REFERENCE ONLY

Multistage Neural Network Ensemble: Adaptive Combination of Ensemble Results

SHUANG YANG

LONDON [illegible]
LIBRARY SERVICES

A thesis submitted in partial fulfillment of the requirements of London Metropolitan University for the degree of Doctor of Philosophy

April 2003

London Metropolitan University

31 1134043 6

Abstract

In the past decade, more and more research has shown that ensembles of neural networks (sometimes referred to as committee machines or classifier ensembles) can be superior to single neural network models, in terms of the generalization performance they can achieve on the same datasets. Combining a set of neural network classifiers whose error distributions are diverse can lead to the generation of superior results than those achieved by any single classifier. Common combination strategies used to combine the results of individual ensemble members are simple averaging, weighted averaging, majorities voting and ranking. These are catalogued as static combination schemes, which require no prior training. One deficiency of such schemes is that weightings for the importance of the output of each ensemble member must be pre-chosen and then applied to produce the combination result. It appears attractive to make the combination process adaptive, so that no a-priori (and possibly incorrect) combination weightings need to be chosen. Therefore, a model is proposed where the procedure of combining ensemble classifiers is turned into the training of another neural network. In this thesis, we propose a novel trainable neural network ensemble combination schema: the multistage neural network ensemble (MNN). Two stages of neural network models are constructed. In the first stage, neural networks are used to generate the ensemble candidates. The second stage neural network model approximates a combination function based on the results generated from the ensemble members from the first stage. A sample of the data sets from the UCI Machine Learning Depository and human gene splice data sets were modeled using MNNs, and significant improvements were obtained by MNN in comparison with the performance of a majority voting scheme. The results suggest that the MNN approach can be used as an alternative ensemble combination method.

Statement of Objectives

The objectives of this research were to develop and explore a novel and alternative approach to combining the results of an ensemble of neural networks: the Multistage Neural Network Ensemble, and demonstrate its improved performance when compared with the most frequently used ensemble combination method: Majority Voting.

Acknowledgement

The author would like to express the deep gratitude to supervisors Dr. Antony Brown and Professor Philip Picton for their dutiful supervisions, invaluable guidance and their constructive suggestions in improving the presentation of this thesis.

Special thanks are due to Dr. B. Hudson for manipulating the splice data sets on the models developed in this study.

Sincere love goes to my husband, Enrong, my son Adam and my daughter Victoria for their support and patience.

Finally, this thesis is dedicated to my parents, who are always there to provide love and support. Without their encouragement, there would be no start of this study and the completion of this dissertation.

Table of Contents

Page

Abstract i

Statement of Objectives ii

Acknowledgement iii

Table of Contents iv

List of Tables vii

List of Figures x

Chapter One

Introduction and Neural Network Ensembles Literature Review

1.1 Introduction 1

1.2 Neural Network Ensembles Literature Review 4

1.3 The Motivation of Generating MNNs 23

1.4 Scope of Thesis 25

1.5 Conclusion 26

Chapter Two

Multistage Neural Network Ensemble

2.1 The Model of Multistage Neural Network Ensemble 28

2.2 Features of Multistage Neural Networks 32

2.3 Conclusions 33

Chapter Three

Methodology for Creating, Training and Testing MNNs

3.1 Introduction 34
3.2 Structures of MNNs 34
3.3 NN Algorithms and Programming Coding 40
3.4 Data 41
3.5 Experimental Procedures 45
3.6 Comparison 59
3.7 Conclusion 67

Chapter Four

Results and Comparisons

4.1 Introduction 68
4.2 Experimental Results from Combining Different Numbers of Ensemble Members 69
4.3 Results of the Other Experiments using MNNs 85
4.4 Comparison Results Regarding the Performance of MNNs 87
4.5 Conclusion 97

Chapter Five

Analysis and Discussions

5.1 Chapter Introduction 98
5.2 Analysis of MNNs Performance 99
5.3 Relationship between Performance of MNNs and its Ensemble Members' Diversity 112
5.4 Discussion of Other Experiments 123
5.5 Conclusions 124

Chapter Six

Conclusions and Future Work

6.1 Conclusions 125

6.2 Suggestions for Future Work 126

Bibliography 132

Appendix A: Algorithm 1 141

(Ensemblemembertrain.m)

Appendix B: Algorithm 2 148

(Ensembletrain3.m)

Appendix C: 155

Ensemble Members of Pima-Diabetes on Ten-Fold Cross-validation

Test Results

Appendix D: Publications (1) 157

Multistage Neural Network Ensembles

Appendix E: Publications (2) 165

Multistage Neural Networks: Adaptive Combination of Ensemble Results

LIST OF TABLES

		Page
Table 3.1	Summary of UCI machine learning depository data sets	43
Table 3.2	Summary of data partition of machine learning data sets	47
Table 3.3	Summary of data partition of splice data	47
Table 3.4	Settings of parameters in the first stage training	49
Table 3.5	Breast-cancer-w ensemble members performance	53
Table 3.6	The selection of ensemble members for breast-cancer-w	53
Table 3.7	T testing samples parameters from MNNs and Majority	63
Table 3.8	Summary of two-sample t test for comparing two populations means	63
Table 3.9	Parameters for variance testing between MNNs and Majority Voting	64
Table 3.10	Summary of two-sample test for comparing two population variances	65
Table 4.1	Ensemble candidates' performances of machine learning data sets	70
Table 4.2	Ensemble candidates' performances of splice data sets	71
Table 4.3	Ensemble Members' structures	72
Table 4.4	Performance of ensemble candidates on Breast-Cancer-W data set	73
Table 4.5	The selection of ensemble members	76
Table 4.6	Comparison of mean results between MNNs and majority voting on machine learning benchmarks, using ten-fold cross-validation (best results in **bold**)	77
Table 4.7	Comparison of MNNs and Voting Performance on Splice Data (best results in **bold**)	78
Table 4.8	Comparison between MNNs, single NN and C5 machine learning methods on splice junction data sets	79

Table 4.9	T test results showing acceptance or rejection of the null-hypothesis for different data sets	80
Table 4.10	Variances of Ensembles Results for benchmark machine learning data sets	83
Table 4.11	F test results on machine learning data sets	84
Table 4.12	Ensemble results by using neural networks with a single layer in combination	85
Table 4.13	Comparison between combining ensemble members' results Along with original source inputs and just combining ensemble members outputs	87
Table 4.14	Performance of each of the five ensemble members on the breast-cancer-w data set used in the diversity experiments	93
Table 4.15	Performance of each of the five ensemble members on the Pima-Diabetes data set used in the diversity experiments	94
Table 4.16	Ensemble members Correlation coefficients for breast-cancer-w data set	94
Table 4.17	Ensemble members Correlation coefficients for pima-diabetes data set	95
Table 4.18	The performances of different diverse groups of ensemble members for the breast-cancer-w data set	96
Table 4.19	The performances of different diverse groups of ensemble members for the pima-diabetes data set	96
Table 5.1	Ensemble inputs and combiner outputs for MNNs and Majority Voting	103
Table 5.2	Ensemble inputs and combiner outputs for majority voting	105
Table 5.3	Ensemble members outputs and the corresponding wrong predictions made by majority voting, where shaded examples are all three ensemble members made the wrong predictions and majority voting's results were wrong as well	106

Table 5.4 The parameters of the MLP combiner 108
Table 5.5 Ensemble outputs, hidden layer activation values and output activation for the four examples where MNN made the correct prediction whilst the majority voting combiner made the wrong decision 109
Table 5.6 10-fold cross validation results of ensemble members on Pima-Diabetes data set 119
Table 5.7 Diversity of pairs of ensemble members for Pima-Diabetes data set 120
Table 5.8 Diversity and accuracy of each ensemble group for Pima-Diabetes data set 121

LIST OF FIGURES

		Page
Figure 1.1	A typical two layers of NN model	1
Figure 1.2	Error contour plot	4
Figure 1.3	A typical NN ensemble using five ensemble candidates	6
Figure 1.4	AdaBoost with three weak learners	12
Figure 1.5	An illustration of stacking	21
Figure 2.1	The illustration of MNN	29
Figure 3.1	NN Model from the first stage of the MNN	35
Figure 3.2	The outline of second stage NN	37
Figure 3.3	Using a single layer NN to combine ensemble members results	38
Figure 3.4	Using original inputs and ensemble members results together as input of second stage NN	39
Figure 3.5	Donor and acceptor junctions	44
Figure 3.6	Data preparation procedure	46
Figure 3.7	Using 10-fold cross-validation in the second stage training (1) - (2)	56
Figure 3.8	Combination of ensemble members by using Majority Voting	61
Figure 3.9	Combining groups with different ensemble members	65
Figure 4.1	Performance of MNNs with increasing numbers of members (1) - (7)	88
Figure 5.1	The signal flow of a second staged MNN	100

Figure 5.2	Actual outputs of test data by three ensemble members in the first ten-fold cross validation experiments, together with their targets	101
Figure 5.3	The 3-D graph of three ensemble members outputs on the whole input data of Pima-Diabetes	102
Figure 5.4	Four cases of three ensemble members' outputs on which MNNs made right predictions while majority voting made wrong predictions	104
Figure 5.5	Ensemble members outputs with the corresponding target values	107
Figure 5.6	Decision boundary of the feature space dividing the four example inputs into two classes	110
Figure 5.7	The decision boundary of the feature space for MLP combiner	111
Figure 5.8	The ranges of three ensemble member's outputs, for data where the MNN combiner gave the correct outputs, whereas majority voting gave the incorrect outputs	113
Figure 5.9	The ranges of five ensemble member's outputs, for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output	114
Figure 5.10	The ranges of nine ensemble member's outputs, for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output	115
Figure 5.11	The ranges of eleven ensemble members' outputs for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output	116
Figure 5.12	The ranges of thirteen ensemble members' outputs for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output	117
Figure 5.13	Relationship between size of ensembles and average diversity of ensemble groups	121

Figure 5.14 Relationship between size of ensembles and average performance of each ensemble group 122

Figure 6.1 MNN model using different part of data for training 127

Chapter One

Introduction and Neural Network Ensembles Literature Review

1.1 Introduction

Neural Networks (NNs) are a multidisciplinary technology that is developed from the study of the human brain's operation, which covers many subjects: neuroscience, computer science, statistics, mathematics and even physics. The designs of NN models (Referred to as classifiers in this thesis) are inspired by imitating biological nervous systems. Thus NNs can simulate some of the functions of the human brain by using certain of its basic structures. The basic structure of a NN is illustrated in Figure 1.1.

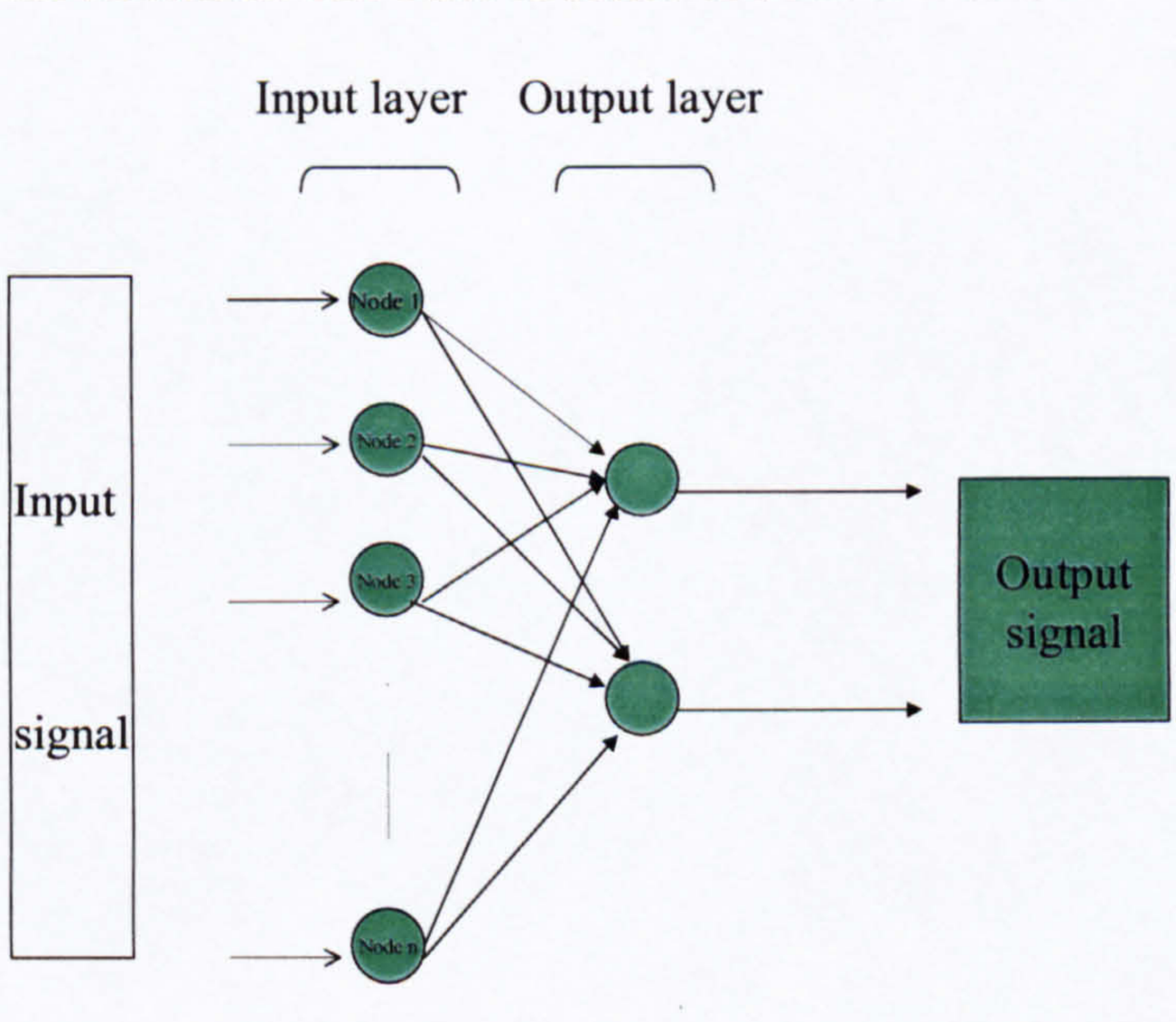

Figure 1.1 A typical two layered NN model

As shown in Figure 1.1, the above NN model has two layers. Each layer includes several neurons and these neurons are linked together via synaptic weights. A NN can be trained to perform a particular function by adjusting the weights between neurons, a procedure which is usually called the learning process. The significant feature of NNs is their ability to learn from input data and hence improve their performance.

In fact, over the past fifty years, significant improvements have been made since the pioneering work of McCulloch and Pitts (1943), both in topologies and applications. It is a rapidly developing field, ranging from the single perceptron to more advanced and complex multilayer perceptrons, using the back-propagation algorithm (Rumelhart, Hinton and Williams, 1986), radial basis function networks (Broomhead and Lowe, 1988), recurrent networks (Elman, 1990), support vector machines (Boser, Guyon and Vapnik, 1992) and many other NN algorithms. The applications of NNs have spread to the fields of pattern recognition, speech, identification, classification, vision and control systems.

In conclusion, NNs have grown to be an interdisciplinary subject in their theory, construction and application. However, NNs are far from being an optimal, firmly established subject. There are several aspects that need to be improved. For example:

- Even the most powerful NN models still can not cope well when they are dealing with complex data sets containing some random errors or insufficient training data, for instance, some complex pattern recognition problems (Ho, 2001). Therefore, the generalization performance of NNs on those data sets may not be as good as expected.

- The generalization of individual networks is not unique. In other words, the NN results are not stable. Different structures of NNs (such as different numbers of hidden layers, different numbers of hidden nodes and arbitrary initial conditions) all may result in different patterns of NN generalization. In particular, changes in the

training data set produce a significant change in the NN performance (Breiman, 1996a). Sharkey (1999) referred to this as: error diversity and neural network ensembles built up on the basis of NN randomness are expected to produce better performance than any single NN.

- The major research on NNs was focussed on how to create a best NN model or a classifier. However the definition of a best classifier is ambiguous. Is the NN model that can generate the most accurate result on the test data the best one we are pursuing? As a test data set only represents a small portion of the original data, for most of the cases, a NN having stable performance with a high prediction rate is expected. Thus once the accuracy and reliability are counted as the two fundamental factors in improving the performance of NNs, we need to find a way to produce more reliable generalization on top of high accuracy.

Generally, how to improve the performance of NNs is always the big issue in the research of this field. There are two ways in achieving this aim. The conventional methods are concentrated on developing more algorithms to optimize the current NN models. Alternatively, instead of just looking for the best classifier, looking for a combination of these best sets of classifiers has become an exciting topic from the beginning of 1990s. Since then the attempts at integrating a set of NN models to benefit from their diversity aroused more and more researchers' interests. These "networks of networks" are known as neural network ensembles.

This chapter will review the main findings in the field of neural network ensembles in the following area.

- The purpose and origin of neural network ensembles
- Methods of constructing effective ensembles

The final section of this chapter will address the research objectives of the study and describe the scope of the rest of thesis.

1.2 Neural Network Ensembles Literature Review

1.2.1 The Motivation for Neural Network Ensembles

Before the 1990s, most of the research in the NN area was focussed on the single NN models. However, the limitations on improving a single NN performance and randomness of a single NN result have hampered the development of better NNs. There are several questions raised that are worth investigating. For example, why does the same training data applied to different NNs or the same training data applied to the same NN with different initialisation result in different performance? What are the major factors affecting this difference?

From a mathematical viewpoint, the function of a single NN is to minimise the gap between the target value and the prediction value. This kind of gap can be expressed quantitatively in mathematics, which is the error. The less error made, the better the NN model is. As it is stated above, the errors made by a group of single NNs on any one application are not always the same. Instead, the error minimum varies. Through further analysis, some errors just settle into local minimum instead of global minimum, where the error should be zero as shown in Figure 1.2.

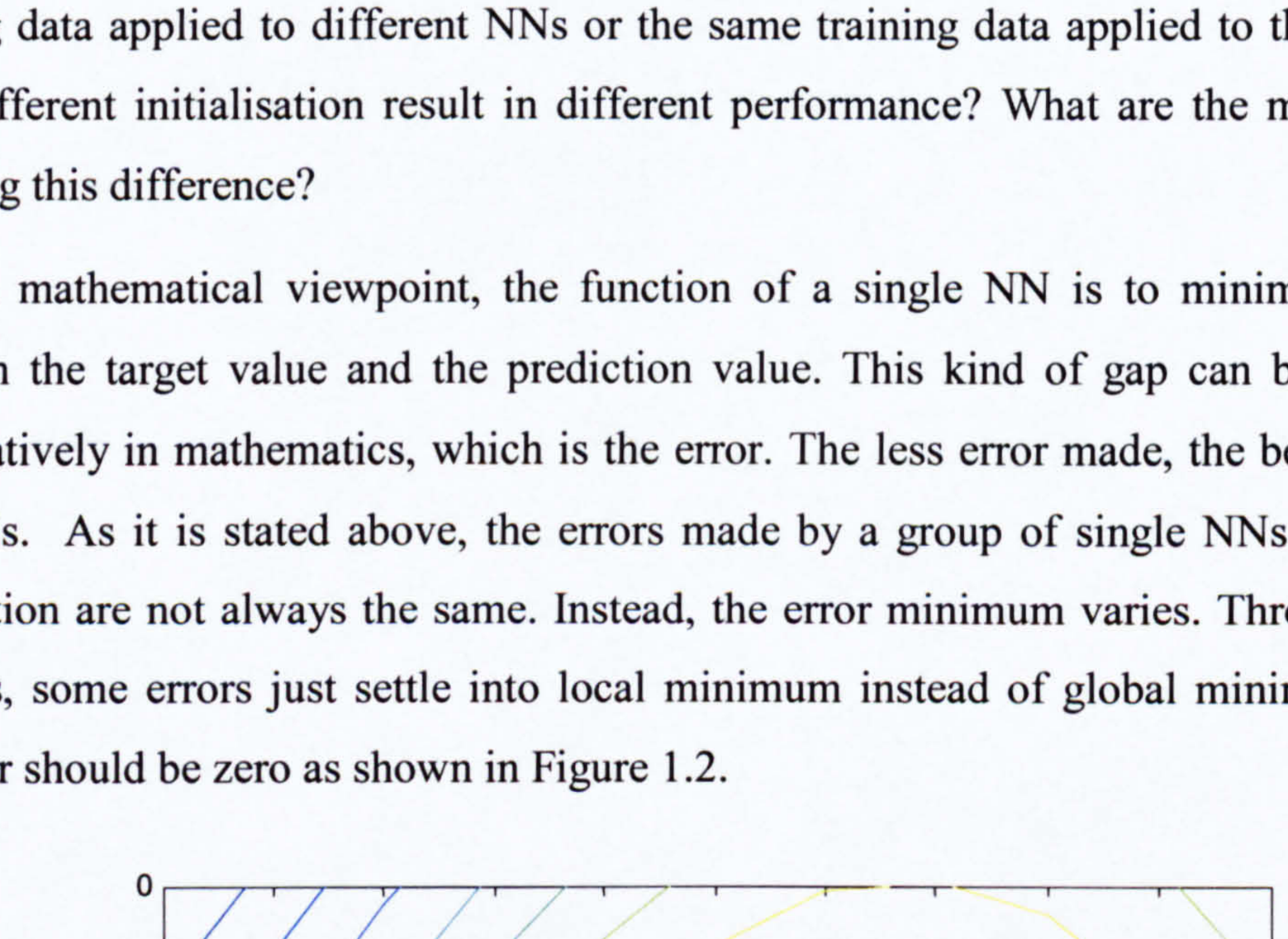

Figure 1.2 Error contour plot

As illustrated in figure 1.2, a single neuron's errors over a range of weight and bias values is plotted. The global error minimum is located in the centre of plot, and the stars represent three local error minimum respectively (Demuth and Beale, 1994). From the plot, it is clearly shown that a NN with different combination of weight and bias values can produce a different error.

Moreover, it is hard to justify which NN error reaches the global minimum while the error rate is not zero. Since the number of different NN models and their potential initialisation are unlimited, the possible number of generated results of any training data set applied to those models is theoretically infinite. Based on the above thinking, the best performance is typically only the best one selected from a limited number of NNs, i.e. a single model with the best generalization to a test set. One interesting point is that for a classification case, other less accurate classifiers may not misclassify the misclassified patterns generated by the most accurate classifier (Sharkey, 1999). As the NNs are a flexible algorithm, a NN model starting from different initialisation condition minimises an error function over the training data and will produce different approximation functions after searching along the error surface. So there exist some differences among the errors made by different NNs.

In short, it is obvious that just selecting the best classifiers according to their assessment performance is not always the optimal choice. More and more researchers realised that just selecting the best classifier will lead to losses of potentially valuable information contained by other less successful classifiers. The limitation of single NNs suggested a different approach to solving these problems by considering those discarded NN models whose performance are less accurate as potential candidates of new NN methods: Neural network ensembles. So the main motivation for developing neural network ensembles is to overcome the shortcomings of the single NN by combining different classifiers when a more reliable and accurate result can be produced.

1.2.2 The Origin of Neural Network Ensembles

There are several definitions of neural network ensembles. Sharkey (1999) stated: "The defining characteristic of an ensemble-based approach is that it involves combining a set

of nets, each of them accomplishes the same task." Dietterich (2000) stated: "An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples." Though the descriptions on neural network ensembles are different, the core meaning is the same. In general, combining several neural network models constitutes the term "neural network ensembles".

The concept of combining neural network models into ensembles can be traced back to the work of Nilsson (1965), however massive neural computing study started in the 1990s. The idea of ensembles of NNs originated from using all the valuable information hidden in classifiers, of which each classifier can contribute to the improvement of generalization. The general ensemble classifiers' structure can be viewed in figure 1.3.

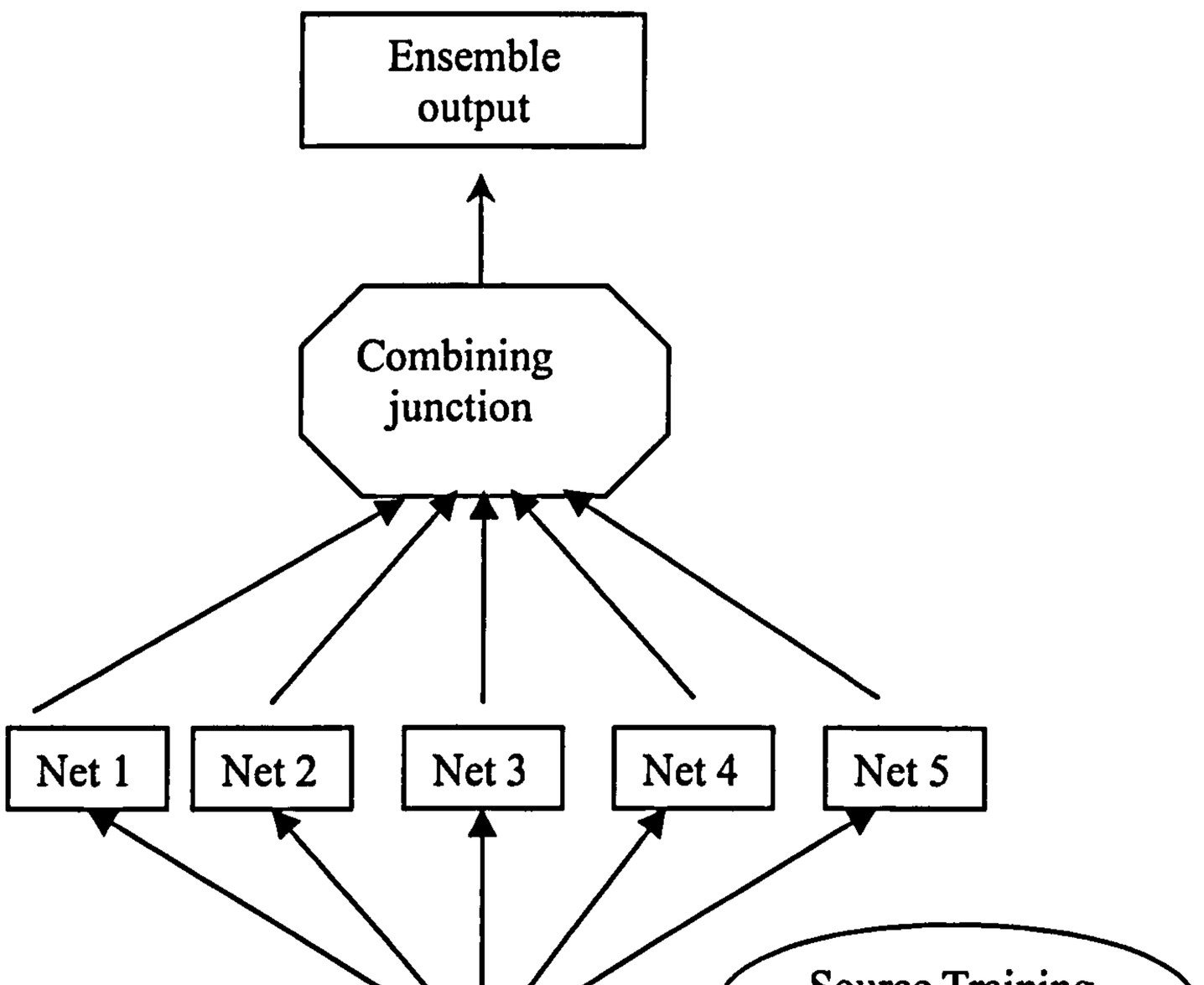

Figure 1.3 A typical NN ensemble using five ensemble candidates

As shown in figure 1.3, Ensemble NNs create several different NN models (called ensemble candidates or ensemble classifiers). None of these ensemble candidates are the same. They may be different in input training data, initialisation or structure. After training, these ensemble candidates will be selected by some criteria, and their generated results will be combined by some method. The final combined result is called the ensemble NN generalization or ensemble output.

However, it is well known that combining the same neural network models will produce no gain (Sharkey, 1999). The aim of neural network ensembles is to be able to produce better results than any single NN does on the same data set. So looking for those ensemble candidates that can generate better results after being combined was the first issue in this research area. Moreover, the concept of an effective ensemble classifier was introduced by Hansen and Salamon (1990), who stated: "A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse." An accurate classifier is one that is well trained and its performance is better than any randomly generated results on the input values. Two classifiers are diverse if they make different errors on the input values (Hansen and Salamon, 1990).

In conclusion, accuracy and diversity are noticed to be two major factors in the improvement of the performance of neural network ensembles in their early stage. With the development of neural network ensembles, these two issues became the key parts in pursuing higher performance. Since then, based on the above theories, a considerable number of ensemble NN models and experiments have been carried out and more deep insight into theoretical explorations gave further strong proof that ensemble methods are useful approaches in terms of improving the performance of NN models.

1.2.3 Methods of Constructing Effective Neural Network Ensembles

In order to find the factors that mainly decide the performance of an ensemble NN, we will look at the statistical analysis of a classifier error function in detail first. Bias-variance decomposition (Geman, Bienenstock and Doursat, 1992) of prediction error provides a clear picture of understanding prediction error. Let vector x denote the input training set and the corresponding target function is $f(x)$, the approximating function realised by the NN is $F(x,w)$, where w is an NN weight parameter matrix applied to the input. The model output is D. The training sample is denoted by $\Gamma = \{(x_i, d_i)\}_{i=1}^{N}$.

The deviation of the squared distance (the following formulation applies to regression problems) between $f(x)$ and $F(x,\Gamma)$ can be written as:

$$L_{av}(f(x), F(x,\Gamma)) = B^2(w) + V(w) \qquad (1.1)$$

where $B^2(w)$ (the *bias*) of the predictor refers to how the average of $F(X,\Gamma)$ differs from the average of $f(x)$, and $V(w)$ (the *variance*) measures the fluctuation of $F(X,\Gamma)$ around its average value $\bar{F}(x,\Gamma)$. The bias and variance are defined respectively by:

$$B(w) = E_{\Gamma}[F(x,\Gamma)] - E[D|X = x] \qquad (1.2)$$

$$V(w) = E_{\Gamma}[(F(x,\Gamma) - E_{\Gamma}[F(x,\Gamma)])^2] \qquad (1.3)$$

The above results are fundamentally important because they provide the mathematical basis for the trade-off between the bias and variance (Breiman, 1999) in order to minimise the error of a single classifier. Equation 1.1 makes it clear that we can either reduce the bias or variance to reduce the NN error. Unfortunately, it is found that for the individual NN concerned, the bias is reduced at the cost of a large variance. However, the variance can be reduced by an ensemble of NNs leaving the bias unchanged (Breiman, 1997). These theoretical findings indicated a bright future for neural network ensemble development, and pointed the way to error reduction for NN systems.

Hansen and Salamon (1990) addressed the idea that if the average error after training is less than 50% and the members of the ensemble are independent in the production of their errors, the ensemble can achieve a generalization error that goes towards zero as the number of networks combined goes to infinity. However, in practise, such an experiment can never be accomplished as the requirement of an infinite number of NN models in the experiments can never be achieved. As Equation 1.1 pointed out, the direction in theory is that ensembles of NNs can easily reduce the variance with less cost from the bias. Since the early 1990s, the research on ensemble classifiers has been very active and several methods have been employed for creating ensembles. Neural network ensembles can be classified by their different functional stages and therefore ensembles can be divided into three components as follows (Sharkey, 1999), which is the method adopted in this thesis:

- How to select diverse training data sets from the original source data.
- How to create different neural network ensemble candidates.
- How to combine the ensemble results.

1.2.2.1 Selecting Diverse Training Data Sets from the Original Source Data

Due to the limitation of the number of training data samples available in most applications, some approaches have been used for selecting samples by varying the data subsets selected or perturbing training sets. Cross validation (Krogh & Vedelsby, 1995), bagging (Breiman, 1996a), boosting (Schapire, 1990), noise injection (Raviv & Intrator, 1996) and input decimation (Tumer and Ghosh, 1996) are the most common techniques.

- *Cross validation* (Krogh & Vedelsby, 1995) is a tool borrowed from statistics. For example, the available data set is randomly partitioned into m disjoint subsets. By selecting one of these subsets as a test data set, the remainders are rejoined as its corresponding training data set. For this case m numbers of overlapping training sets and m independent test sets are obtained. As each training set is different somehow, the error they generated after training is expected to fall in different local error

minimum and therefore lead to different results. Classifier performance is measured on the corresponding test data set. This approach (ideally) requires that the number of partitions is the same as the number of data instances. Practically, 10-fold and 20-fold cross validations are adopted (where *m* equals 10 and 20 respectively).

The cross validation method offers an effective resample technique to obtain the different data sets by reducing data overlapping. In practice, when the source data set is small and different training sets are required, this approach is usually adopted, as is the case in our experiments.

- *Bagging* (Bootstrap Aggregating) was the first widely used method of selecting learning sets for ensemble classifiers. The idea of bagging came from attempts to reduce the error variance, and bagging is one of the ways to achieve this. In Breiman's paper (Breiman, 1999), he found that perturbing the learning set a little generates a different predictor. Suppose there are T sets of training data W, represented by $\{W_1, W_2, ...W_T\}$. Bootstrap approximation is used to create a new learning set of size N by taking N independent samples of instances with replacements from T. This method is especially useful when the training process faces data shortage. Though the generated training data may contain repetition or overlap, we can generate as many groups of data as we want. Suppose data set T_1, T_2, T_3, ... T_m each has size N of training data drawn from the original source data set is used as the illustration to demonstrate the bagging algorithm:

Given: size of the source data W is T, size of new training data is N, number of new training data is m;

1. Initialise $t = 1$, iterate while $t <= m$.
2. Initialise $i = 1$, iterate while $i <= N$.

   Let *Randrow* = *T*rand()*.

   If *Randrow* <= *T*

   let $T_t(i,:) = W(Randrow, :)$;

   Else back to step 2;

3. Back to step 1.

4. Output the final training data sets generated by bagging method: T_1, T_2, T_3, ... Tm.

The Bagging method is very efficient in constructing a reasonable size of training set while the original source data size is small due to the feature of its random sampling with replacement. We also used the Bagging method in our experiments when we encountered a small data set (see section 3.5.1). Therefore, Bagging is a useful data preparation method in neural network ensembles. But it must be noted that as the function of Bagging is supplying the training data to the ensemble classifiers, so the performance of Bagging is closely related to the choice of base classifiers.

- *Boosting* was originated by Schapire (1990), and works by using filters to modify the training data and force the learning algorithm to focus on the harder to learn parts of the data. Schapire (1990) proved that accuracy and diversity were not equal key factors that influence ensemble classifiers' performance. He found that diversity has the main role in constructing ensemble NNs. The paper pointed out that strong and weak NN models are equivalent in terms of generalization. Schapire (1990) proved that no matter whether the learner is strong or weak (generalization just above 0.5), the final result is surprisingly equivalent. So the training emphasis is put on those training data where performance is poor and retrain them to get better and better results. Typically, the final combined hypothesis is a weighted vote of the weak hypotheses. Based on this, Freund and Schapire (1995) developed another type of boosting algorithm, termed the adaptive boosting algorithm (AdaBoost). This algorithm can actively pick training data according their probability. Those data whose predicted value is close to the target value are adjusted to low probability of selection, and those data that are hard examples have a higher probability of selection assigned. This procedure will give the hard examples more chance to be retrained and thus the overall accuracy can be better than a single NN.

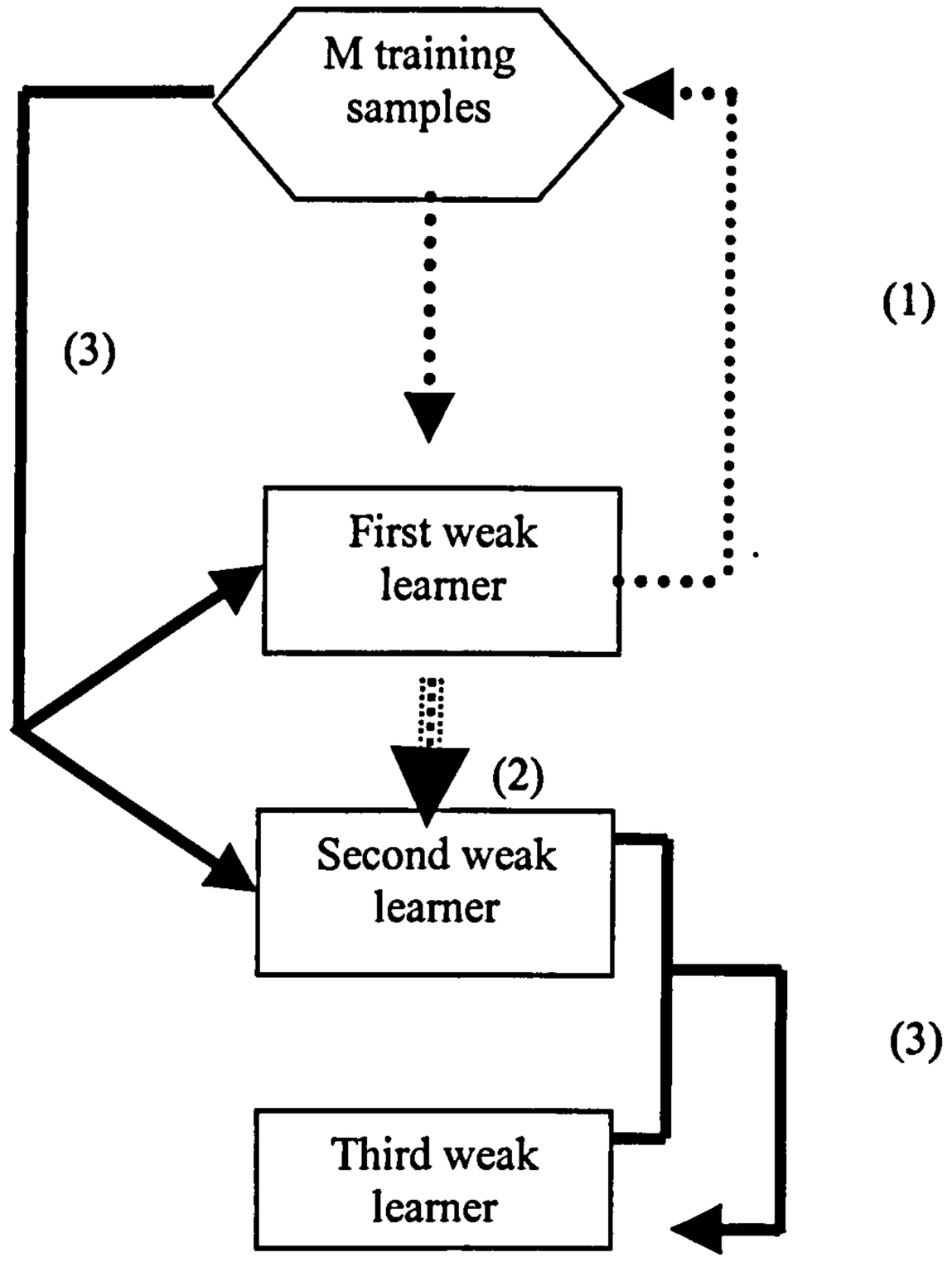

Figure 1.4 AdaBoost with three weak learners

Consider three weak learners, each with an error rate of less than 0.5. In Figure 1.4, dotted lines (1) represent the first step, where the first weak learner takes the data from the source training data repetitively and trains. The result is that it will pass the data (of which about half is classified correctly and about half is misclassified) to the second weak learner (2). Then, the second weak learner will be trained on these data items. After the second trainer's training, the data from the source will be passed to the first and second weak learners. If there is

disagreement between the first and the second classifiers on the same input, that set of input data will be passed to the third learner (3). The third learner will be trained on these groups of data which are difficult to train on. The final result will be dependent on the output of the three learners. For a classification problem, we can use majority voting and for a regression problem the result with lowest error rate is selected. Boosting algorithms have been shown to be very effective in many experiments (Drucker, Cortes, Jackel, LeCun and Vapnik, 1996; Jackson and Craven, 1996; Freund and Schapire, 1996; Breiman, 1996b).

The character of Boosting is that it requires a committee of weak classifiers to learn different features of the training data space and it forces the classifiers to learn on the very hard training data. So when the training data contains some noisy data, this method wastes too much attention on the data part which does not contribute much to the final results, therefore it is easy to understand why Boosting performs badly in domains containing noisy training data (Quinlan, 1996). In addition, when the ensemble classifiers are strong, Boosting fails to deliver high performance (Schapire, 1999; Wickramaratna, Holden and Buxton, 2001). So there is always an argument whether we want the ensembles be built on the weak classifiers instead of strong classifiers (Ho, 2001). The aim is to increase the performance beyond the high accuracy classifiers, but Boosting fails to do so in such circumstances.

- *Noise injection* was described by Raviv and Intrator (1996). Adding noise to the input during the training with ensemble averaging can effectively reduce the variance, since the injection of noise increases the independence among the different training sets derived from source data sets. The result of injecting noise is to push ensemble nets to different local error minimum so as to produce more diverse errors. However, caution must be taken when applying this method. Dietterich (2000) found that adding noise to the training data can cause over-fitting and some of the methods did not do as well as the experiments shown.

- *Input decimation* (Tumer and Ghosh, 1996) is an important pre-processing technique, based on the theory of lowering correlation among classifiers. It focuses on reducing the dimensionality of a training set in order to reduce the correlation between classifiers. An average error rate E_{add}^{ave} of combined classifiers can be given by:

$$E_{add}^{ave} = E_{add}(\frac{1+\delta(N-1)}{N}) \qquad (1.4)$$

Here, E_{add} is the sum of the error rates of all the classifiers and δ is the parameter that represents the correlation among the N classifiers. In Equation 1.4, we can see that if the errors are independent, then the part $\delta(N-1)$ is zero and the combined error is reduced by N. However, if each of the classifiers have high correlation (close to 1), then the average combined error is equal to the original errors and there is no gain in combining. Since the effect of each single input dimension on each output class of the classifier can be measured by their correlation, by pruning the input's dimensions that have the least effect on the output, one can train a classifier with partial inputs without compromising the overall performance. In experiments to avoid pruning too many inputs (which may lead to substantial reduction in discriminating power), we can use N classifiers (where N is the number of output classes). For each class, a subset of inputs with low correlation to that class can be removed. The resulting N classifiers will each have seen a slightly different feature set.

Though input decimation will cut off the internal link among input dimensions where each single dimension may look irrelevant to the output classes, attention must be paid on its application. Since some of the input dimensions may have no direct effect on any output classes individually, they may affect the general performance through the link with other input dimensions. If these input elements interact with some other input elements, they may play important roles in

generating the final results. Therefore caution must be used in applying the input decimation method to ensembles of NNs to avoid over pruning.

In summary, the above approaches can all make up the shortages of input signals for the purpose of generating diverse training data via different means. Cross-validation can generate the diverse training data by dividing the original source data into different subsets. Bagging is the most frequently used method in ensembles. One of the advantages of bagging is that each ensemble net can be trained independently (i.e. in parallel) compared with another popular approach (boosting). However, most experiments have shown that in the majority of cases boosting algorithms generated better results than bagging (Breiman, 1997; Drucker, Cortes, Jackel, LeCun and Vapnik, 1996). The application of adaboosting also showed that it only gives good performance in low-noise cases (Dietterich, 2000). Since it puts too much weight on the misclassified sample, of which most of them are, noise data in the high-noise cases leads to over-fitting. The noise injection approach introduces the injection of noise data into the source data, where it not only reduces the variance of error but also increase the size of training data. This could be counted as another benefit of the noise injection method. However, as we addressed already, the application of this approach did not always help on improving the ensemble performance, sometimes even degrading the generalization (Dietterich, 2000). Input decimation is useful on those data whose input dimension is large and where not all of them are relevant to the output classes. Despite successful empirical experiments on all above methods, further theoretical exploration and experiments are needed to prove their usage and compare their relative performance.

1.2.2.2 Creating Different Ensemble Members

a) How to create ensemble members

Krogh and Vedelsby (1995) found the relation amongst ensemble nets outlined in equation (1.8) indicates how to combine ensemble members and improve their performance. They used the term ambiguity (called diversity here) to measure the

disagreement amongst the networks. If $\hat{o}$ is the ensemble output, and $o(x)$ is output generated by each of the ensemble members, then the diversity term d_i of network i on input x is defined as:

$$d_i = [o_i(x) - \hat{o}(x)]^2 \qquad (1.5)$$

The error squared of network i and of the ensemble are:

$$\varepsilon_i(x) = [o_i(x) - f(x)]^2 \qquad (1.6)$$

$$e(x) = \hat{[}\hat{o}(x) - f(x)]^2 \qquad (1.7)$$

Where $f(x)$ is the target value of input x. The $\hat{E}$, E_i and D_i refer to the averages over the input distribution of $e(x), \varepsilon(x), d(x)$ respectively. Therefore the ensemble's error can be defined as:

$$\hat{E} = \overline{E} - \overline{D} \qquad (1.8)$$

where $\overline{E} = \sum_i w_i E_i, \overline{D} = \sum_i w_i D_i$, $\overline{E}$ is the weighted average of the individual network's output error and $\overline{D}$ is the weighted average of the diversity among members of the ensemble.

