	49.82
9	94.79	32.69	34.28	41.27	44.53	19.62	18.94
10	111.26	70.39	216.26	33.47	672.73	43.36	44.51
11	103.14	47.40	24.03	26.08	32.01	51.22	48.07
12	209.76	27.74	236.34	58.35	43.63	131.42	30.13
13	67.97	48.11	23.01	24.61	338.88	23.27	71.93
14	175.39	28.49	58.20	87.31	22.86	28.15	32.50
15	92.92	28.19	48.88	180.07	93.47	84.72	39.48
16	362.58	17.49	41.83	45.69	517.76	61.24	48.63
17	259.92	42.69	57.04	25.17	61.70	47.09	33.30
18	145.49	92.94	151.03	68.52	136.78	50.69	68.51
19	116.47	50.76	135.19	41.22	49.14	33.50	69.60
20	87.00	34.33	67.22	447.18	256.34	30.02	119.20
21	171.09	13.97	23.91	60.74	49.75	45.74	51.48
22	70.95	273.73	54.18	1405.30	355.82	87.65	26.26
23	56.31	107.75	21.24	100.81	42.75	53.05	33.72

Table B.5: Time Complexity Required of Different Algorithms : Tic-Tac-Toe database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)
24	86.71	24.74	58.82	59.14	41.36	38.93	51.31
25	337.10	44.36	693.93	32.91	20.10	18.62	53.10
26	73.25	36.40	38.04	37.76	31.72	73.06	24.83
27	124.03	32.80	51.03	155.27	23.25	28.34	66.00
28	102.02	48.47	837.00	369.24	27.59	35.46	77.78
29	98.12	142.06	80.12	76.54	25.07	28.68	72.84
30	71.81	46.73	15.84	46.06	49.17	45.40	69.29
31	124.85	106.02	43.29	38.42	73.77	46.62	54.01
32	92.42	33.00	43.56	58.68	94.82	40.60	19.08
33	791.02	27.15	61.21	81.36	48.26	15.72	84.27
34	235.15	84.13	116.57	28.34	144.52	41.97	77.58
35	132.06	18.32	87.62	92.77	34.97	35.41	36.16
36	124.30	229.22	23.08	208.13	48.35	39.45	26.29
37	380.35	29.62	29.76	81.59	366.53	52.99	39.84
38	121.68	63.88	44.68	69.74	38.38	37.17	68.77
39	131.92	31.18	14.88	35.00	19.89	26.07	17.14
40	186.49	90.44	20.63	52.51	96.94	24.12	34.48
41	332.76	63.15	2442.05	106.69	114.62	68.09	29.81
42	57.05	47.38	28.27	43.48	38.32	22.32	48.94
43	160.45	57.99	47.28	116.96	82.30	47.13	108.48
44	185.97	32.04	109.69	35.45	55.37	43.63	19.13
45	239.36	30.37	41.74	346.53	47.46	278.36	25.90

Table B.5: (cont'd) Time Complexity Required of Different Algorithms : Tic-Tac-Toe database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	4	4	3	2	2	2	2	1
2	5	5	2	2	1	2	1	1
3	1	1	2	2	2	2	1	2
4	5	5	1	2	1	2	4	1
5	3	3	2	5	2	2	1	1
6	1	1	6	2	1	4	1	1
7	2	2	1	5	2	1	3	2
8	2	2	3	2	2	3	2	1
9	3	3	2	2	1	3	1	1
10	3	3	3	3	2	2	1	2
11	4	4	2	2	2	2	2	3
12	2	3	2	5	2	3	2	2
13	3	3	2	2	1	2	2	3
14	5	5	1	4	3	1	2	1
15	3	2	2	2	2	2	3	2
16	2	2	1	3	2	1	1	2
17	2	2	3	3	1	1	1	1
18	2	2	4	3	4	2	1	2
19	4	4	2	4	2	3	1	3
20	3	3	1	4	1	1	1	1
21	4	4	1	1	1	2	2	2
22	4	4	7	2	2	1	1	1
23	4	4	2	1	3	2	1	2