The above equations show that an ensemble consisting of more accurate nets with much disagreement will be more likely to have good performance. Thus, how to generate the diversity within an ensemble is the key path to the creation of an effective ensemble. There are several methods for generating diversity, including:

- Initialisation of different starting weights for each of the ensemble members.
- Varying the architecture of nets, as those with different number of hidden units or different number of hidden layers might generate different results.
- Using different training algorithms, such as the Back-propagation (Rumelhart, Hinton and Williams, 1986), Radial-Basis Function (Broomhead and Lowe, 1988), and Bayesian regression (MacKay ,1992) algorithms.

Though there are many ways of training them, the above approaches are not isolated and not necessarily independent. In fact most of the time they are employed together within one ensemble model. A common approach of creating ensemble NNs is to train a group of different single nets using the methods stated above and then use selection criteria (diversity and accuracy) to pick some of them to construct an ensemble model. Though there are many ways being offered to create the diverse NN models, the most significant approach among them is varying the initial conditions (Parmanto, Munro and Doyle, 1996b). Therefore, we adopted this method in our experiments in order to make the ensemble training simply in the mean time without affecting the ensemble performance.

b) The choices on number of ensemble members being created

After training, each member of the ensemble has generated its own result. If the individual members of ensemble nets are diverse, disagreeing results for the same inputs are expected. Before combining these ensemble candidates, strategies on deciding the number of ensemble members being created must be proposed. These strategies are generally classified in two types:

- Creating an exact number of ensemble members.
- Overproducing ensemble candidates and then selecting a subset of these.

For the first strategy, several ensemble approaches (Bagging, Boosting, etc.) can be employed to generate the right number of diverse ensemble members needed for combining directly. Therefore, no selection procedure will be used, and all the members will be used in combination. The aim of the second strategy is to create a large set of ensemble candidates and then choose those most diverse nets to combine. This criteria of selecting ensemble nets is described by Partridge and Yates (1996), where some error diversity measures used to choose diverse ensemble nets are introduced. Since the first type of ensemble creation is based on the idea of creating diverse nets at the early stage of design, it is better than the second method in the authors' view in situations where access to powerful computing resources is restricted. This is because the second approach can not avoid occupying much computing time and storage used while creating a large

number of ensemble candidates, some of which are to be later discarded. However, access to powerful/distributed computing resources can make this approach more attractive. In our experiments, both the above schemas are adopted, although there was a preference for the first approach. We did not want to deliberately select ensemble members, as more ensemble candidates were available in some circumstances due to the feature of our experimental design.

1.2.2.3 Combining Ensemble Results

Once the ensemble candidates are selected, the last step is to combine these ensemble classifiers' results. The combination method can be classified into two categories: static combination and adaptive combination strategies.

a) Static Combination

Static combination means that the combination stage does not involve the original input data and requires no prior training. The combination of ensembles members is based on their generated results. Common static strategies used to combine these single nets' results and then produce the final output are Simple Averaging (Tumer and Ghosh, 1995; Lincoln and Skrzypek, 1990), Weighted Averaging (Jacobs, 1995), Majority Voting (Hansen and Salamon, 1990) and Ranking (Ghoneim and Vijaya Kumar, 1995; Ho, Hull and Srihari, 1994).

- *Simple Averaging* is the one of the most frequently used combination methods. After training the members of the ensemble, the final output can be obtained by averaging each output of the ensemble members. Some experiments have shown that simple averaging is an effective approach (Breiman, 1996b; Hansen and Salamon, 1990). It is more useful when the variance of ensemble members is different, in other words, the local minimum of ensemble nets are different. Different local minimum means that ensemble candidates are diverse. Thus averaging can reduce the ensemble variance. However, this method treats each ensemble member equally, i.e. it doesn't

stress those ensemble members who can make more contribution to the output generalization. If the variances of ensemble nets are very different, we shall not expect a better result by averaging, as we know that combining the same classifiers will produce no gain.

- *Weighted Averaging* is where the final ensemble result is calculated based on individual ensemble member's performance and a weight attached to each individual member's output. For instance, if the gross weight is 1, each member of the ensemble is entitled to a portion of this gross weight according to their performance, or diversity. There are different methods of computing the classifier weights (Hashem and Schmeiser, 1993; Lincoln and Skrzypek, 1990; Merz and Pazzani, 1997). Although, weighted averaging provides more flexibility in terms of combination weights compared with simple averaging, the weights are actually assigned to a unit represented by a classifier, rather than a per data instance basis. In other word, the general performances of classifiers are the base on which weights are to be calculated. For example, a classifier that gains the largest portion of weights in a NN ensemble performs well only on the particular part of input data. The final ensemble result generated by the weighted averaging strategy can only show the effectiveness on that part of data and fails to reflect its advantage of the flexibility on the other part of the data. So it is clear that the flexibility offered by weighted averaging has its limitation.

- *Majority Voting* is the most popular combination method for classification problems because of its easy implementation. Members of an ensemble net vote to decide the value of each output dimension. The result for which over half of the ensemble members agree is to be accepted as the final output of the ensemble (regardless of the diversity and accuracy of each net's generalization). Majority voting ignores the fact that some networks that lie in the minority sometimes do produce the correct results. At this stage of combination, it ignores the existence of diversity that is the motivation for ensembles.

- *Ranking* is where the members of ensemble are called low level classifiers (LLC) and they produce not only a single result but a list of choices ranked according to their likelihood. Then the high level classifier (HLC) chooses from this set of classes using additional information that is not usually available to or well represented in a single LLC. The difficulty of applying this approach is it requires the classifiers produce not only the outputs but also the rank scores along with the outputs. Few classifiers are designed to implement this function and it may be this reason that less research has been explored in this subject.

There are several approaches that can be chosen while combining the ensemble members. However, there are no unique selection criteria on the usage of all the above combination methods. The choice mainly depends on the features of the particular application. For example, the nature of the application (classifier or regression), the size and quality of training data, the generated errors on the region of the input space, etc. Using the same combination method to apply to on the ensemble of a regression problem may generate good results. However, it may not work on a classification problem and vice versa. In addition, different classifiers will have an influence on the selection of which combination to use. Thus, empirical experiments so far can not find an optimal method for selection of the combination strategy. More theoretical development and experiments are needed to explore this field.

b) Adaptive Combination Method

One alternative to the static combination strategy, adaptive combination, means original input data are used again during the combination stage and a training procedure is introduced at the stage of combining. Stacking (Wolpert, 1992) is the earliest and typical approach in this type of combination.

- *Stacking*: The feature of stacked generalization (or stacking) proposed by Wolpert is that the information supplied to the original generalizers comes from multiple partitioning of the original learning set, all of which split up that learning set into two

subsets. Every generalizer is trained on one part of the partition and the rest of the parts are used to generate the outputs of the original generalizers (these outputs will be used as second space generalizers inputs). Then the second space generalizers are trained with those original generalizer's outputs and the second space generalizer's output is treated as the correct prediction. In fact, stacked generalization works by combined classifiers with weights according to individual performance, to find a best combination of generalizers. Generally stacking uses two ideas, preparing data and ensemble combination.

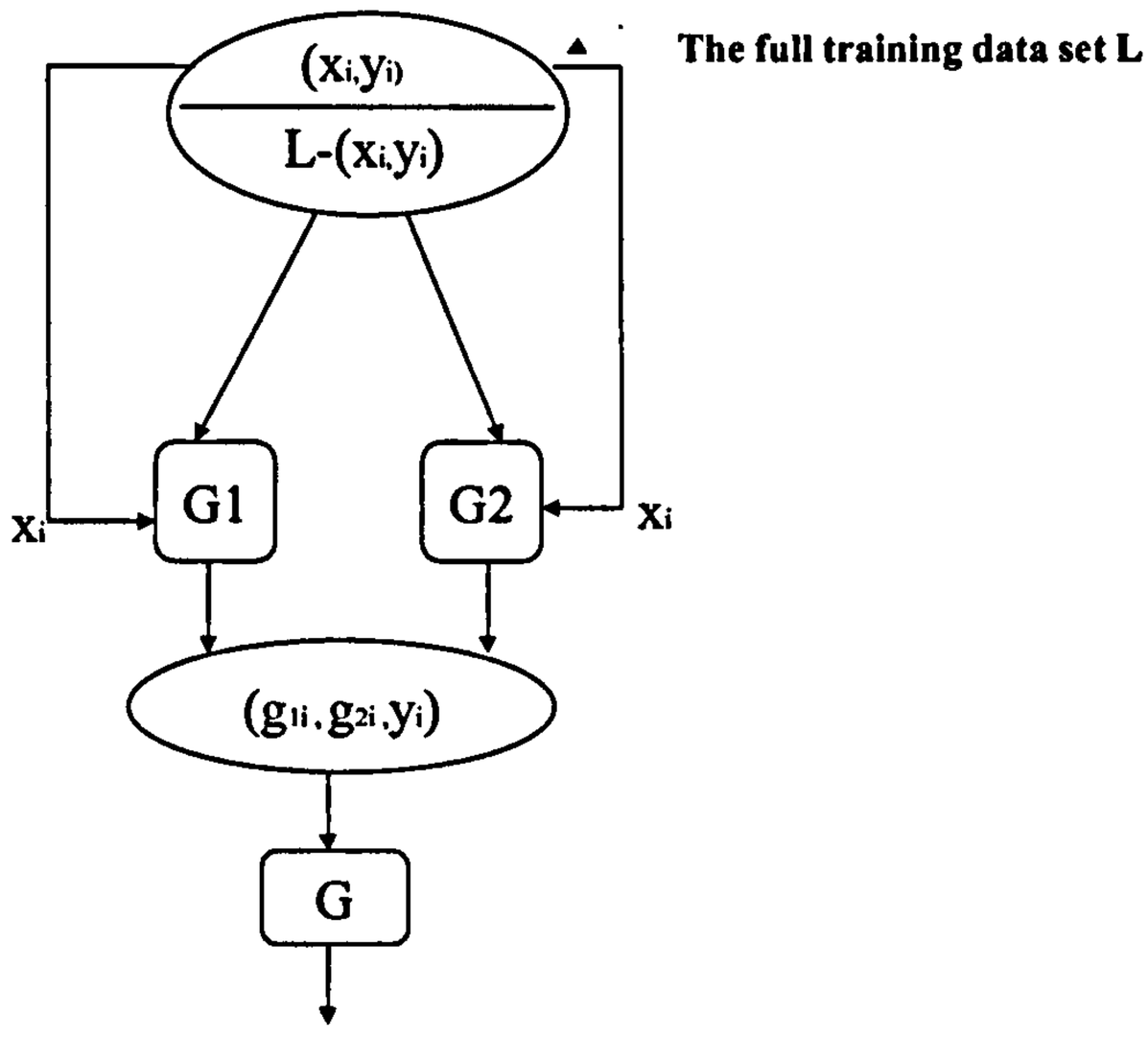

Figure 1.5 An illustration of stacking

Stacking is shown in Figure 1.5, where firstly this uses the idea of cross validation to get the training data sets for generalizers. Secondly, it explores the way of using the second generalizers to combine the first generalizers. In figure 1.5, the training data (*N* instances) is separated into a single data set (x_i, y_i) and L -(x_i, y_i). It uses *N*-fold cross-validation

(*N* partitions), and (x_t, y_t) just represents one of data partitions from *L*. The generalizers *G1* and *G2* are trained on the data: *L* -(x_t, y_t). After this cycle of training, x_t is put into *G1* and *G2*, and their output is the prediction of y_t: g_{1t} and g_{2t} respectively. The data set (g_{1t}, g_{2t}, y_t) will be used as one of the inputs of generalizer *G*. After all the partition has been trained in level 0, the generalizer *G* in level 1 starts training using the outputs of level 0. Any test data can be applied to *G* and level 1's generalizers outputs are treated as final results.

Stacking is a different type of combining approach compared with those static combination methods like simple averaging, ranking, majority weighting and so on. It introduced the general concept of using a dynamic model to perform the functionality of combination.

Inspired by stacking, there were some attempts using adaptive combination for some specific applications. As early as 1993, some experiments were done in digit recognition (Lee and Srihari, 1993) by using a single layer network to combine ensemble classifiers. Unfortunately, these experiments did not show any performance gain compared with other combination strategies. It was claimed the failure was due to the very high accuracy of all the individual classifiers being combined.

In 1995, Partridge and Griffith (1995) presented a selector-net approach. The selector-net was defined as a network, which used the outputs from a group of different trained nets as its input. The experiments based on this idea demonstrated that selector-net's performance was better than the populations of networks they were derived from. It clearly confirmed that this kind of ensemble method is better than individual NNs. But no further work has been done to compare the performance of this strategy with any other ensemble method.

In 1997, Wesolkowski and Hassanein (1997) used a NN model as a combiner on a digital recognition application based on the classifiers generated by using the boosting approach.

It was found that this combination method outperformed non-adaptive approaches like majority voting. However, no clear picture could be seen whether the gain was due to the usage of an adaptive combination approach or due to the weak performance of majority voting. As usual, the recommended combination schema applied on Boosting is the weighted averaging instead of majority voting (Schapire, 1999).

More recently, Kittler (1998) stated that: ``It is possible to train the output classifier separately using the outputs of the input classifiers as new features". However he did not implement this idea.

Very recently, Zeng and Martinez (2000) used a single NN as an approximator for voting classifiers. It was claimed that storage and computation could be saved, at the cost of a little less accuracy. However, here a NN was being used to approximate the behaviour of the ensemble, instead of using it as part of the ensemble components.

Though several of the previous experiments mentioned above have used the idea of stacking on some applications, they either used it as one of the tools to apply on one specific case (e.g., Lee and Srihari, 1993; Wesolkowski and Hassanein, 1997) or just use it to simulate a well-established ensemble model such as Zeng and Martinez (2000). The research so far is limited by the very narrow range of the applications. For example, the NNs performances as an ensemble combiner were only reported on a few examples of digital recognition data (Lee and Srihari 1993; Wesolkowski and Hassanein, 1997). Therefore, applications to a wide range of data that can show consistent performance need to be explored in this research area. Moreover, the questions can be asked: "Is there any standard rule to construct an optimal neural network combiner in order to apply universally?", and "How much potential extra overall generalization abilities can such kind of models demonstrate?" The lack of both theoretical analysis and extensive experimental support leaves this field with much space to explore.

1.3 The Motivation of Generating MNNs

It is really necessary to focus in more detail on neural network ensembles using the idea of stacked generalization. Therefore, we propose to systematically investigate a novel two layer ensemble model: Multistage Neural Network Ensembles (MNNs) (Yang, Browne and Picton, 2002; Yang, Browne, Picton, Hudson and Whitley, 2002). By doing this, this thesis extends the idea of stacking and investigates the use of a single neural network model to combine the ensemble member's results. Detailed investigations on the model itself are presented. A wide variety of data sets were selected for the experiments. Our hypothesis is that by using another second-stage NN to combine the results of ensembles as an adaptive combiner, superior generalization performance can be demonstrated compared to when the most frequently used non-adaptive ensemble combination technique (majority voting) is used. This implies that our null hypothesis is that there is no performance difference between these adaptive and non-adaptive ensemble combination techniques.

In summary, there are three motivations in generating a multistage ensemble approach. Firstly, the motivation to construct multistage ensembles stems from deficiencies inherent in current ensemble combination methods. With the exception of majority voting, other ensemble combination approaches all use weights assigning to the ensemble members while combining ensemble outputs. Thinking about the way that most ensembles use static combination, it leads us to think that using neural networks to generate the combining weights automatically may be better than assigning the weights manually. Secondly, once the outputs of neural network members are considered as the inputs of another neural network, it is very natural to employ neural network architectures to combine these. It appears attractive to make the combination process adaptive, so that no a-priori combination weightings need to be chosen. Thirdly, there has never been a thorough investigation done in exploring the creation of such an adaptive combination model, in monitoring the performance of such adaptive combination model, in comparing such model with other ensemble methods, and in analysing its performance into some depth. Therefore, a multistage ensemble model is proposed where the procedure of combining ensemble classifiers is turned into the training of another neural network model.

MNN models inherit some features of stacking. However there are some distinctions between two models. First, MNNs implement the idea of stacking and mainly emphasises on using NNs to construct two stages of ensemble models. Further, the way of manipulating data between two approaches is different. Stacking suggests using cross validation method to prepare the training data for the second stage. MNNs go even further and just uses the outputs of first staged training data as the inputs of second stage. This change means the training of two stages of ensembles can be separated to some extent. There is no need to take some special techniques such as the cross validation method to prepare the training data for the second staged training any more. Therefore the role of the combination function that the second staged MNNs plays emerges, and unlike stacking it is just a part of an ensemble model which has to be linked with the first stage training closely. Once the second stage MNNs is recognised as a combiner it makes MNNs more easily and widely applied and also leaves more flexibility for the first stage MNNs to select its own training style. So the MNNs method provides a convenient format to combine the ensembles. In addition, the author developed several models of MNNs. The comparisons among these models and also with other ensemble method, e.g. majority voting, were conducted. In particular, the combination functionality and the performance of this strategy were investigated in more depth in this thesis. The experiments showed that improved generalization could be gained by using MNNs. Besides the exploration on the structures of MNNs gave suggestions on how to construct such a simple and efficient model.

1.4 Scope of Thesis

A considerable amount of research effort has been spent to develop the model of MNNs and the details are reported in the rest of the thesis, which can be split into four parts.

a) Chapter two outlines the model of MNNs and introduces the idea on how to construct such a model.

b) The methodology of MNNs and our experimental details on how to apply MNNs to a wide variety data are presented in chapter three.
c) The experimental results of applying MNNs and some statistic comparison are demonstrated in chapter four.
d) The working features of MNNs are analysed in chapter five.
e) Finally, chapter six delivers the conclusions of the study and proposes the areas which need further research and investigation.

1.5 Conclusion

In this chapter, the history of neural network ensembles and its development has been introduced. The purpose of neural network ensembles is to improve the generalization ability and the reliability. This aim is achieved by various ensemble methods, through generating more diverse training data sets, more diverse net structures and more output error diversities. There are several approaches that have been used in neural network ensembles. These methods are classified into three groups by the functionality they perform. These three categories are:

- Generating diverse training data sets from the original source data, including Bagging, Boosting, Noise Injection, Cross Validation and Input Decimation.
- Creating different structures of neural network ensemble candidates, for example, initalising randomly, varying the inner structure of ensemble candidates and applying different neural network algorithms are the most common methods used.
- Combining the ensemble results. Two types of ensembles combination were introduced:
    - Static combination, including simple averaging, weighted averaging, majority weighting and ranking.
    - Adaptive combination, where stacking is the typical method of this type of combination schema.

Based on these techniques, some empirical experiments have been done and the results are encouraging. The experiments using ensemble methods mentioned in this chapter

demonstrated that these ensemble methods are effective ways compared with the conventional monolithic NNs. However, none of the above methods has been proved universal to all problems in practice, and exploration of new approaches is still very active. Among them, developing an adaptive combination strategy would appear to be a fruitful area for further research. The author has proposed a new dynamic ensemble combination approach called multistage NN. The structure of multistage ensembles and the implementation on improving the performance of this model are demonstrated in this thesis. The aim is to find a more general and effective way of combining neural network ensemble members. The experiments that were applied to a wide variety of data (including benchmark data from a machine learning data site and biology data from human gene bank) using multistage NNs imply that the model proposed can be used as a new combination method in neural network ensembles.

Chapter Two

Multistage Neural Network Ensemble

In chapter one a novel neural network ensemble was proposed: the Multistage Neural Network Ensemble (MNN). The model of the MNN is outlined and the features of this model are introduced in this chapter.

2.1 The Model of Multistage Neural Network Ensemble

Assume that we have a source data set $S\ \{s_1, s_2, ...s_n\}$ with its corresponding target data set $T\ \{t_1, t_2 ..t_n\}$. There are W number of NNs in the first stage represented as: $N_1, N_2 ... N_W$. The data for each member in the first stage are labelled as $Data_1$, $Data_2$, ...$Data_w$, which are usually partitioned into three parts: test data, training data and validation data after a data preparation procedure. The NN in the second stage is represented as N. If using matrix O represents the inputs for the second stage and matrix T represents the associated target outputs, matrix O and vector T can be expressed as the following:

$$O = \begin{bmatrix} o_{11} & o_{12} & \cdots & o_{1w} \\ o_{21} & o_{22} & \cdots & o_{2w} \\ \cdots & \cdots & \cdots & \cdots \\ o_{n1} & o_{n2} & \cdots & o_{nw} \end{bmatrix}, \qquad T = \begin{bmatrix} t_1 \\ t_2 \\ \cdots \\ t_n \end{bmatrix}$$

Where each column of matrix O stands for the vector which contains the output values of the corresponding NN in the first stage. The outline of a MNN model is illustrated as Figure 2.1.

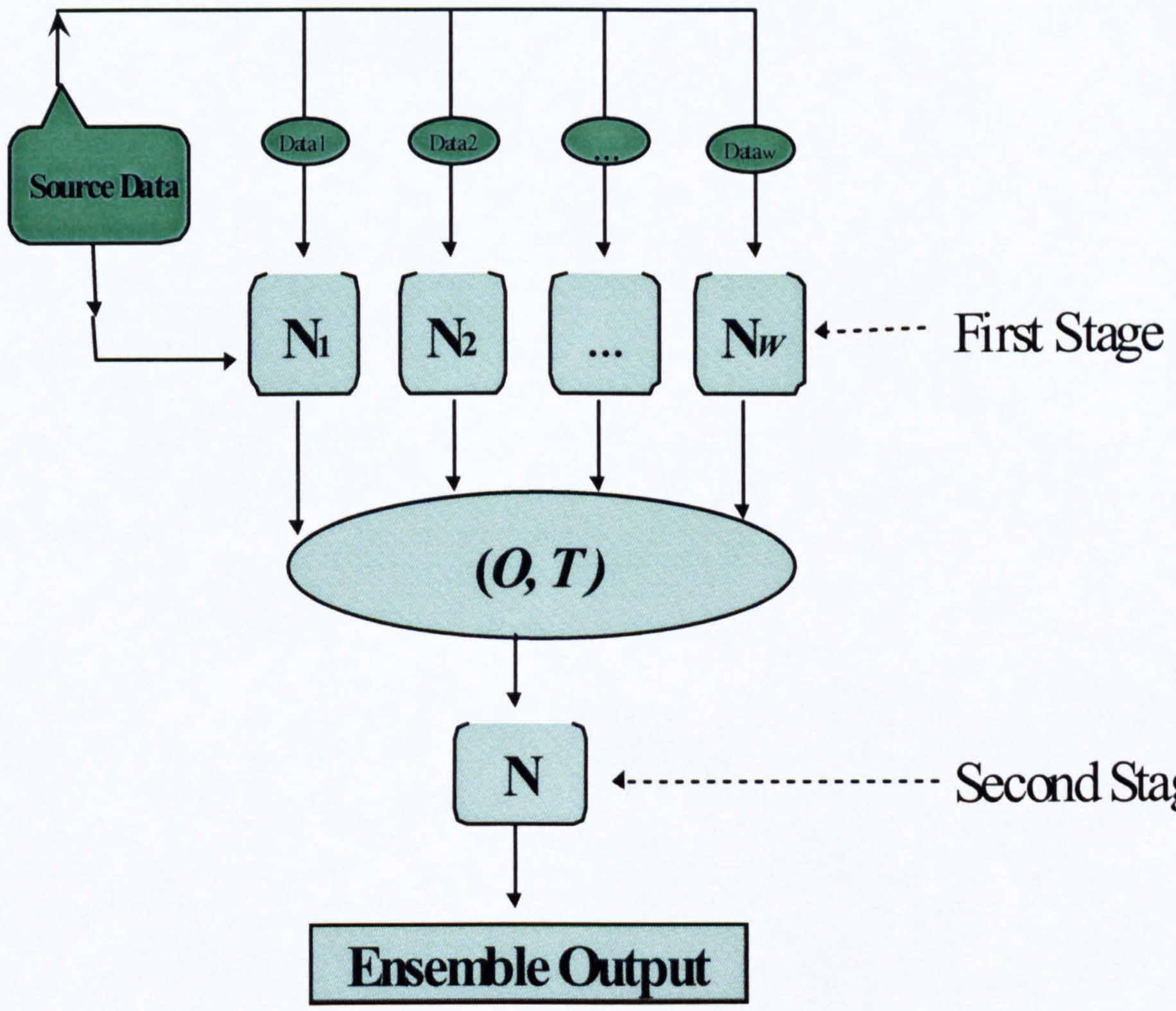

Figure 2.1 An illustration of MNN

As shown in figure 2.1, the structure of MNN generally includes three major components. The first part is the data, used for training, validation and testing. The second part is the set of ensemble candidate networks in the first stage and the third part is the NN model in the second stage, which serves as a combiner.

The data for each NN's training in the first stage are usually different due to the fact that the training data set, validation data set and test data set for each NN are randomly picked from the source data. The purpose of using a different training data set for each NN in the first stage is to keep the error diversity between ensemble members as much as possible. There is an important observation on ensembles that the variance of the prediction can be reduced while keeping the bias unchanged (Tumer and Ghosh, 1995, 1996). The ability of ensembles to achieve this goal relies on the error correlation among the ensemble candidates. Different classifiers provide different generalizations by making different decision boundaries (Ghosh and Tumer, 1994). Therefore, it is essential to make diverse ensemble candidates. Furthermore, earlier experiments (Parmanto, Munro and Doyle, 1996b) prove that training ensemble members with different training sets is more effective in reducing the error correlation than using the same training sets. This general rule is followed in designing MNN model. As shown in figure 2.1 that $Data_1$, $Data_2$, ... $Data_w$ is used to represent the different data inputs for each ensemble candidates in the first stage. Thus the input feature space that each ensemble member works in is different and each ensemble member is expected to simulate a different approximation function to some extent. Once the first stage training is completed, the training data (again randomly picked from the source data) are injected into the first stage models and the subsequent generated outputs along with the corresponding target values are used as the inputs of the second staged NN. After being trained by the outputs of first stage ensemble members along with the associated target values, the outputs of second stage neural network model are taken as the final ensemble results.

As noticed, no any particular ensemble techniques in preparing data for training in both stages of MNN is employed comparing with stacking approach. The normal procedures of preparing data in ensembles were followed in our experiments (details addressed in

chapter three). There are two purposes for doing so. Firstly, to simplify the training procedures in the first stage. For instance, as in the application of a cross validation method, the times of training for each model in the first level of stacking is equal to the size of training data . In contrast, only one time training is enough for each NNs in the first stage of MNN. Secondly, the second stage training does not rely closely on the first staged outputs. Therefore for each stage they can have more choices on selecting ensemble techniques. For example, the training of first stage ensemble candidates can be more flexible. We don't necessarily use one-fold cross validation in order to prepare the inputs for the second stage training. Furthermore, we can select whatever ensemble approach to use as long as it can be beneficial to create the diverse ensemble members.

As the name of MNN implies, there can be several stages of NNs in this model. From figure 2.1, we can see that only two stages NNs are employed. Compared to other no prior training needed combination methods, like simple averaging, majority voting, we have the reason to believe that two stages of NNs, which can generate more precise weightings after training are powerful enough to accomplish the combination function. More over, to avoid complexity and much time consuming, just two stages of MNN were constructed in this thesis. The first stage of MNN includes several neural network models. The number of neural network models in the first stage can be varied from two to a large number. Only one NN is used in the second stage as adding one more NN in this stage means we need another stage to combine them and more stages means more training needed. The second staged NN can be clearly seen as a combiner to combine the first stage outputs.

Due to the stochastic searching features of NNs, several techniques were used to create diverse ensemble candidates during the first stage training, such as random initial weightings, different subsamples of source input used as training data, randomly disturbing the sequential order of training data, and different structures of NNs. The performance of ensemble candidates trained under such a design can differ in a large amount (Ho, 2002), the aim of creating diversity among ensemble candidates can be achieved.

2.2 Features of Multistage Neural Networks

MNN actually use the same type of models to apply to the data set twice. It is not the simple sum, but has a two-stage application. In the first stage, the models try to simulate the functionality of source data set. There are several NN models in this stage. In the second stage, the model tries to simulate the combination function based on the results generated by the first stage models. Only one NN model is used at the current stage. The features of using MNN as one of the ensemble techniques are:

- For each ensemble candidates, the normal data preparation methods can be applied to them and ensemble candidates can be trained separately by using various neural network models and algorithms. The generalizations of these first stage NNs are expected to be as diverse as possible in order to get the best ensemble result. Because of this, the choices for the first stage NN training are very flexible and allow a wide range of network architectures and training algorithms.

- The second stage of a single NN is used as an ensembles' combiner to apply different weights to those first stage ensemble members in order to approximate a best combination function. Due to the power of NNs, which adjust connection weights towards minimizing errors, there is a strong reason to believe that it can more accurately adjust the weightings to be assigned to ensemble members than manual methods.

- MNN can be widely used, both for classification and regression problem, unlike some other ensemble combination techniques, such as majority voting which can only be applied to classification problems.

2.3 Conclusions

The author propose a standard ensemble model by using multistage neural networks in this thesis, where the adaptive properties of a second layer network are used to combine the outputs of the individual ensemble members. The model of MNN is illustrated and its structure is introduced. The design of such a model is based on the theory of diversity that exists among the ensemble classifiers and therefore several techniques that were used generating different ensemble classifiers of MNNs in the experiments were introduced. Moreover, the features of MNN models, which make it distinguished from other ensemble combination methods are discussed.

There is an expectation that it can offer enhanced performance over a simple voting based combination method. The expected improvements on the generalization of ensembles rely on the neural network's ability to adaptively assign weights to those ensemble members automatically.

Chapter Three

Methodology for Creating, Training and Testing MNNs

3.1 Introduction

In order to prove the effectiveness of MNNs, a wide variety of data on different MNN models was applied. Firstly, the structures of MNN models used in these experiments are described in section 3.2. Secondly, the NN algorithm and program coding used in my experiments are addressed in section 3.3 and experimental data applied to MNN models are introduced in section 3.4. Next, the experimental details based on the above models are presented in section 3.5. Finally the comparison procedures used when comparing the multistage combination strategy to the majority voting strategy are explained in section 3.6.

3.2 Structures of MNNs

As mentioned in chapter two, two stages of NN models are used in MNNs. The function of each NN model in the first stage is to approximate the functionality of the original source data. The purpose of adopting a second stage model is to combine the outputs of first stage models. In the experiments, there were several types of NN architectures, designed and implemented to explore the ability of the multistage neural network combination schema. The NN models were presented in the form of stages. The NN structures used in the first stage training are described in the first part. Next, the second part emphasizes the organization of the second stage NN. Last, some trials employing other structures of MNNs are introduced in part three. As within this part only the structures of MNNs were discussed, correspondingly the experimental details using these kinds of models are reported in section 3.5.

I. NN Models in the First Stage

Most classification cases in the real world are non-linear separable problems. Therefore NN models with one hidden layer, which are capable of approximating any continuous functional mapping (Bishop, 1995), were used in my experiments. In the first stage of MNNs, the standard Multi Layer Perceptrons (MLPs) with one hidden layer using the Back-Propagation algorithm (Rumelhart, Hinton and Williams, 1986) were employed. Justification for this is presented in section 3.3. As illustrated in Figure 3.1, the source nodes just receive information from the input signal; therefore, no computation is performed in that stage. The number of neurons in the hidden layer for each ensemble member was varied with the purpose of creating diversity among the ensemble members. The number of output neurons was set to one.

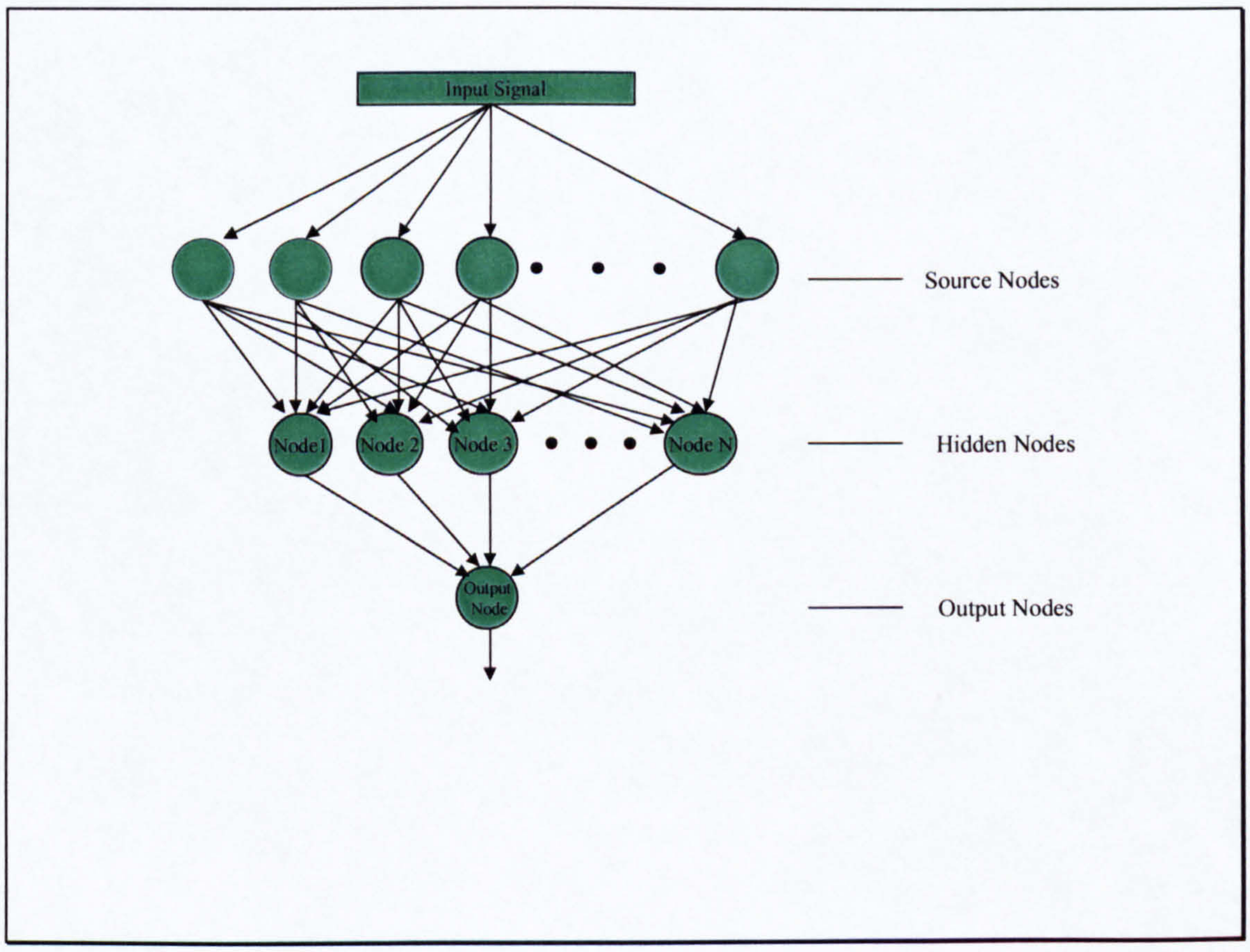

Figure 3.1 NN Model from the first stage of the MNNs

In my experiments, nineteen ensemble candidates using the above NN models combined with the back-propagation algorithm and pre-processing data technique (section 3.5.1) were implemented in creating ensemble members. The selection of ensemble numbers was decided by the results of MNNs. The initial combination number started from three and was gradually increased to nineteen. The training details using the above basic structure of MLPs are addressed later in Section 3.5.

II. Structure of the Second Stage MNNs

The combinations of different ensemble members have been investigated by using the multistage neural network combination approach in my experiments. The structure of MNNs is a top-down style, thus the combination stage was after the completion of first stage training. As mentioned in part I, nineteen ensemble candidates were created in the first stage and therefore the maximum of ensemble members that could be combined in the second stage were nineteen. However in order to assess the performance of MNNs by combining different numbers of ensemble members, the experiments in the combination stage were done step by step. The numbers of combined ensemble members were all odd numbers in order that the performances generated by MNNs could be compared to majority voting, which prefers the odd number of ensemble members to avoid the half-wrong and half-right occasion. These numbers are three, five, seven, nine, eleven, thirteen, fifteen, seventeen and nineteen. The number of ensemble members was gradually increased starting from three and ended up at number nineteen where the improvement of MNNs performance on most of data sets could not be made by just increasing the number of ensemble members.

A fully connected two layered backpropagation NN was used to combine the outputs of ensemble members as illustrated in Figure 3.2. There were two considerations while selecting the number of hidden neurons. First, the aim was to construct the second stage of MNNs as simply as possible. As the application of MNNs has to be on the basis of extra training for the second stage, so it is necessary to minimize such training cost. In addition, during the experiments, there was a found that the second stage NN was less

sensitive. Several combinations of learning parameter and number of hidden neurons could all reach to the same generalization. Based on above thoughts, two hidden neurons were adopted, after initial experimental testing showed that two hidden neurons were powerful enough to perform the combination function based on the data sets used in my experiments. (Further detail is presented in section 3.5). The results generated by the output net are counted as the final ensemble results.

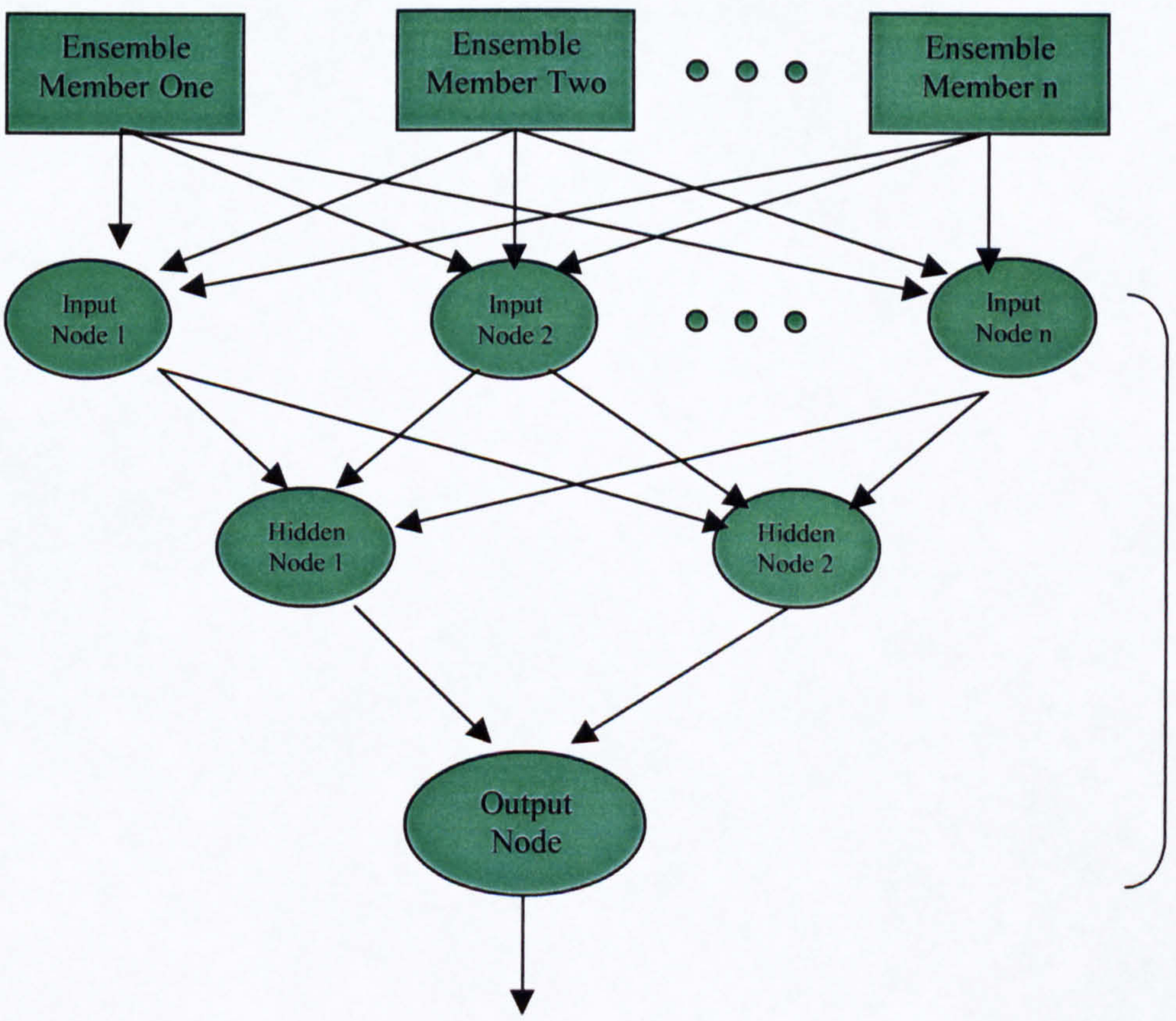

Figure 3.2 The outline of second stage NN

III. Other experiments using different MNN models

Apart from the NN models introduced in part I and part II, some other structures of MNNs were tried, which failed to deliver better results. However, the experimental details related to these trials were still presented and such failure may help us to construct the optimal MNN model in future work. The structures of the NN models used in these MNNs experiments are shown in Figure 3.3 and Figure 3.4.

a) Combining Ensemble Results Using a One Layer NN

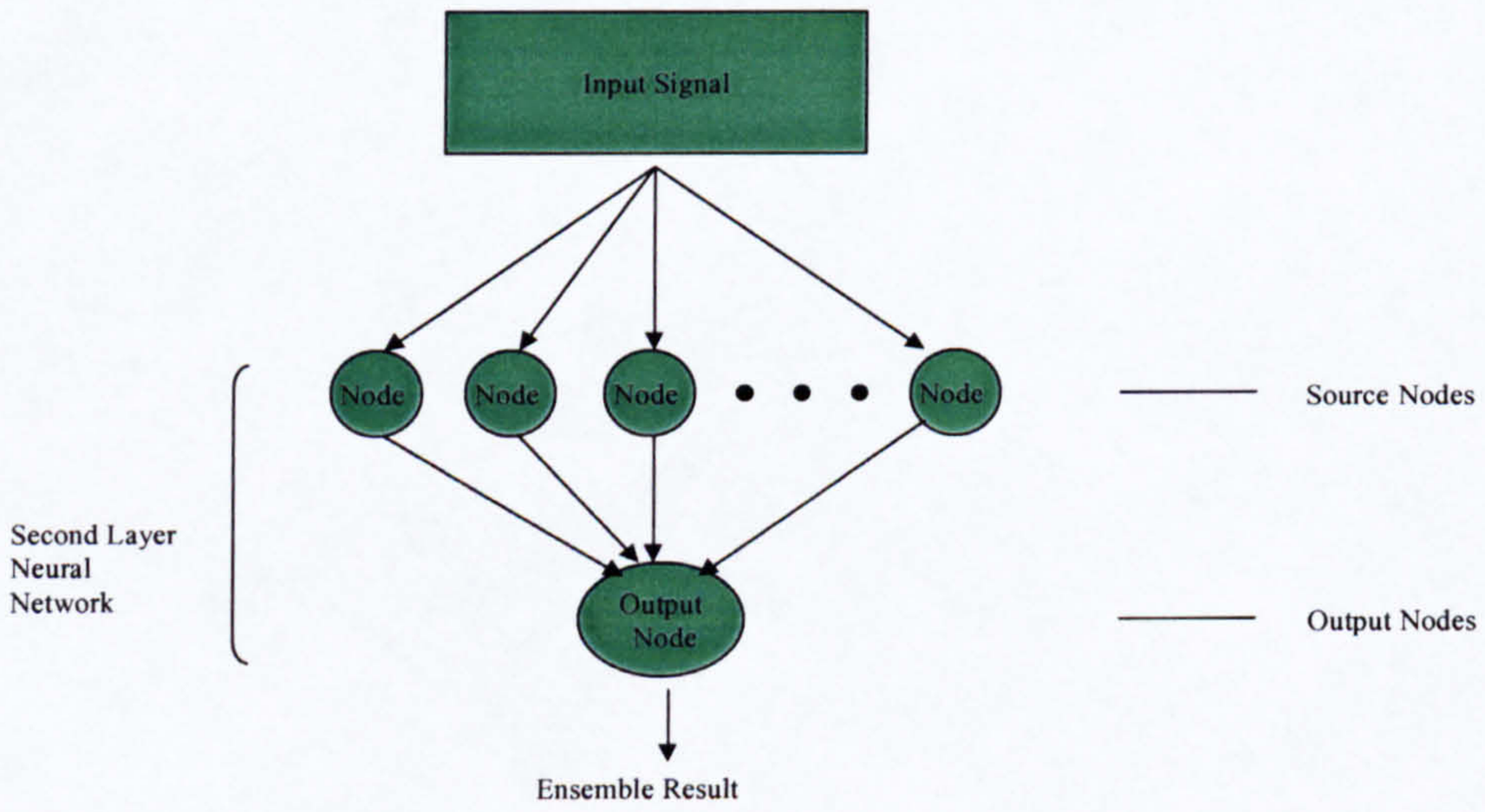

Figure 3.3 Using a single layer NN to combine ensemble members results

As shown in Fig. 3.3, a single layer NN is applied as a second layer combiner. The input data are assigned to the source nodes and only the output net performs the combination function in generating the final ensemble results. As this kind of MNN is the simplest one, it would be the ideal model if it could implement the combination function. Unfortunately, my experiments using this MNN model failed and experimental details are reported in section 4.3.

b) Combining Ensemble Members Results along with Original Input Code

After using the outputs of ensemble members as the input of the second stage MNNs, there was a thought that the original input to the second stage MNNs along could be provided with the outputs of those ensemble members. The aim was to tell the combiner what part of the decision space it was in, as some ensemble members may give more reliable predictions in some areas of the space. The original input, which was used to generated these ensemble members outputs were concatenated with the corresponding ensemble members results. The joint parts are used as the input for the second layer neural networks training as illustrated in Figure 3.4. The results on this model was reported in section 4.3.2.

In summary, different structures of neural network models were applied in the ensemble experiments. From single layer to two layers neural networks, that have various number of hidden neurons, have been constructed. The details of the experiments using different structures of MNNs presented in this section are described in section 3.5. Further, the comparison on the performance of these MNN models detailed in Chapter four will show the effectiveness of these models.

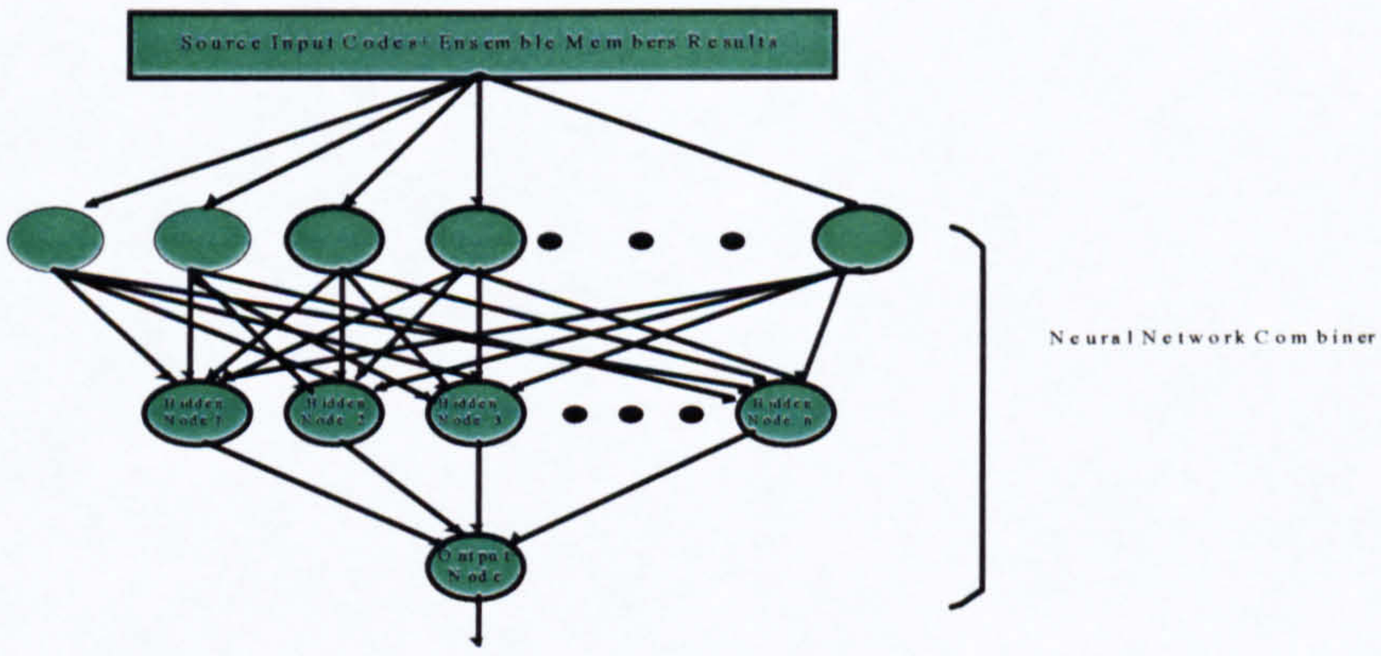

Figure 3.4 Using original input and ensemble members results together as input of second stage NN

3.3 NN Algorithms and Programming Coding

The main concern in this thesis is whether MNNs can be used as one of ensemble combination techniques and furthermore is MNNs generalization better than majority voting or can it at least achieve the same performance level. For this purpose, any of the NN training algorithms can be used, as long as the ensemble members used for these two ensemble combination approaches are the same. Choosing a NN algorithm which can be easily applied and widely used is ideal, so three universal NN algorithms were considered: Radial-Basis function network (RBF) (Powell, 1985), Support Vector Machines (SVMs) (Boser, Guyon and Vapnik, 1992) and the Back-Propagation algorithm applied to MLPs (Rumelhart, Hinton and Williams, 1986).

The RBF network requires sufficient data that represent all aspects of the problem being solved in order to generate an accurate performance (Callan, 1999). It is not my purpose in this thesis to investigate whether the information provided by the original input data is enough or not for training using such a NN model. Instead, the real concern is to explore the combination capability of MNNs. For this reason RBF networks was not selected in the experiments, which would cost much effort in training.

The support vector algorithm's running time is slower than a NN using Back-Propagation in achieving a similar generalization performance (Haykin, 1999). After considering this fact the Back-Propagation algorithm was selected.

The Back-Propagation algorithm was formally proposed in 1986(Rumelhart, Hinton and Williams, 1986). It can be easily implemented and can be obtained from many NN software tools. As the Back-Propagation algorithm is the most frequently used algorithm in NNs, for the purpose of future testing and applications on MNNs, the Back-Propagation algorithm was used through the whole experiments. Programming code was implemented by using the Neural Network Toolbox and Aston Netlab software under Matlab Version 13.0 for Windows.

3.4 Data

There are two groups of data sets being employed in the experiments. All the data sets are two-classification problem except the iris data set, which is a three-classification problem. One group of data sets is benchmark data from the machine learning website. Another group of data is human DNA sequences (Thanaraj, 1999). Their details are described in the section 3.4.1 and section 3.4.2 respectively. The efforts in using a wide variety of data to apply to the multistage neural network models will prove the effectiveness of my novel ensemble models for a wide variety of applications. Furthermore, the application of the gene splice sites data to my MNN models attempts to extend the research in the gene prediction field by using ensemble methods.

3.4.1 Machine Learning Data

The UCI (ftp://ftp.ics.uci.edu/pub/machine-learning-databases) maintains the international machine learning database repository, an archive of over 100 databases used specifically for evaluating machine learning algorithms. From there, five frequently used data sets were selected. The details of these data sets are described below.

- Wisconsin Breast Cancer Database

This breast cancer databases was obtained from the University of Wisconsin. There are 699 instances. Every instance includes clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses attributes, which are represented by 1 through 10. Each instance has one of 2 possible classes: benign or malignant represented by 0 and 1 respectively.

- BUPA liver disorders (Bupa-Liver)

The source was from BUPA Medical Research Ltd. Each line in the BUPA data file constitutes the record of a single male individual. Attributes include mean corpuscular volume, alkaline phosphotase, alanine aminotransferase, aspartate aminotransferase, gamma-glutamyl transpeptidase, number of half-pint equivalents of alcoholic beverages, selector field used to split data into two sets: order or disorder represented by 1 or 2.

- Johns Hopkins University Ionosphere database (Ionosphere)

This is a binary classification task. All 34 predictor attributes are continuous. The 35th attribute is either "good" or "bad". "Good" radar returns are those showing evidence of some type of structure in the ionosphere which is defined 1. "Bad" returns are those that do not; their signals pass through the ionosphere, which is defined 0.

- Iris Plants Database

The data set contains 3 classes stand by 1, 2 and 3. (Iris Setosa, Iris Versicolour, Iris Virginica) of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other. Attributes are sepal length in cm; sepal width in cm, petal length in cm, petal width in cm. Class Distribution is 33.3% for each of 3 classes.

- Pima Indians Diabetes Database (Pima)

Original owners: National Institute of Diabetes and Digestive and Kidney Diseases. All samples were taken from patients who are females with at least 21 years old of Pima

Indian heritage. Attributes are number of times pregnant, Plasma glucose concentration in a 2 hours oral glucose tolerance test, Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), 2-Hour serum insulin (mu U/ml), Body mass index (weight in kg/(height in m)^2), Diabetes pedigree function, Age (years). Class value 1 (500 instances) is interpreted as "tested positive for diabetes" and class value 0 (268 instances) is interpreted as "tested negative for diabetes".

It was noticed that there are some missing values in some of data sets and thus the instances that contain missing values have been removed from the source data. The summary of these data set introduced above is shown in Table 3.1.

Data	Number of Cases	Number of Input features	Number of Output features
Breast-cancer –w	682	9	2
Bupa-Liver	345	6	2
Ionosphere	351	34	2
Iris*	150	4	3
Pima-Diabetes	768	8	2

Notations: * - multi-class data set.

Table 3.1 Summary of UCI machine learning depository data sets

3.4.2 Human Eukaryotic DNA Sequence Data

A splice data set from human DNA sequences (Thanaraj, 1999) was used for these experiments. Using computational tools to identify human gene structural elements (e.g. translation start/stop and splice sites) is one of two major parts in genome sequencing projects (another part involves using biological experiments). Furthermore, it is well recognized that pinpointing exactly the splice site signals in a DNA sequence plays a key role in prediction of gene structures. Several computational approaches (Thanaraj, 2000), such as discriminant function, single NNs, hidden Markov models and decision trees (Hudson, Whitley, Ford and Browne, 2002), have been employed, since predicting splice junction sites is crucial in finding gene coding. Using the splice data set in this thesis

explores the effectiveness of the multistage neural networks in the application of finding human gene sequences.

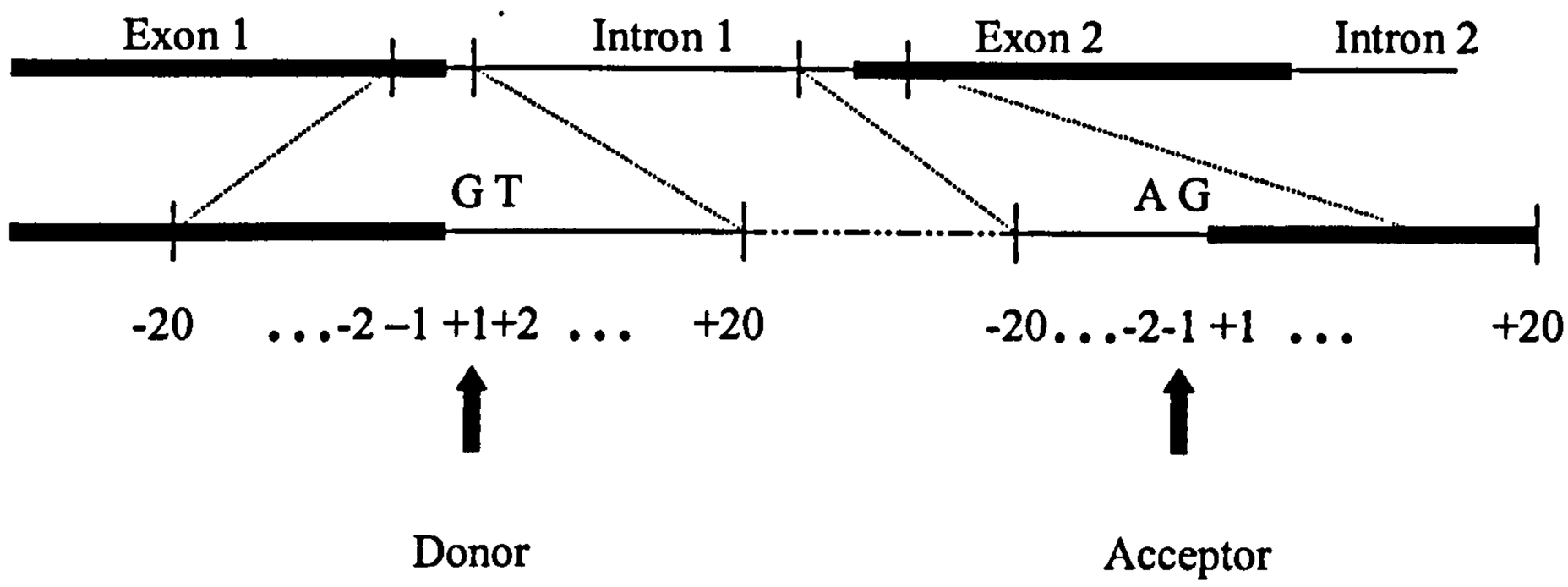

Figure 3.5 Donor and Acceptor Junctions

As shown in Figure 3.5, the splice site refers to the consensus dinucleotide sequences starting from GT and ending with AG. Here the observations of splice are nucleotides and each letter represents one type of nucleotides. Therefore GT and AG pairs with a DNA sequence are indications for splice junctions. Identifying the real splice sites can be split into identifying real donor (e.g. Exon-Intron) junctions and real acceptor (e.g. Intron-Exon) junctions. However, not all the sequences containing GT are marked as donor and also not all the sequences containing AG are marked as acceptor. To find the genuine donor and acceptor site from the mixture of real and non-functional junction sites is my task.

There are 567 real donor sites along with 943 false donor sites and 637 real acceptors along with 468 false acceptors in the data set. These data set were processed by Thanaraj (1999) and are standard data sets used to compare the extensive research in this area, and have been placed on the European Bioinformatics Institute website for use by other researchers. The test data set was prepared by (Hudson, Whitley, Ford and Browne, 2002) according to the information provided by Thanaraj (1999).

In my experiments, the nucleotide regions starting from position -6 and ending at position +3 (where donor signal GT is at position +1 and +2) in the donor junction were used; while for acceptor junctions, the nucleotide regions between position -11 to +1 (where the AG marker is at positions -2 and -1) were used. In experiments, the donor signal GT and acceptor signal AG were omitted (as these are invariant) and all the nucleotide nodes represented by letters were coded into binary categorical form. For example, T, C, A, and G were defined as:

T→ (1 0 0 0) C→(0 1 0 0) A→(0 0 1 0) G→(0 0 0 1)

In general, two major types of data set were used in my experiments. One type is benchmark machine learning data sets. Another type is a standard bioinformatic data. The performances of all the data sets on MNNs are compared with other researchers' results in chapter four.

3.5 Experimental Procedures

3.5.1 Data Preparation

The sequence of each source data set has been randomized first and the training data for each NN model are randomly picked from it, as randomizing the order of presentation of training samples tends to make the search in weight space stochastic over the NN training. Therefore, the different sequential training data makes it more likely different weight updating possible during the training (Haykin, 1999). Moreover, as the training data is designed to be randomly picked from the source input for each ensemble candidates, each such training data set can be counted as a subsample of the source input. The learning of classifiers on these subsamples that contain different features of source input can be different (Breiman, 1996a). As a result, diverse ensemble candidates can be created. The general procedure of preparing data is illustrated in Figure 3.6.

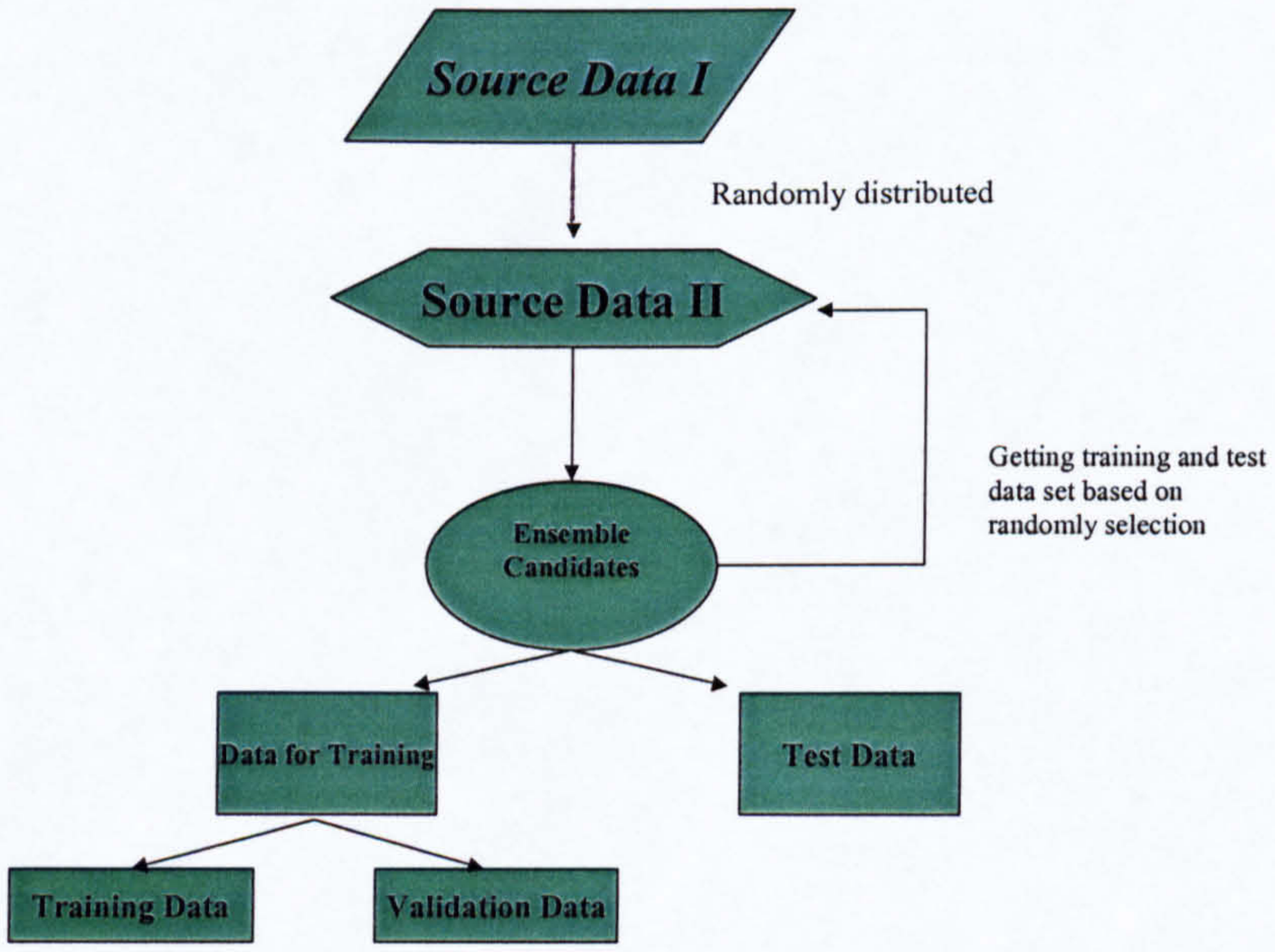

Figure 3.6 Data preparation procedure

As shown in figure 3.6, the sequences of each source data set were randomly distributed first in order to get the average distributive training and test data set. For each ensemble candidate, 4/5 of the source data were randomly taken for the purpose of training, and 1/5 of the source data were taken randomly as the test data. To prevent over-fitting problems occurring, the training set was further partitioned into two disjoint subsets: training data set was used in training and validation data set was used to stop the training when the validation error rate started to increase instead of decreasing. The size of the validation data was about 1/3 of the data for training. The test data set was used to assess the performance of each of the ensemble candidates. All the ensemble candidates followed the above rule to get the data from the source data set. Therefore the size of training data, validation data and test data kept unchanged for each ensemble candidates.

In my experiments, all the data have been partitioned into three subsets (training/validation/test) in the approximate ratio of 2:1:1 except the Iris data set, which

used the bagging method to generate the training, validation and test data due to the small size of source data. The size of each group for Iris data was the same.

Data	Number of Cases	Number of training data	Number of validation Data	Number of test data
Breast-cancer –w	682	400	150	132
Bupa-Liver	345	150	95	100
Ionosphere	351	200	75	76
Iris*	150	150	150	75
Pima-Diabetes	768	400	200	168

Table 3.2 Summary of data partition of machine learning data sets

All the above data in Table 3.2 did not contain duplicated values except the Iris data set, whose data sets were generated using the bagging (Breiman, 1996a), which is introduced in chapter one. For each ensemble candidate, the data procedures stated above were followed in generating the above number of group data for each data set.

For the splice data set, as the size of data is larger than any of above data, an independent test data set was applied in the experiments. However, the rules selecting training and validation data set were still same as illustrated in figure 3.6 for ensemble members. Table 3.3 showed the detail of data partition for the splice data.

Data	Number of Cases	Number of training data	Number of validation data	Number of test data
Splice-Donor	2112	1010	500	602
Splice-Acceptor	1555	805	300	450

Table 3.3 Summary of data partition of splice data

After completing the random selection of training, validation and test data (except splice sets, which has the independent test data already), all the training data was randomly

distributed again to distribute the data evenly. Then the data groups were ready for training.

3.5.2 Training procedures

3.5.2.1 Ensemble Candidates Training

All the ensemble candidates were trained by the back-propagation algorithm using one hidden layer feedforward multilayer perceptrons (MLPs). As detailed in the data preparation section (3.5.1), training data, validation data and test data are taken randomly from the source, and prepared for training. Within each ensemble candidates training, the MLP was trained by the combination of changing the learning rate within a range along with changing the number of hidden neurons. Each such kind of NN model was trained 20 times with random starting weight initialization. So the total number of training permutations of NN models for each ensemble candidate is:

*Total number of neural network model choices for each ensemble candidate = number of parameters changing * number of hidden neurons selection* number of random initial weight initializations*

The actual changing range of the learning rate is case dependent, and so is the number of hidden neurons. Very rough empirical training was done before these parameters could be set in a reasonable range for experimentation. The details of parameter settings are illustrated in table 3.4.

Data	Learning Rate Range (λ)	Learning Rate Step Size (Δλ)	Range of Hidden Neurons	Hidden Neurons Step Size
Breast-cancer –w	0.1 - 8	0.2	2 -10	1
Bupa-Liver	0.1 - 1	0.1	2 - 10	1
Ionosphere	0.2 - 3	0.3	3 - 8	1
Iris*	0.1 - 1	0.1	2 - 10	1
Pima-Diabetes	0.1 - 3	0.1	2 - 10	1
Splice-Acceptor	0.1 - 1	0.1	2 - 10	1
Splice-Donor	0.1 - 1	0.1	2 - 10	1

Table 3.4. Settings of parameters in the first stage training

During the training the validation data set served as a monitor to check the progress of training after each 5 epochs. The training was stopped once the validation accuracy rate decreased and the best result achieved was saved. The test data set was then used to test its performance. The best result after iterations was saved. The NN model that produced such a result was treated as one of the ensemble candidates. Below, the experimental procedure is laid out as Algorithm 1, the Matlab code for which can be found in Appendix A.

Algorithm 1:

1. *Distribute the sequences of source data and taking training data, validation data and test data in the ratio of 2:1:1 randomly into different subsets.*
2. *Set up the basic structure of an ensemble neural network model (including the number of input and output neurons, number of training cycles, etc.).*
3. *Set up a loop for learning parameters:*

   3.1 FOR (λ_0 : increment ($\Delta\lambda$): λ) (λ_0, λ stand for the real value of learning rate, increment stands for the increasing units of each time.

   3.2 Setting up a loop for number of hidden neurons: FOR (number of hidden neurons = 1: increment: m).

      3.3 Setting up a loop: FOR (runtimes = 1:1:μ), running random initialization.

3.4 Training the ensemble candidate using the above initialization, learning parameter and number of hidden neurons from the above loops.

After one cycle of training is finished: Apply the test data to the NN model, get the test results and save the NN model details, training results, validation results and test results into T1, T2, T3, T4 files respectively;

If runtimes == 1, save the current T1, T2, T3, T4 file as the best B1, B2, B3, and B4 file; otherwise Compare the validation results with the one from the best file B3. If error rate is still getting less, save the information from T1, T2, T3, and T4 files into best files B1, B2, B3 and B4 separately; otherwise go to step 3.5.

3.5 Back to step 3.3. If the number of random initializations reaches μ, go to step 3.6.

3.6 Back to step 3.2. If the number of hidden neurons is already increased to m, go to step 3.7.

3.7 Back to step 3.1. If the value of learning parameters has been λ, then the whole loop's training is finished.

4. Finish the first stage training of ensemble candidates.

The training procedures of each of the NN ensemble candidates are illustrated above. Since the training data sets for each candidate are different, different generalization results are expected. Among these candidates, those models with high prediction rate and showing different error rate on the test data set will be selected as the ensemble members. The rule on selecting ensemble members are explained in next subsection. Based on well-trained NN ensemble members, another NN model (called the second stage NN) will be

applied as a combination mechanism to combine these ensemble members' result with the aim of producing final results.

3.5.2.2 Selection of the Ensemble Members

There are two main considerations while selecting ensemble members. One is how many ensemble members need to be created. Another one is how to select ensemble members from ensemble candidates.

After the generation of ensemble members, experiments combining three, five, seven, nine, eleven, thirteen, fifteen, seventeen and nineteen ensemble members using MNNs and majority voting were explored. The reason that selecting odd numbers of ensemble members to be combined is related to using the majority voting method. As majority voting takes the majority of the most ensemble members with the same outputs as the final ensemble results, an odd numbers of ensemble members were used in experiments to avoid confusion when half of the ensemble members agree on one result while another half disagree (in a two classification problem). For instance, in figure 3.8, all the ensemble members outputs have to be either fall in the group marked as 1 whose value exceeds 0.5 or fall in group 0 whose value less or equal to 0.5. Here the threshold is set as 0.5. As the number of ensemble members is odd, there is no chance that two groups have the same number of ensemble members. Therefore there must be a winner in this case. Several ensemble techniques were applied, like data pre-processing, randomly initialization combination weights and using different structures of NNs with different parameters to prevent the same ensemble candidates to be created. Thus the intention was to select the second ensemble members' selection strategy (section 1.2.2.2(b)), which is creating an exact number of ensemble members. The ensemble members in my experiments were actually created step by step according the ensemble performance.

In my experiments, for each data set, though nineteen ensemble candidates were created, however these ensemble candidates were not created at the same time. Initially, only five ensemble members were generated. Three ensemble members were needed to be

selected from these five candidates. Those ensemble candidates whose generalization performances were good and diverse from others were chosen as the ensemble members. One should keep in mind that **Accuracy and Diversity** are two key factors to consider when choosing the best ones. Unfortunately there are no standard tools to be applied to select ensemble members, in the application two factors that affect the performance of ensembles can only be used as the general rule. The general rule stated above was applied while selecting the ensemble group containing three members.

The other ensemble groups that include five, seven... nineteen members didn't go through the selection procedure as the size of ensemble groups was increased gradually according to the performance of MNNs. Thus two more ensemble candidates were created with the increment of the size of ensemble group each time. Under such circumstance, there was no choice to select. Moreover, the major reason for doing so is the diverse of ensemble candidates could be created during their training, which involved data pre-processing, different weight random initialization and different parameters' selection. Also it is more realistic to create the diverse ensemble candidates during their training instead of selecting them from a large number of candidates. The breast-cancer data set was used as an illustration to show how the ensemble members were selected in the experiments and how the diverse of ensemble candidates were created during their training. Table 3.5 lists the performance of each ensemble members and table 3.6 presents the constitutuents of each ensemble group.

Breast-cancer-w	Single Neural Networks Performance (%)									
No. of ensemble member	1	2	3	4	5	6	7	8	9	10
Performance	97.21	97.50	97.21	96.62	96.76	97.06	97.06	97.35	97.50	97.65
No. of ensemble member	11	12	13	14	15	16	17	18	19	
Performance	97.50	97.35	97.47	96.47	96.91	97.50	95.88	96.62	97.65	

Table 3.5 Breast-cancer-w ensemble members performance

Data	The actual ensemble members for each groups of ensemble models								
	3	5	7	9	11	13	15	17	19
Breast-cancer –w	1,2,5	1,2,3,4,5	1-7	1-9	1-11	1-13	1-15	1-17	1-19

Table 3.6. The selection of ensemble members for breast-cancer-w

As the third ensemble candidate's performance is same as the first one, to avoid use of two identical members, it was discarded. The fifth ensemble candidate prediction rate is slightly better than the forth one, so the fifth one was selected as one of the members of group of three. The more details of these ensemble candidates that can further show that none of them are same can be found in table 4.4 in the next chapter. That's why all the ensemble members were adopted for the ensemble group whose size is bigger than five.

Once the ensemble members are selected, the second stage MNNs training can be started based on the information provided by these ensemble members.

3.5.2.3 Second Stage Training

All the training data were injected into each of the ensemble members, and the corresponding results along with the target values were used as the input for the second stage training. The MLP in the second stage was trained by these inputs and the test data was applied to the MLP when training finished. The general procedures were similar as the first stage ensemble members training. However, there are some small differences because of the different functionality of the two stages. The purpose of generating the ensemble candidates was to create as much diversity amongst ensemble members as possible. So the emphasis was put on how to generate diversity among ensemble members in the first stage. In contrast, the second stage neural net performs the combination functions where more accurate results are expected. Hence, there are two noticeable differences between two stages training. First, to reflect the general performance of MNNs and due to the limitation of the small size of most data sets (except splice data set which had independent test data), a cross validation approach was used in the second stage training. In addition, the structures and parameters setting of MLPs in the two stages were different as well. In the first stage, structures and parameters of each ensemble candidate were selected automatically by Algorithm 1. Meanwhile in the second stage training, the structure and most of parameters of the NN model were fixed. The reason for using a fixed structure and MLP in the second stage was to make the combination procedure as simple as possible without affecting the

combination performance. More discussions about this point are stated later in this section.

As 10-fold cross-validation were applied to most of the data set, the averaging of results based on 10-fold cross-validation were counted as the final ensemble results. The processing of second stage training using 10-fold cross-validation are illustrated in Figure 3.7 and Figure 3.8.

As illustrated in Figure 3.7, the whole source data set was divided equally into ten parts. The first part was used as the test data set and therefore was not involved into any second layer training at this stage. In the mean time, the other parts of the data (nine parts altogether) were injected into the selected ensemble members (e.g. three ensemble members were used in Fig. 3.7 (1)). Afterwards, the results that each ensemble member generated were concatenated together, combined with the corresponding target value as the input for the second layer neural network. Once the training of second stage MLP finished, the test data was applied and the result was saved. While completing the whole process as stated above, the next cycle would begin and this time the second part of data was used as the test data (see Fig. 3.7 (2)). The whole loop would finish till the test data set reached to the last part of source data.

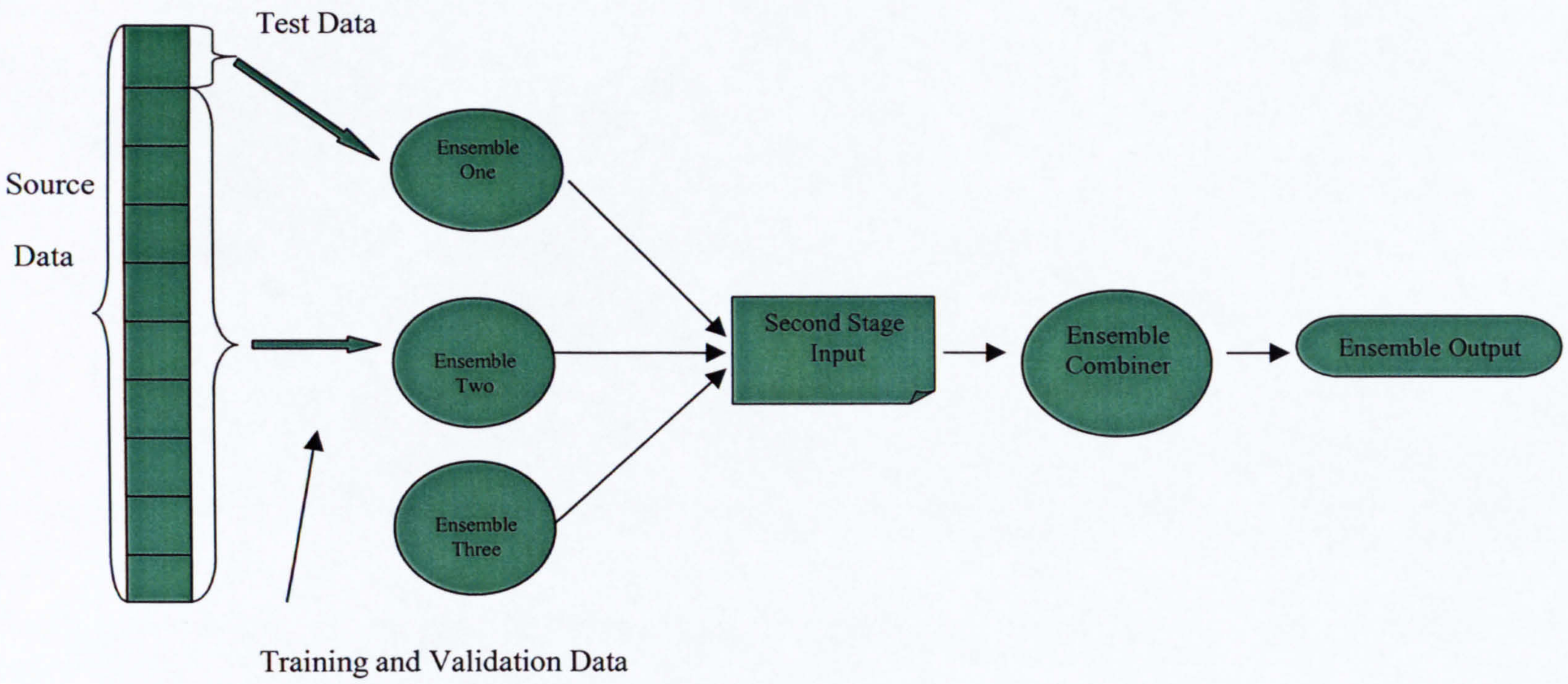

Figure 3.7 Using 10-fold cross-validation in the second stage training (1)

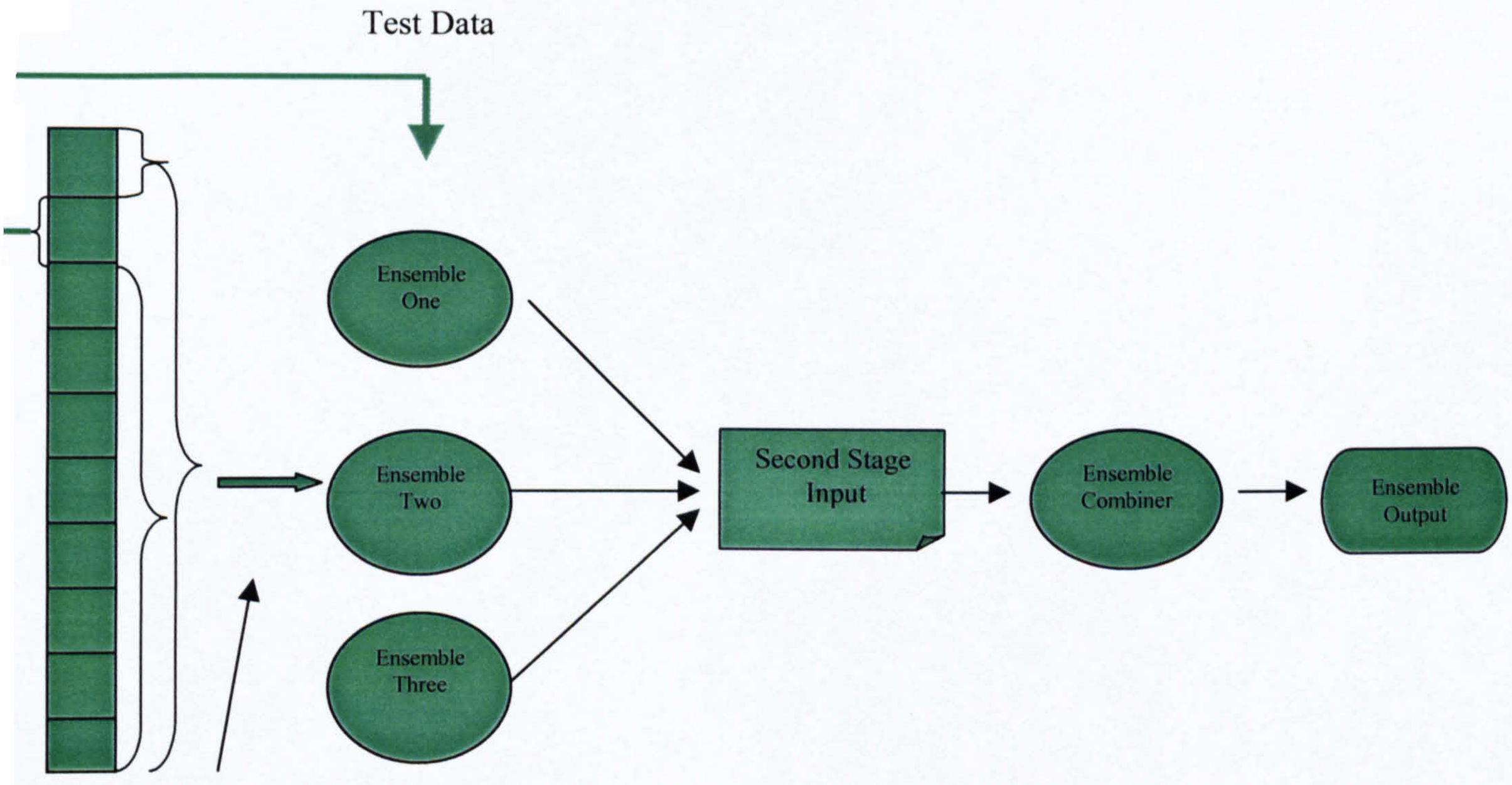

Figure 3.7 Using 10-fold cross-validation in the second stage training (2)

Figure 3.7 (2) shows that the second part of source data was used as the test data and the rest of the parts were used as a whole and injected into the ensemble members.

For each cycle of cross-validation, the training procedure of the MLP in the second stage was the same as the training of ensemble candidates in the first stage. However, some settings of the MLP were fixed. For instance, just one MLP with one hidden layer was employed in the second stage. The number of neurons in the hidden layer was set to two and the learning rate was fix to 0.1. The reason of using these fixed setting was

demonstrated by extensive empirical experiments. In our initial experiments, the optimal numbers of hidden neurons and the learning rates were set to be picked up by the machine automatically within given ranges. After extensive simulations the optimal number of hidden neurons and learning rate were chosen as two and 0.1 respectively, and then used on all the data sets used in our experiments. Therefore, these setting were fixed and further training based on them was carried out. The training procedure of the second stage neural net is explained below as Algorithm 2, the Matlab code for which can be found in Appendix number B.

Algorithm 2

1) Distributing the whole source data randomly.

2) Dividing the source data into ten parts equally.

3) Setting up the basic structure of ensemble neural network model where learning rate was 0.1 and number of hidden neurons was 2

4) Setting up loop For (datapart = 1:1:10), taking one part of data as test data and the rest of source data were split into training and validation data in the ratio of 2:1 randomly.

5) Injecting the training data, validation data and test data into each ensemble member model respectively and the generated results by each ensemble members were combined together along with the corresponding target values as the input training, validation and test data for the second layer.

6) Setting up a loop: FOR (runtimes = 1:1:1000) , running random initialization. Starting training by taking random initialization in above loops.

7) After one cycle of training finish: Applying the test data into neural network model and getting the test results. If runtimes == 1, saving the net information, training results, validation results and test results into FN, FV, FU, FT files; Otherwise move to step 8.

8) Saving the net information, training results, validation results and test results into N, V, U, T files separately and comparing the validation

results with the results in FU files. If better, updating FN, FV, FU, FT by N, V, U, T files; otherwise move to step 9.

9) Taking the test results from file FT and adding it to file SUMTEST. While number of random initialization <1000, Back to step 6; Otherwise go to step 10.

10) Back to step 4 until datapart reaches 10;

11) Finishing the second layer training and averaging the results saved in file SUMTEST.

The above training procedures were applied to the second stage NN training, no matter how many ensemble members were going to be combined. The results generated by the MNNs were compared with the results generated by the simple majority voting combination method.

3.6 Comparison

In order to test the performance of MNNs, the most frequently used classification combination method, majority voting, was applied to combine the same ensemble members used by MNNs. The detail of using majority voting is described in Figure 3.9. In addition, the comparison of the MNNs performances by using different number of ensembles was conducted as well. The optimal number of ensemble members being combined for any particular case through these further experiments was expected to be found. Furthermore, for testing the effect of selecting different ensemble members, the comparison of MNNs performance was done by using different groups of ensemble members where the number of ensemble members was fixed to three.

- **Comparison with Majority Voting**

In my experiments, each part of the source data (equally divided into ten parts) was tested by using majority voting and MNNs. Thus, ten test results of these two ensemble methods were obtained for the whole data set. There might be substantially different performance over different parts of the data space, therefor averaging the sum of ten test

results was the desirable way in comparing the general performance of MNNs and majority voting. Equation number 3.1 below displayed shows how to calculate the means.

$$\bar{x} = \frac{1}{n}\sum_{i}^{n} x_i \qquad (3.1)$$

In my experiments n equals to ten, the x_i represents the result of each cross-validation and $\bar{x}$ stands for the value of means for each ensemble group.

- Introduction to the experiments using Majority Voting

The majority voting's working procedures in my experiments are illustrated in Figure 3.8. As shown in Figure 3.8, the source data set was divided equally to 10 parts as previously done in second stage training. Each part was used as the test data, which was injected into well-trained ensemble members. The data in other parts was used as the training data. The results generated by each ensemble member would go to either of groups Sum1 or Sum0, according to their real values. The loop continued until all the ensemble members' results were classified. Next, the sum of the number of ensemble members in the Sum1 group was compared with the sum of the number of ensemble members in the Sum0 group. The one with the greater number was the winner, and its corresponding label was the final ensemble result of majority voting for this part of test data.