Table B.6: Network Size of Different Algorithms : Tic-Tac-Toe database

Sample	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
24	4	4	1	3	3	2	3	2
25	5	5	3	8	1	1	1	2
26	4	4	2	2	2	2	3	1
27	2	2	2	3	2	1	2	2
28	4	5	2	8	3	1	2	3
29	3	4	5	5	3	2	2	2
30	5	4	3	1	1	2	1	2
31	5	5	4	2	1	3	3	2
32	2	1	1	2	2	3	2	1
33	3	3	2	2	1	2	1	2
34	1	1	2	3	1	1	2	2
35	1	1	1	2	1	1	2	2
36	1	1	2	2	1	2	2	2
37	1	1	2	3	1	1	2	1
38	5	5	2	4	2	3	3	1
39	4	4	2	1	1	1	1	1
40	3	3	2	2	1	1	2	2
41	1	1	2	11	1	2	2	2
42	5	5	3	1	2	2	1	2
43	2	2	2	3	2	2	1	2
44	2	2	2	3	2	1	2	1
45	1	1	2	2	1	3	1	1

Table B.6: (cont'd) Network Size of Different Algorithms : Tic-Tac-Toe database

Bibliography

[1] Tam, K. Y., and Kiang, M. Y., "Managerial applications of neural networks : the case of bank failure predictions," *Management Science*, vol. 38, pp. 926–947, July 1992.

[2] Mangasarian, O. L., and Wolberg, W. H., "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, pp. 1–18, Sept. 1990.

[3] Mangasarian, O. L., and Wolberg, W. H., "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the National Academy of Sciences, U.S.A.*, vol. 87, pp. 9193–9196, Dec. 1990.

[4] Mangasarian, O. L., Setiono, R., and Wolberg, W. H., "Pattern recognition via linear programming: Theory and application to medical diagnosis," *Large-scale numerical optimization*, pp. 22–30, 1990.

[5] Wilson, R. L., and Sharda, R., "Bankruptcy prediction using neural networks," *Decision Support Systems*, vol. 11, pp. 545–557, 1994.

[6] Michie, D., Spiegelhalter, D. J., and Taylor, C. C., eds., *Machine Learning, Neural and Statistical Classification.* Artificial intelligence (Englewood Cliffs, N.J.), Englewood Cliffs, N.J. : Prentice Hall, 1994.

[7] Denœux, T., "Pattern classification," in *Neural Network Applications*, pp. F1.2:1–F1.2:8, IOP Publishing Ltd and Oxford University Press, 1997.

[8] Han, I., Chandler, J. S., and Liang, T. P., "The impact of measurement scale and correlation structure on classification performance of inductive learning and statistical methods," *Expert Systems With Applications*, vol. 10, no. 2, pp. 209–221, 1996.

[9] Chung, F. L., and Lee, T., "A node pruning algorithm for backpropagation networks," *International Journal of Neural Systems*, vol. 3, no. 3, pp. 301–314, 1992.

[10] Ragsdale, C. T., and Stam, A., "Introduction discriminant analysis to business statistics curriculum," *Decision Science*, vol. 23, pp. 725–745, 1992.

[11] Yarnold, P. R., and Soltysik, R. C., "Refining two-group multivariable classification models using univariate optimal discriminant analysis," *Decision Science*, vol. 22, pp. 1159–1164, 1991.

[12] Winston, P. H., *Artificial intelligence.* Addison Wesley, 3rd ed., 1992.

[13] "An overview of neural computing technology," tech. rep., DTI NeuroComputing Web, URL:http://www.clients.globalweb.co.uk/nctt/guidelines, Mar. 1996.

[14] Archer, N. P., and Wang, S., "Application of the backpropagation neural network algorithm with monotonicity constraints for two-group classification problems," *Decision Sciences*, vol. 24, no. 1, pp. 60–75, 1993.

[15] Sarle, W. S., "Neural network faq, part 3 of 7 : Generalization," tech. rep., periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL:ftp://ftp.sas.com/pub/neural/FAQ3.html, 1997.

[16] Sietsma, J., and Dow, R. J. F., "Neural net pruning - why and how," *IEEE International Conference Neural Networks*, vol. 1, pp. 325–332, 1988.

[17] Kwok, T. Y., and Yeung, D. Y., "Constructive algorithms for structure learning in feedforward neural networks for regression problems(to be published)," *IEEE Transactions on Neural Networks*, 1997.

[18] Eberhart, R. C., and Dobbins, R. W., *Neural Network PC Tools : A Practical Guide.* Academic Press, 1990.

[19] Patterson, D. W., *Artificial Neural Networks : Theory and Applcations.* Prentice Hall, 1996.