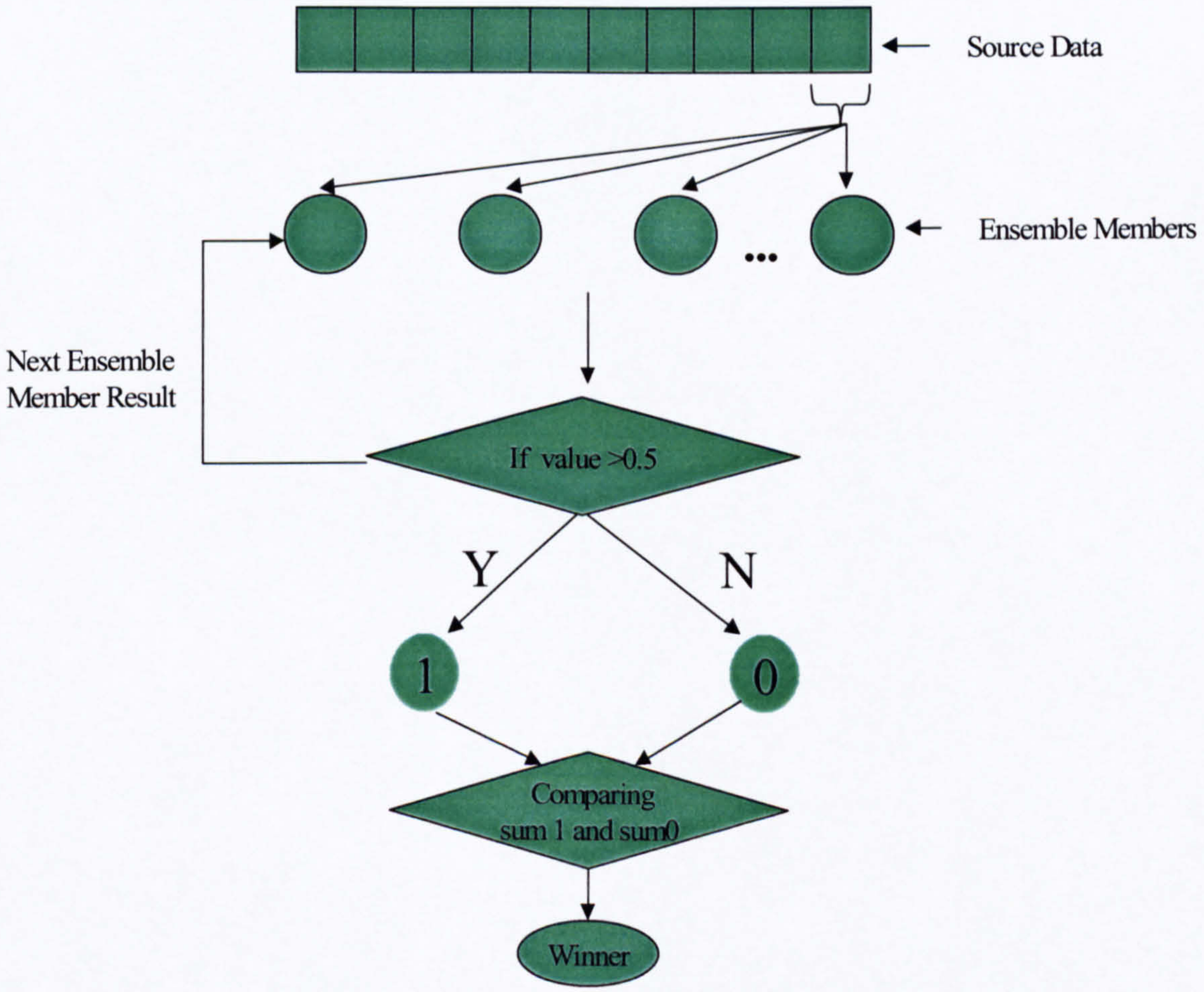

Figure 3.8 Combination of ensemble members by using Majority Voting

- **Comparison Using Statistical Tools**

To measure the significance of differences between the ensemble combination schema, which in MNNs and majority voting are not due simply to random variation in the experiments, two statistical methods were employed to interpret the results of two ensemble techniques. In order to investigate the average ensemble performance of MNNs and majority voting to find which is better, instead of testing the generalization of two ensemble methods separately, a one-tailed t test was used to compare the mean differences of ensemble results. To make sure the t test results are valid, the variances of ensemble results of MNNs and majority voting must be equal statistically. Thus one-tailed F test was used to compare two samples' variances.

➢ Differences in Means

All the results reported in chapter four were the means of the results based on 10-fold cross-validation. Also we know that the distribution of means for a small sample (sample size less than 30) is a t distribution (Mendenhall and Ott, 1980), which means one can use the t test to compare the achievement of MNNs with the performance of majority voting. There are two types of t test. One is called two tailed t test, which is used to assess that two populations are different. Another is called one tailed t test, which is used to examine whose strength of means between two populations is strong or weak statistically. As all my results of MNNs are equal to or better than majority voting, therefore one tailed t test was used to prove that the performance of MNNs is significantly better than that of majority voting. The hypothesis of a one tailed t test and the formula used for the test is illustrated in table 3.8, where samples from two populations were taken: MNNs and majority voting. Table 3.7 lists the parameters used in t test.

	Population	
	MNN	Majority Voting
Population Mean	μ_1	μ_2
Samples from two population	y_1	y_2
Sample Size	n_1	n_2
Sample Mean	$\bar{y}_1$	$\bar{y}_2$
Sample Variance	s_1^2	s_2^2

Table 3.7 T testing samples parameters from MNN and Majority Voting

Null hypothesis (H_0): $\mu_1 - \mu_2 = 0$

Alternative hypothesis: $\mu_1 - \mu_2 > 0$

Test statistic: $$t = \frac{\bar{y}_1 - \bar{y}_2}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$ where

$$S = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$ where

$$s_1^2 = \frac{\sum y_1^2 - (\sum y_1)^2 / n_1}{n_1 - 1}$$

Rejection region: For a specified value of confidence value α, for freedom degree $df = n_1 + n_2 - 2$ and a one tailed test:

Reject H_0 if $t > t_\alpha$ or $t < -t_\alpha$

Table 3.8 Summary of two-sample t test for comparing two populations means

Setting up a null hypothesis assuming MNNs make no difference to the performance compared with majority voting. However, if t falls in the region of rejection which was given in the table 3.8 as $t > t_\alpha$ or $t < -t_\alpha$, rejecting the null hypothesis and accepting the alternative hypothesis that MNNs do make a significant improvement.

- Difference in variances

The investigation on the variances of the ensemble results was shown in table 4.10. Each result in the table was computed on averaging 10-fold cross validation results. Therefore, the variances of these ten 10-fold cross validation results were calculated to check whether the ensemble results for each group generated by MNNs produce more variability than majority voting, or vice versa. The reason using this test was to prove that the improved performance made by MNNs was not at the cost of increase of variability of the ensemble results. Here, a one tailed F statistical test with confidence value was set to 0.05. The corresponding parameters and testing procedures are presented in tables 3.9 and 3.10.

	Population	
	Population with large variance	Population with small variance
Population Variation	σ_1	σ_2
Sample Size	n_1	n_2
Degrees of freedom	$v_1 = n_1 - 1$	$v_2 = n_2 - 1$
Sample Variance	s_1^2	s_2^2

Table 3.9 Parameters for variance testing between MNNs and Majority Voting

Null hypothesis (H_0):	$\sigma^2{}_1 = \sigma^2{}_2$
Alternative hypothesis:	$\sigma^2{}_1 > \sigma^2{}_2$
Test statistic:	$F = \frac{s_1^2}{s_2^2}$
Rejection region: For a given value of α,	
Reject H_0	if $F > F_\alpha$

Table 3.10 Summary of two-sample test for comparing two population variances

Note that in the formula $F = \frac{s_1^2}{s_2^2}$, s_1^2 is the larger of the two sample variances. After checking the F table when using α equals to 0.05, the rejection region was located, the area where value of F should greater than value 3.18.

- **Comparison of Performance Using Different Number of Ensemble Members**

As the performance MNNs was affected by the number of ensemble members being combined, the results of different groups of ensembles with 3, 5, 7, 9, 11, 13, 15,17, 19 ensemble members respectively were studied. The ensemble results of the above nine groups could reflect the trend on how number of ensemble members affecting the performance of MNNs in some degree.

- **Comparison of Performance When Selecting Different Ensemble Members**

The attempt in this experiment was to explore how much influence ensemble diversity would have on the ensemble results. Obviously the diversity among ensemble members of each group was not the same (providing that each of the ensemble members were not the same). Thus, the ensemble combination using MNNs would lead to different results. The experiment would show how much difference could be caused by employing different groups of ensemble members. There was an expectation that the results would

give a guide on selecting ensemble members when using MNNs, or other ensemble combination schemas in the future. As the ensemble diversity is directly tied to the amount of correlation among the classifiers (Tumer and Ghosh, 1996; Breinman, 1996a), different diverse ensemble groups were used to discuss the influence of the diversity on the MNNs performance. The illustration of the process is shown below in Figure 3.9.

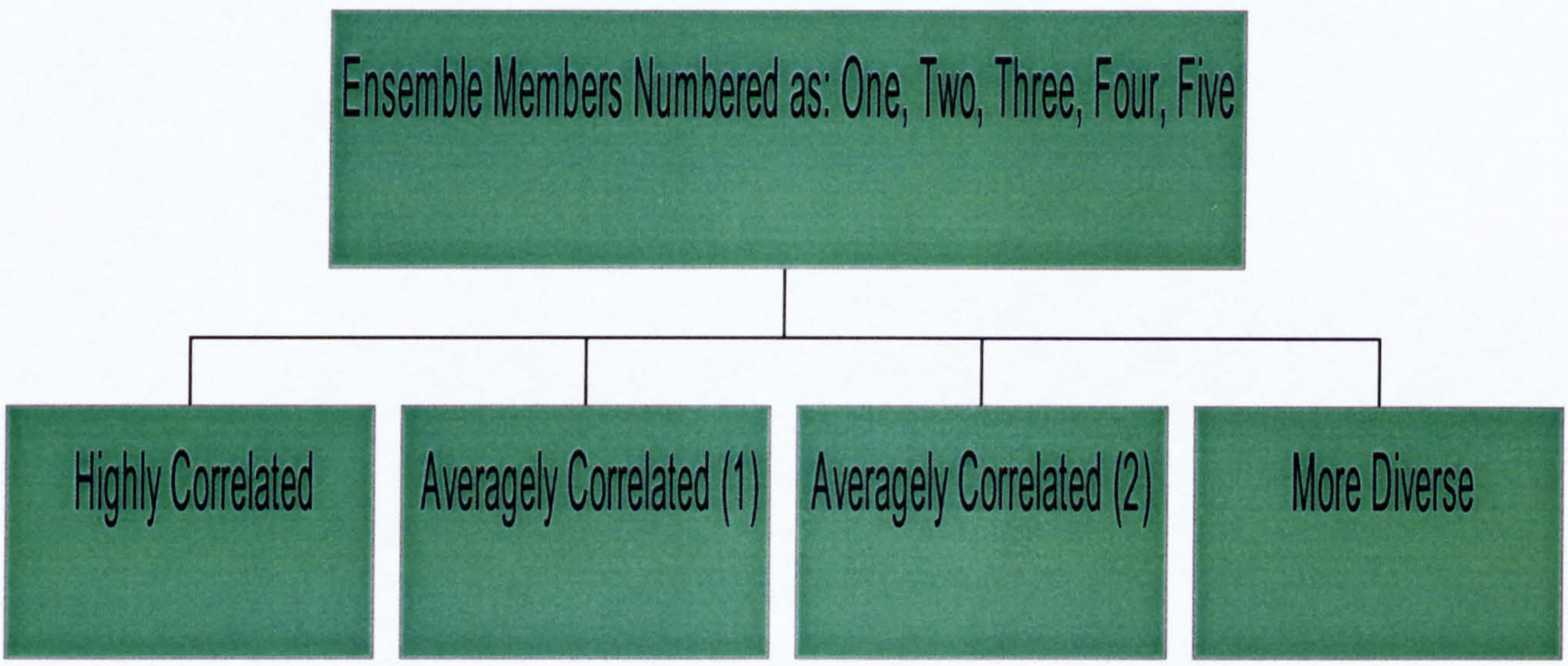

Figure 3.9 Combining groups with different ensemble members

The comparison was conducted using five ensemble candidates. The diversity of each ensemble group was measured by averaging the correlation coefficients of each two members within the group. To save the time on computing, the smallest size of MNNs was used in the experiments, which only constituting three ensemble members. According to the average degree of diversity for each group, the selected four groups of ensembles each having three members were catalogued into highly correlated, average correlated (1), average correlated (2) and more diverse groups. Further experiments

based on the combinations of these ensemble candidates, which have different diversity features, were explored.

3.7 Conclusion

For the assessment of MNN models, the methodology of creating, training and testing ensembles was introduced in this chapter. The structures and settings of NN models, the data sets used in the experiments, the experimental procedures, were all described in detail. In order to compare the performance of MNNs with the most frequently used combination approach: majority voting, some statistic tools were borrowed to test the significant improvement made by MNNs. By applying these procedures, two types of data from different sources were compared, by using MNNs and majority voting. Meanwhile, to explore the strength of MNNs, comparisons on the ensemble results generated by combining different number of ensemble candidates and comparisons on the performance of MNNs by using different ensemble members within certain group were conducted as well. In summary, MNNs is an ensemble combination method, which is technically easy to be applied. Due to the features of the design of MNN models, the improved ensemble performance compared with majority voting is expected.

Chapter Four

Results and Comparisons

4.1 Introduction

Chapter three described the experimental materials and procedures based on MNNs. In this chapter, the corresponding experimental ensemble results using MNNs technique are presented, and comparisons with the frequently used ensemble combination method of majority voting are provided. The significant improvements shown by using MNNs are proved using several statistical tools.

Within this chapter, section 4.2 demonstrates the experimental results from combining three, five, seven, nine, eleven, thirteen, fifteen, seventeen and nineteen ensemble members in MNNs and the corresponding combining results from majority voting, on the machine learning data sets and human gene splice data set. Section 4.3 presents the other experimental results that were generated by the MNN models introduced in part III of section 3.2. Some comparisons are conducted in section 4.4, including getting the optimum number of combining ensemble members by applying MNNs (on some of the data sets) and obtaining different results by combining different group of ensemble members as stated in section 3.6 in chapter three. Later, some conclusions are drawn in section 4.5. Thus, this chapter is concerned with presenting the experimental results achieved by using MNNs. Analysis of the reasons behind these improvements is explored in chapter five.

4.2 Experimental Results from Combining Different Numbers of Ensemble Members

Two types of data set were applied to MNNs and majority voting, including machine learning benchmarks and human gene splice data sets (detailed in section 3.4). Nineteen ensemble candidates were created independently, following the procedures described in section 3.5.2.1. As the test data set was randomly picked up for each ensemble member, to reflect the general performance of these single nets, all the source data was applied on testing these single NNs' performances instead of using its own test data (except the splice data set where independent data test data sets were available). Thus, the results shown in table 4.1 were those of these ensemble members on the whole source data. In contrast, the results shown in table 4.2 were based on independent test data. The structures and parameters of these NNs are reported in table 4.3.

The results generated by MNNs and majority voting are shown in tables 4.6 and 4.7. In order to get the general view of performance of both ensemble combination approaches, ten-fold cross-validation was applied on ensembles; so all the results shown in these tables are the average of ten results rather than the result based on a single test data set. In fact, all the final ensemble results shown in this thesis were all based on the ten-fold cross-validation approach unless it is stated specifically to the contrary.

4.2.1 Ensemble Members' Results

In tables 4.1 and 4.2, all the results are presented as percentage of accurate prediction rates for each ensemble member. As stated in section 3.5.2.1, all the individual nets were created by various ensemble techniques to generate a selection of nets as diverse as possible. Though some of the percentage rates shown in the above tables were the same within one data set (for instance in Breast–cancer-w the first and the third ensemble members' results were same), their nets' initialization and structures were different, so both ensemble members were not actually the same NNs.

Data	Single Neural Networks Performance (%)									
	1	2	3	4	5	6	7	8	9	10
Breast –cancer-w	97.21	97.50	97.21	96.62	96.76	97.06	97.06	97.35	97.50	97.65
Bupa-Liver	72.06	72.65	71.76	73.82	72.06	72.94	71.76	72.35	72.65	73.53
Ionosphere	95.43	94.86	95.14	93.14	94.29	93.71	94.86	90.57	92.57	94.00
Iris	97.60	96.80	98.4	96.00	96.00	98.00	98.40	99.20	97.60	98.40
Pima- Diabetes	73.03	74.08	72.50	74.21	71.32	69.87	68.68	70.00	69.74	71.71

Table 4.1 Ensemble candidates' performances of machine learning data sets (1)

Data	Single Neural Networks Performance (%)								
	11	12	13	14	15	16	17	18	19
Breast –cancer-w	97.50	97.35	97.65	96.47	96.91	97.50	95.88	96.62	97.65
Bupa-Liver	72.94	74.71	73.82	73.53	71.47	73.53	73.82	74.12	72.65
Ionosphere	95.43	95.14	95.43	93.14	90.86	90.00	92.29	92.57	95.14
Iris	98.40	98.00	97.60	98.80	98.80	99.20	98.40	98.80	98.80
Pima- Diabetes	71.84	73.42	72.37	70.53	74.47	70.13	72.63	75.53	73.68

Table 4.1 Ensemble candidates' performances of machine learning data sets (2)

Data	Single Neural Networks Performance (%)									
	1	2	3	4	5	6	7	8	9	10
Splice-Acceptor	86.0	88.00	88.44	87.11	89.33	87.33	88.44	87.56	88.22	89.11
Splice-Donor	94.52	94.35	94.85	94.86	94.19	94.68	94.19	94.85	95.02	94.52

Table 4.2 Ensemble candidates' performances of splice data sets (1)

Data	Single Neural Networks Performance (%)								
	11	12	13	14	15	16	17	18	19
Splice-Acceptor	88.89	88.67	88.89	89.11	88.67	88.67	88.89	88.67	87.33
Splice-Donor	95.02	94.52	95.02	94.52	94.19	94.85	94.85	95.18	95.02

Table 4.2 Ensemble candidates' performances of splice data sets (2)

Data	Single Neural Networks Number of Hidden Neurons and Learning Rate									
	1	2	3	4	5	6	7	8	9	10
	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ
Breast–cancer-w	4/0.2	3/0.2	3/0.2	8/1	8/6	3/0.5	4/3.7	2/0.7	4/0.5	3/0.1
Bupa-Liver	3/2.3	5/1.1	5/2.3	4/2.3	2/2.3	9/0.9	9/0.6	9/1	5/0.8	10/0.9
Ionosphere	3/0.2	3/0.2	4/0.2	3/1.1	7/0.2	8/0.1	5/0.3	8/0.1	7/0.1	5/0.5
Iris *	2/0.2	2/0.2	3/0.2	3/0.2	2/0.2	10/0.9	7/0.9	3/0.2	2/0.1	2/0.2
Pima- Diabetes	5/1.7	3/2.3	5/1.4	4/2.9	4/0.8	9/1	2/1	7/0.3	2/0.6	3/1.1
Splice-Acceptor	8/0.6	4/0.3	9/0.8	3/0.9	8/0.4	2/0.1	4/0.8	3/0.1	8/0.4	2/0.5
Splice-Donor	4/0.4	8/0.6	8/0.9	5/0.9	2/0.3	3/0.9	9/0.1	8/0.5	4/0.3	4/0.1

Table 4.3 Ensemble Members' structures where N refers to the number of hidden neurons and λ refers to the learning rate (1)

Data	Single Neural Networks Number of Hidden Neurons and Learning Rate								
	11	12	13	14	15	16	17	18	19
	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ	N/λ
Breast–cancer-w	4/0.1	5/1.3	6/0.9	2/0.1	7/0.3	4/0.1	9/0.3	4/0.1	5/0.4
Bupa-Liver	7/0.6	8/0.3	7/1	4/0.6	8/0.3	6/0.4	7/0.6	3/0.9	6/0.3
Ionosphere	3/0.2	4/0.2	3/0.2	4/0.3	10/0.6	4/0.3	7/0.6	5/0.9	7/0.3
Iris *	2/0.2	2/0.2	3/0.2	5/0.2	7/1.0	10/0.9	10/1.0	9/1.0	3/1.0
Pima- Diabetes	4/1.4	5/0.7	6/2	7/1.4	9/1.1	3/1.0	6/1.9	9/1.9	3/1.1
Splice-Acceptor	10/0.3	4/0.2	10/0.3	10/0.8	10/0.2	10/0.1	10/0.1	6/0.7	10/0.8
Splice-Donor	3/0.5	3/0.9	3/0.1	4/0.4	2/0.3	8/0.9	8/0.9	7/0.1	10/0.8

Table 4.3 Ensemble Members' structures where N refers to the number of hidden neurons and λ refers to the learning rate (2)

Source data part	Ensemble candidates performance on each part of source input data (%)									
	1	2	3	4	5	6	7	8	9	10
1	91.18	95.59	91.18	91.18	88.24	92.65	92.65	94.12	92.65	95.59
2	98.53	98.53	98.53	95.59	98.53	98.53	1.00	98.53	98.53	1.00
3	98.53	97.06	97.06	95.59	98.53	95.59	98.53	97.06	98.53	98.53
4	94.12	94.12	94.12	95.59	94.12	94.12	95.59	92.65	94.12	94.12
5	94.12	97.06	95.59	97.06	94.12	95.59	94.12	97.06	95.59	94.12
6	1.00	98.53	1.00	1.00	98.53	1.00	97.06	1.00	1.00	1.00
7	97.06	97.06	98.53	98.53	97.06	97.06	97.06	97.06	97.06	95.59
8	1.00	98.53	97.06	97.06	98.53	98.53	97.06	98.53	98.53	1.00
9	1.00	98.53	1.00	95.59	1.00	1.00	98.53	1.00	1.00	1.00
10	98.5	1.00	1.00	1.00	1.00	98.53	1.00	98.53	1.00	98.53
Average	97.21	97.50	97.21	96.62	96.77	97.06	97.06	97.35	97.50	97.65

Table 4.4 Performance of ensemble candidates on Breast-Cancer-W data set (1)

Source data part	Ensemble candidates performance on each part of source input data (%)								
	11	12	13	14	15	16	17	18	19
1	97.06	95.59	94.12	88.24	88.24	95.59	91.18	89.71	95.59
2	1.00	98.53	1.00	98.53	98.53	97.06	97.06	98.53	1.00
3	98.53	98.53	98.53	97.06	98.53	98.53	95.59	98.53	98.53
4	92.65	94.12	95.59	94.12	94.12	94.12	92.65	92.65	94.12
5	95.59	94.12	94.12	92.65	95.59	95.59	94.12	95.59	95.59
6	98.53	98.53	98.53	1.00	1.00	98.53	94.12	98.53	98.53
7	97.06	98.53	98.53	97.06	97.06	98.53	97.06	97.06	98.53
8	97.06	98.53	98.53	98.53	98.53	98.53	98.53	97.06	97.06
9	1.00	98.53	98.53	1.00	98.53	98.53	98.53	98.53	98.53
10	98.53	98.53	1.00	98.53	1.00	1.00	1.00	1.00	1.00
Average	97.50	97.35	97.65	96.47	96.91	97.50	95.88	96.62	97.65

Table 4.4 Performance of ensemble candidates on Breast-Cancer-W data set (2)

Even when the structures and parameters were the same, the two ensemble members were not exactly same due to the different initialization conditions and different training data set. For example, also in Breast–cancer-w, the second and third ensemble member's number of hidden neurons and learning rate were all the same. However these two NNs were not the same as their prediction rates are different. Table 4.4, which gives the precise view of all NNs performance on the same input of breast-cancer-w data, would help us to understand the error diversity between them. The source breast-cancer-w data was equally divided into ten parts and each part of data was applied on all the ensemble candidates. The data in each cell of table 4.4 is the performance of each ensemble candidates on the corresponding part of source data as indicated in the first column. If the two ensemble members are identical, the performance of them should be same on each part of data set, in other word, each cell of the two ensemble members results should be same. Comparing the performance of these NNs on each part of data, one could find that none of these ensemble candidates are exactly same. Only based on the diversity offered by these ensemble candidates, better ensemble results can be generated.

4.2.2 MNNs Combination Results Compared with Majority Voting Performance

4.2.2.1 Selecting of Ensemble Members

The ensemble members being combined for each group (there were nine ensemble groups including using three, five, seven, nine, eleven, thirteen, fifteen, seventeen and nineteen ensemble members) were selected from the nineteen single NNs whose performances are shown in table 4.1, according to the general rule of diversity and accuracy. Two factors needed to be considered while selecting the ensemble members. Not only those with highest accuracy were selected, but also those whose prediction rates and structures were different with others. However, there are no standard tools or arithmetic calculations that can help us to select the optimum ensemble members combination. Further explorations (section 4.4.2) were done to reveal the effect of different combinations of ensemble members on ensemble results. Those results indicated that just selecting the ensemble

members according to the general rule of accuracy and diversity as long as the ensemble members were created using different methods for pursuing maximum diversity among ensemble members was good enough to generate reasonable ensemble results. In the same way as the process of selecting ensemble members on breast-cancer-w data was illustrated in section 3.5.2.2, the ensemble groups for the rest of data sets were selected. The details of these selected ensemble members in each ensemble group on the rest of data sets are shown in table 4.5.

Data	The actual ensemble members for each groups of ensemble models								
	3	5	7	9	11	13	15	17	19
Breast-cancer –w	1,2,5	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Bupa-Liver	1,2,3	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Ionosphere	2,3,4	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Iris *	1,2,3	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Pima- Diabetes	2,3,4	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Splice-Acceptor	1,4,5	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19
Splice-Donor	1,2,3	1,2,3,4,5	1 - 7	1 - 9	1 - 11	1 - 13	1 - 15	1 - 17	1 - 19

Table 4.5 The selection of ensemble members

4.2.2.2 Ensemble Results

The ensemble results generated by using different ensemble members referred to in table 4.5 are shown in tables 4.6 and 4.7. The data shown in each cell were calculated according to equation 3.1, which is addressed in section 3.6.

Dataset		Ensemble Performance (%)								
		3	5	7	9	11	13	15	17	19
Breast-cancer -w	MNNs	97.65	97.65	97.65	97.8	97.79	97.79	97.79	97.79	97.79
	Voting	**97.21**	**97.35**	**97.50**	**97.65**	**97.79**	**97.65**	**97.79**	**97.65**	**97.65**
Bupa-liver	MNNs	73.82	74.12	74.41	74.41	74.71	75.29	75.29	74.71	74.71
	Voting	**72.65**	**73.53**	**74.12**	**73.82**	**74.12**	**75.29**	**74.71**	**74.71**	**74.71**
Ionosphere	MNNs	96.0	96.29	96.57	96.57	96.86	97.43	97.14	97.14	97.14
	Voting	**96.0**	**96.0**	**96.57**	**95.72**	**95.71**	**97.14**	**97.14**	**96.57**	**96.86**
Iris *	MNNs	98.8	98.8	98.40	99.2	98.8	98.8	98.8	98.8	98.8
	Voting	**98.8**	**97.6**	**98.0**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**
Pima- diabetes	MNNs	75.13	75.53	75.92	76.58	75.79	75.53	75.0	75.92	76.84
	Voting	**74.47**	**73.95**	**73.42**	**73.42**	**73.42**	**74.47**	**74.08**	**73.68**	**73.95**

Table 4.6 Comparison of mean results between MNNs and majority voting on machine learning benchmarks, using ten-fold cross-validation (best results in **bold**).

Dataset		Ensembles Performances (%)								
		3	5	7	9	11	13	15	17	19
Splice-Acceptor	MNNs	89.33	89.33	89.33	89.78	89.33	89.33	89.56	89.11	88.89
	Voting	**89.11**	**88.89**	**88.67**	**88.67**	**89.11**	**88.89**	**89.11**	**88.89**	**88.89**
Splice-Donor	MNNs	94.52	95.02	95.02	95.02	94.85	95.02	95.02	95.02	95.02
	Voting	**94.52**	**95.02**	**94.85**	**95.02**	**94.68**	**95.02**	**95.02**	**95.02**	**95.02**

Table 4.7 Comparison of MNNs and Voting Performance on Splice Data (best results in **bold**)

Table 4.6 shows that the prediction rates of MNNs were better or equal to the corresponding performances of majority voting, on a wide variety of data sets. Therefore, it strongly indicated that MNN is a better combination schema than the most frequently used combination method of majority voting.

In addition, table 4.7 shows the prediction rates of MNNs and majority voting on the same splice test data sets. The values also favor the same results: prediction rates of MNNs were better or equal to the corresponding performances of majority voting. Apart from comparing majority voting and MNNs, MNNs performances on splice test data were compared with the other machine learning methods. The data shown in table 4.8 is taken from Hudson, Whitley, Ford and Browne (2003) where exactly same splice test data sets were applied. From table 4.8, it can be observed that a dramatic improvement has been made using MNNs.

Method	Splice-Donor (%)	Splice-Acceptor(%)
C5 ruleset	93.7	86.9
MLP	92.3	83.9
MNNs	**95.02**	**89.78**

Table 4.8 Comparison between MNNs, single NN and C5 machine learning methods on splice junction data sets

In conclusion, the MNNs generalization compared with majority voting or other data mining techniques on a wide variety of data sets using different numbers of ensemble members was presented. The results have demonstrated that MNN ensemble methods have the potential to offer improved performance when compared with majority voting as an ensemble combination method.

4.2.3 Statistical Comparisons of MNNs versus Majority Voting

It can be seen that most of the MNNs generalizations were better than the results obtained by majority voting (in the form of percentage accuracy). However, statistical tests were needed to investigate the differences between MNNs and majority voting's average results. Firstly, the T-test (or student's test) was used to test how significant the improvements were made using MNNs compared with majority voting. In addition, the standard deviation was calculated to monitor the variances of these means.

4.2.3.1 Differences in Means

The results shown in table 4.6 are actually the means of 10-fold cross-validation results using different ensemble members generated by the two ensemble methods. Hence the statistical test of a hypothesis was about the difference between MNNs and majority voting means. The sizes of two samples were all nine (results offered by ensemble groups with ensemble members three, five, seven, nine, eleven, thirteen, fifteen, seventeen and nineteen). The value of the test statistic t (α equals to 0.05 and 0.1 respectively) and corresponding test results for each data set are given in table 4.9.

	Values of t	Testing results (α =0.05, t_α = 1.746)	Testing results (α =0.1, t_α = 1.337)
Breast-cancer –w	2.3463	Reject	Reject
Bupa-liver	1.3701	Accept	Reject
Ionosphere	1.5461	Accept	Reject
Iris *	1.3484	Accept	Reject
Pima- diabetes	7.8351	Reject	Reject

Table 4.9 T test results showing acceptance or rejection of the null-hypothesis for different data sets

An estimation for the generalization difference between MNNs and majority voting is shown in table 4.9. The first column lists the data sets involved in t testing. The second

column gives the computed value of t. The corresponding testing results indicate whether to accept or reject the null hypothesis corresponding to a value of α equals to 0.05 and α equals to 0.1 (which represent confidence values are 95% and 90% respectively) are shown in columns three and four respectively. It was clearly noticed that when α was 0.05, two of the five data sets rejected the null hypothesis. There was sufficient evidence to support the result that MNNs made significant level of improvement compared with majority voting on these two data sets. Furthermore, all of the five data sets rejected the null hypothesis when α was 0.1. So there was strong evidence to suggest a dramatic improvement made by MNNs when the confidence value is 0.1, which means the chance of null hypothesis is rejected when it should be true is 10%. Based on the above t test results, it is claimed that MNNs can make some significant improvement on MNNs results when compared with majority voting with a reasonable error probability.

4.2.3.2 Difference in Variances

As stated that all the data shown in table 4.6 of machine learning data sets are computed on averaging ten-fold cross validation results. Thus, the test results obtained reflect a general and average performances of MNNs and majority voting. Based on these test results for each ensemble groups, the variances of each ensemble groups were the values in each cells shown in table 4.10 multiplied by unit 10^{-4}.

Furthermore, F is computed by using the larger variance divided by the small variance taken from the variances of MNNs and the corresponding majority voting provided in table 4.10, that applied the same number of ensemble members for each data set.

The values of F for each such ensemble group are the figures in each cells presented in table 4.11 multiplied by unit 10^{-4}. It is noticed that almost all the values of F except one in table 4.11 are less than 3.18, which means most of the values didn't fall in the rejection region. Therefore, there is enough evidence to accept the null hypothesis that the two populations were not different, i.e. MNNs and majority voting showed no difference in their performances variances. In addition, one exception (in the pima-diabetes data set

where the number of ensemble members was seven) that rejected the null hypothesis was checked. This was the one where MNNs obtained small variance when compared with majority voting. So, there is the reason to believe that in this case MNNs performance was more stable (showed less variance) than majority voting.

Data		Variances (s) of Ensembles Results for each group($* 10^{-4}$)								
		3	5	7	9	11	13	15	17	19
Breast-cancer -w	MNN	1.5379	2.0185	1.0573	1.0813	2.0185	2.0425	2.5231	2.0425	1.5619
	Voting	**4.0609**	**2.3068**	**1.4658**	**1.5379**	**1.5619**	**1.5619**	**1.5619**	**2.0185**	**2.0185**
Bupa-Llver	MNN	20	22	52	18	54	45	50	48	70
	Voting	**54**	**58**	**63**	**49**	**67**	**54**	**52**	**64**	**64**
Ionosphere	MNN	22	22	18	8.7075	9.8866	9.8866	5.4422	9.0703	9.0703
	Voting	**17**	**9.4331**	**11**	**13**	**13**	**9.0703**	**9.0703**	**11**	**12**
Iris *	MNN	3.7333	3.7333	4.2667	2.8444	3.7333	4.2667	3.7333	3.7333	3.7333
	Votig	**3.7333**	**11**	**4.4444**	**3.7333**	**3.7333**	**3.7333**	**3.7333**	**3.7333**	**3.7333**
Pima-diabetes	MNN	28	14	7.0022	16	13	19	12	24	18
	Voting	**34**	**33**	**40**	**34**	**35**	**39**	**33**	**36**	**31**

Table 4.10 Variances of Ensembles Results for benchmark machine learning data sets

Data	F test results for each group(* 10^{-4}, α = 0.05, F = 3.18)								
	3	5	7	9	11	13	15	17	19
Breast-cancer -w	2.6405	1.1428	1.3864	1.4223	1.2923	1.3077	1.6154	1.0119	1.2923
Bupa-Liver	**2.7000**	**2.6364**	**1.2115**	**2.7222**	**1.2407**	**1.2000**	**1.0400**	**1.3333**	**1.0938**
Ionosphere	1.2941	2.3322	1.6364	1.4930	1.3149	1.0900	1.6667	1.2127	1.3230
Iris *	**1.0000**	**2.9465**	**1.0416**	**1.3125**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Pima- Diabetes	1.2143	2.3571	5.7125	2.1250	2.6923	2.0526	2.7500	1.5000	1.7222

Table 4.11 F test results on machine learning data sets

4.3 Results of the Other Experiments using MNNs

Apart from the experiments combining different number of ensemble members using a one hidden layer MLP, some other experiments using different inputs for the second stage combiner or using a different combiner (as mentioned in section 3.2) were carried out.

4.3.1 Experimental Results Using a Single Layer Neural Network to Combine

As described in part III in section 3.2, a single layer neural network without any hidden neurons was used in the second layer of ensembles as a combiner. The experiments were first tried on the splice data sets only, for the purpose of easy implementation on the data sets whose test data are independent. However the combination results were totally beyond the degree of acceptance, there was no point to apply such model on more data sets further.

The learning rate was changed from 0.1 to 5 in the increment of 0.1 each training cycle. The experiments were repeated by combining three, five, seven, nine ensemble members each time initially where details of ensemble members for each group were listed in table 4.4 respectively. The results on the test data sets are illustrated in table 4.12. As the initial trial already showed single layer neural networks were not capable of being a combiner in such occasion, therefore no more ensemble members were applied in such experiments.

Data set	Number of Ensemble Members being Combined			
	3	5	7	9
Splice-Acceptor	52.67	52.67	52.67	52.67
Splice-Donor	61.96	61.96	61.96	61.96

Table 4.12 Ensemble results by using neural networks with a single layer in combination

It is interesting to see all the ensemble results for the same data set were the same, no matter how many ensemble members were being combined (as table 4.12 shows). These results suggested that the single layer neural network didn't perform well in the combination stage. Further analysis on the generated test data outputs showed that all the outputs generated by the single layer neural networks combiner were all classified as 1, and that explained why all the test prediction percentages were the same, even though the number of ensemble members being combined was different. In short, this kind of neural network was not powerful enough to perform the combination function. This is due to the single layer neural network's weak computation capability.

4.3.2 Experimental Results on Combining Ensemble Members Results along with Original Inputs

In part b of section 3.2, there was a discussion on the experiments on combining ensemble members along with the original inputs. These experiments were done on the splice data again for the purpose of easy application because of its independent test data sets. Unlike the other experiments described before, here the inputs for the second stage NNs were the ensemble members' results along with the original inputs that generated these ensemble results. With the complexity of inputs increased, the cost of training increased as well. Therefore only three ensemble members were combined initially and the number of ensemble members being combined was increased gradually. However, once found that the performances could not achieve the generalization of MNNs using just the outputs of ensemble members, such experiments were not carried out further.

In the second stage, there was a fully connected MLP with two hidden neurons (more hidden neurons were tried, e.g., 10, 20, 30 and 50, but all failed to deliver better results), trained using the backpropagation algorithm. The learning rate was set to 0.1 and the training cycle was repeated 1000 times (with different initialization). The ensemble members selected for each group are shown in table 4.4. and the results on the test data sets are presented in table 4.13.

Data	Number of Ensemble Members being Combined							
	3		5		7		9	
	Original Input	**Just Output**	**Original Input**	**Just Output**	**Original Input**	**Just Output**	**Original Input**	**Just Output**
Splice-Acceptor	87.33	89.33	88.67	89.33	88.67	89.33	88.67	89.78
Splice-Donor	93.69	94.52	94.52	95.02	95.02	95.02	94.68	95.02

Table 4.13 Comparison between combining ensemble members' results along with original source inputs and just combining ensemble members outputs

From the results shown in table 4.13, no improvements were made by adding more information in the inputs to the second stage NNs' training. Therefore, just the ensemble members results was used as the second stage inputs in the experiments.

4.4 Comparison Results Regarding the Performance of MNNs

In section 4.2 and section 4.3, the results generated by MNNs method were reported. Improvements have been observed compared to majority voting. In this section, further investigation on the potential of MNNs are conducted, and more comparisons are presented in order to explore the optimum MNN structures.

4.4.1 Comparison of Performance Using Different Numbers of Ensemble Members

The intention of using different number of ensemble members being combined was to find whether the number of ensemble numbers will effect the performance MNNs. Especially it would be ideally if one can find the optimal ensemble members. To give a clearer picture of MNNs performances, several figures were drawn according to the information supplied in tables 4.6 and 4.7.

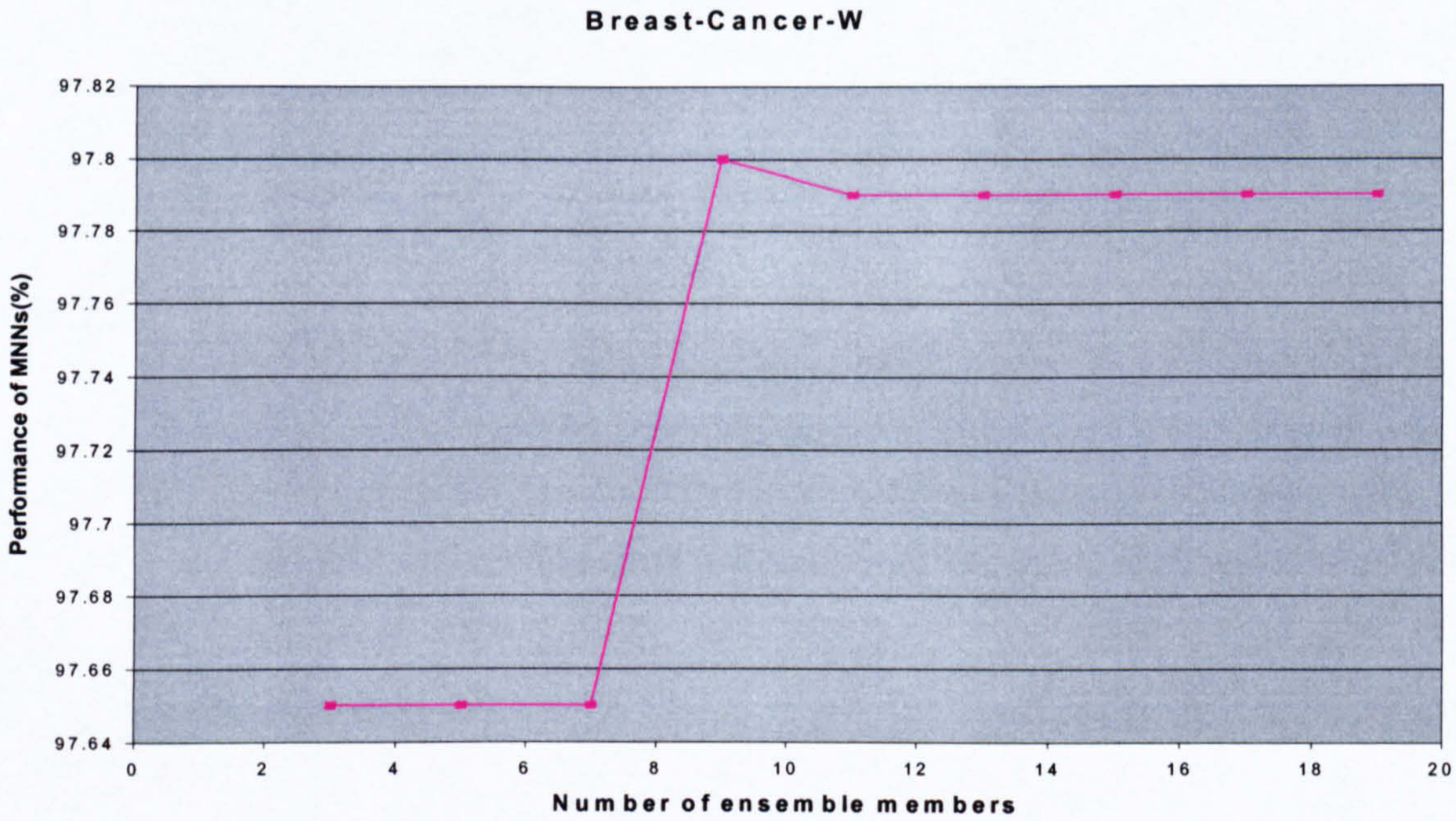

Figure 4.1 Performance of MNNs with increasing numbers of members (1)

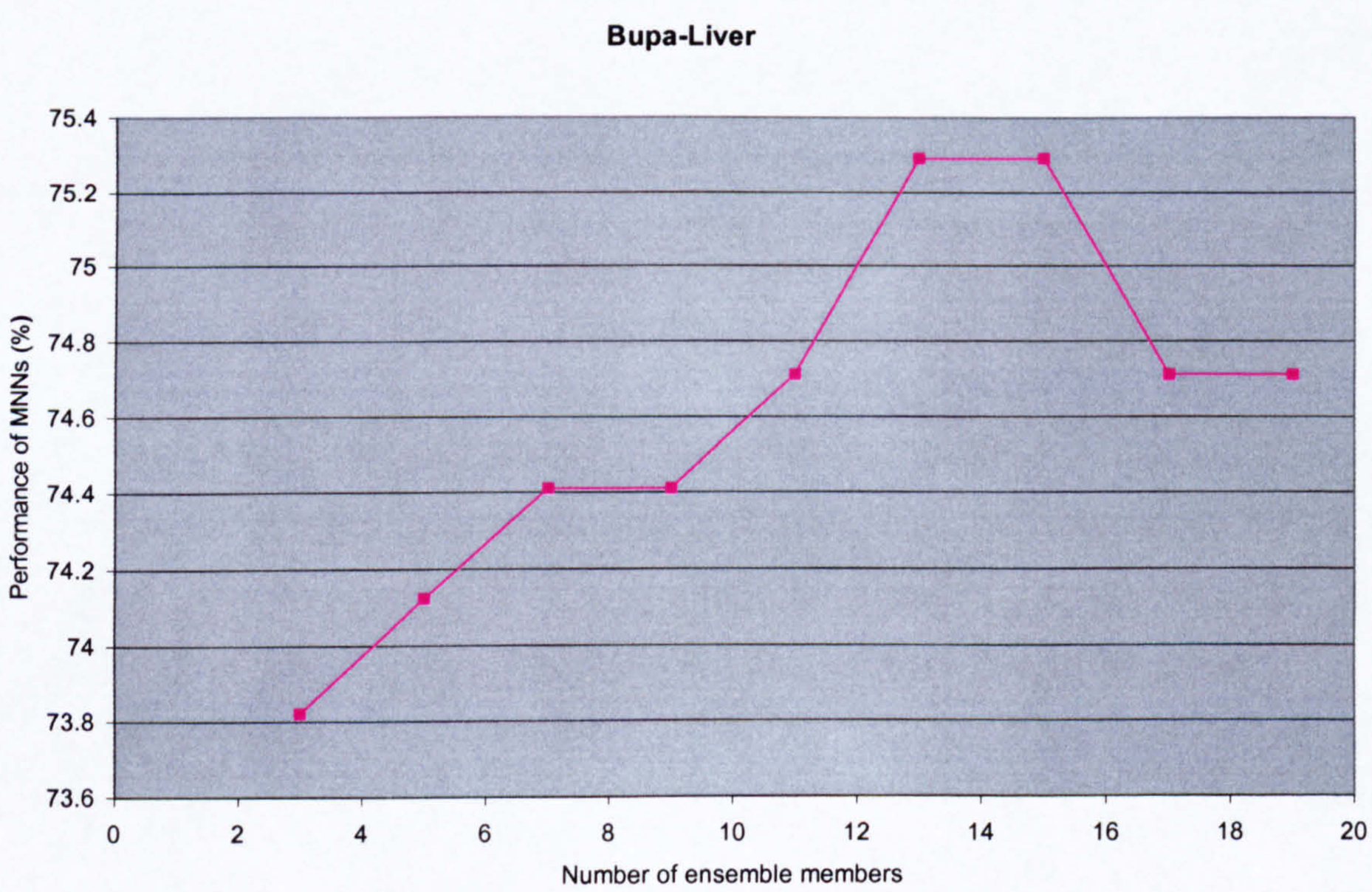

Figure 4.1 Performance of MNNs with increasing numbers of members (2)

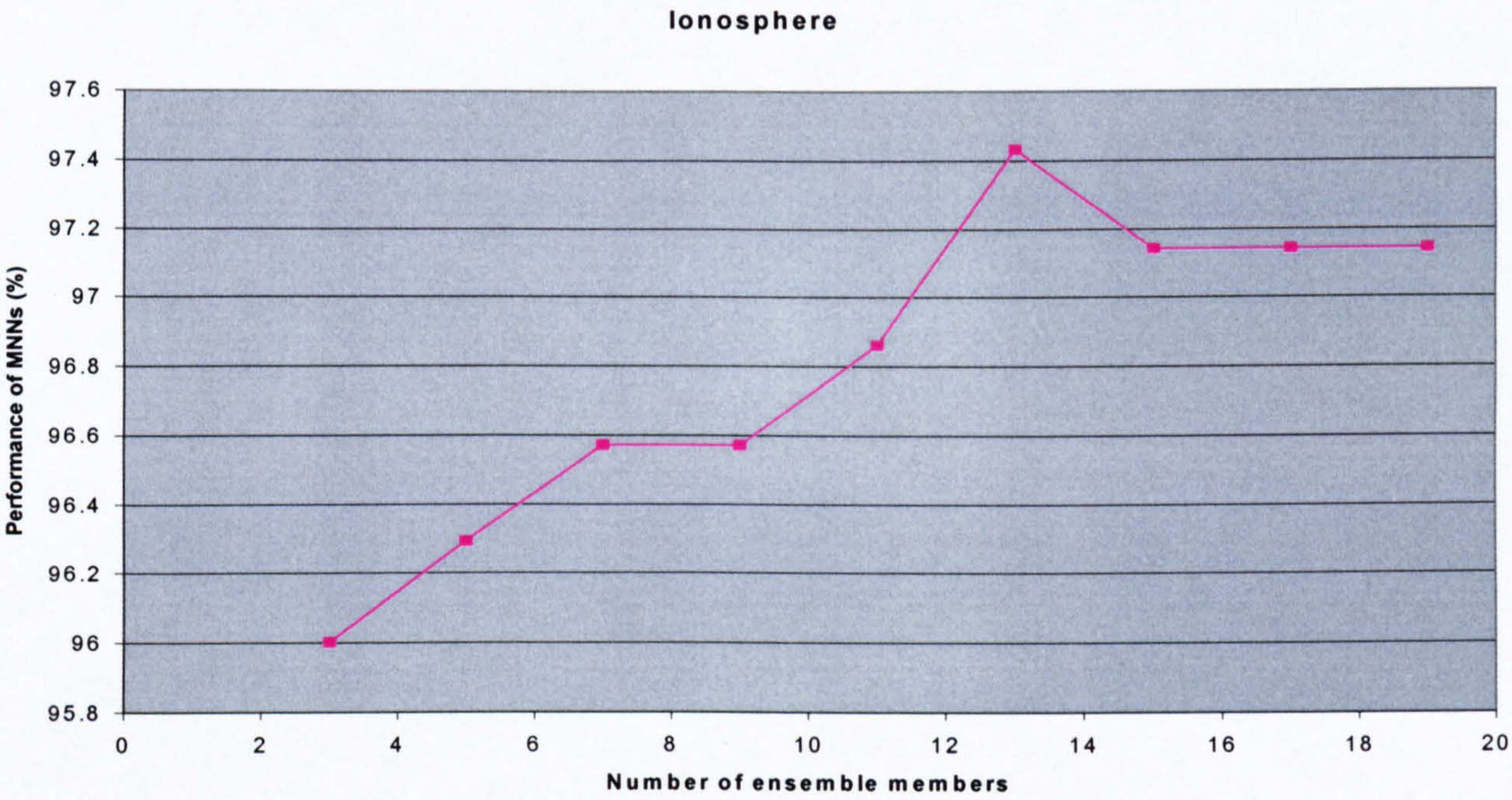

Figure 4.1 Performance of MNNs with increasing numbers of members (3)

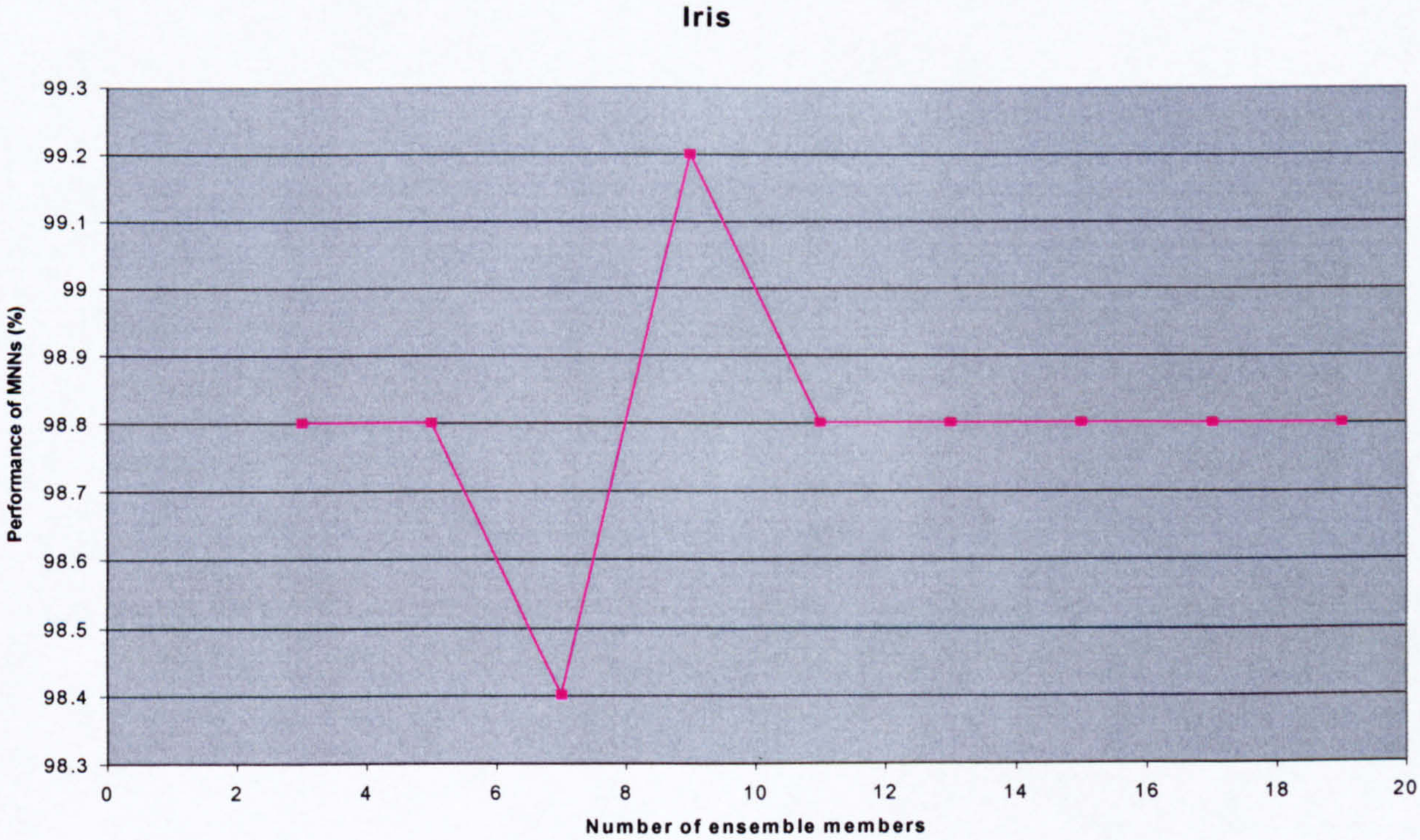

Figure 4.1 Performance of MNNs with increasing numbers of members (4)

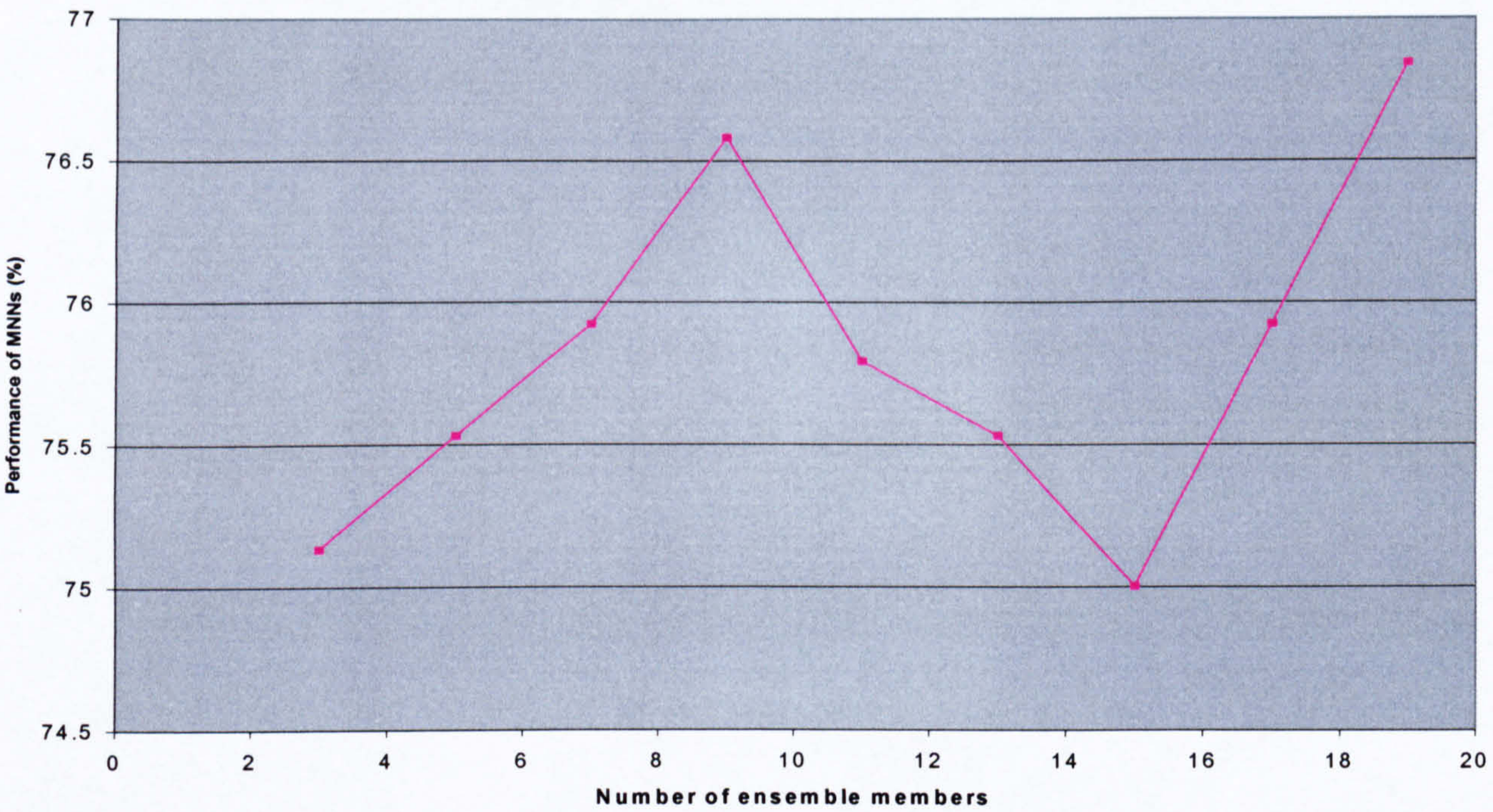

Figure 4.1 Performance of MNNs with increasing numbers of members (5)

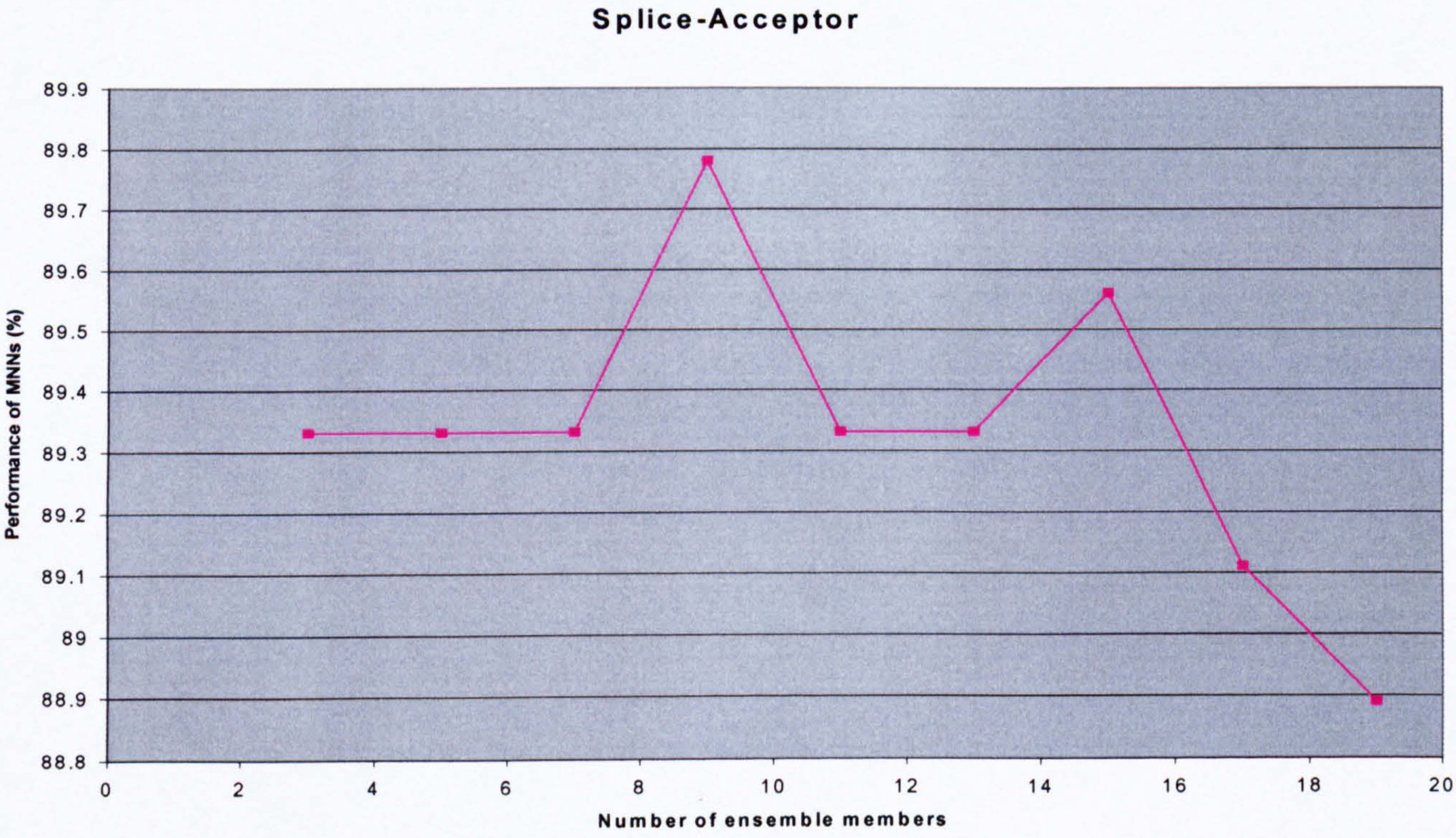

Figure 4.1 Performance of MNNs with increasing numbers of members (6)

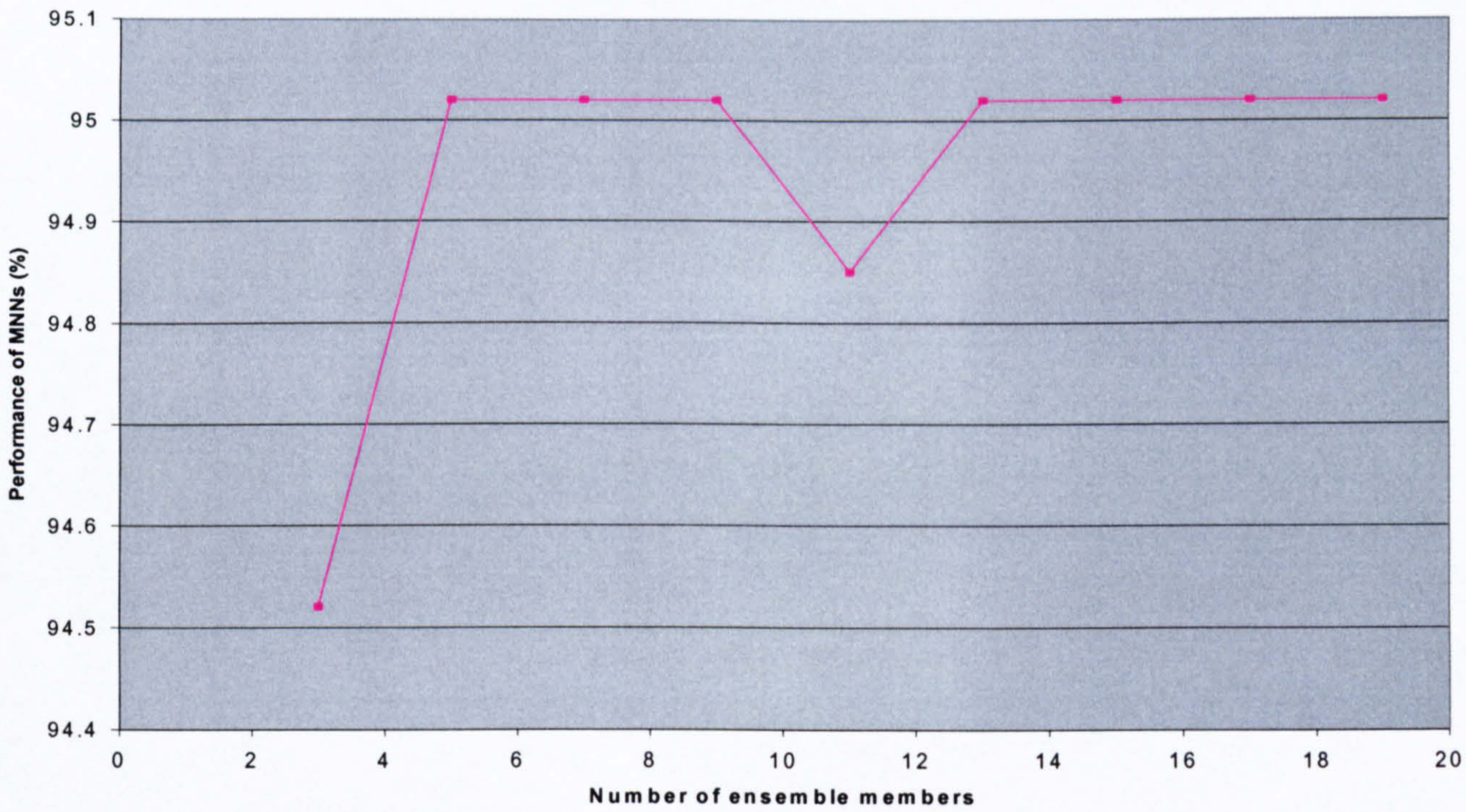

Figure 4.1 Performance of MNNs with increasing numbers of members (7)

Figures 4.1 (1)-(7) were drawn to reflect the trend of MNNs performance. The above diagrams have something in common, in that the performance lines start from low points and climbed gradually to one high point, then slip down afterwards. Therefore, there is a general trend that can be drawn, i.e. that for any particular case there is a peak performance that can be obtained by combining a certain number of ensemble members using the MNNs method. Though the number of ensemble members were not the same where the performance of the MNNs reached their peak, from the fig. 4.1(1)-(7) it is shown that there must be an optimum number of ensemble members existing which can give the best MNNs performance. Also notice that after the peak point, not all MNNs performance went down or remained unchanged. In figure 4.1 (5), the line picked up again, it indicated that MNNs performance reached another higher point. Therefore the

first peak point does not necessarily mean where the best MNNs performance is, instead there may be more peak point afterwards. So how to interpret this phenomenon?

As the performances of ensembles lie in their ensemble members' diversity and accuracy as addressed in section 3.5.2.2, the optimum ensemble number of a MNNs may be related to the diversity among the ensemble members and their individual prediction rates. There are two sides on this issue. First, selecting different ensemble members will result in different ensemble generalizations when keeping the number of ensemble members constant. So the optimum number of ensemble members depends on the quality of the ensemble members and can be found within the certain number of ensemble candidates. The diagrams in Figure 4.1 all show the peak points, which gave us the full evidence. On the other hand, as the size of ensembles can be infinite, there may be always possible that optimum number of ensembles can never be found as long as keep increasing the number of ensemble candidates. Considering the interaction of diversity and accuracy, it is understood that these two factors result in the performance of MNNs going up and down. There is further discussion regarding this issue in chapter five.

4.4.2 Comparison of Performance When Selecting Different Ensemble Members

Diversity is the one factor that affects the performances of ensembles. Therefore, several methods were used for generating ensemble members in the experiments. It was found that after using these, the ensemble members were all different in the experiments. However, there must still be different diversity existing among the different ensemble groups created by combining different ensemble members whilst keeping the number of ensemble members unchanged. The purpose of this experiment was to find out how much diversity among ensemble members affects the ensemble results (even after all the ensemble members were generated in the way of pursuing as much diversity as possible among them).

Five ensemble members (the first five ensemble members and their prediction performances of ten -fold cross validation are presented in table 4.14 and table 4.15) were selected from those trained on the breast-cancer-w and pima-diabetes data. The correlation coefficients (denoted by ρ) of each pair of ensemble members were calculated. The quantity ρ can be computed by using formula below.

$$\rho_{xy} = \frac{s_{xy}}{\sqrt{s_{xx} s_{yy}}} \qquad (4.1)$$

Where $s_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$ and $s_{xx} = \sum x^2 - \frac{(\sum x)^2}{n}$

Where n is the number of samples and x, y represents the each ten-fold cross validation results of the corresponding pair's ensemble group.

Input data part	Ensemble candidates performance on each part of input data (%)				
	1	2	3	4	5
1	91.18	95.59	91.18	91.18	88.24
2	98.53	98.53	98.53	95.59	98.53
3	98.53	97.06	97.06	95.59	98.53
4	94.12	94.12	94.12	95.59	94.12
5	94.12	97.06	95.59	97.06	94.12
6	1.00	98.53	1.00	1.00	98.53
7	97.06	97.06	98.53	98.53	97.06
8	1.00	98.53	97.06	97.06	98.53
9	1.00	98.53	1.00	95.59	1.00
10	98.5	1.00	1.00	1.00	1.00
Average	97.21	97.51	97.21	96.62	96.766

Table 4.14 Performance of each of the five ensemble members on the breast-cancer-w data set used in the diversity experiments

Input data part	Ensemble candidates performance on each part of input data (%)				
	1	2	3	4	5
1	64.47	67.11	68.42	65.79	60.53
2	80.26	80.26	77.63	81.58	77.63
3	67.11	68.42	72.37	69.74	67.11
4	67.11	72.37	63.16	72.37	68.42
5	71.05	72.37	69.74	69.74	69.74
6	72.37	72.37	76.32	72.37	72.37
7	78.95	82.89	81.58	84.21	80.26
8	77.63	80.26	73.68	82.89	78.95
9	75.00	75.00	73.68	72.37	71.05
10	76.32	69.74	68.42	71.05	67.11
Average	73.03	74.08	72.50	74.21	71.32

Table 4.15 Performance of each of the five ensemble members on the pima-diabetes data set used in the diversity experiments.

Ensemble members no.	1	2	3	4	5
1	1	0.9554	0.9614	0.9230	0.9839
2		1	0.9490	0.9426	0.9523
3			1	0.9487	0.9487
4				1	0.9262
5					1

Table 4.16 Ensemble members Correlation coefficients for breast-cancer-w data set

Ensemble members no.	1	2	3	4	5
1	1	0.8744	0.7007	0.7986	0.8092
2		1	0.6622	0.8550	0.7982
3			1	0.7207	0.7207
4				1	0.8098
5					1

Table 4.17 Ensemble members Correlation coefficients for pima-diabetes data set

Tables 4.16 and 4.17 show the correlation coefficient between each pair of ensemble members for two data sets (half of each table is shaded because of duplicated values). The strength of the linear relationship between each two members can be determined by the value of the correlation. It was noticed that the values of the correlation coefficient in table 4.16 were quite high, due to the high accuracy of the prediction rates for the breast-cancer-w data set. In contrast, lower values of correlation coefficient were observed alongside the relatively lower prediction rates for the pima-diabetes data set.

According to the values above, several groups of ensembles were organized. These were labeled as highly correlated, averagely correlated and more diverse (which reflected the different features of diversity existing within each group). Details are shown in tables 4.18 and 4.19, where the means of correlation coefficients were computed by averaging the sum of correlation coefficients between any two members within each group.

	Highly Correlated	Averagely Correlated	Averagely Correlated	More Diverse
Group Members	1,3,5	1,2,5	1,4,5	3,4,5
Mean of correlation	0.9647	0.9639	0.9444	0.9412
Majority Voting (%)	96.91	97.21	97.06	97.35
MNN (%)	97.06	97.35	97.35	97.50

Table 4.18 The performances of different diverse groups of ensemble members for the breast-cancer-w data set

	Highly Correlated	Averagely Correlated	Averagely Correlated	More Diverse
Group Members	1, 2, 4	3, 4, 5	2, 3, 4	2, 3, 5
Mean of correlation	0.8427	0.7504	0.7460	0.7270
Majority Voting (%)	74.34	72.89	74.47	73.55
MNN (%)	74.34	75.39	75.13	75.53

Table 4.19 The performances of different diverse groups of ensemble members for the pima-diabetes data set

The results shown in the above two tables are all the average performances (based on the ten-fold cross validation method). From the performances shown in tables 4.18 and 4.19, it can be seen that diversity did affect the ensemble results. If comparing the results between highly correlated groups with those from more diverse groups, a larger difference can be observed. However, it is also interesting to see that not much gain can be observed when comparing averagely correlated groups with more diverse groups. As all the individual ensemble members were created by various ensemble methods as mentioned in section 3.5.2 in order to get the maximum diversity among them as possible. Therefore, there was not much space left (in terms of diversity) after combining these ensemble members. Also, it was noticed that the means of correlation coefficients between the averagely correlated groups and more diverse groups were not significant, so

it is easy to understand why little improvement can be made by using more diverse ensemble groups.

4.5 Conclusion

In this chapter, the ensemble results generated by MNNs were presented, along with the corresponding results generated by majority voting using ensemble groups consisting of the same ensemble members (on a wide variety of applications). Statistical analyses of these results were conducted to investigate the effectiveness of MNNs. Based on the above results, it can be concluded that the significant improvement can be made by MNNs in terms of ensemble generalization compared with the most frequently used combination method, majority voting. In addition, the explorative results with other experiments using different structures of MNNs presented in section 4.3 suggested that a single layer NN is not power enough to perform the combination function in MNNs. Also the use of original inputs along with the outputs of ensemble members as the inputs of second stage MNNs did not show any improvement compared with just using the MNNs' first stage outputs. Furthermore, some comparisons for the purpose of exploring the optimum MNNs structures showed that the diversity and accuracy of ensemble members play the key role in affecting the ensemble performance.

From information provided in the above results presented in this chapter, further analysis are performed in chapter five.

Chapter Five

Analysis and Discussions

5.1 Chapter Introduction

The model of MNNs was presented in chapter two and the experimental details of using MNNs were described in chapter three. Chapter four demonstrated the experimental results on a wide variety of applications, including machine learning benchmarks and human gene data sets. Using MNNs on some of the data sets showed that a significant improvement in ensemble generalization could be obtained using MNNs compared with majority voting. Thus, the explanation of some of the MNNs results in chapter four was delivered. However, it is necessary to look further into MNNs to gain a better understanding. In section 5.2, a detailed analysis of using MNNs using the pima-diabetes data set is conducted, on which the most significant improvement has been made. Next, a theoretical explanation of the observed trend of MNNs performances with increasing ensemble size is given in section 5.3. Furthermore, some analysis is performed on the other experimental results of chapter four in section 5.4.

5.2 Analysis of MNNs Performance

When comparing the second stage of MNNs (combiner) with a single MLP, it was found that the most influential factor in the performance of the MNNs second stage with a single MLP is the origin of the source data for training. For a conventional MLP, the training data are usually from the real world or are created using mathematical tools. The training data for the second stage of MNNs are the output prediction data after being approximated by the first stage MLPs making up the ensemble. Consequently, the second stage of MNNs acts as a classifier, separating all input patterns (i.e. ensemble outputs) into two categories (for a two-classification problem).

Figure 5.1 shows the m-2-1 signal flow graph of a single MLP, which was employed in the second stage combiner of MNNs in the experiments. The m outputs on the left represent the outputs from the corresponding ensemble members. Therefore, for the second stage of this model all the inputs are in m-dimensional form, regardless of what kinds of original input features are presented to the first stage of MNNs. The standard MLP used in the second stage of MNNs in the experiments has two hidden neurons (denoted by neuron *1* and *2* respectively) and one output neuron as shown in the figure. The synaptic weights of a neuron are denoted by w with the corresponding index. The activation function is represented by $\varphi(.)$ and the hyperbolic tangent function is used by the hidden neurons. The output neuron uses the linear combination function. The corresponding bias for each neuron is labeled as b with the associated index. For convenience of analysis, the simplest MNN model using three ensemble members is used in the analysis (MNNs using five or more ensemble members all operate in the same way). Thus, the three dimensional inputs will be transformed into two dimensional patterns on the hidden units, and then transformed into a single dimensional output pattern through the output unit.

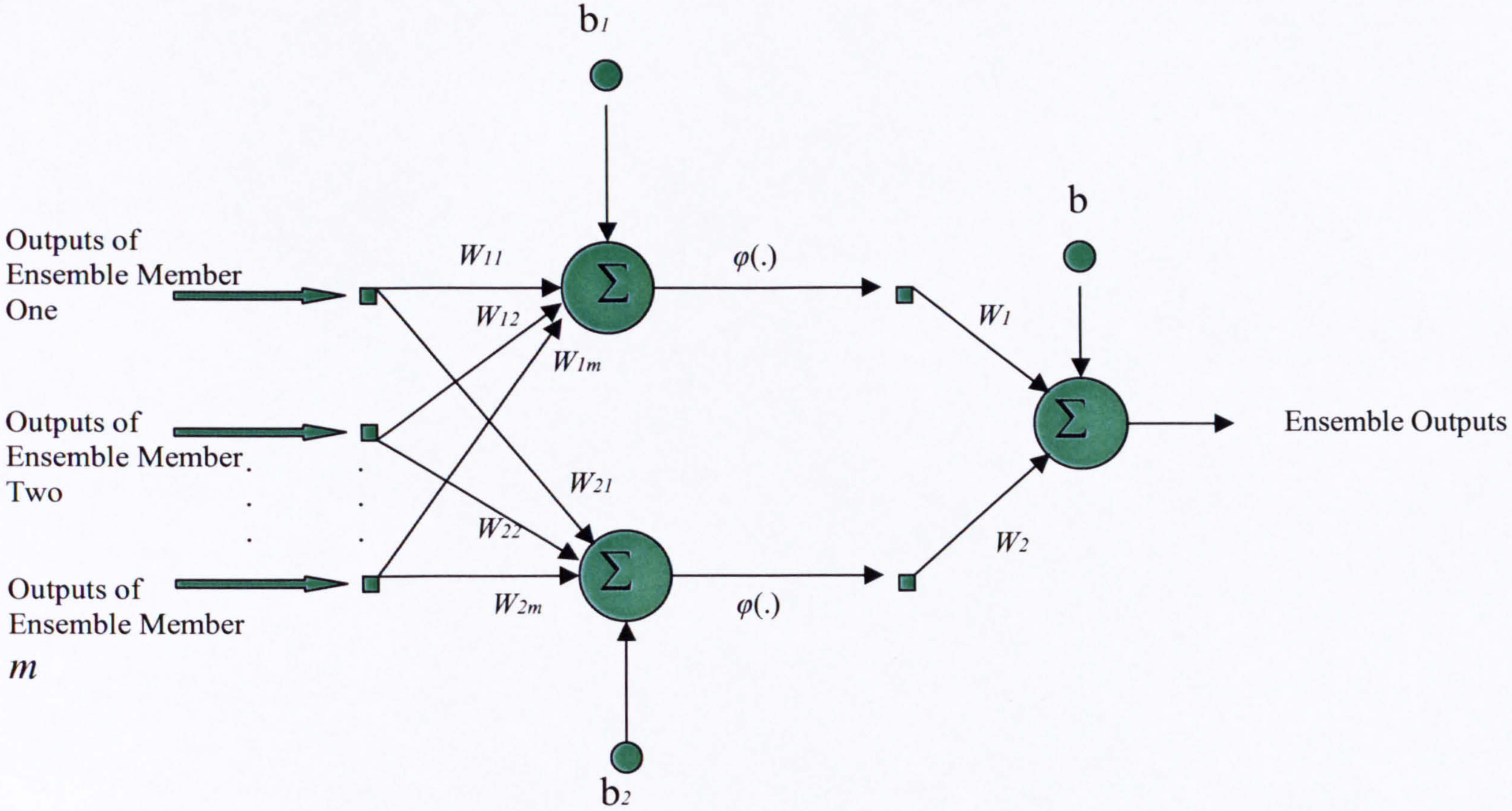

Figure 5.1 The signal flow of a second staged MNN

5.2.1 Introduction to Data Used in the Analysis

As ten-fold cross-validation on the ensemble models of the pima-diabetes dataset was applied, there are ten models of MNNs. Due to the limitation of the size of this thesis, only the MNN model trained on the first part of the cross-validation data was used in the analysis. The outputs of three ensemble members (the second, third and fourth) used in the experiments, whose ensemble performance is shown in table 4.5, were used as the illustrative inputs for the second stage of MNNs. Their outputs, together with their associated target values are shown in Figure 5.2.

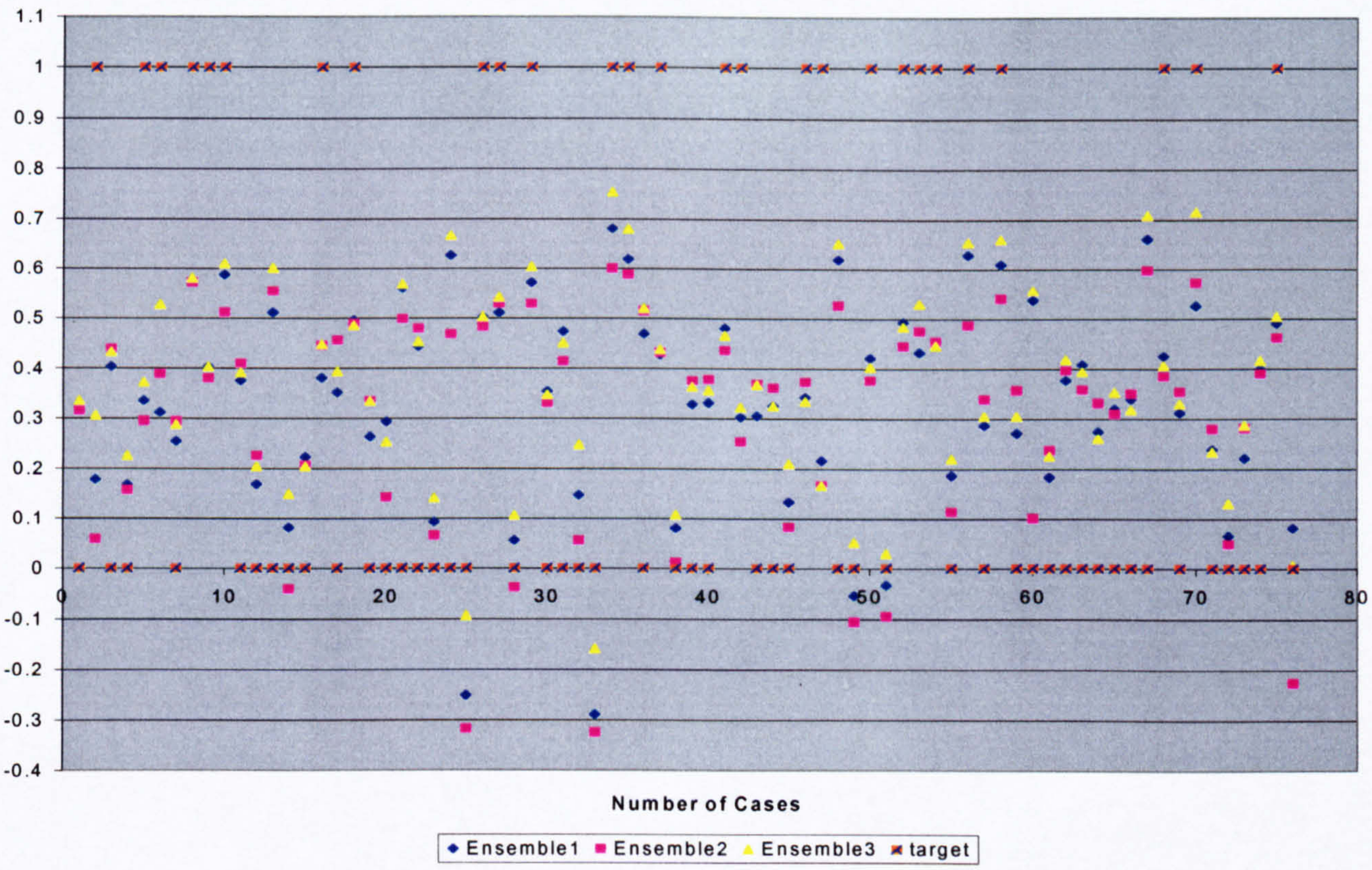

Figure 5.2 Actual outputs of test data by three ensemble members in the first ten-fold cross validation experiments, together with their targets

As shown in Figure 5.2, the outputs of ensemble members were mainly spread in the range between -0.3 and 0.8. Since three ensemble members were employed in the experiments, this is a three dimensional space problem if let each input represent a coordinate system for that space and the set of coordinates gives the position of the feature in that space (Picton, 2000). Therefore, there exists an approximation function within three-dimensional space to separate the ensemble members' input features (Callan, 1999). Figure 5.3 shows these three ensembles outputs in the three dimensions on the whole input data.

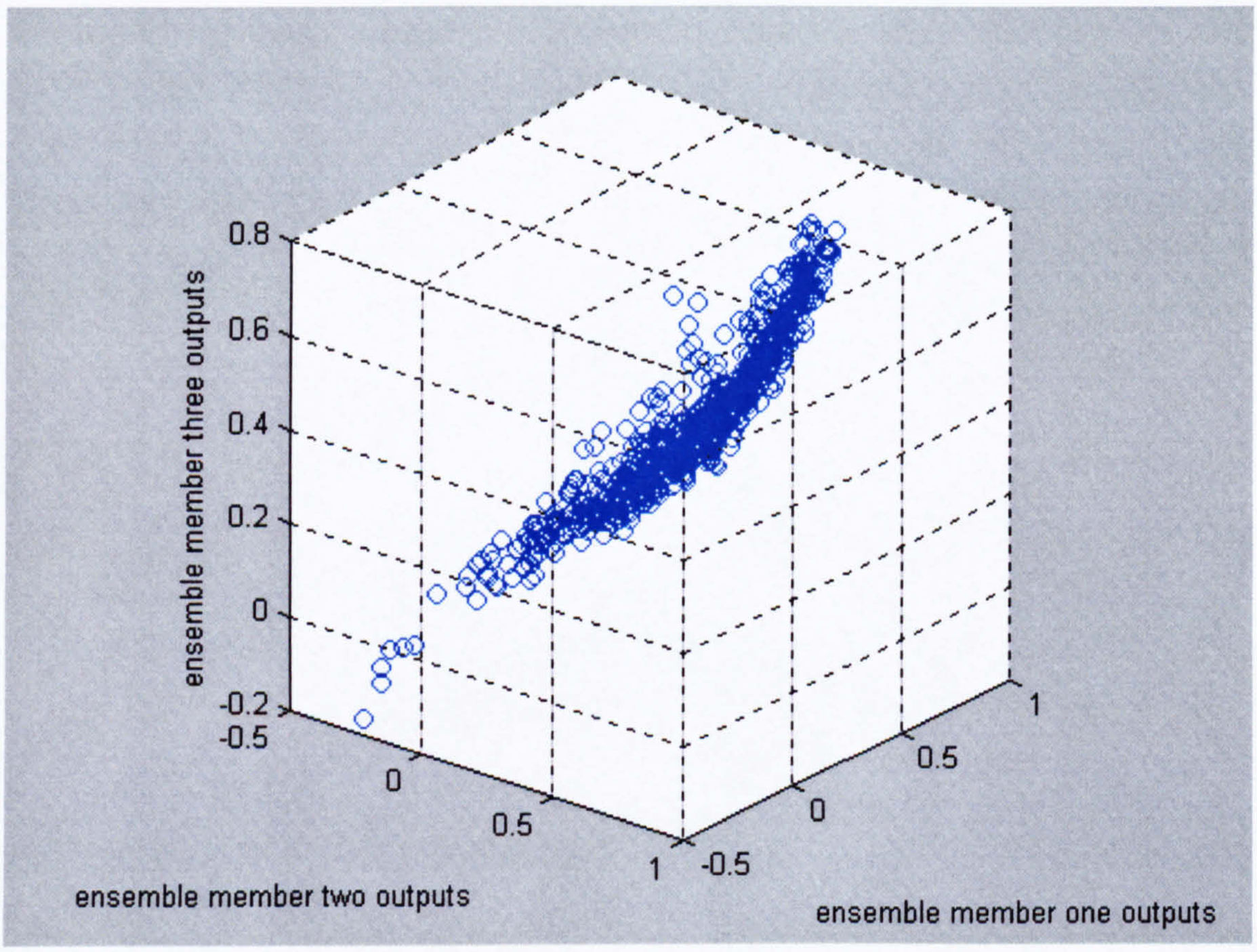

Figure 5.3 The 3-D graph of three ensemble members outputs on the whole input data of Pima-Diabetes

By using a MLP in the second stage of MNNs, the network transforms the three-dimensional input data or multiple dimensional inputs (when more ensemble members are used) into a two-dimensional space on its two hidden neurons. Then it is possible to draw a line to separate the input patterns.

All four cases among the 76 test data on which MNNs made the correct prediction whilst on the contrary majority voting method made the wrong predictions were selected to demonstrate how the two ensemble methods (majority voting and MNNs) work on them. Figure 5.4 gives the three ensemble members' outputs with the data shown in the table 5.1.