[20] Narazaki, H., Member, IEEE, Watanabe, T., and Yamamoto, M., "Reorganizing knowledge in neural networks : An explanatory mechanism for neural networks in data classification problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 26, pp. 107–117, Feb. 1996.

[21] Sarle, W. S., "Neural network faq, part 1 of 7 : Introduction," tech. rep., periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL:ftp://ftp.sas.com/pub/neural/FAQ.html, 1997.

[22] Devroye, L., Györfi, L., and Lugosi, G., "A probabilistic theory of pattern recognition," in *NY:Springer*, 1996.

[23] Faragó, A., and Lugosi, G., "Strong universal consistency of neural network classifiers," *IEEE Transactions on Information Theory*, vol. 39, pp. 1146–1161, 1993.

[24] Lugosi, G., and Zeger, K., "Nonparametric estimation via empirical risk minimization," *IEEE Transactions on Information Theory*, vol. 41, pp. 677–678, 1995.

[25] Freeman, J. A., and Skapura, D. M., *Neural Networks : Algorithms, Applications, and Programming Techniques.* Addison Wesley, 1991.

[26] Gallant, S. I., *Neural Network learning and Expert Systems.* The MIT Press, Cambridge, London, England, 2nd ed., 1994.

[27] Hinton, G. E., "Connectionist learning procedures," *Aritifical Intelligence*, pp. 185–234, 1989.

[28] Tanaka, H., and Ishibuchi, H., "Possibilistic regression analysis based on linear programming," in *Fuzzy Regression Analysis* (Kacprzyk, J., and Fedrizzi, M., eds.), pp. 47–60, Physica-Verlag, Heidelberg, 1992.

[29] Sakawa, M., and Yano, H., "Fuzzy linear regression and its applications," in *Fuzzy Regression Analysis* (Kacprzyk, J., and Fedrizzi, M., eds.), pp. 61–80, Physica-Verlag, Heidelberg, 1992.

[30] Peters, G., "Fuzzy linear regression with fuzzy intervals," *Fuzzy Sets and Systems*, vol. 63, pp. 44–55, 1994.

[31] Savic, D. A., and Pedrycz, W., "Evaluation of fuzzy linear regression models," *Fuzzy Sets and Systems*, vol. 39, pp. 51–63, 1991.

[32] Mache, N., *Stuttgart Neural Network Simulator User Manual.* http://www-ra.informatik.uni-tuebingen.de: University of Stuttgart, version 4.1 ed.

[33] Hansen, L. K., and Pedersen, M., "Controlled growth of cascade correlation nets," *Proceedings of the International Conference on Artificial Neural Networks*, vol. 1, pp. 797–800, 1994.

[34] Thimm, G., "Evaluating pruning methods," *Proceedings of the International Symposium on Artificial Neural Networks*, Dec. 1995.

[35] Hassibi, B., and Stork, D. G., "Second order derivatives for network pruning: Optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 5, pp. 164–171, 1993.

[36] Goutte, C., "On the use of a pruning prior for neural networks," in *IEEE Workshop on Neural Networks for Signal Processing*, (http://www.ei.dtu.dk/staff/groutte/PUBLIS/nnsp96.html), 1996.

[37] Pedersen, M. W., Hansen, L. K., and Larsen, J., "Pruning with generalization based weight saliencies : γobd, γobs," in *Advances in Neural Information Processing Systems*, vol. 8, pp. 521–528, Cambride, Massachusetts, MIT Press, 1996.

[38] Mozer, M. C., and Smolensky, P., "Skeletonization : A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, vol. 1, pp. 107–115, 1989.

[39] Tanaka, H., Uejima, S., and Asai, K., "Linear regression analysis with fuzzy model," *IEEE Trans. Systems, Man and Cybernet*, vol. 12, pp. 903–907, 1982.

[40] Merz, C. J., and Murphy, P. M., "Uci repository of machine learning databases," tech. rep., Irvine, CA : University of California, Department of Information and Computer Science, http://www.ics.uci.edu/ mlearn/MLRepository.html, 1996.

[41] Schrage, L., *LINDO : An Optimization modeling system.* The Scientific Press, fourth ed., 1991.

[42] de la Maza, M., "Splitnet : Dynamically adjusting the number of hidden units in a neural network," *Proceedings of the International Conference on Artificial Neural Networks*, vol. 1, pp. 647–651, 1991.