Three Ensemble Members Outputs			MNNs Results	Final Majority Voting Results	Targets
0.4957	0.4888	0.4851	1	0	1
0.4931	0.4450	0.4835	1	0	1
0.5372	0.1004	0.5565	0	1	0
0.4903	0.4620	0.5031	1	0	1

Table 5.1 Ensemble inputs and combiner outputs for MNNs and Majority Voting

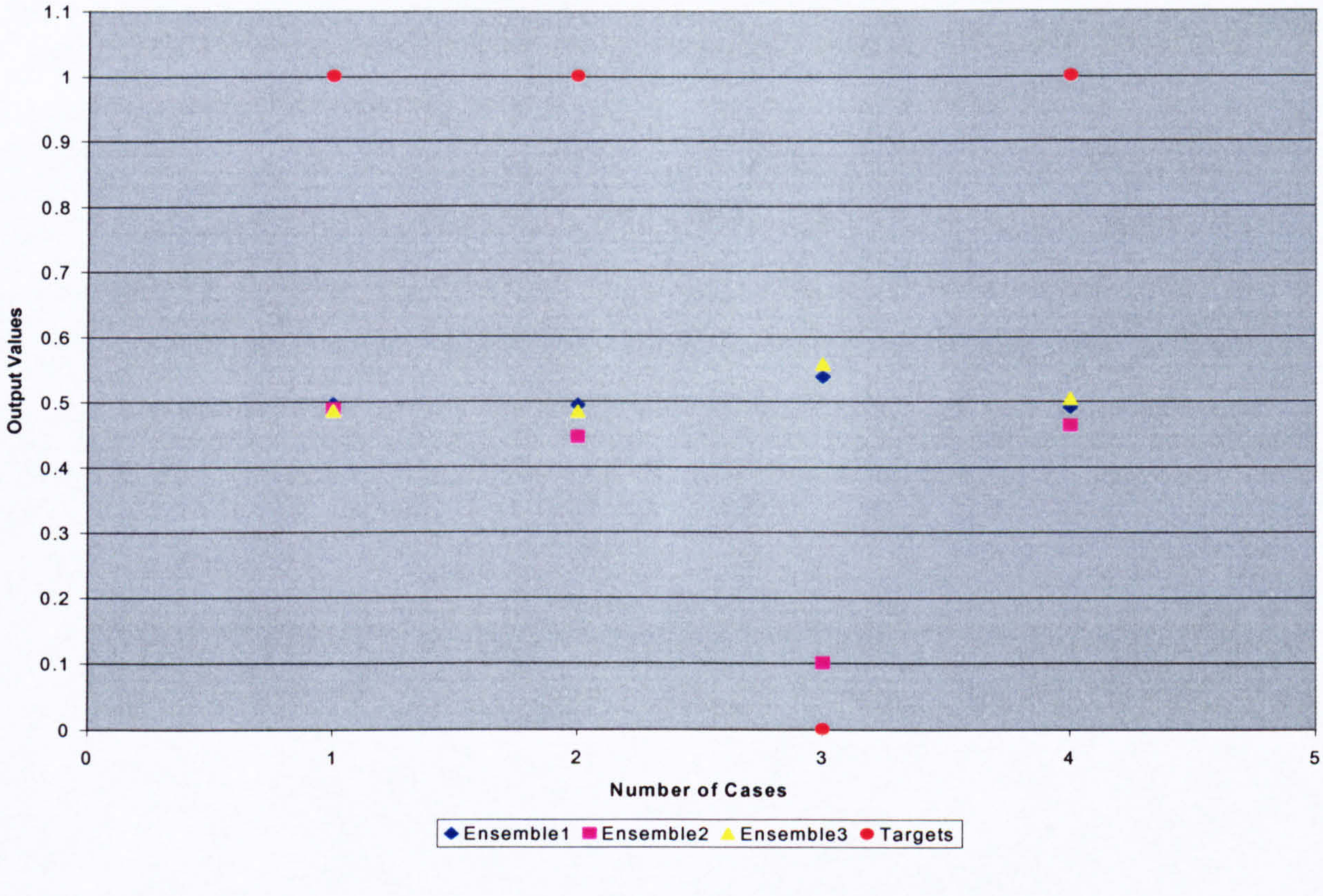

Figure 5.4 Ensemble members outputs for the four cases on which MNNs made the correct predictions whilst majority voting made the wrong predictions

5.2.2 Error Analysis on Majority Voting's Performance

Table 5.2 shows how majority voting made the decision on the three ensemble members' outputs, which led to the wrong results whilst MNNs made the right predictions.

Three Ensemble Members Outputs			Numbers of Agreed on		Final Majority Voting Results	Targets
			(1)	(0)		
0.4957	0.4888	0.4851	0	3	0	1
0.4931	0.4450	0.4835	0	3	0	1
0.5372	0.1004	0.5565	2	1	1	0
0.4903	0.4620	0.5031	1	2	0	1

Table 5.2 Ensemble inputs and combiner outputs for majority voting

All the inputs less than 0.5 were classified as 0 and all the inputs above or equal to 0.5 were classified as 1.

Table 5.3 lists those ensemble members' outputs where majority voting failed to make the right decision on the first part of cross validation test data, where the last four rows were the examples used in this section and in section 5.2.3. Figure 5.4 draw the range of ensemble members' output based on the table 5.3.

The majority voting combination method has two flaws. One, obviously when most or all of the ensemble members make the wrong predictions, majority voting's results must be wrong as well. The cases shown in shading in table 5.3 are such examples. In addition, most wrong predictions were made by majority voting when the outputs of ensemble members fall in the region around the decision boundary (0.5). For instance, it appears that most of the ensemble members' outputs fell in the range between 0.3 and 0.7 in figure 5.5. In contrast, MNNs has the ability to classify some of these data by adjusting second stage MLP's weights.

Case Number	Ensemble Members Outputs			Majority Voting Results	Targets
	One	Two	Three		
1	0.177666	0.058015	0.305598	0	1
2	0.334568	0.294824	0.37087	0	1
3	0.311472	0.38842	0.526464	0	1
4	0.403095	0.380284	0.402525	0	1
5	0.511222	0.554198	0.600046	1	0
6	0.37995	0.445687	0.447429	0	1
7	0.559735	0.498712	0.567454	1	0
8	0.624964	0.467565	0.664626	1	0
9	0.468205	0.514794	0.519233	1	0
10	0.426681	0.431814	0.437934	0	1
11	0.479817	0.435977	0.465674	0	1
12	0.301383	0.253323	0.321276	0	1
13	0.341823	0.373011	0.33353	0	1
14	0.215304	0.16669	0.16581	0	1
15	0.617164	0.526273	0.649589	1	0
16	0.420793	0.376663	0.40254	0	1
17	0.432602	0.475534	0.528901	0	1
18	0.447123	0.454176	0.445611	0	1
19	0.659796	0.597025	0.70814	1	0
20	0.424679	0.38451	0.405593	0	1
21	0.495685	0.488836	0.485098	0	1
22	0.493125	0.445048	0.483544	0	1
23	0.537216	0.10042	0.556505	1	0
24	0.490276	0.461954	0.503087	0	1

Table 5.3 Ensemble members outputs and the corresponding wrong predictions made by majority voting, where shaded examples are all three ensemble members made the wrong predictions and majority voting's results were wrong as well

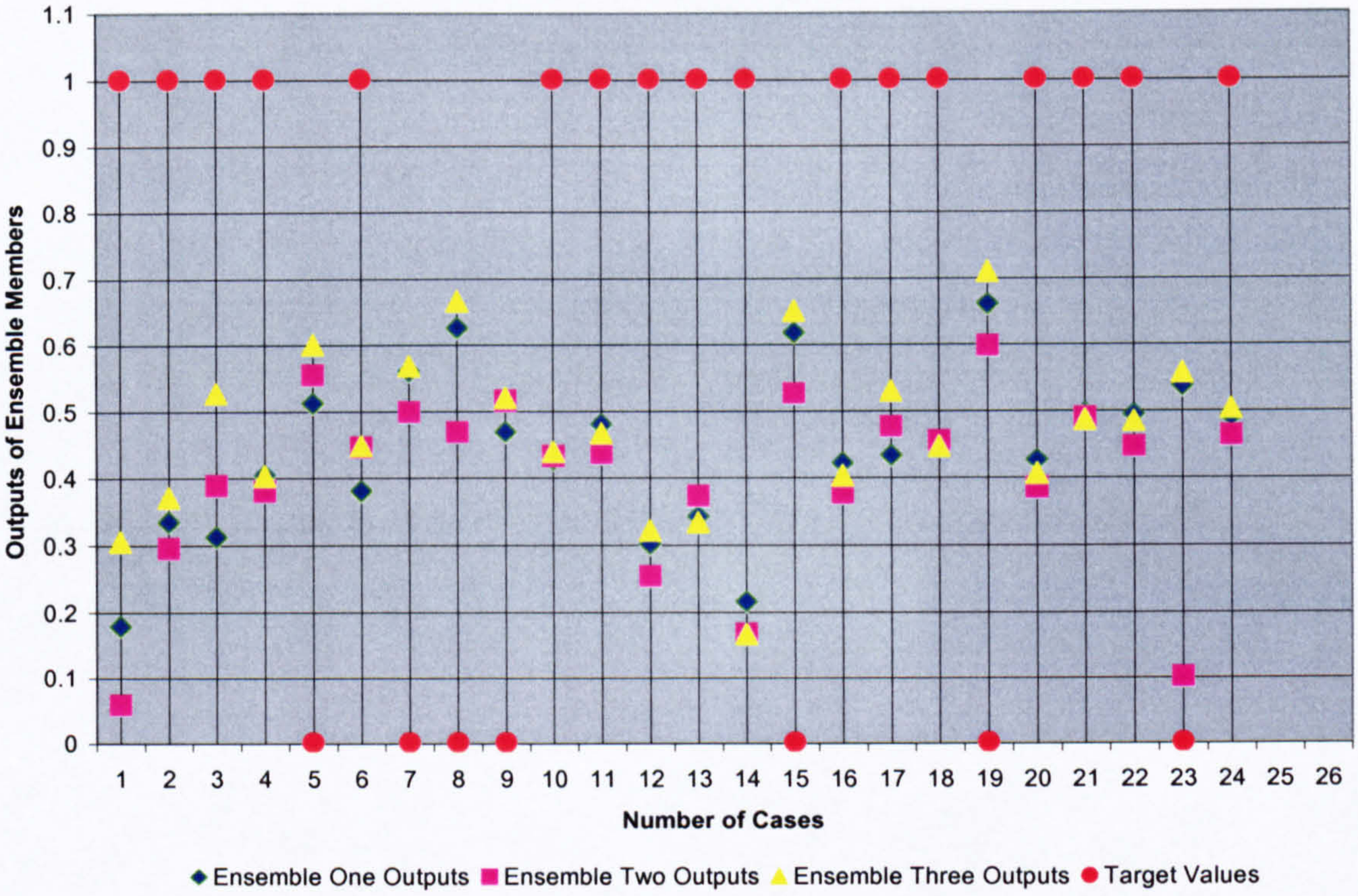

Figure 5.5 Ensemble members outputs with the corresponding target values

5.2.3 Analysis of the Performance of MNNs

Firstly, the parameters of the MLP used as the second-stage combiner were checked. The weights and biases for the hidden neurons and the output neurons are shown in table 5.4.

Hidden layer weights and bias		Output layer weights and bias	
B	W	B	W
[1.8319, 0.4704]	$\begin{bmatrix} -1.0521 & 0.4963 \\ -1.2147 & -0.7780 \\ -0.5404 & -0.3172 \end{bmatrix}$	[0.8952]	$\begin{bmatrix} -1.3444 \\ 1.1345 \end{bmatrix}$

Table 5.4 The parameters of the MLP combiner

In the experiments, the results for the neuron inputs $neuron_i$ and $neuron_j$ of the hidden units i and j used the TANH activation function, and the final ensemble results were summed in the neuron input $neuron_k$ of output unit k used and put through a linear activation function:

For the calculation of the activations of these two hidden neurons i and j

$$[\text{HiddenNeuronOutput}_i \;\; \text{HiddenNeuronOutput}_j] = \tanh(z) \qquad \text{where}$$

$$z = [\text{HiddenNeuronInput}][\text{FirstLayerWeight}] + [\text{FirstLayerBias}]$$

$$= \begin{bmatrix} 0.49568483 & 0.48883647 & 0.48509806 \\ 0.49312517 & 0.44504833 & 0.48354448 \\ 0.53721561 & 0.10042012 & 0.55650527 \\ 0.49027603 & 0.46195383 & 0.50308656 \end{bmatrix} \begin{bmatrix} -1.0521 & 0.4963 \\ -1.2147 & -0.7780 \\ -0.5404 & -0.3172 \end{bmatrix} + [1.8319 \; 0.4704]$$

(5.1)

For the calculation of $Neuron_k$:

$$[\text{NeuronOutPut}_k] = [\text{HiddenNeuronOutput}_i \; \text{HiddenNeuronOutput}_j][\text{SecondLayerWeight}]$$

$$+ [\text{SecondLayerBias}]$$

$$= [\text{HiddenNeuronOutput}_i \ \ \text{HiddenNeuronOutput}_j] \begin{bmatrix} -1.3444 \\ 1.1345 \end{bmatrix} + [0.8952] \tag{5.2}$$

Table 5.5 gave the data values for the four examples where MNNs made the correct prediction whilst majority voting made the wrong decision.

Input			HiddenNeuronOutput $_i$	HiddenNeuronOutput $_j$	NeuronOutPut $_k$
0.4957	0.4888	0.4851	0.4256	0.1802	0.5275
0.4931	0.4450	0.4835	0.4709	0.2122	0.5029
0.5372	0.1004	0.5565	0.6879	0.4481	0.4788
0.4903	0.4620	0.5031	0.4487	0.1923	0.5101

Table 5.5 Ensemble outputs, hidden layer activation values and output activation for the four examples where MNN made the correct prediction whilst the majority voting combiner made the wrong decision

The output of the MLP actually modeled a line since it has only two inputs (Callan, 1999).

As the weights and bias for the output neuron k was already known in table 5.4, the following equations were given to calculate the weighted sum for the input of Neuron *k*:

$$\text{OutPutNeuron}_k = \text{HiddenNeuronOutput}_i (1.3444) + \text{HiddenNeuronOutput}_j * 1.1345 + 0.8952 \tag{5.3}$$

Further the value of output neuron *k* was classified at the point 0.5, therefore the separating line could be obtained by setting OutPutNeuron_k equals to 0.5:

$$\text{HiddenNeuronOutput}_i * (-1.3444) + \text{HiddenNeuronOutput}_j * 1.1345 + 0.8952 = 0.5 \tag{5.4}$$

Where the separating line has the equation:

$$\text{HiddenNeuronOutput}_j = (1.3444/1.1345)\ \text{HiddenNeuronOutput}_i + (0.5-0.8952)/1.1345 \quad (5.5)$$

Which equals:

$$\text{HiddenNeuronOutput}_j = 1.185015\text{HiddenNeuronOutput}_i - 0.34835 \quad (5.6)$$

Based on equation 5.6, one could clearly see that the relationship between $\text{HiddenNeuronOutput}_i$ and $\text{HiddenNeuronOutput}_j$ is linear. Therefore the feature positions dominated by these two coordinates were linearly separable. The weights and biases that are shown in equations 5.3, 5.4 and 5.5 for the MLP of MNNs actually determine the separating line. Figure 5.6 shows how a line can separate the feature space with $\text{HiddenNeuronOutput}_i$ and $\text{HiddenNeuronOutput}_j$ being the two axes.

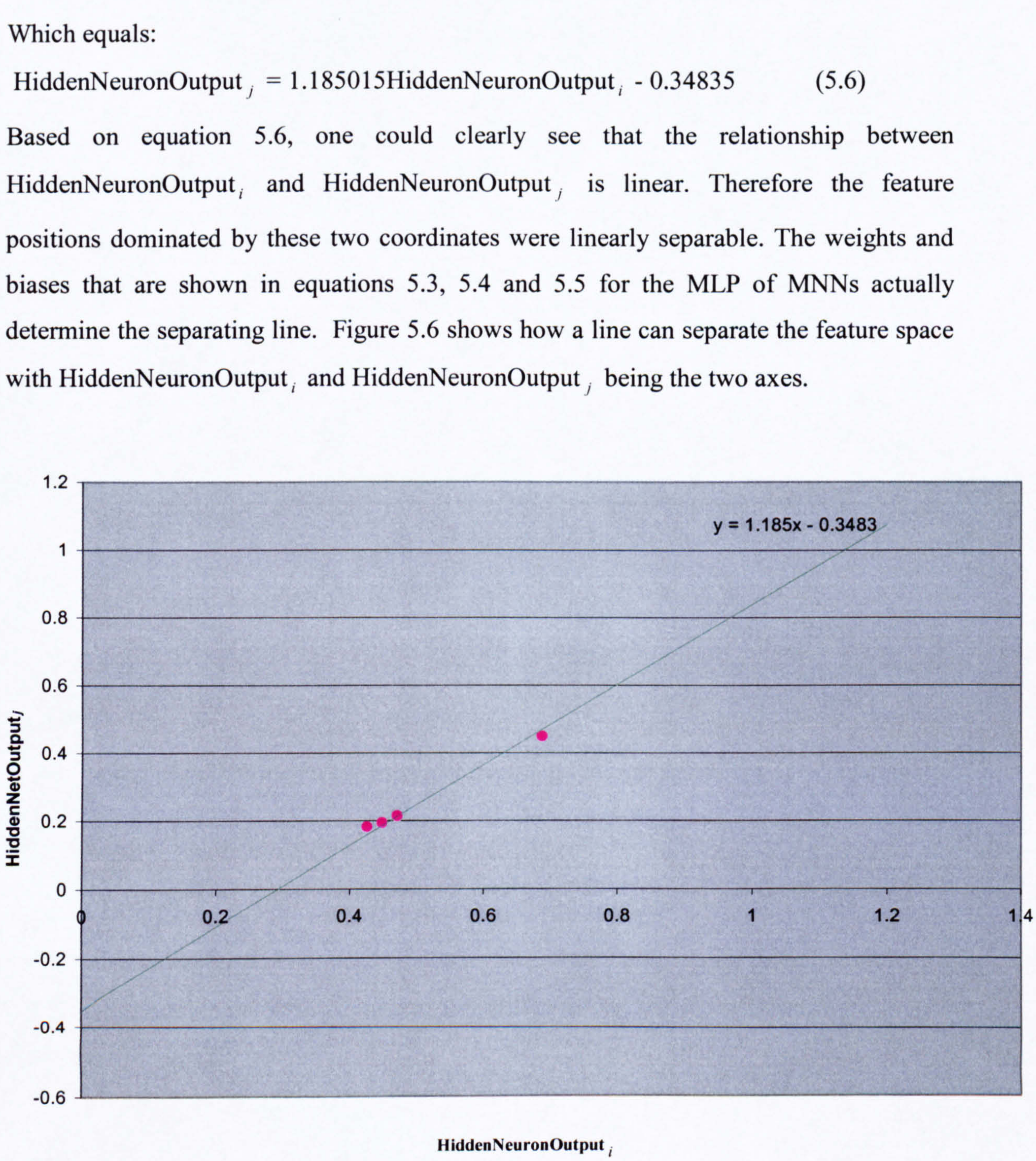

Figure 5.6 Decision boundary of the feature space dividing the four example inputs into two classes

Figure 5.6 showed the separating line divided the input space of the output neuron into two parts. The line was drawn based on equation 5.6 where the X axis represents $\text{HiddenNeuronOutput}_i$ and the Y axis represents $\text{HiddenNeuronOutput}_j$. The space above the line is taken as 1, and the part below the line is taken as 0. Any inputs from Neuron_i and Neuron_j would fall into the area either above the line or underneath. It can be seen that the activation function of Neuron_k (the output unit) classified the two feature spaces into two parts according to the information provided by two inputs (i.e. the hidden neurons' outputs).

According to equation 5.6, Figure 5.7 shows how the decision boundary divided the 76 test data for this problem into two classes. For comparison, the corresponding target values are also shown.

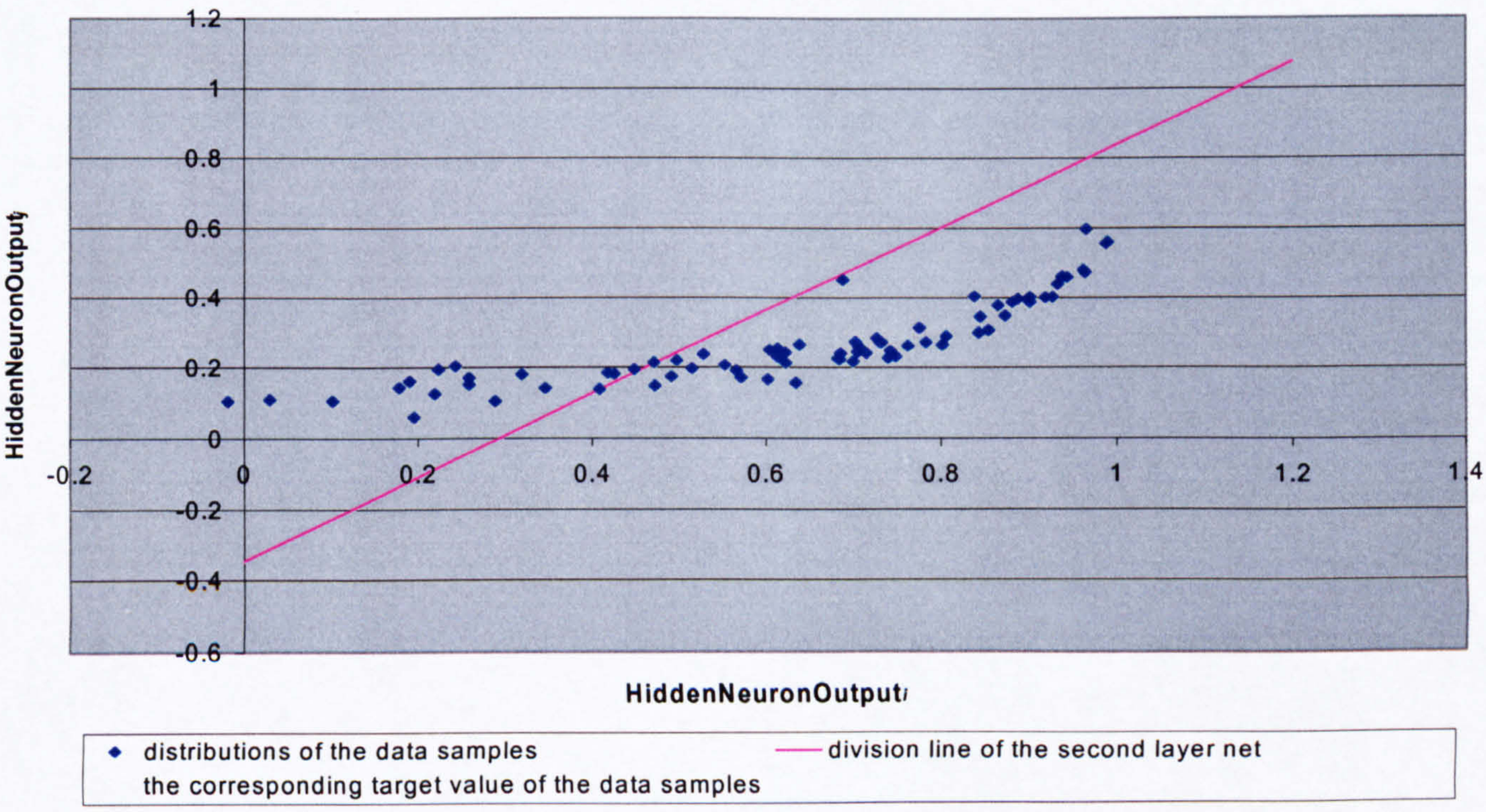

Figure 5.7 The decision boundary of the feature space for MLP combiner

Similar to Figure 5.6, all the 76 test data feature positions calculated by the activation function of hidden units are shown in figure 5.7. For comparison, the corresponding target values were drawn on this figure as well. One part that plays an important role during networks training is the quality of the training data (Callan, 1999). The representation of the training data must reflect the whole features of the problem. Therefore, though some data in this case were not correctly classified, it wasn't clear that this failure was due to a limitation of MNNs or was because of the quality of the source data. As each ensemble method is related to the quality of ensemble members' performances to a certain degree, so it is not difficult to understand that MNNs method can only work well on some part of data sets.

In this section, the MNNs with three ensemble members was used as an illustration to demonstrate how MNNs classified data. As the structures of MLP for different ensemble groups used in the experiments were the same. Therefore, all these MLPs worked in the same way though the number of input patterns was different. Regardless of how many dimensions the inputs have, they were all transformed into two-dimensional patterns by the two hidden neurons.

5.3 Relationship between Performance of MNN and its Ensemble Members' Diversity

It has been noticed that there is a trend regarding the performance of MNNs in chapter four. The generalization of MNNs moved up and down on all the data sets whilst increasing the numbers of ensemble members steadily. It was interested to know reasons that could result in this trend. To discuss this question, figures 5.8, 5.9 and 5.10 were created based on the test results of pima-diabetes' 10-fold cross validation and provided the corresponding data in Appendix C. Each of these figures displays the ensemble outputs for data where the MNN combiner produced the correct outputs, but majority voting was wrong.

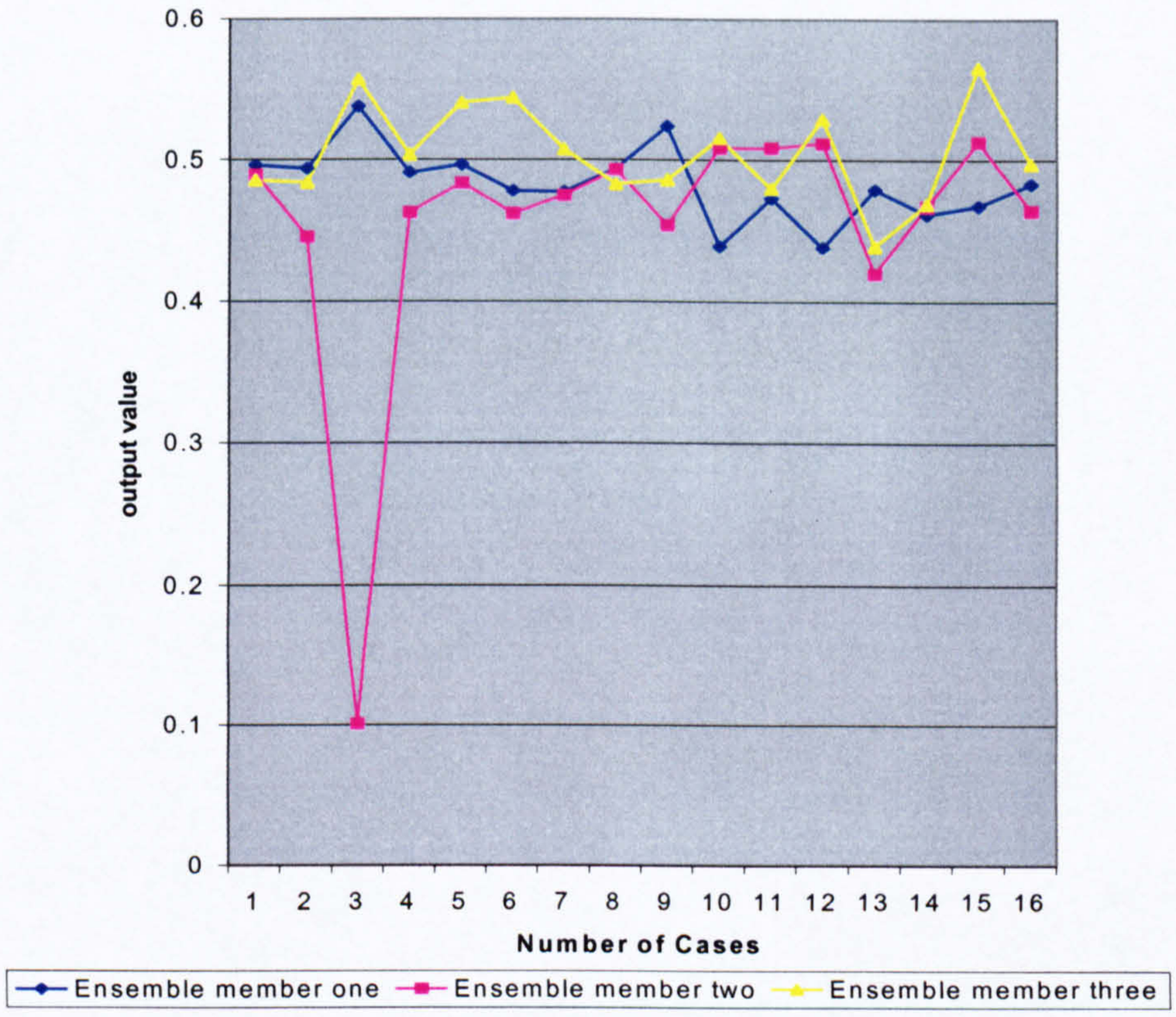

Figure 5.8 The ranges of three ensemble member's outputs, for data where the MNN combiner gave the correct outputs, whereas majority voting gave the incorrect outputs

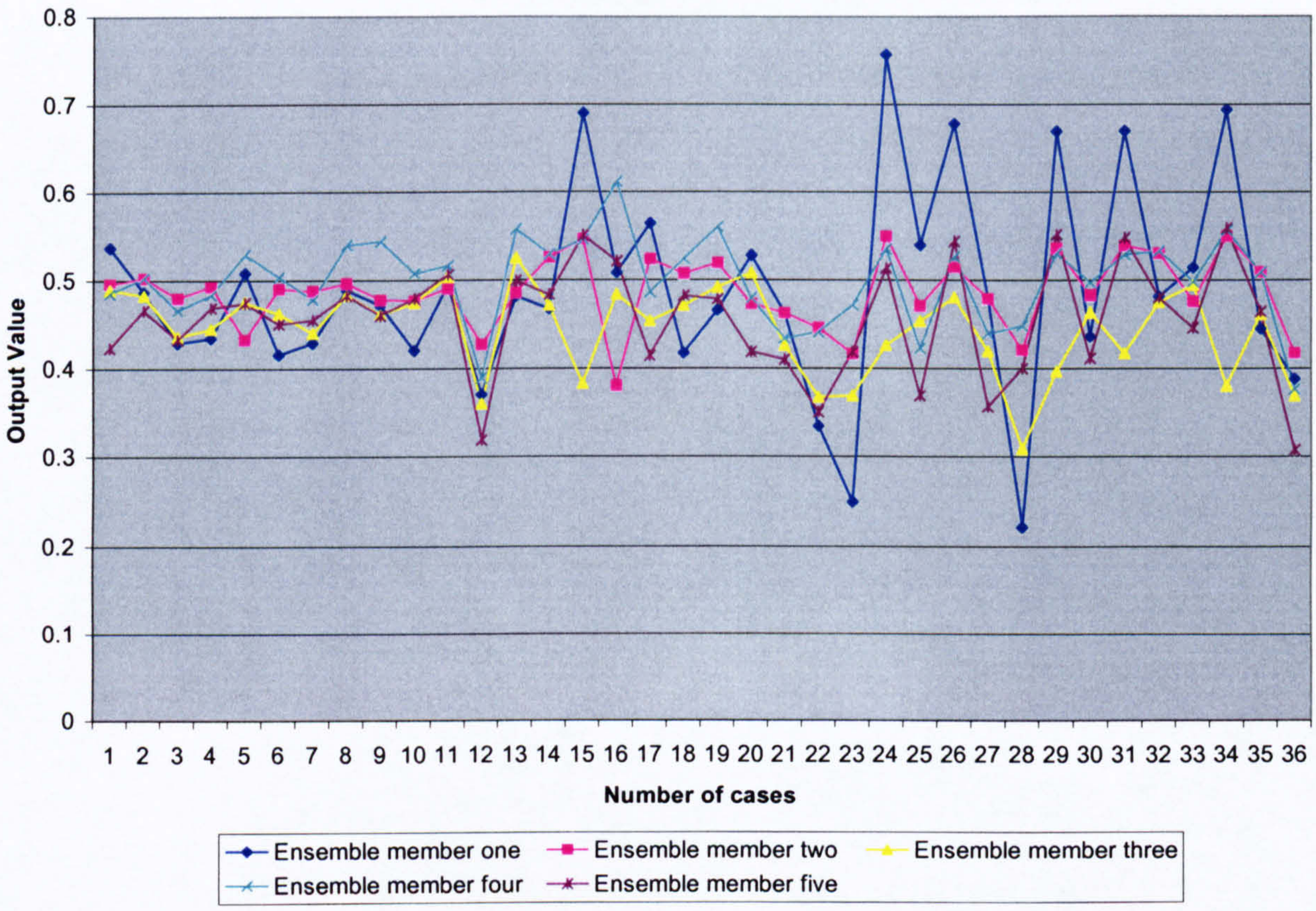

Figure 5.9 The ranges of five ensemble member's outputs, for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output

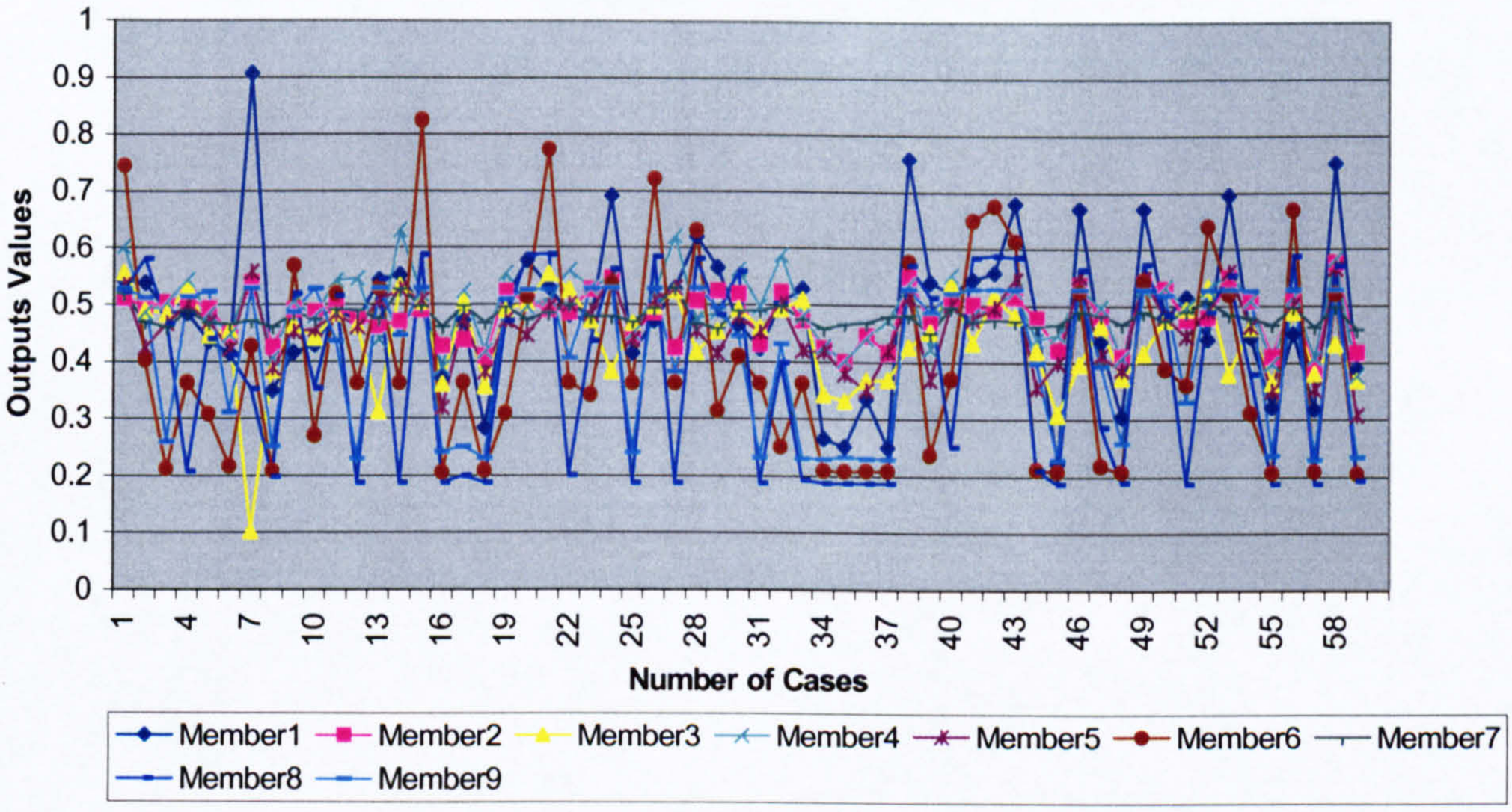

Figure 5.10 The ranges of nine ensemble member's outputs, for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output

From figures 5.8, 5.9 and 5.10, increased ranges of ensemble members outputs can be observed as the number of MLPs in the ensemble increases. In figure 5.8, the ranges were between 0.4 - 0.6 (with just one exception). In figure 5.9, the ranges increased to between 0.3 and 0.7. In figure 5.10, most of ranges fell between 0.2 and 0.8 (with very few exceptions). The number of cases continued to increase (from 36 to 59), but the steps of increasing could be seen slow down. The ranges of eleven and thirteen ensemble members are illustrated in figures 5.11 and 5.12.

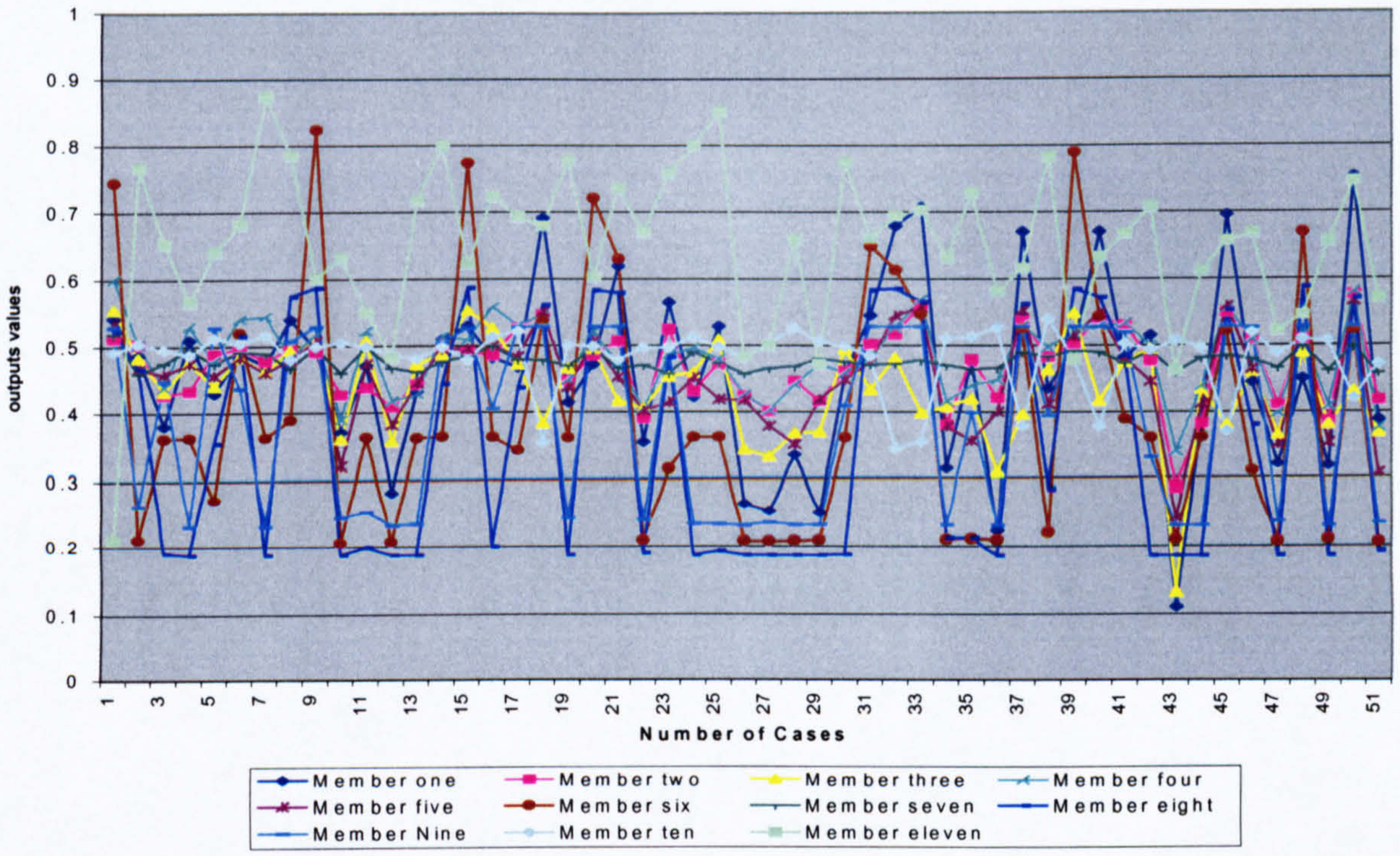

Figure 5.11 The ranges of eleven ensemble members' outputs for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output

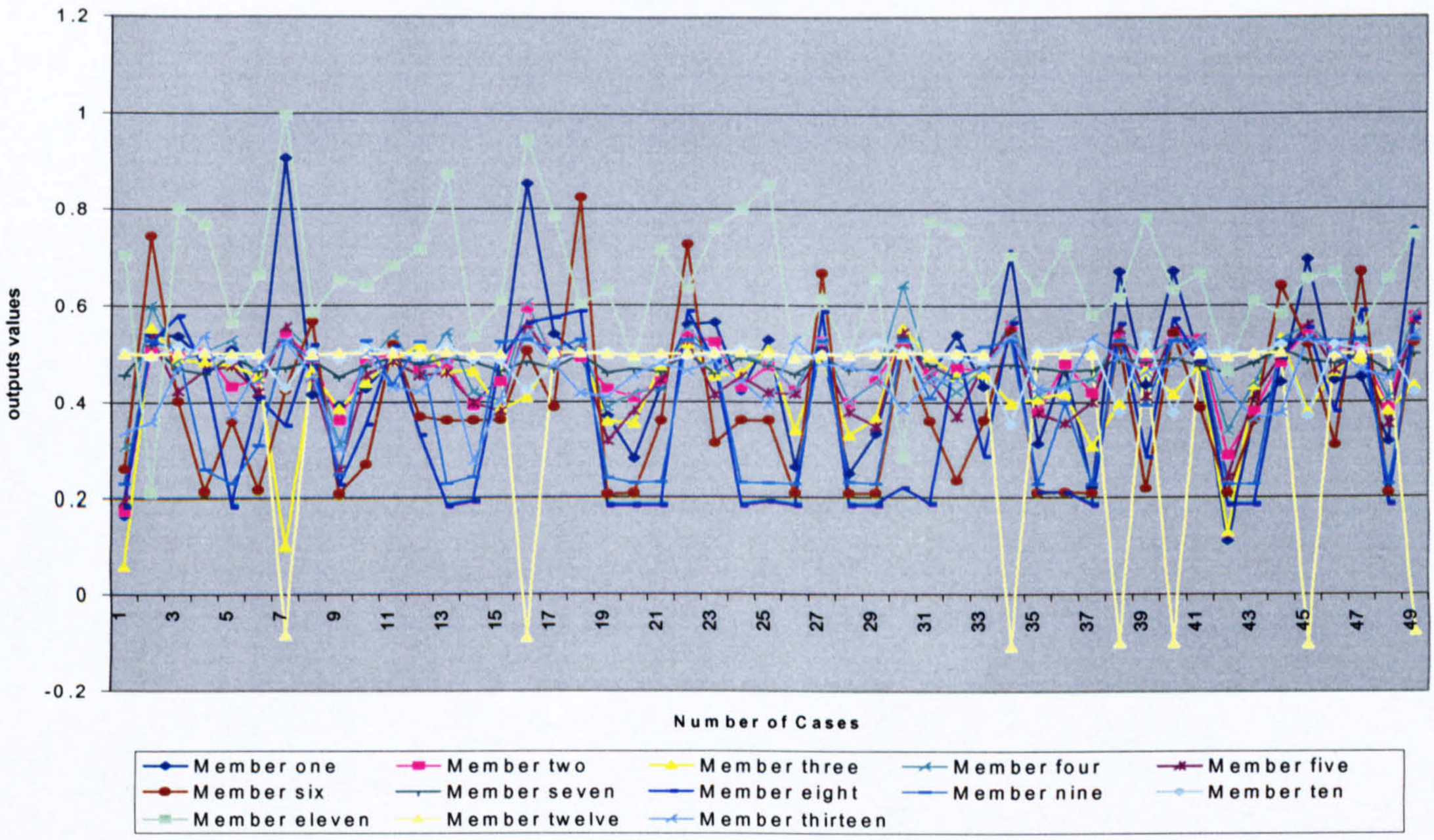

Figure 5.12 The ranges of thirteen ensemble members' outputs for data where the MNN combiner gave the correct output, whereas majority voting gave the incorrect output

An interesting thing to see is that there is not so large a jump observed when moving from figure 5.10 to figure 5.11. Noticing that in figure 5.11 and figure 5.12, the ranges of data where the MNN made the correct prediction whilst majority voting was wrong were still spread mainly in the range between 0.2 and 0.8. However, the number of cases dropped compared with the MNNs using nine ensemble members.

From figures 5.8 to 5.12, it is observed that in the ranges of the ensemble members' outputs from 0.2 to 0.8, MNNs worked better than majority voting. Still it has to be admitted that MNNs' abilities were not powerful enough to solve some of the extremely difficult data. So that resulted in the further question: why MNNs performances stopped at some point?

The importance of the diversity among the ensemble members and ensemble members' accuracy has been stressed throughout the thesis. For an ensemble model diversity is a big issue directly related to the ensemble performance (as stated in chapter one). Therefore for the second staged ensemble model, one of the extents of MNNs performance depends on the amounts of correlation among the errors of these ensemble members' outputs and their accuracy to some degree. Another factor that needs to be considered which effects the MNNs' performance is the number of ensemble members that are combined in the second stage of MNNs as proved in section 4.4.1. To illustrate the relationships among the size of ensembles, ensemble diversity and ensemble members' accuracy, the pima-diabetes data was used once more to demonstrate these interactions.

Firstly, in table 5.6, results on each part of the ten-fold cross validation test data are presented for all the nineteen ensemble members used in the experiments. Then, the correlation coefficients, which represent the diversity of each pair of ensemble members, are calculated according to equation 4.1. The results of diversity between each pair of ensemble members are listed in table 5.7. Next, the average diversity and average accuracy of each ensemble groups and results were computed, which are reported in table 5.8. Finally, the relationships between ensemble diversity, accuracy with the size of ensembles were drawn in figures 5.13 and 5.14 based on the data from table 5.8.

Part of Data	Ensemble Members Performance																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0.6447	0.6711	0.6842	0.6579	0.6053	0.6316	0.6184	0.5921	0.6184	0.6316	0.6184	0.6842	0.6316	0.6316	0.6447	0.5921	0.6316	0.6579	0.6579
2	0.8026	0.8026	0.7763	0.8158	0.7763	0.7632	0.7763	0.7237	0.7500	0.7763	0.7500	0.7763	0.7895	0.8026	0.8026	0.7763	0.7895	0.8026	0.8289
3	0.6711	0.6842	0.7237	0.6974	0.6711	0.6842	0.6053	0.6711	0.6579	0.7237	0.6447	0.6974	0.6711	0.6316	0.6974	0.6184	0.6842	0.7368	0.6579
4	0.6711	0.7237	0.6316	0.7237	0.6842	0.6974	0.5921	0.7368	0.6974	0.6579	0.6579	0.5921	0.7105	0.6711	0.6974	0.7237	0.6711	0.7237	0.7105
5	0.7105	0.7237	0.6974	0.6974	0.6974	0.6579	0.6316	0.6842	0.6842	0.7237	0.7105	0.7237	0.7368	0.6974	0.7368	0.6974	0.7105	0.7237	0.7105
6	0.7237	0.7237	0.7632	0.7237	0.7237	0.7105	0.6711	0.7368	0.6842	0.7368	0.7500	0.8026	0.7105	0.6842	0.7237	0.6711	0.7368	0.7500	0.7368
7	0.7895	0.8289	0.8158	0.8421	0.8026	0.7763	0.8289	0.7500	0.7368	0.7895	0.7368	0.7632	0.8158	0.7763	0.8289	0.7895	0.8421	0.8421	0.8421
8	0.7763	0.8026	0.7368	0.8289	0.7895	0.6711	0.7895	0.6711	0.7105	0.7500	0.7895	0.7763	0.7632	0.7368	0.7895	0.7105	0.8026	0.8158	0.7895
9	0.7500	0.7500	0.7368	0.7237	0.7105	0.7500	0.7237	0.7368	0.7763	0.7105	0.7632	0.7500	0.7105	0.7237	0.7632	0.7500	0.7237	0.7763	0.7500
10	0.7632	0.6974	0.6842	0.7105	0.6711	0.6447	0.6316	0.6974	0.6579	0.6711	0.7632	0.7763	0.6974	0.6974	0.7632	0.6842	0.6711	0.7237	0.6842
Average	0.7303	0.7408	0.7250	0.7421	0.7132	0.6987	0.6869	0.7000	0.6974	0.7171	0.7184	0.7342	0.7237	0.7053	0.7447	0.7013	0.7263	0.7553	0.7368

Table 5.6 10-fold cross validation results of ensemble members on Pima-Diabetes data set

No. of members	Diversity between pairs of ensemble members																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		0.8264	0.6826	0.8265	0.8416	0.5859	0.8467	0.5409	0.6991	0.7410	0.8847	0.7451	0.8320	0.9289	0.9660	0.7859	0.8211	0.8427	0.8401
2			0.7004	0.9650	0.9598	0.7294	0.9390	0.5442	0.7838	0.8145	0.6422	0.4296	0.9478	0.9182	0.8857	0.8620	0.9571	0.9384	0.9816
3				0.6836	0.7442	0.7150	0.8269	0.3833	0.5317	0.8786	0.5057	0.7355	0.6606	0.6718	0.6953	0.4821	0.8356	0.7742	0.7519
4					0.9570	0.6526	0.9108	0.4890	0.6632	0.8064	0.6238	0.4408	0.9151	0.8677	0.8730	0.7732	0.9436	0.9388	0.9354
5						0.7022	0.8934	0.6089	0.7400	0.9017	0.7167	0.5355	0.9455	0.8700	0.8976	0.8019	0.9802	0.9687	0.9417
6							0.6838	0.7757	0.8544	0.7068	0.3749	0.2687	0.6945	0.7198	0.6494	0.8001	0.7083	0.7651	0.7905
7								0.3846	0.7126	0.7992	0.6741	0.6153	0.8318	0.8770	0.8624	0.7252	0.9383	0.9066	0.9349
8									0.7408	0.5204	0.5219	0.2076	0.6296	0.5691	0.6085	0.7911	0.5241	0.6270	0.5900
9										0.6350	0.6238	0.3032	0.7223	0.7969	0.7441	0.8955	0.6987	0.7927	0.7961
10											0.5778	0.6164	0.8491	0.7599	0.8052	0.6335	0.9252	0.8882	0.8202
11												0.7926	0.6330	0.7246	0.8177	0.6244	0.6641	0.7005	0.6461
12													0.4264	0.5477	0.6335	0.2640	0.5796	0.5234	0.4848
13														0.9179	0.9110	0.8838	0.9272	0.8956	0.9373
14															0.9266	0.8996	0.8563	0.8448	0.9442
15																0.8510	0.8821	0.9132	0.8717
16																	0.7511	0.8040	0.8719
17																		0.9594	0.9444
18																			0.9101
19																			

Table 5.7 Diversity of pairs of ensemble members for Pima-Diabetes data set

Average	Ensemble Groups								
	3	5	7	9	11	13	15	17	19
Diversity	78.30	81.87	80.88	73.44	72.94	71.27	74.13	75.51	77.56
Accuracy	73.60	73.03	71.96	71.49	71.55	71.75	71.85	71.80	72.09

Table 5.8 Diversity and accuracy of each ensemble group for Pima-Diabetes data set

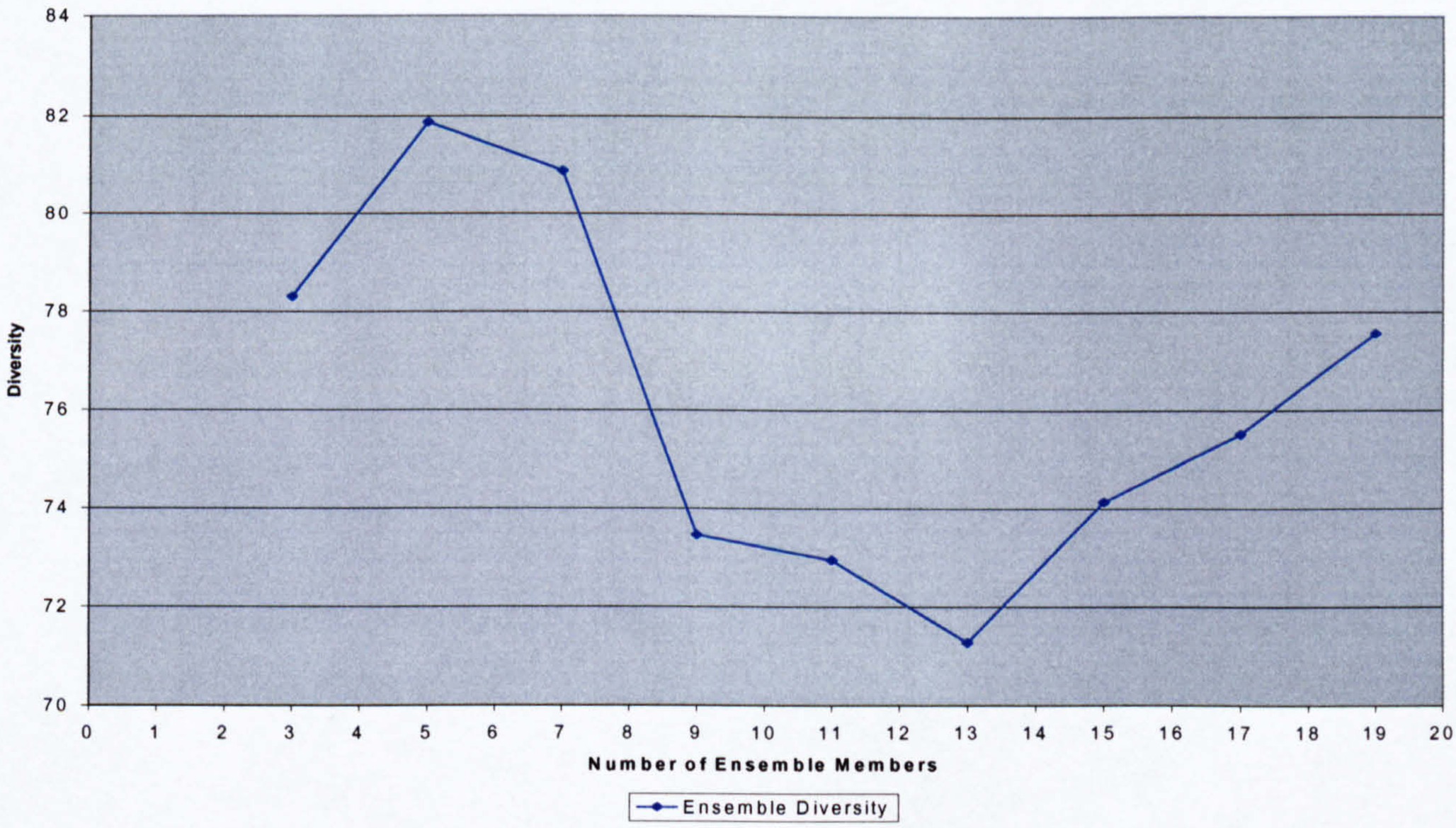

Figure 5.13 Relationship between size of ensembles and average diversity of ensemble groups

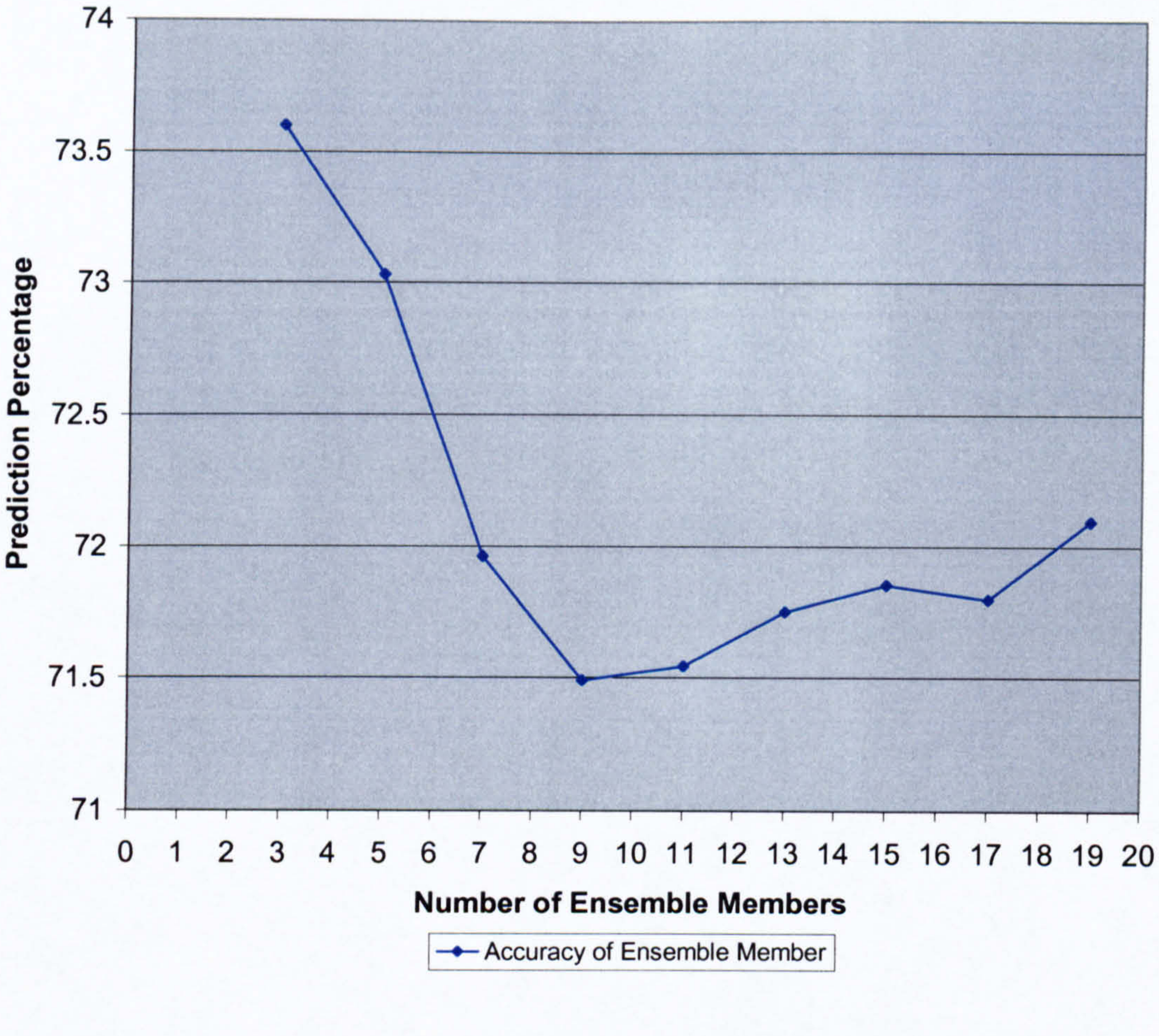

Figure 5.14 Relationship between size of ensembles and average performance of each ensemble group

Figures 5.13 and 5.14 have something in common. Both figures start from high point then slip down and pick up again steadily. From Figure 5.13, note that ensemble diversity researches the lowest level which represents the high diversity as the number of ensemble members is thirteen. However, at this point, MNNs did not make a peak performance. In figure 5.14, the relatively high average accuracy with the increased size of ensembles occurrs at the point where the number of ensemble members is nineteen. However, in figure 5.13, the corresponding diversity at that point is quite high. Still the best generalization of MNNs were obtained when the number of ensemble members is nineteen.

From the above observation, it is found that there is conflict between increasing the number of ensemble members and decreasing the correlation among them at some point. In the meantime, the performance of MNNs is partly related to the performance of its ensemble members. However the relationships among these factors were also noticed: number of ensemble members, ensemble diversity and accuracy are complex which is difficult to reveal their actual interactions in quantity. The interactions of these links result in different performances of MNNs. The issue of how much effect diversity, accuracy and the number of ensemble members have on the generalization of ensembles is a really big issue in the research of ensembles.

5.4 Discussion of Other Experiments

5.4.1 Why does a single layer network not work well as a combiner?

A single layered MLP can only implement a linearly separable problem and most two binary input applications are linearly separable (Picton, 2000). Once classifying multi-dimensional inputs with more than two dimensions, a hyperplane in the pattern space is needed to separate the classes (Picton, 2000 and Callan, 1999). When returned to the experiments, it was known that the smallest number of inputs for a second staged MLP was three. Therefore the classification difficulties were well beyond the capability of a single layered MLP and accordingly such an MNNs model could not produce good ensemble performances.

5.4.2 Why does having the original inputs (as extra inputs to the combiner) not help?

The output results of ensemble members along with the original inputs used as the inputs for the second stage MNNs did not show improvement in ensemble accuracy. There are two possible reasons that may lead to such results. Firstly, the knowledge contained in

this second stage MNNs training sets could not supply more information to be learned than just using ensemble members outputs. Secondly, the second staged MNNs could not exhaust more information from the training sets limited by the power of MLP in the second stage. As reported in section 4.3.2, several more complex MLP modelswere served as second stage combiners trained on such inputs and none of them achieved better results than just using inputs from ensemble members outputs. Therefore its more likely that the first reason result in no improvement can be made.

5.5 Conclusions

Through the analysis discussed in chapter five, it was found that the reason for MNNs achieving a better performance lies within the differences between the kinds of decision surfaces a second staged network can model when compared to those decision surfaces that can be produced by majority voting. Moreover, how diversity among ensemble members affected the performance of MNNs was analyzed. The interactions between ensemble members and their diversity were demonstrated. One can hope that all of these analyses will help us to understand the working mechanism of MNNs and improve the performance of MNNs further in future work.

Chapter Six

Conclusions and Future Work

6.1 Conclusions

In this thesis, a novel ensemble model was proposed originated on the concept of adaptive combination.

Firstly, the author has demonstrated a model of multistage ensembles, where the adaptive properties of a second stage network are used to combine the outputs of the individual ensemble members. The recommended structures of MNNs are two stages, of which a one two-layered MLP with a certain number of (two in the experiments) hidden neurons is used in the second stage. The concept of MNNs was first introduced and a model of MNNs was first explicitly constructed.

Secondly, an implementation of MNNs on various data was described by the author, such as several machine learning data sets and the human-gene splice data sets. The comparisons between MNNs with the most frequently used ensemble method, majority voting, were conducted. In the experiments, MNNs showed significant improvement over majority voting using a confidence level of 0.05 on some of data sets and on all machine learning data sets at a confidence level of 0.1. The fact has been established that MNNs is an effective method of improving ensemble generalization.

Thirdly, the author found the performances of MNNs are related mainly to the diversity among ensemble members, accuracy of ensemble members and the number of ensemble members. Therefore, the best MNNs ensemble performance can be obtained at some

combination point of these factors. The successful identification of the main factors that affect MNNs performance provide guidelines for constructing and implementing MNNs in future applications.

Finally, through analysis the author found that MNNs achieve better performance because of the difference between the kinds of decision surfaces a second staged network can model when compared to those decision surfaces that can be produced by majority voting. Such analysis results allow us to apply more effective NNs in the second stage MNNs when solving different problems and result in better performance in future work.

In summary, the author's major contributions in this thesis are: an alternative ensemble combination method: explorations regarding creating, training and testing this MNNs were conducted. Recommendations based on the experiments can be made for setting up an optimum MNNs model.

There is an expectation that MNNs can be tested and improved through further experiments and applications, especially in the quickly developing area of bioinformatics. The successful results on splice data sets indicate that it is well worth doing more explorations using this novel method in the human genome project.

6.2 Suggestions for Future Work

Being a novel model of ensemble methods, MNNs can be developed and implemented further in many ways. Some suggestions for future work on MNNs explorations are provided. The recommendation includes the following aspects.

a) Data

As mentioned before, some relatively small data sets were applied on the MNN models. Due to the limitation of the size of data set, the data used in the second stage MNNs were

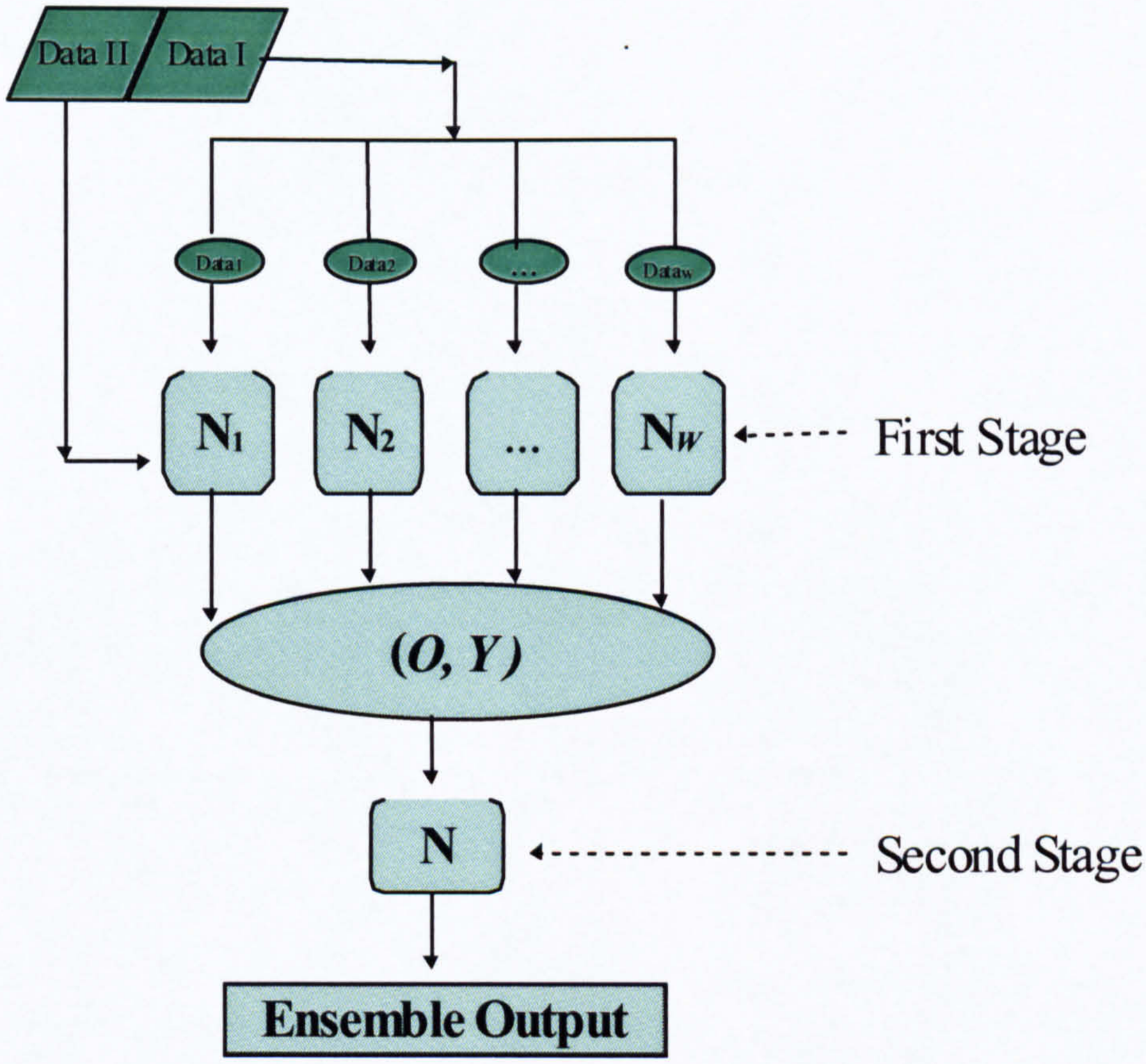

Figure 6.1 MNN model using different part of data for training

overlapped with the data in the first stage training. Therefore it may affect the enhancement of MNNs performance. Here, another MNN model which uses a different part of the data for training is presented.

The difference between figure 6.1 and the standard model presented in figure 2.1 is the data part taken from the source data for two-stage training. In figure 6.1, the source data is split into two parts: Data I for the use of the first stage training, Data II for the second stage training. There is no any overlap between these two parts of the data, thus the data which will be used in the second stage training does not go through any procedures of the first stage training. Under such circumstances, the second stage training should not be affected by the first stage training and should reflect the features contained in the source data more naturally. However only a large data set can be implemented using this model.

b) Creating Ensemble Members

There is a deficiency of standard and effective criteria on how to create ensemble MNNs. Though the diversity among ensemble members has one of the key roles in constructing an ensemble, in many situations this factor could not be measured using available techniques. In addition, the exploration on creating diverse ensemble members is inadequate. Current methods only provide the choices for creating diverse ensemble members, but no standard procedures and measurements can be followed. Exploration in this field will not only be of benefit to the MNNs technique, but has universal application to ensemble research.

c) Options of Second Stage MNNs

The choices of parameters for the second stage MNNs, such as number of units in hidden layers and learning rate are arbitrary set by trial and error. Therefore the question of the optimal parameters settings for the second stage MLP still remains. Though the number

of hidden neurons in the second stage of MNNs may vary in different applications, my experiences recommended that the simple structures of the MLP are good enough to implement the combination function. However further development and implementation of other NN models or other activation combination functions in the second stage MNNs can be explored. For example, Radial-Basis Function networks (RBF) (Powell, 1985) and support vector machines (Boser, Guyon and Vapnik, 1992) can all be options used in the second stage. In addition, the choice of MLP architecture for the second stage combiner should be investigated. For instance, the implementation of more hidden units and other activation combination functions, such as softmax and the logistic function can be investigated and compared with the MLP model used in this thesis.

d) More Stages Ensembles

An MLP with one hidden layer was applied in the second stage MNNs. On top of that, another MLP may be added as the third stage MNNs. Since MNNs derives its name in the fact that multiple stages can be applied, therefore more stages can be created. However, with the increase of the number of stages, the complexity of this kind of model must be considered.

The training of second staged MNNs did cost more time compared with other conventional methods whose combination procedures do not require prior training. But with the help of current high-speed computers, the time taken for training the combiner can be almost ignored. For instance, it only took a few minutes for the training of the second stage of the MNNs on each of the data sets used in the experiments. However it must be pointed out that being an adaptive combination method, the cost on training of second stage MNNs did exist. Therefore there is the question, how many stages is optimum? It may depend on the nature of source data or depend on the power of the model used as combiner. Whatever it will be, one criteria is always correct that if the increase of a model's complexity does not improve the performance compared with the simple model, then it is not worth exploring such a model. On the other hand, if the increase of a model's complexity does improve the performance, then is it worth adopting

such a model, which requires more cost? So the balance between complexity and improvement gained by using MNNs needs to be well considered in the future research.

e) Application on Other Machine Learning Method

The idea of multistage ensembles is not only limited to the field of NN ensembles. It can be expanded to the area of other machine learning approaches, such as decision trees, genetic algorithms or similarity measure approaches (and so on). Furthermore, the implementation of multistage ensembles is not only limited to one method, but also can be applied to several different types of machine learning approaches. The application on mixing different machine learning approaches within one ensemble using the idea of MNNs is an attractive topic.

f) Analysis

Theoretical exploration, especially mathematical justification for the approximation of MNNs will offer the potential for creating more effective models.

g) Applications

A wide range of applications on large scale, real-world data sets are needed to be explored to make MNNs a really useful ensemble method, especially the practical applications for MNNs in pattern recognition and bioinformatics research areas.

The MNNs model is just one of many possible explorations on methods for developing ensemble ANNs. The major concern in the research of ensembles is to find an effective and standard way to create and organize an ensemble ANN model. In order to be adopted in practice, some systematic design methodology for ensemble NNs must be proposed.

There is still a long way to go until formal methods for specifying ensemble NNs to optimize their application in the real world are developed.

Bibliography

Bishop, C. M. (1995) Neural Networks for Pattern Recognition, *ISBN 0 19 853849 9, Oxford University Press Inc, New York,* 116.

Boser, B., Guyon, I. and Vapnik, V. N. (1992) A Training Algorithm for Optimal Margin Classifiers, *Fifth Annual Workshop on Computational Learning Theory*, San Mateo, CA: Morgan Kaufmann, 144-152.

Breiman, L. (1996a) Bagging Predictors, *Machine Learning*, 26(2), 123-140.

Breiman, L.(1996b) Bias, Variance, and Arcing Classifiers, *Technical Report 460,* Statistics Department, University of California at Berkeley.

Breiman, L.(1997) Prediction Games and Arcing Classifiers, *Technical Report 504,* Statistics Department, University of California at Berkeley.

Breiman, L. (1999) Combining Predictors, In *Combining Artificial Neural Nets-Ensemble and Modular Multi-Net Systems*, Springer-Verlag, A. J. C. Sharkey (Ed.), 31-50.

Broomhead, D. S. and Lowe, D. (1988) Multivariable Functional Interpolation and Adaptive Networks, *Complex Systems*, 2, 321-355.

Callan, R. (1999) *The Essence of Neural Networks*, Prentice Hall Europe ISBN 0-13-908732-X, 27.

Demuth, H. and Beale, M. (1994) *Neural Network Toolbox*, The Math Works Inc., 4-15.

Dietterich, T. G. (2000) Ensemble Methods in Machine Learning, *Lecture notes in computer science*: *Proceedings of MCS 2000*, Kittler, J. and Roli, F. (Eds.), ISBN 185233004X 1857, 1-15.

Drucker, H., Schapire, R. and Simard, P. (1993) Boosting Performance in Neural Networks, *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4), 705-719.

Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y. and Vapnik, V. (1996) Boosting and Other Ensemble Methods, *Advances in Neural Information Processing Systems*, 8, 479-485.

Elman, J. L. (1990) Finding Structure in Time, *Cognitive Science*, 14, 179-211.

Freund, Y. and Schapire, R. E. (1995) A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting, In *Computational Learning Theory: Second European Conference, Euro COLT '95* : Springer-Verlag, 23-37.

Freund, Y. and Schapire, R. E. (1996) Experiments with a New Boosting Algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156.

Geman, S., Bienenstock, E. and Doursat, R. (1992) Neural Networks and the Bias/Variance Dilemma, *Neural Computation*, 4, 1-58.

Ghoneim, K. and Kumar Vijaya B V. K. (1995) Learning Ranks with Neural Networks, In *Applications and Science of Artificial Neural Networks*, Proceedings of the SPIE, 2492, 446-464.

Ghosh, J. and Tumer, K. (1994) Structural Adaptation and Generalization in Supervised Feedforward Networks, *Journal of Artificial Neural Networks*, 1, 1-55.

Hansen, L. and Salamon, P. (1990) Neural Network Ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12, 993-1001.

Hashem, S. and Schmeiser, B. (1993) Approximating a Function and Its Derivatives using MSE-Optimal Linear Combinations of Trained Feedforward Neural Networks, *Proceedings of the Joint Conference on Neural Networks*, New Jersey, 87, 617-620.

Haykin, S. (1999) Neural Networks-A Comprehensive Foundation: Prentice-Hall, Inc. ISBN 0-13-908385-5, 156-203.

Ho, T. K., Hull, J. J., and Srihari, S. N. (1994) Decision Combination in Multiple Classifier Systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 16(1), 66-76.

Ho, T. K. (2001) Data Complexity Analysis for Classifier Combination, *Multiple classifier systems: Second International Workshop 2001,* Cambridge, Kittler, J. and Roli, F (Eds.), ISBN 3-540-42284-6 53-67.

Ho, T. K. (2002) Multiple Classifier Combination: Lessons and Next Steps, *Hybrid Methods in Pattern Recognition*, Kandel, A. and Bunde, H. (Eds.), World Scientific, 171-198.

Houle, G.F., Aragon, D.B., Smith, R.W., Shridhar, M. and Kimura, D. (1998) A Multilayered Corroboration-based Check Reader. *Document Analysis Systems* II, World Scientific. Hull, J. J. and Taylor, S. L. (Eds.), 495-546.

Hudson, B., Whitley, D., Ford, M. G. and Browne, A. (2003) Biological Data Mining: High Fidelity Rule Extraction from Trained Neural Networks, *Proceedings of the 12th European Symposium on Quantitative Structure-Activity Relationships*, Ford, M.G. and Livingstone, D. (Eds.), Bournemouth, UK (in press).

Jackson, J. C. and Craven, M. W. (1996) Learning Sparse Perceptrons, *In Advances in Neural Information Processing Systems,* 8, 654-660.

Jacobs, R. A. (1995) Methods for Combining Experts' Probability Assessments, *Neural Computation*, 7, 867-888.

Kittler, J. (1998) Combining Classifiers: A Theoretical Framework, *Pattern Analysis and Applications,* 1, 18-27.

Kreinovich, V. Y. (1991) Arbitrary Nonlinearity is Sufficient to Represent all Functions by Neural Networks: A Theorem, *Neural Networks,* 4(3), 381-383.

Krogh, A. and Vedelsby, J. (1995) Neural Network Ensembles, Cross Validation and Active Learning, *Advances in Neural Information Processing Systems*: MIT press, Tesauro, G., Touretzky, D. S. and Leen, T. K. (Eds.), 7, 231-238.

Lee, D. S., Srihari, S. N. (1993) Handprinted Digit Recognition: A Comparison of Algorithms.*Proc. Third International Workshop On Frontiers In Handwriting Recognition, Buffalo, USA,* 153-162.

Lincoln, W. and Skrzypek, J. (1990) Synergy of Clustering Multiple Back Propagation Networks, *Advances in Neural Information Processing Systems-2*: Morgan Kaufmann, Touretzky, D. (Ed.), 650-657.

Liu, y. and Yao, X. (1997) Negatively Correlated Neural Networks Can Produce Best Ensembles, *Australian Journal of Intelligent Information Processing Systems,* 4(3/4), 176-185.

MacKay, D. J. C. (1992) The Evidence Framework Applied to Classification Problems, *Neural Computation,* 4(5), 720-736.

Maclin, R. and Opitz, D. (1997) An Empirical Evaluation of Bagging and Boosting, *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 546-551.

McCulloch, W. S. and Pitts, W. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biophysics*, 5, 115-133.

Mendenhall, W and Ott, L. (1980) *Understanding Statistics*, California: Wadsworth, ISBN 0-87872-241-6. 219-220.

Merz, C. J. and Pazzani, M. J. (1997) Combining Neural Network Regression Estimates with Regularized Linear Weights, *Advances in Neural Information Processing Systems9,* Mozer, M. C., Jordan, M. I. and Petsche, T. (Eds.), MIT Press, 564-570.

Nilsson, N. J. (1965*) Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, New York: McGraw-Hill.

Oza, N. C. and Tumer, K. (2001) Input Decimation Ensembles: Decorrelation through Dimensionality Reduction, *Proceedings of the Second International Workshop on Multiple Classifier Systems*, Cambridge, U.K., Lecture Notes in Computer Science 2096, Heidelberg, Springer, ISBN 3-540-42284-6, 238-247.

Parmanto, B., Munro, P. W. and Doyle, H. R. (1996a) Improving Committee Diagnosis with Resampling Techniques, *Advances in Neural Information Processing Systems8,* Cambridge, MA: MIT press, Mozer, M. and Hasselmo, M. (Eds.).

Parmanto, B., Munro, P. W. and Doyle, H. R. (1996b) Reducing Variance of Committee Prediction with Resampling Techniques, *Connection Science,* 8, 405-425.

Partridge, D. and Griffith, N. (1995) Strategies for Improving Neural Net Generalisation *Neural Computing and Applications,* 3, 27-37.

Partridge, D. and Yates, W. B. (1996) Engineering Multiversion Neural-net Systems, *Neural Computation*, 8, 869-893.

Perrone, M. P. and Cooper, L. N. (1993) Learning From What's Been Learned: Supervised Learning In Multi-neural Networks Systems, *Proceedings of the World Congress on Neural Networks III*, INNS Press, 354-357.

Picton, P. (2000) Neural Networks (Second Edition), Palgrave: New York, ISBN: 0-333-80287-X, 32-37.

Powell, M. J. D. (1985) Radial Basis Functions for Multivariable Interpolation: A Review, *IMA Conference on Algorithms for the Approximation of Functions and Data,* RMCS, Shrivenham, England, 143-167.

Quinlan, J. R. (1996) Bagging, Boosting and C4.5, *Proceedings of the 13th National Conference on Artificial Intelligence*, MIT Press, 725-729.

Raviv, Y. and Intrator, N. (1996) Bootstrapping with Noise: An Effective Regularization Technique, *Connection Science,* 8(3&4), 355-372.

Roli, F., Raudys, S. and Marcialis, G. L. (2002) An Experimental Comparison of Fixed and Trained Rules for Crisp Classifiers Outputs, *Proceedings of the Third International Workshop on Multiple Classifier Systems*, Cagliari, Italy, Lecture Notes in Computer Science 2364, Heidelberg, Springer, ISBN 3-540-431818-1, 232-241.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986) Learning Representations of Back-propagation Errors, *Nature (London),* 323, 533-536.

Schapire, R. E. (1990) The Strength of Weak Learnability, *Machine Learning*, 5, 197-227.

Schapire, R. E. (1999) A Brief Introduction to Boosting, *16th International Joint Conference on Artificial Intelligence*, Dean, T. (Ed.), Morgan Kaufmann, 1401-1406.

Schwenk, H. and Bengio, Y. (1997) Adaboosting Neural Networks, *Application to Online Character Recognition*, 1327, 967-972.

Sharkey, A. J. C. (1999) Multi-Net Systems, *Combining Artificial Neural Nets-Ensemble and Modular Multi-Net* Systems: Springer-Verlag, A. J. C. Sharkey (Ed.), 1-30.

Sharkey, A. J. C., Sharkey, N. E. and Chandroth, G. O. (1996) Neural Nets and Diversity, *Neural Computing and Applications,* 4, 218-227.

Sharkey, A. J. C. (2002) Types of Multinet System, *Proceedings of the Third International Workshop on Multiple Classifier* Systems, Cagliari, Italy, Lecture Notes in Computer Science 2364, Heidelberg, Springer, ISBN 3-540-431818-1, 108-117.

Thanaraj, T. A. (1999) A Clean Data Set of EST-Confirmed Splice Sites from Homo Sapiens and Standards for Clean-up Procedures, *Nucleic Acids Res.*,27(13), 2627-2637.

Thanaraj, T. A. (2000) Positional Characterisation of False Positives From Computational Prediction of Human Splice Sites, *Nucleic Acids Research,* 28 (3), 744-754.

Tumer, K. and Ghosh, J. (1995) Order Statistics Combiners of Neural Classifiers, In *Proceedings of the World Congress on Neural* Networks, Washington D.C.: INNS press, 31-34.

Tumer, K. and Ghosh, J. (1996) Error Correlation and Error Reduction in Ensemble classifiers, *Connection Science. Special Issue on Combining Artificial Neural Ensemble Approaches*, 8(3&4), 385-404.

Wesolkowski, S and Hassanein, K. (1997) Comparative Study of Combination Schemes for An Ensemble of Digit Recognition Neural, *Proceedings of IEEE International Conference on Systems, Man And Cybernetics,* 3534-3539.

White, H. (1990) Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings, *Neural Networks,* 3(5), 535-549.

Wickramaratna, J. , Holden, S. and Buxton, B. (2001) Performance Degradation in Boosting, *Proceedings of the Second International Workshop on Multiple Classifier Systems*, Cambridge, UK, Lecture Notes in Computer Science 2096, Heidelberg, Springer, ISBN 3-540-42284-6, 11-21.

Wolpert, D. H. (1992) Stacked Generalization, *Neural Networks*, 5, 241-259.

Yang, S., Browne, A. and Picton P. D. (2002) Multistage Neural Network Ensembles, *Proceedings of the Third International Workshop on Multiple Classifier Systems*, Cagliari, Italy, Lecture Notes in Computer Science 2364, Heidelberg, Springer, ISBN 3-540-431818-1, 91-97.

Yang, S., Browne A., Picton, P.D., Hudson B. D. and Whitley, D. (2002) Multistage Neural Networks: Adaptive Combination of Ensemble Results, *The Proceedings of the Fourth International Conference on Recent Advances in Soft Computing*, Nottingham, U.K. ISBN 1-84233-0764, 55-60.

Yang, S., Browne A., Picton, P.D. (2003) Multistage Neural Network Ensemble: Adaptive Combination of Ensemble Results, *Applications and Science in Soft Computing,* Springer-Verlag, (In press).

Zeng, X., Martinez, T. R. (2000) Using a Neural Network to Approximate an Ensemble of Classifiers, *Neural Processing Letters*, 12, 225-237.

Appendix A: Algorithm 1

(Ensemblembertrain.m)

```
%pima data sets include 768 instances and 8 attributes with two classes choice
%create ncandidate number of ensemble members

ncandidate=5

for (ensemblenum=1:ncandidate)

if ensemblenum==1
    randdata;
    randtraindata;
    cd 1;
else if ensemblenum==2
   cd ..;
    randdata;
    randtraindata;
    cd 2;
else if ensemblenum==3
    cd ..;
    randdata;
    randtraindata;
    cd 3;
    else if ensemblenum==4
    cd ..;
    randdata;
    randtraindata;
cd 4;
else if ensemblenum==5
cd ..;
randdata;
randtraindata;
cd 5;
end
```

```
    end
  end
end
```

```
load  c:\matlab6p1\work\pima-indian\filedata.dat;
load  c:\matlab6p1\work\pima-indian\validdata.dat;
load  c:\matlab6p1\work\pima-indian\testdata.dat;

datafile=filedata;
[row,col]=size(datafile);
trainnum=row;
[validrow,validcol]=size(validdata);
validnum=validrow;
[testrow,testcol]=size(testdata);
testnum=testrow;

Y=(datafile(:,col))';
ma=max(Y);  % maximum value of output class
mi=min(Y);  % minumal value of output class
traindata=datafile;  %data used for training including validation data
vadata=validdata;   %   data used for validation
tedata=testdata; %  data used for testing

traininput=datafile(:,1:col-1);
traintarget=datafile(:,col);
vainput=vadata(:,1:col-1);
vatarget=vadata(:,col);
testinput=tedata(:,1:col-1);
testtarget=tedata(:,col);
```

```
firstrun=0;
j=1;
m=0;
n=0;
total=0;
nout=0;       % 1 output parameters
ndata=row-1; % number of input instances
```

```
OPTIONS=zeros(1,18);
OPTIONS(1)=1;     %provides display error values
OPTIONS(14)=5;    %number of training cycles
nin=col-2;        % number of input parameters


testpercentage =0;
algorithm='QUASINEW';

for (A=0.1:0.1:3)
    alpha=A;
    for (B=2:1:10)
        nhidden=B;
        j=1;
        m=0;

%TRAIN THE MODEL FOR SOME EPOCHS TILL BEST RESULT IS GENERATED

while(j<=20)

NET=mlp(nin+1,nhidden,1,'linear',alpha);
n=1;

% control the training cycle until the best result obtained

while( n==1 )

   m=m+1;
   [NET,OPTIONS] = netopt(NET,OPTIONS, traininput, traintarget, algorithm);

%The generated result after training
trainper=0;
vaper=0;
testper=0;
% generating training result
TOUT = MLPFWD(NET, traininput);
VAOUT=MLPFWD(NET,vainput);
%let the output decimal belong to its nearest integer
for i=1:trainnum
     TOUT(i,1)=round(TOUT(i,1));
   if TOUT(i,1)>=ma
```

```
    TOUT(i,1)=ma;
end

if TOUT(i,1)<mi
    TOUT(i,1)=mi;
end

if TOUT(i,1)==traintarget(i,1)
    trainper=trainper+1;
end
end
innertrainper=trainper/trainnum;
for i=1:validnum
    VAOUT(i,1)=round(VAOUT(i,1));
    if VAOUT(i,1)>=ma
       VAOUT(i,1)=ma;
    end
   if VAOUT(i,1)<mi
       VAOUT(i,1)=mi;
    end

    if VAOUT(i,1)==vatarget(i,1)
        vaper=vaper+1;
    end
 end
 innervaper=vaper/validnum;


if m==1  %save the firist inter cycle result

   save ftrainper.dat innertrainper -ascii ;
   save  fvaper.dat innervaper  -ascii;
   save fnet.net NET;
else  load  ftrainper.dat;   %load the previous best cycle result
      load   fvaper.dat;
%compare this cycle with the best cycle result
if innertrainper>ftrainper & innervaper>fvaper

save  ftrainper.dat innertrainper  -ascii;
save  fnet.net NET;
save   fvaper.dat innervaper -ascii;
```

```
   else n=0;
   end
end  %finish the jth training cycle



% compare this jth cycle result with the previous j-1 best one we got so far
load  ftrainper.dat;
load  fvaper.dat;
testper=0;
load  fnet.net NET -mat;
TESTOUT = MLPFWD(NET, testinput);

 for i=1:testnum
    TESTOUT(i,1)=round(TESTOUT(i,1));
    if TESTOUT(i,1)>=ma
       TESTOUT(i,1)=ma;
    end

    if TESTOUT(i,1)<mi
       TESTOUT(i,1)=mi;
    end
    if TESTOUT(i,1)==testtarget(i,1)
        testper=testper+1;
    end
end

testpercentage=testper/testnum;
if j==1 %save the first result of each external while [100(j)]cycle
     save  temptrainper.dat ftrainper -ascii;
     save tempvaper.dat fvaper -ascii;
     save  temptestper.dat testpercentage -ascii;
end

if firstrun==0   %record the first external training  cycle result
  save testpercen.dat testpercentage -ascii;
  save  trainper.dat ftrainper -ascii ;
  save  vaper.dat fvaper  -ascii;
  save  finalnet.net NET;
  firstrun=1;
```

```
else
        load  trainper.dat;
        load  vaper.dat;
        load  testpercen.dat;
        load  temptrainper.dat;
        load  tempvaper.dat;
        load  temptestper.dat;

       if testpercentage>=testpercen  % save the best result of whole loop
         if ftrainper>=trainper | fvaper>=vaper
         save  testpercen.dat testpercentage -ascii;
         save  finalnet.net NET ;
         save  trainper.dat ftrainper -ascii ;
         save vaper.dat fvaper  -ascii;
         end
       end
%compare the jth result with the previous best result within inner while loop
    if fvaper>temptrainper
         if fvaper>tempvaper
             save  temptrainper.dat ftrainper -ascii;
             save tempvaper.dat fvaper -ascii;
             save  temptestper.dat testpercentage -ascii;
          end
     end
 end


end  %end the internal loop

j=j+1
m=0;

end  % end the while external loop

% just run the result
% save the best result of this running
load  temptrainper.dat ;
load  tempvaper.dat;
load  temptestper.dat;
curroutput=[nhidden alpha temptrainper tempvaper temptestper];
```

```
if total==0
    output1=zeros(10,6);
    output1=curroutput;
    total=1;
    save output1.dat -ascii;
else load  output1.dat;
    output1=[output1;curroutput];
end
save  output1.dat output1 -ascii;
end
end

end % end the all the ensemble members training
```

Appendix B: Algorithm 2

(Ensembletrain3.m)

```
%load the source input data
load firsttraindata.dat;
%distrubute the data
randdata;

% ten-fold cross validation is applied
for( num=1:1:10)                    %loop 1
    fulltraindata=firsttraindata;
    thistestdata=fulltraindata(((num-1)*76+1):((num-1)*76+76),:);
    fulltraindata(((num-1)*76+1):((num-1)*76+76),:)=[];
    thistraindata=fulltraindata;
save temptraindata.dat thistraindata -ascii;

randtraindata;

load crosstraindata.dat;
load crossvaliddata.dat;

%read the data file
datafile=crosstraindata;
[row,col]=size(datafile);
Y=(datafile(:,col))';
ma=max(Y);  % maximum value of output class
mi=min(Y);  % minumal value of output class
traindata=datafile ;
vadata=crossvaliddata;
testdata=thistestdata;
trainnum=row;
[validrow,validcol]=size(crossvaliddata);
validnum=validrow;
[testrow,testcol]=size(thistestdata);
testnum=testrow;
```

```
%prepare the training input and output data sets
traininput=traindata(:,1:col-1);
traintarget=traindata(:,col);
vainput=vadata(:,1:col-1);
vatarget=vadata(:,col);
testinput=testdata(:,1:col-1);
testtarget=testdata(:,col);
% execute the trained three ensemble members nets and get their
%outputs.
load c:\matlab6p1\work\pima-indian\2\finalnet.net NET -mat;
net1input=MLPFWD(NET, traininput);
net1vainput=MLPFWD(NET,vainput);
net1testinput=MLPFWD(NET,testinput);

load c:\matlab6p1\work\pima-indian\3\finalnet.net NET -mat;
net2input=MLPFWD(NET, traininput);
net2vainput=MLPFWD(NET,vainput);
net2testinput=MLPFWD(NET,testinput);

load c:\matlab6p1\work\pima-indian\4\finalnet.net NET -mat;
net3input=MLPFWD(NET, traininput);
net3vainput=MLPFWD(NET,vainput);
net3testinput=MLPFWD(NET,testinput);


% combine the above three nets outputs as the new network model %inputs
entraininput=[net1input net2input net3input ];
entraintarget=traintarget;

envainput=[net1vainput net2vainput net3vainput];
envatarget=vatarget;

entestinput=[net1testinput net2testinput  net3testinput ];
entesttarget=testtarget;
n=1;
```

```
m=1;
first=1;
nout=0;       % 1 output parameters
ndata=row-1; % number of input instances
nin=2;        %number of input instances

OPTIONS=zeros(1,18);
OPTIONS(1)=1;     %provides display error values
OPTIONS(14)=5;    %number of training cycles

algorithm='QUASINEW';

% try different alpha value and different number of hidden neurons
%combinations. each NN model will try j numbers of random
%initialization and each initialization will run till reaching the
%best result.


for( alpha=0.1)       %loop 2
   for(nhidden=2)     %loop 3
 trainper=0;
 vaper=0;
 testper=0;
 votetestper=0;
 j=1;

%TRAIN THE MODEL FOR SOME EPOCHS TILL BEST RESULT IS GENERATED
while(j<=1000)         %loop 4
n=1;
NET=mlp(nin+1,nhidden,1,'linear',alpha); %initialization

% control the inner training cycle until the best result obtained

while( n==1)          %loop 5

  [NET,OPTIONS] = netopt(NET,OPTIONS, entraininput, entraintarget,
algorithm);
```

```
%The generated result after training
trainper=0;
vaper=0;
testper=0;
TOUT =mlpfwd(NET, entraininput);% generating training result
VAOUT=mlpfwd(NET,envainput);

%let the output decimal belong to its nearest integer

for i=1:trainnum
     TOUT(i,1)=round(TOUT(i,1));
   if TOUT(i,1)>=ma
       TOUT(i,1)=ma;
   end

   if TOUT(i,1)<=mi
       TOUT(i,1)=mi;
   end

   if TOUT(i,1)==entraintarget(i,1)
    trainper=trainper+1;
   end
end

%generating validation result
for i=1:validnum
    VAOUT(i,1)=round(VAOUT(i,1));
    if VAOUT(i,1)>=ma
       VAOUT(i,1)=ma;
    end
if VAOUT(i,1)<=mi
    VAOUT(i,1)=mi;
 end

 if VAOUT(i,1)==envatarget(i,1)
     vaper=vaper+1;
```

```
    end
 end


trainper=trainper/trainnum;
validper=vaper/validnum;

if first==1
    save tempvalidper.dat validper -ascii;
    save tempnet.net NET ;
    first=0;
else load tempvalidper.dat;

    if validper<=tempvalidper
        n=0;
    else if m>=30  %prevent the dead loop
            n=0;
        else if validper>tempvalidper
                save tempnet.net NET;
            end
        end
    end
end
m=m+1;

end  %end loop 5
j=j+1;
end  %end loop 4

end  %end loop 3
end  %end loop 2

load tempnet.net NET -mat;
switch (num)
case 1
    save newmnn3net1.net NET;
 case 2
```

```
    save newmnn3net2.net NET;
case 3
    save newmnn3net3.net NET;
case 4
    save newmnn3net4.net NET;
case 5
    save newmnn3net5.net NET;
case 6
    save newmnn3net6.net NET;
case 7
    save newmnn3net7.net NET;
case 8
    save newmnn3net8.net NET;
case 9
    save newmnn3net9.net NET;
case 10
    save newmnn3net10.net NET;
end
```

```
load tempnet.net NET -mat;
TESTOUT =mlpfwd(NET, entestinput);

 for i=1:testnum
    TESTOUT(i,1)=round(TESTOUT(i,1));
    if TESTOUT(i,1)>=ma
       TESTOUT(i,1)=ma;
    end

    if TESTOUT(i,1)<=mi
       TESTOUT(i,1)=mi;
    end
    if TESTOUT(i,1)==entesttarget(i,1)
        testper=testper+1;
    end
end

testpercentage=testper/testnum;
```

```
if num==1

    save ensem3crosstest.dat testpercentage -ascii;

else load ensem3crosstest.dat;

    en3cross=[ensem3crosstest;testpercentage];

save ensem3crosstest.dat en3cross -ascii;

end
end % end loop 1

%test results
test3crossresult;

display('The whole programme:newensemtrain3.m has finished running!');
```

Appendix C:

Ensemble Members of Pima-Diabetes on Ten-Fold Cross-Validation

Test Results

Part of Source Data	Ensemble Memebers' Performance									
	1	2	3	4	5	6	7	8	9	10
1	0.6447	0.6711	0.6842	0.6579	0.6053	0.6316	0.6184	0.5921	0.6184	0.6316
2	0.8026	0.8026	0.7763	0.8158	0.7763	0.7632	0.7763	0.7237	0.75	0.7763
3	0.6711	0.6842	0.7237	0.6974	0.6711	0.6842	0.6053	0.6711	0.6579	0.7237
4	0.6711	0.7237	0.6316	0.7237	0.6842	0.6974	0.5921	0.7368	0.6974	0.6579
5	0.7105	0.7237	0.6974	0.6974	0.6974	0.6579	0.6316	0.6842	0.6842	0.7237
6	0.7237	0.7237	0.7632	0.7237	0.7237	0.7105	0.6711	0.7368	0.6842	0.7368
7	0.7895	0.8289	0.8158	0.8421	0.8026	0.7763	0.8289	0.75	0.7368	0.7895
8	0.7763	0.8026	0.7368	0.8289	0.7895	0.6711	0.7895	0.6711	0.7105	0.75
9	0.75	0.75	0.7368	0.7237	0.7105	0.75	0.7237	0.7368	0.7763	0.7105
10	0.7632	0.6974	0.6842	0.7105	0.6711	0.6447	0.6316	0.6974	0.6579	0.6711
Average	0.73027	0.74079	0.725	0.74211	0.71317	0.69869	0.68685	0.7	0.69736	0.71711

Continues

Part of Source Data	Ensemble Memebers' Performance								
	11	12	13	14	15	16	17	18	19
1	0.6184	0.6842	0.6316	0.6316	0.6447	0.5921	0.6316	0.6579	0.6579
2	0.75	0.7763	0.7895	0.8026	0.8026	0.7763	0.7895	0.8026	0.8289
3	0.6447	0.6974	0.6711	0.6316	0.6974	0.6184	0.6842	0.7368	0.6579
4	0.6579	0.5921	0.7105	0.6711	0.6974	0.7237	0.6711	0.7237	0.7105
5	0.7105	0.7237	0.7368	0.6974	0.7368	0.6974	0.7105	0.7237	0.7105
6	0.75	0.8026	0.7105	0.6842	0.7237	0.6711	0.7368	0.75	0.7368
7	0.7368	0.7632	0.8158	0.7763	0.8289	0.7895	0.8421	0.8421	0.8421
8	0.7895	0.7763	0.7632	0.7368	0.7895	0.7105	0.8026	0.8158	0.7895
9	0.7632	0.75	0.7105	0.7237	0.7632	0.75	0.7237	0.7763	0.75
10	0.7632	0.7763	0.6974	0.6974	0.7632	0.6842	0.6711	0.7237	0.6842
Average	0.71842	0.73421	0.72369	0.70527	0.74474	0.70132	0.72632	0.75526	0.73683

Appendix D: Publications (1)

Multistage Neural Network Ensembles

Yang, S., Browne, A. and Picton P. D.

Proceedings of the Third International Workshop on Multiple Classifier Systems, Cagliari, Italy, Lecture Notes in Computer Science 2364, Heidelberg, Springer, ISBN 3-540-431818-1, 91-97, 2002.

Multistage Neural Network Ensembles

Shuang Yang[1], Antony Browne[1] and Philip D. Picton[2]

[1]School of Computing, Information Systems and Mathematics, London Guildhall University, London EC3N 1JY, UK
Email: syang@lgu.ac.uk
Tel: (+44) 0207 320 1705, Fax: (+44) 0207 320 1707
[2]School of Technology and Design, University College Northampton, Northampton NN2 6JD, UK

Abstract. Neural network ensembles (some times referred to as committees or classifier ensembles) are effective techniques to improve the generalization of a neural network system. Combining a set of neural network classifiers whose error distributions are diverse can lead to generating more accurate results than any single network. Combination strategies commonly used in ensembles include simple averaging, weighted averaging, majority voting and ranking. However, each method has its limitations, dependent either on the application areas it is suited to, or due to its effectiveness. This paper proposes a new ensembles combination scheme called multistage neural network ensembles. Experimental investigations based on multistage neural network ensembles are presented, and the benefit of using this approach as an additional combination method in ensembles is demonstrated.

1 Introduction

Combining the outputs of diverse classifiers can lead to an improved result [1, 2]. Common ensemble combination strategies include simple averaging [8, 9], weighted averaging [10], majority voting [11], and ranking [12,13].

There are no unique criteria on the usage of all the above combination methods. The choice mainly depends on the nature of the application the ensemble is being used for, the size and quality of training data, or generated errors on different regions of the input space. One combination method applied on the ensemble of a regression problem may generate good results, but may not work on a classification problem and vice versa. In addition, different classifiers will have an influence on the selection of the appropriate combination method. However, empirical experiments reported to date cannot find an optimal method for selecting the combination strategy to be used. More theoretical development and experiments are needed to explore in this field. The ensemble combination technique most related to the work reported in this paper is stacking, as the new model outlined here inherits some ideas from stacking and develops them further. In this paper, we will propose a new model for ensemble combination method based on another neural network layer.

1.1 Stacking

Stacking [6] covers two areas of ensemble construction: preparing data and ensemble combination. Generally, stacking deals with two issues. Firstly, it uses the idea of cross-validation to select training data for ensemble members. Secondly, it explores the notion of using the second level generalizers to combine the results of the first level generalizers (ensemble members). A feature of stacked generalization is that the information supplied to the first-level ensemble members comes from multiple partitioning of the original dataset, which divides that dataset into two subsets. Every ensemble member is trained by one part of the partitions, and the rest of parts are used to generate the outputs of the ensemble members (to be used as the second space generalizers (i.e. combiners) inputs). Then the second level generalizers are trained with the original ensembles outputs and the second level generalizers output is treated as the correct guess. In fact, stacked generalization works by combined classifiers with weights according to individual classifier performance, to find a best combination of ensemble outputs. Based on the idea of the combining method of stacking, we propose a new type of ensemble neural network model called multistage ensemble neural networks.

2 Multistage neural network ensembles

Inspired by stacking, some researchers have realized that it is possible to construct a new combination method using a similar idea.

As early as 1993,some experiments were done in digit recognition [14] by using a single layer network to combine ensemble classifiers. Unfortunately, these experiments did not show any performance gain compared with other combination strategies. It was claimed the failure was due to the very high accuracy of all the classifiers being combined.

In 1995, Partridge and Griffith presented a selector-net approach [5]. The selector-net was defined as a network which used the outputs from a group of different trained nets as its input. The experiments based on this idea delivered that selector-net's performance was better than the populations of networks they were derived from. It clearly confirmed that this kind of ensemble method is better than individual neural networks. But no further exploitation has been done to compare the performance of this strategy with any other ensemble method.

More recently, Kittler [15] stated that:"it is possible to train the output classifier separately using the outputs of the input classifiers as new features".

Very recently, Zeng [7] used a single neural network as an approximator for a voting classifiers. It was claimed that storage and computation could be saved, at the cost of a little less accuracy. However, it is noticed here a neural network being used to approximate the behavior of the ensemble, instead of using it as part of the ensemble components.

This paper extends the idea of stacking and investigates the use of a single neural network model as a combiner to combine the ensemble members results.

The experimental results demonstrate that it is an improved approach which achieves the better generalisation performance of neural network ensembles. The major improvement of this combination method is it offers an alternative neural network ensemble model, which is proved effective after the generalisation improvement obtained, compared with majority voting.

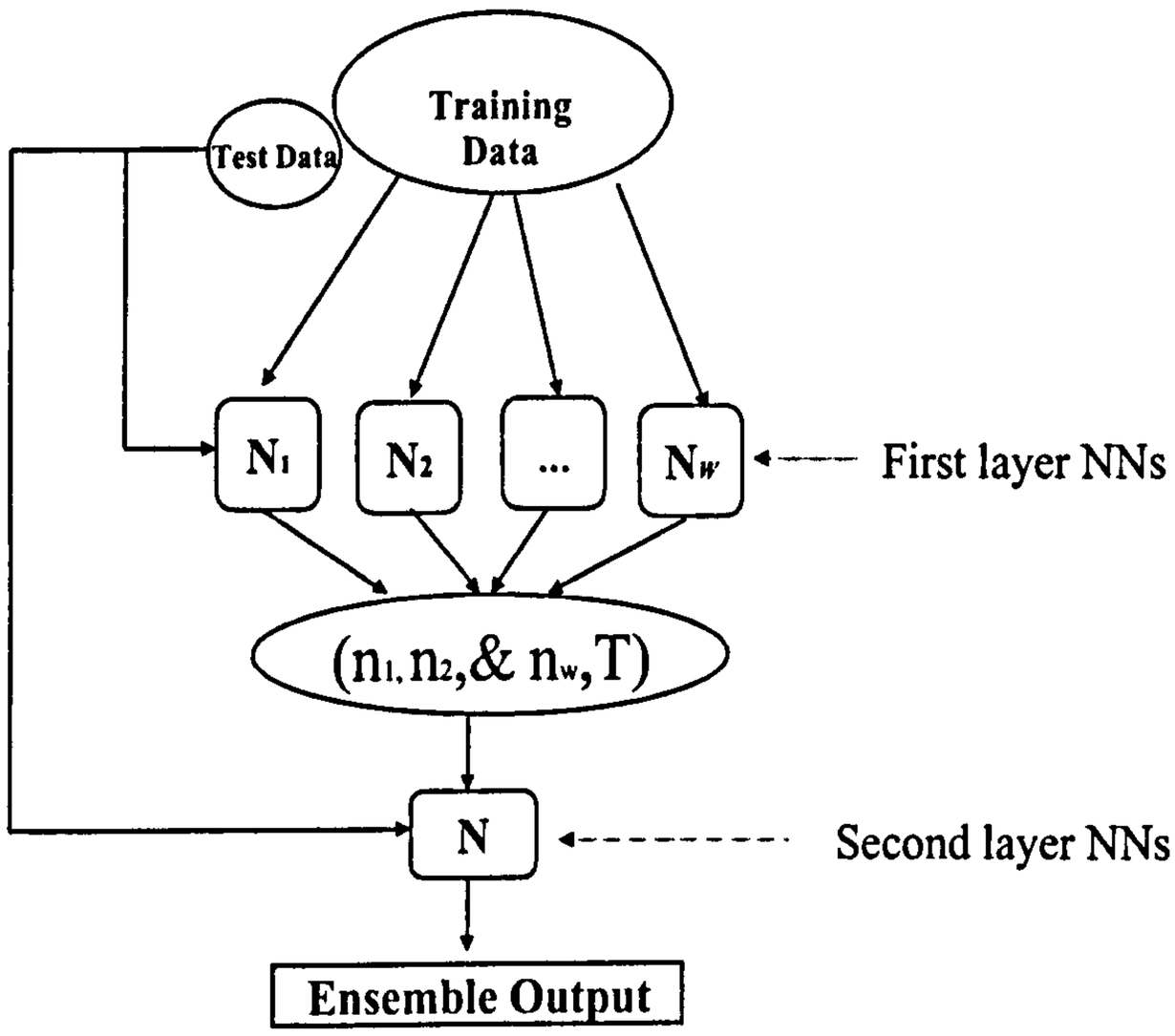

Fig. 1. An illustration of a multistage ensemble neural network.

The experiments on multistage neural network ensembles is based on a well trained group of diverse single neural networks (so called ensemble candidates). A single neural network is trained to combine these well trained neural nets results by concatenating their outputs together as its input. The reason of employing another neural networks to combine ensemble candidates is relying on neural networks capability. A neural network can be trained to prerform complex functions by asjusting the connection weights. Except majority voting, other approaches all adopt weights while combining. If so, why not use neural networks to assign weights to those ensemble members automatically instead of employing some traditional mathematical method manually? The advantage of a neural network is its ability to automatically adjust the connection weights. Therefore,

Table 1. Summary of UCI machine learning depository data sets, where '*' signifies a multi-class data set.

Data Set	No. of Cases	No. of Input features	No. of Output features
Breast-cancer-w	682	9	2
Bupa-Liver	345	6	2
Glass*	214	9	6
Ionosphere	351	34	2
Iris*	150	4	3
Pima-Diabetes	768	8	2

it is very natural to think of using a neural network to combine ensemble results. An illustration of the multistage neural network model described here is shown in Figure 1.

In Figure 1, suppose there is a source data set $S\{s_1, s_2, ...s_n\}$ and its corresponding target data set $T\{t_1, t_2, ...t_n\}$, which are partitioned into two parts: test data and training data. The training data usually will be preprocessed by various methods in order to generate diverse results before they being applied to the first layer's neural network models: $N_1, N_2, ...N_w$. The preprocessing methods on the training data set include distributing sequences randomly, noise injection, bagging, boosting or other methods. After training, the test data set will be applied to these ensemble candidates to access their performance. Afterwards, the whole training data set will be applied and each first layer neural networks' corresponding results $(n_1, n_2, ...n_w)$ are used as the second layer neural network model's inputs. The second layer neural networks, was trained by using the first layers generated results on the whole training data as inputs combined with their target data set. Advantages of multistage ensemble neural networks include:

- Multistage ensemble neural networks can be applied to both classification and regression problems.
- For each ensemble candidate, normal data preparation methods can be applied to them, and ensemble candidates can be trained separately by using various different neural network models and algorithms.
- Generalization of these first layer neural networks when using this model can be tuned to be as diverse as possible, so the choices for the first layer neural network training are very flexible and allow a wide range of selections.

3 Experiments

To investigate the performance of multistage ensembles, six classification data sets are taken from UCI Machine Learning Depository to construct experimental datasets. Details of these data sets are listed in Table 1, where datasets marked with '*' are multi-class datasets.

4 Experiments

4.1 Data preparation

For each ensemble candidates' training, each data set was randomly partitioned into three parts: training, validation and test data. The sequences of training data were randomly distributed before they were applied to the neural networks model (i.e., steps were taken to prepare the most diverse data among ensemble members for training). There were no overlapping data instances inside the training data sets. For most of the data sets listed in Table 1, this approach was effective in generating diverse training data sets. The exception to this were the Glass and Iris data set, where the bagging [3] method was applied, due to source data's small size.

4.2 Experimental procedures

Five single hidden layer neural networks trained by the backpropagation algorithm were generated as ensemble candidates for each ensemble. These five candidates were constructed with different sequences of training data, different neural network structures (numbers of hidden neurons) and different initialization. Each neural network model was trained 20 times with random initialization of starting weights. The number of hidden neurons is changed from one to the number of inputs of each data set (time considerations prevented the exploration of networks with hidden layers larger than this). During training, the validation data set was used to prevent overfitting. As the experiments in this paper is concentrated on comparing the performance of two ensemble combination methods. To make the things simpliest, the test data set is applied to the ensemble candidates after their training. Those ensemble candidates with best generalisation performance were kept. Majority voting and multistage neural networks then applied to these same ensemble members to generate the combination results.

After the training of the first layer's ensemble candidates, the whole source data is randomly disturbed again. 10-fold cross-validation [4] is applied to the second layer's neural network training in order to estimate the average performance. First the training data is injected into the ensemble members and their outputs are concatenated together with the corresponding target values as the input of the second layer's neural network. The training procedure and parameter setting for a neural network combiner are the same as an ensemble candidate's training.

The result of the majority voting applied to 3 and 5 ensemble members are then compared with the performance of multistage neural networks by averaging over 10-fold cross-validation.

5 Results

Table 2 shows the results for these different combination strategies when combining the best three ensemble candidates and all five ensemble candidates. In this

Table 2. Percentage correct performance on test set of voting versus multistage networks on UCI datasets. S1..S5 signify single networks, where each '#' indicates one of the three ensemble members selected for three-member combination. V1 and V5 signify combination by voting with three and five ensemble members respectively, whilst M3 and M5 signify combination by multistage neural network with three and five ensemble members respectively.

Data	S1	S2	S3	S4	S5	V3	M3	V5	M5
Breast-cancer-w	97.73#	98.48#	95.45	96.21	96.97#	97.21	97.35	97.35	97.5
Bupa-Liver	75.0#	78.0#	77.0#	77.0	78.0	72.65	73.82	73.53	73.82
Glass	42.19	46.88#	50.0#	45.31#	45.31	58.17	58.50	54.29	54.76
Ionosphere	85.53	94.74#	93.42#	86.84#	86.84	96.0	96.29	96.0	96.29
Iris	93.33#	92.0#	94.67#	90.67	90.67	97.60	98.0	97.6	98.0
Pima-Diabetes	72.02	73.81#	72.02#	75.60#	69.64	74.47	75.13	73.95	74.61

table, the actual ensemble candidates selected for ensembles of three networks are marked with '#'.

From these results, it can be seen that multistage neural networks always perform better than majority voting based on the same ensemble members, regardless the number of ensemble members used in combination is 3 or 5.

6 Conclusions

This paper has demonstrated that, on a wide range of datasets (including simple categorization and multiple categorization), multistage neural network ensembles offer improved performance when compared with majority voting as an ensemble combination method. The experimental results clearly show that statistic improvement can be made by using this new ensemble model on this range data of sets, when compared with another widely used ensemble technique such as majority voting. Currently, experiments are being carried out to investigate the exact theoretical reason for this performance improvement offered by multistage neural networks. The reason probably lies within differences between the kinds of decision surfaces a second layer network can model when compared to those decision surfaces that can be produced by majority voting. However, a clearer analysis and description of this area needs to be developed. In the future the intention is to develop and implement further experiments to investigate if the performance of multistage neural networks can be enhanced by using more ensemble members in the first layer, choice of training, validation and test datasets, and choice of neural network for the second layer combiner. It may be that these factors interact, and will allow this research to push the performance of such ensembles even further.

References

1. Tumer, K., Ghosh, J.: Error correlation and error reduction in ensemble classifiers. Connection Science: Special Issue on Combining Artificial Neural Ensemble Approaches, 8(3&4), 385-404, 1999.
2. Sharkey, A. J. C., Sharkey, N. E., Chandroth, G. O.: Neural nets and diversity. Neural Computing and Applications, 4, 218-227, 1996.
3. Breiman, L.: Bagging predictors. Machine learning, 24 123-140, 1996.
4. Krogh, A., Vedelsby, J. : Neural Network Ensembles, Cross Validation, and Active Learning. Advances in Neural Information Processing Systems, 7, MIT press, Editors: Tesauro, G., Touretzky, D.S. and Leen, T.K. pp.231-238, 1995.
5. Partridge, D., Griffith, N. : Strategies for Improving Neural Net Generalisation. Neural Computing and Applications, 3, 27-37, 1995.
6. Wolpert, D. H.: Stacked generalization. Neural Networks, 5, 241-259, 1992.
7. Zeng, X., Martinez, T. R.: Using a Neural Network to Approximate an Ensemble of Classifiers. Neural Processing Letters, 12, 225-237, 2000.
8. Tumer, K., Ghosh, J.: Order statistics combiners of neural classifiers. In Proceedings of the World Congress on Neural Network, INNS press, Washington DC, 31-34, 1995.
9. Lincoln, W., Skrzypek, J.: Synergy of clustering multiple back propagation networks. Advances in Neural Information Processing Systems-2, Touretzky, D., (ed.), Morgan Kaufmann, 650-657, 1990.
10. Jacobs, R. A.: Methods for combining experts' probability assessments. Neural Computation, 7, 867-888, 1995.
11. Hansen, L., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Analysis and Machine Intell, 993-1001, 1990.
12. Al-Ghoneim, K., Kumar Vijaya B. V. K.: Learning ranks with neural networks. In Applications and Science of Artificial Neural Networks: Proceedings of the SPIE, 2492, 446-464, 1995.
13. Ho, T. K., Hull, J. J., Srihari, S. N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(1), 66-76, 1994.
14. Lee, D. S., Srihari, S. N.: Handprinted Digit Recognition: A Comparison of Algorithms. Pre-Proc. 3RD International Workshop On Frontiers In Handwriting Recognition, Buffalo, USA, 153-162, 1993.
15. Kittler, J.: Combining Classifiers: A Theoretical Framework. Pattern Analysis and Applications, 1, 18-27, 1998.

Appendix E: Publications (2)

Multistage Neural Networks: Adaptive Combination of Ensemble Results

Yang, S., Browne A., Picton, P.D., Hudson B. D. and Whitley, D.

The Proceedings of fourth International Conference on Recent Advances in Soft Computing, Nottingham, U.K. ISBN 1-84233-0764, 55-60, 2002.

Multistage Neural Networks: Adaptive Combination of Ensemble Results

Shuang Yang [1], **Antony Browne** [1], **Philip Picton** [2]

[1] School of Computing
London Metropolitan University
London EC3N 1JY, UK
E-mail: (syang@lgu.ac.uk)
[2] School of Technology and Design
University College Northampton, Northampton NN2, 6JD, UK

Brian D. Hudson [3] **and David Whitley** [3]

[3] Centre for Molecular Design
University of Portsmouth, Portsmouth PO1 2DY, UK

Abstract. *In the past decade, more and more research has shown that ensembles of neural networks (some times referred to as committee machines or classifier ensembles) can be superior to single neural network models, in terms of the generalization performance they can achieve on the same datasets. In this paper, we propose a novel trainable neural network ensemble combination schema: multistage neural network ensembles. Two stages of neural network models are constructed. In the first stage, neural networks are used to generate the ensemble candidates. The second stage neural network model approximates a combination function based on the results generated by the ensemble members from the first stage. A sample of the data sets from UCI Machine Learning Depository and a human Gene data set are modeled using multistage neural networks and a comparison of the performance between multistage neural networks and a majority voting scheme is conducted.*

Keywords: *neural network ensemble, combination strategy, multistage ensembles.*

1. Introduction

It is well known that combining a set of neural network classifiers whose error distributions are diverse can lead to the generation of superior results than those achieved by any single classifier. The most common combination strategies used to combine the results of individual ensemble members are simple averaging [10,13], weighted averaging [6], majority voting [3] and ranking [2,4]. These are catalogued as fixed combination schemes, which require no prior training. One deficiency of such schemes feature is that weightings for the importance of the output of each ensemble member must be pre-chosen and then applied to produce the combination result.

There is a trend to explore trainable combination methods which aim to find the optimal weights to be combined [7,9,11,15,16]. In this paper, we propose a new model for ensemble combination, based on another neural network layer, called multistage neural network ensembles. Unlike the conventional combination schema, the weights assigned to ensemble members will be generated

automatically by another neural network layer. The new model outlined here inherits some ideas from stacking [14] and develops them further.

2. Multistage Neural Networks

It appears attractive to make the combination process adaptive, so that no a-priori (and possibly incorrect) combination weightings need to be chosen. Therefore, a model is proposed where the procedure of combining ensemble classifiers is turned into the training of another neural network model. The structure of the multistage neural network ensembles is shown in Figure 1.

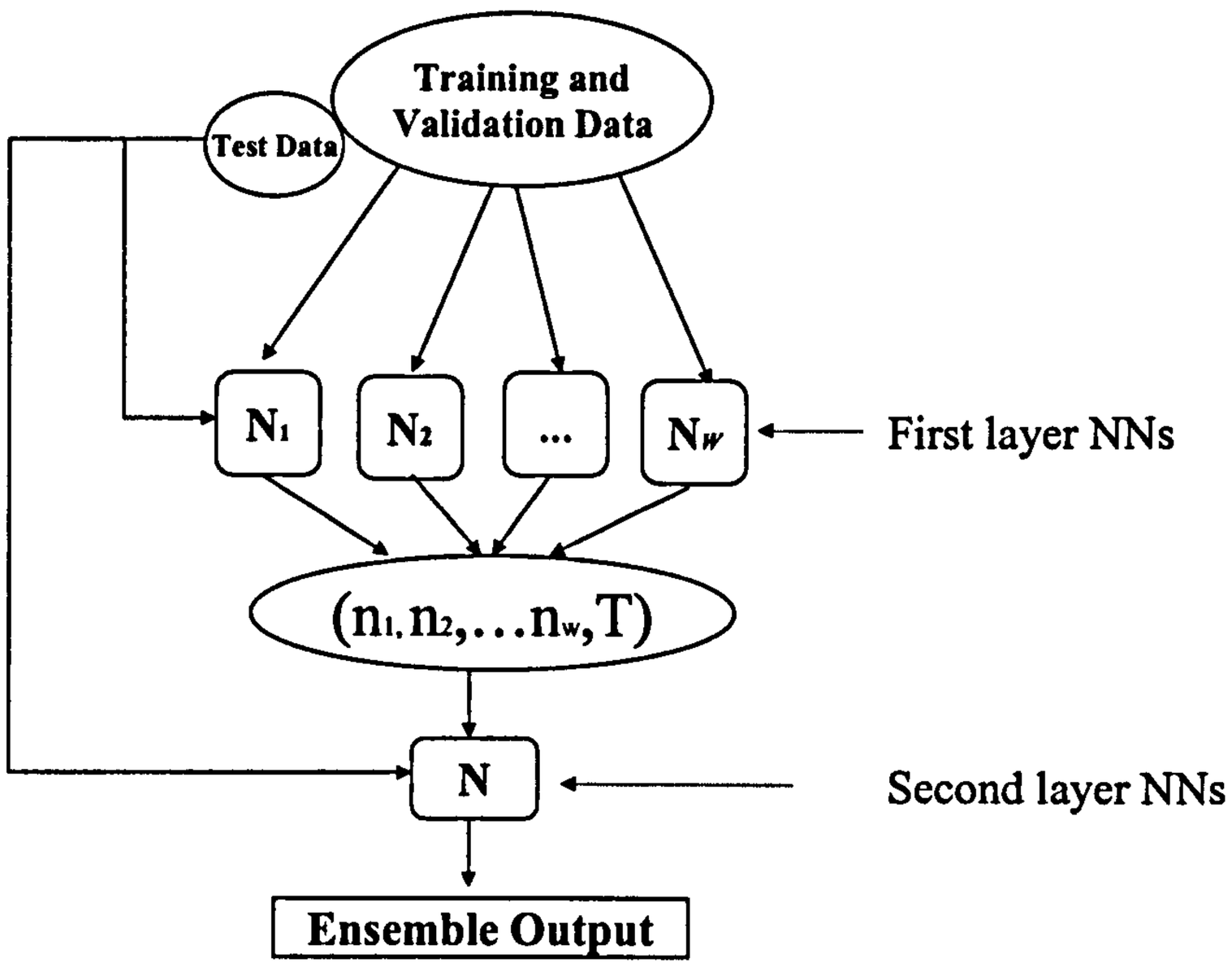

Figure 1. Illustration of multistage ensemble neural networks

Referring to figure 1, suppose there is a source data set $S\{s_1, s_2, ...s_n\}$ with its corresponding target data set $T\{t_1, t_2 ..t_m\}$ are partitioned into three parts: test data, training data and validation data. The training data usually are pre-processed by various methods in order to generate diverse results before they are applied to the first layer's neural network models: $N_1, N_2 ...N_W$. The pre-processing methods on training data set include distributing sample presentation sequences randomly, noise injection, bagging, boosting or other methods. After training, the test data set are applied to these ensemble candidates to access their performance. Afterwards the whole training data is applied and each first layer neural networks' corresponding results $(n_1, n_2 ...n_w)$ are used as the second layer neural network model's inputs. The second layer neural networks is trained by using the first layers generated results on the whole training data as inputs combined with their

target data set.

The advantages of multistage ensemble neural networks are:

- For each ensemble candidate, the normal data preparation methods can be applied and ensemble candidates will be trained separately by using various different neural network models and algorithms. The generalization of these first layer neural networks are expected to be as diverse as possible in order to get the best ensemble result. Because of this, the choices for the first layer neural network training are very flexible and allow a wide range of selections.
- The second layer of a single neural network is used as an ensembles' combiner to adaptively apply different weights to those first layer ensemble members in order to approximate a best combination function. There is a strong reason to believe that it can more accurately adjusting the weights which are assigned to ensemble members than manual methods.
- Multistage ensemble neural networks can be applied to both classification and regression problem, unlike some other ensemble combination techniques.

3. Experimental Results

Five classification data sets taken from UCI Machine Learning Depository (http://www.ics.uci.edu/mlearn/MLRepository.html) and a splice data set from human DNA sequences [5,12] were used for these experiments. Since predicting splice junction sites is crucial in finding gene coding. The attempt using splice data set in this paper would explore the effectiveness of the multistage neural networks in the application of finding human gene sequences. The details of these data sets are listed in Table 1.

Table 1. Summary of the data sets

Data	Number of Cases	Number of Input features	Number of Output features
Breast-cancer –w	682	9	2
Bupa-Liver	345	6	2
Ionosphere	351	34	2
Iris*	150	4	3
Pima-Diabetes	768	8	2
Splice-Acceptor	1555	10	2

Notations: * - multi-class data set.

For each ensemble candidates' training, each data set was randomly partitioned into three parts: training, validation and test data. The sequences of training data were randomly distributed before they were applied to the neural networks model (i.e., steps were taken to prepare the most diverse data among ensemble members for training). There were no overlapping data instances inside the training data sets. For most of the data sets listed in Table 1, this approach was effective in generating diverse training data sets. The exception to this was Iris data set, where the bagging [1] method was applied, due to the source data's small size.

Nine single hidden layer neural networks trained by the backpropagation algorithm were generated as ensemble candidates for each ensemble. These nine candidates were constructed with different sequences of training data, different neural network structures (numbers of hidden

neurons) and different initialization. Each neural network model was trained 20 times with random initialization of starting weights. The number of hidden neurons was changed from two to the number of inputs of each data set (time considerations prevented the exploration of networks with hidden layers larger than this). During training, the validation data set was used to prevent overfitting. As the experiments in this paper was concentrated on comparing the performance of two ensemble combination methods, to make the things simpler, the test data set was applied to the ensemble candidates after their training. Those ensemble candidates with the best generalization performance were kept. Majority voting and multistage neural networks were then applied to these same ensemble members to generate the combination results. These nine single neural network's average performances of each data set (except Splice-Acceptor data whose performance was scaled on its own independent test data set) on the test data, which were generated from ten-fold cross validation approach are illustrated in table 2.

Table 2. Ensemble candidates performance

Data	Single Neural Networks Performance (%)								
	1	2	3	4	5	6	7	8	9
Breast –cancer-w	97.21	97.50	97.21	96.62	96.76	97.06	97.35	97.06	97.50
Bupa-Liver	72.06	72.65	71.76	73.82	72.06	72.94	71.76	72.35	72.65
Ionosphere	95.43	94.86	95.14	93.14	94.29	93.71	94.86	90.57	92.57
Iris *	97.60	96.80	98.4	96.00	96.00	98.00	98.40	99.20	97.60
Pima- Diabetes	73.03	74.08	72.50	74.21	71.32	69.87	68.68	70.00	69.74
Splice-Acceptor	88.44	88.00	88.44	87.11	89.33	87.33	86.00	87.56	88.22

After the training of the first layer's ensemble candidates, the whole source data was randomly distributed again. Ten-fold cross-validation [8] was applied to the second layer's neural network training in order to estimate the average performance. The exceptional case was the Splice-Acceptor data set. For this data set, there were independent 1105 instances of training data and 450 instances of test data respectively. Hence it was not necessary to use any cross-validation approach. The procedure of the second layer of neural network model training is: First, the training data was injected into the ensemble members and their outputs were concatenated together with the corresponding target values as the input of the second layer's neural network. Second, The parameters of the second layer neural network model were chosen to be fixed: learning rate was 0.1 and the number of hidden neurons was two. These parameter settings were based on the empirical observation of a large number of training runs. It was noticed that the changing of parameters setting didn't effect the final combination result much, based on the experiments using the above datasets. In most of cases the best parameters combination was: learning rate of 0.1 and two hidden neurons. However the initialization does contribute strongly to the final result. Therefore, the second layer neural network model was trained 1000 times with random initialization of starting weights.

The result of the majority voting applied to three, five and nine ensemble members were then compared with the performance of multistage neural networks by averaging over ten-fold cross-validation. In Table 4, all the results shown for the voting and multistage neural network are based on the same ensemble members for each group. The details for each combination group can be seen in Table 3. The figures present the average performance of the two ensemble combination methods except the Splice-Acceptor data set who has independent test data were shown in Table 4.

Table 3. The selection of ensemble members

Data	Th actual ensemble members for each groups of ensemble models		
	3	5	9
Breast-cancer –w	1,2,5	1,2,3,4,5	1,2,3,4,5,6,7,8,9
Bupa-Liver	1,2,3	1,2,3,4,5	1,2,3,4,5,6,7,8,9
Ionosphere	2.3.4	1,2,3,4,5	1,2,3,4,5,6,7,8,9
Iris *	1,2,3	1,2,3,4,5	1,2,3,4,5,6,7,8,9
Pima- Diabetes	2,3,4	1,2,3,4,5	1,2,3,4,5,6,7,8,9
Splice-Acceptor	4,5,7	2,3,4,5,7	1,2,3,4,5,6,7,8,9

Table 4. Ensemble performance comparison between multistage neural networks and majority voting

Data	Voting Accuracy (%)			Multistage Neural Networks Accuracy ((%)		
	3	5	9	3	5	9
Breast-cancer -w	97.21	97.35	97.65	97.35	97.5	97.8
Bupa-Liver	72.65	73.53	73.82	73.82	74.12	74.41
Ionosphere	96.0	96.0	95.72	96.0	96.29	96.57
Iris *	98.8	97.6	98.8	98.8	98.8	99.2
Pima- Diabetes	74.47	73.95	73.42	75.13	75.53	76.58
Splice-Acceptor	89.11	88.89	88.67	89.33	89.33	89.78

From these results, it can be seen that multistage neural networks always perform better than majority voting based on the same ensemble members, regardless the number of ensemble members used in combination was three, five or nine. The results have demonstrated that, on a wide range of data sets (including simple categorization and multiple categorization), multistage neural network ensembles offer improved performance when compared with majority voting as an ensemble combination method.

The intention of using different number of ensemble members being combined was to find whether the ensemble numbers will effect the performance of multistage neural networks. The current experimental results show that the multistage approach generalization improved steadily while the number of ensemble numbers was increased.

4. Conclusion

This paper has demonstrated that multistage ensembles, where the adaptive properties of a second layer network are used to combine the outputs of the individual ensemble members, offer enhanced performance over a simple voting based combination method. Currently, experiments are being carried out to investigate the exact theoretical reason for this performance improvement. The reason probably lies within differences between the kinds of decision surfaces a second layer network can model when compared to those decision surfaces that can be produced by majority

voting. However, a clearer analysis and description of this area needs to be developed. In future, the effect of the choice of training, validation and test data sets, and choice of neural network architecture for the second layer combiner will be investigated. It may be that these factors interact, and will allow this research to push the performance of such ensembles even further.

References

[1] L. Breiman (1996). Bagging predictors, *Machine learning*, vol. 26(2), pages 123-140.

[2] K. Ghoneim and B.V.K. Vijaya (1995). Learning ranks with neural networks, *Applications and Science of Artificial Neural Networks: Proceedings of the SPIE*, vol. 2492, pages 446-464.

[3] L. Hansen and P. Salamon (1990). Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 12, pages 993-1001.

[4] T.K. Ho, J.J. Hull and S. N. Srihari (1994). Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16(1), pages 66-76.

[5] B. Hudson, D. Whitley and A. Browne (2002). Biological data mining:high fidelity rule extraction from trained neural networks. *Proceddings of the Fourteenth European Symposium on Quantitative Structure-Activity Relationships,* Bournemouth, UK (in press).

[6] R.A. Jacobs (1995). Methods for combining experts' probability assessments. *Neural Computation*, vol. 7, pages 867-888.

[7] J. Kittler (1998). Combining Classifiers: A Theoretical Framework. *Pattern Analysis &* Application, vol. 1, pages 18-27.

[8] A. Krogh and J. Vedelsby (1995). Neural Network Ensembles, Cross Validation, and Active Learning. *Advances in Neural Information Processing Systems*, MIT press, G. Tesauro, D.S. Touretzky and T.K. Lee (eds.), vol. 7, pages 231-238.

[9] D.S. Lee and S.N. Srihari (1993). Handprinted Digit Recognition: A Comparison of Algorithms. *Pre-Proc. 3RD International Workshop On Frontiers In Handwriting Recognition,* Buffalo, USA. pages 153-162.

[10] W. Lincoln and J. Skrzypek (1990). Synergy of clustering multiple back propagation networks. *Advances in Neural Information Processing Systems-2,* Morgan Kaufmann, D. Touretzky (ed.), pages 650-657.

[11] D. Partridge and N. Griffith (1995). Strategies for Improving Neural Net Generalisation. *Neural Computing and Applications,* vol. 3, pages 27-37.

[12] T.A. Thanaraj (1999). A clean data set of EST-confirmed splice sites from Homo sapiens and standards for clean-up procedures. *Nucleic Acids Res.*,vol. 27(13), pages 2627-2637.

[13] K. Tumer and J. Ghosh (1995). Order statistics combiners of neural classifiers. *Proceedings of the World Congress on Neural Networks* , Washington D.C.: INNS press, pages 31-34.

[14] D.H. Wolpert (1992). Stacked generalization. *Neural Networks,* vol. 5, pages 241-259.

[15] S. Yang, A. Browne and P.D. Picton (2002) Multistage neural network ensembles. *Lecture Notes in Computer Science:Proceedings of the Third International Workshop on Multiple Classifier Systems*, Cagliari, Italy, Heidelberg, Springer., vol. 2364, Pages 91-97.

[16] X. Zeng and T.R. Martinez (2000). Using a Neural Network to Approximate an Ensemble of Classifiers. *Neural Processing Letters,* vol. 12, pages 225-237.